Web aplikacija za oglašavanje i najam objekata u turizmu

Koraca, Sandro

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **University of Rijeka / Sveučilište u Rijeci**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:195:896554

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-19**

Sveučilište u Rijeci
Fakultet informatike i digitalnih tehnologija

Repository / Repozitorij:

Repository of the University of Rijeka, Faculty of Informatics and Digital Technologies - INFORI Repository

zir.nsk.hr

DIGITALNA KNJIŽNICA

DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

Sveučilište u Rijeci – Fakultet informatike i digitalnih tehnologija

Preddiplomski sveučilišni studij Informatika

Sandro Koraca

Web aplikacija za oglašavanje i najam objekata u turizmu

Završni rad

Mentor: Doc. dr. sc. Lucia Načinović Prskalo

Rijeka, 16.9.2022.

FAKULTET INFORMATIKE I DIGITALNIH TEHNOLOGIJA
SVEUČILIŠTE U RIJECI

Rijeka, 22.4.2022.

Zadatak za završni rad

Pristupnik: Sandro Koraca

Naziv završnog rada: Web aplikacija za oglašavanje i najam objekata u turizmu

Naziv završnog rada na eng. jeziku: Web application for advertising and renting facilities in tourism

Sadržaj zadatka: Zadatak završnog rada je izrada web aplikacije za oglašavanje apartmana te opis svih koraka izrade aplikacije. Također, u radu će biti opisani svi alati koji su se koristili prilikom izrade aplikacije te njihove značajke. Naposljetku će se prikazati i opisati funkcionalnosti izrađene web aplikacije.

Mentor

Lucia Načinović Prskalo

Voditelj za završne radove

Doc. dr. sc. Miran Pobar

Zadatak preuzet: 25.5.2022.

(potpis pristupnika)

Sažetak

Kroz ovaj završni rad detaljno je opisan proces izrade aplikacije za iznajmljivanje apartmana u alatu Django. Detaljno su pojašnjene funkcionalnosti alata Django i njegovi dodaci – dodatak za obrasce Crispy forms, dodatak za autentifikaciju Allauth te Tailwind za izradu *frontend*-a. Aplikacija podržava tri vrste korisnika, a to su neregistrirani korisnik, registrirani korisnik te administrator. Svaka od tri vrste ima određeni skup dozvola i mogućnosti. Registrirani korisnik primjerice može dodavati nove apartmane te uređivati i brisati svoje, a administrator može odobravati novododane ili uređene apartmane.

Ključne riječi

Django, Crispy forms, Allauth, Tailwind, Heroku, Python, apartman, aplikacija, web aplikacija, model

Sadržaj

1. Uvod 5
1.1 Opis tehnologija 5
1.1.1 Django 5
1.1.2 Tailwind 6
1.1.3 Heroku 6
1.2 Paradigma MVC 6
2. Instalacija, priprema i opis elemenata 8
2.1 Instalacija i priprema projekta 8
2.2 Models.py 15
2.3 Admin.py 16
2.4 Forms.py 17
2.5 Views.py 18
2.6 Main/urls.py 22
2.7 AptBooker/urls.py 23
2.8 HTML datoteke 24
3. Prikaz kreirane aplikacije 27
3.1 Neprijavljeni korisnik 27
3.2 Prijavljeni korisnik 32
3.3 Korisnik prijavljen kao administrator 36
4. Zaključak 40
5. Popis slika 41
6. Literatura 42
7. Popis priloga 43

1. Uvod

Kroz ovaj završni rad opisan je postupak izrade aplikacije za iznajmljivanje apartmana u alatu Django. Isto tako, opisano je i kako se koristi dodatak za forme koji se zove Crispy forms te dodatak za autentifikaciju Allauth. Krajnji rezultat rada je "AptBooker" - web aplikacija koja pruža korisnicima pregled i dodavanje apartmana. Kako je Hrvatska turistička zemlja, ovakva aplikacija olakšala bi korisnicima rezervaciju svog smještaja u Hrvatskoj. Isto tako olakšala bi najmodavcima oglašavanje svog apartmana kako bi ga mogli vidjeti potencijalni klijenti.

Aplikacija ima tri vrste korisnika, a to su neregistrirani korisnik, registrirani korisnik te administrator. Neregistrirani korisnik može samo pregledavati objavljene apartmane. Registrirani korisnik osim pregledavanja objavljenih apartmana može i dodavati nove apartmane, te uređivati i brisati svoje apartmane. Administrator može sve što mogu neregistrirani i registrirani korisnici no ima dodatnu funkciju, a to je da može odobravati apartmane kako bi postali javni da ih korisnici mogu vidjeti. Ovo znači da kada registrirani korisnik doda novi apartman ili uredi postojeći on je neobjavljen te mora proći kroz kontrolu administratora koji odluči hoće li objaviti apartman ili će ga obrisati. Dodatna provjera je uvedena kako bi se spriječilo objavljivanje neželjenog sadržaja preko aplikacije.

1.1 Opis tehnologija

U nastavku su opisane tehnologije koje su se koristile u izradi aplikacije.

1.1.1 Django

Django je softver otvorenog koda te koristi paradigmu MVC (model-view-controller) pomoću koje Django prikazuje ekrane korisniku aplikacije. Iza njega stoji neprofitabilna nezavisna zaklada Django Software Foundation (DSF). Kako bi zaklada financijski funkcionirala, a pošto je korištenje Djanga besplatno, postoje korporativni članovi koji pomažu po tom pitanju. Obzirom da je Django otvorenog koda, ako DSF zaključi da je neki volonter napravio nešto korisno što se može implementirati u Django, zajednica DSF nagradi tog volontera. Programski jezik kojim se Django koristi je Python (Django Software Foundation, 2022).

1.1.2 Tailwind

Tailwind je besplatan CSS alat otvorenog koda koji olakšava izradu dizajna web stranica. Kompanija koja razvija Tailwind je Tailwind Labs, a programski jezik u kojem rade je JavaScript. Ima predefinirane klase koje je dovoljno dodati u HTML-u u "class" varijablu. Na taj način je puno lakše dizajniranje jer nije potrebno u vlastitoj CSS datoteci definirati klase za svaki element. Također, pošto je otvorenog koda moguće je prilagoditi klase po vlastitoj potrebi (Tailwind CSS, 2022).

1.1.3 Heroku

Heroku je online platforma u oblaku na koju se može postaviti vlastite aplikacije da budu dostupne online. Heroku podržava više jezika među kojima su Noje.js, Ruby, Java, PHP, Go, Scala, Clojure te Python koji je potreban za ovu aplikaciju zato što Django koristi Python. Za neprofitabilne aplikacije Heroku je besplatan, ali moguće je i za veće aplikacije platiti te dobiti veće pogodnosti. (Heroku, 2022)

Aplikacija se nalazi na poveznici https://aptbooker.herokuapp.com/, a podaci za administratora su:

- Korisničko ime: administrator
- Lozinka: admin

Kod aplikacije dostupan je na linku: https://github.com/SandroKoraca/AptBooker

1.2 Paradigma MVC

Django koristi paradigmu model-view-controller (MVC) koju poziva model-template-view (MTV) (Petrović, 2022). Podaci se na ekranu prikazuju na način da se prolazi kroz tri elementa: model, pogled i kontroler. Točnije, u praksi, prvo se koristi kontroler, zatim pogled te na posljetku, ukoliko je potrebno, može se koristiti model.

Model se koristi za definiranje klasa, odnosno modela, koje se spremaju u bazu podataka. Točnije, u modelu se definiraju klase koje označavaju tablice u bazi podataka. Za svaku klasu definiraju se varijable koje se odnose na tu klasu. Također, u bilo kojem trenutku moguće je dodati nove modele ili izmijeniti postojeći model. Sve što je potrebno na završetku je migrirati izmjene.

View se koristi za korisničko sučelje. Točnije, koristi se za generiranje HTML-a na način da Django uzima zahtjev, zatim s tim zahtjevom napravi što je potrebno te vraća odgovor na taj zahtjev.

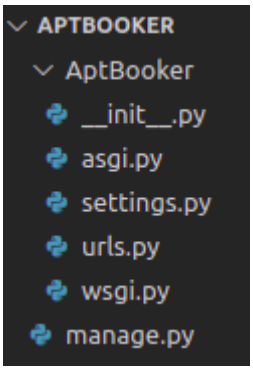
Controller se koristi za logiku oko povezivanja svega u cjelinu u svrhu pružanja odgovora na korisnički zahtjev. Controller također uspostavlja vezu s bazom podataka i učitava datoteke.

2. Instalacija, priprema i opis elemenata

2.1 Instalacija i priprema projekta

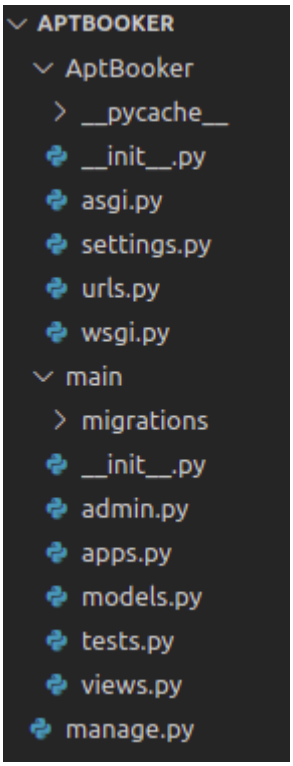
Instalacija Djanga izvršava se na način da se pokrene naredba “pip3 install Django”. Pomoću nje se instalira Django paket te je nakon završetka instalacije Django spreman za uporabu.

Kako bi se kreirao novi projekt, pozicioniramo se u direktorij koji želimo te pokrenemo naredbu za kreiranje projekta “django-admin startproject AptBooker”. Nakon kreiranja projekta pozicioniramo se u naredni direktorij, odnosno u direktorij “AptBooker”. Unutar tog direktorija postoji “manage.py” datoteka pomoću koje u terminalu pokrećemo razne naredbe vezane za aplikaciju. Postoji još jedan direktorij s istim nazivom. Taj direktorij je glavni dio aplikacije, odnosno on je srž aplikacije pomoću koje se sve pokreće. Unutar tog direktorija postoje dvije ključne datoteke, a to su “setting.py” i “urls.py”. U datoteku “setting.py” postavljaju se razne postavke koje su ključne za aplikaciju, dok se u datoteku “urls.py” pišu sve URL putanje koje su vezane za aplikacije unutar glavne aplikacije. Sadržaj aplikacije prikazan je slici 1.



Slika 1 Početne datoteke

Sljedeći korak je kreiranje aplikacije “main” pomoću koje će funkcionirati cijela glavna aplikacija. Kreiranje aplikacije radi se na način da se pokrene naredba “./manage.py startapp main”. Pomoću ove naredbe kreirala se mapa “main” koja označava kreiranu aplikaciju u kojoj se nalaze razne datoteke koje će biti od koristi. Nadalje, kako bi se prikazivale URL putanje koje će se definirati u datoteci “main/urls.py” potrebno je u datoteku “AptBooker/urls.py” dodati liniju koda unutar varijable urlpatterns koja glasi “path('', include('main.urls')),” kao na slici 20. Unutar datoteke “AptBooker/settings.py” unutar varijable “INSTALLED_APPS” još dodamo “'main.apps.MainConfig',”. Sadržaj aplikacije nakon dodavanja „main“ dijela prikazan je slici 2.



Slika 2 Datoteke nakon dodavanja "main" aplikacije

Kako bi sve html datoteke bile na jednom mjestu u datoteci “AptBooker/settings.py”, unutar varijable “TAMPLATES” gdje je “DIRS” unutar vitičastih zagrada doda se “'./templates'” (slika 3). Potrebno je kreirati direktorij “templates“ unutar direktorija “AptBooker”. Ovaj direktorij koristit će se za sve html datoteke, točnije unutar tog direktorija bit će sve html datoteke vezane za rad aplikacije.

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': ['./templates'],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

Slika 3 Postavljanje direktorija za templates

Na ovaj način pripremljena je aplikacija za daljnje razvijanje, a unutar ovog projekta koriste se slike u modelu, Crispy forme, sustav autentifikacije te Tailwind. Također potrebno je dodati postavke od kuda će se učitati statičke datoteke poput css-a i slika koje nisu vezane za apartmane.

Kako bi se u modelu mogla koristiti “ImageField” opcija te kako bi se slike spremale na jedno određeno mjesto potrebno je unutar “AptBooker/setting.py” dodati liniju koda “MEDIA_ROOT = os.path.join(BASE_DIR, 'media')”. Na ovaj način omogućilo se da se sve slike koje korisnik postavi za sliku apartmana spreme u direktorij “media”. Kako bi se na njih moglo kliknuti i pokazati uvećano, potrebno je dodati i liniju koda “MEDIA_URL = '/media/'”. Prikazano na slici 4.

```
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
MEDIA_URL = '/media/'
```

Slika 4 Postavljanje "media" direktorija

Nadalje, u ovom projektu koristit će se Crispy forme koje olakšavaju način korištenja formi. Za instalaciju Crispy forms potrebno je pokrenuti naredbu “pip install django-crispy-forms". Nakon instalacije unutar datoteke “AptBooker/settings.py” unutar varijable “INSTALLED_APPS” potrebno je dodati “'crispy_forms',” kao što je prikazano na slici 5. Na ovaj način je pripremljeno korištenje Crispy formi.

Kako se u ovom projektu koristi sustav autentifikacije, potrebno je instalirati i Allauth. To se radi na način da se pokrene naredba “pip install django-allauth". Nakon instalacije potrebno je unutar datoteke “AptBooker/settings.py” dodati neke dodatne opcije. Unutar varijable

"INSTALLED_APPS" dodaje se "'django.contrib.sites', 'allauth', 'allauth.account', 'allauth.socialaccount'," kao na slici 5.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'main.apps.MainConfig',
    'crispy_forms',
    'django.contrib.sites',
    'allauth',
    'allauth.account',
    'allauth.socialaccount',
]
```

Slika 5 Postavljanje instaliranih aplikacija

Također, u "AptBooker/settings.py" potrebno je dodati i "SITE_ID = 1" kako bi se označilo da je identifikator stranice broj 1. Ovo je važno jer ako toga nema neće se prikazivati stranice vezane za autentifikaciju. Potrebno je dodati i putanje na koje će se preusmjeravati prilikom prijave, kreiranja profila i odjave. Zato je u postavkama potrebno dodati sljedeće tri linije: "ACCOUNT_LOGOUT_REDIRECT_URL = '/'", "ACCOUNT_SIGNUP_REDIRECT_URL = '/'" te "LOGIN_REDIRECT_URL = '/'" kao na slici 6. Nadalje, u "AptBooker/urls.py" potrebno je u varijablu urlpatterns dodati "path('accounts/', include('allauth.urls'))," kako bi se prikazivale stranice autentifikacije. Na ovaj način pripremljeno je sve za korištenje autentifikacije.

```
SITE_ID = 1

ACCOUNT_LOGOUT_REDIRECT_URL = '/'
ACCOUNT_SIGNUP_REDIRECT_URL = '/'
LOGIN_REDIRECT_URL = '/'
```

Slika 6 Postavljanje poveznica za autentifikaciju

Kako bi se olakšalo dizajniranje *frontend*-a, koriti se Tailwind. Pomoću tog sustava nije potrebno raditi zasebne klase kako bi se uredio izgled stranice već se koriste predefinirane klase.

Kako bi se instalirao Tailwind, u terminal je potrebno pokrenuti naredbu “python -m pip install django-tailwind". Nakon instalacije u datoteci “AptBooker/settings.py” unutar varijable “INSTALLED_APPS” treba dodati “'tailwind',” kako bi se Tailwind mogao pokrenuti. Nakon toga, unutar mape “AptBooker” u terminalu je potrebno kreirati aplikaciju Tailwind CSS koristeći naredbu “./manage.py tailwind init”. Prilikom pokretanja naredbe pita se da se unese naziv aplikacije te se tu može staviti naziv po želji, a u ovom projektu je to “theme”. Zatim je tu aplikaciju potrebno staviti u instalirane aplikacije unutar postavki. Zato je u datoteku “AptBooker/setting.py” unutar varijable “INSTALLED_APPS” potrebno dodati naziv kreirane Tailwind aplikacije, u ovom slučaju “theme”. Također, unutar postavki potrebno je dodati i “TAILWIND_APP_NAME = 'theme'” kako bi se označilo kako se zove Tailwind aplikacija. Zatim je u postavke potrebno dodati “INTERNAL_IPS = ["127.0.0.1",]”. Kako ne bi trebalo stalno na novo učitavati stranicu, može se koristiti Djangovo automatsko učitavanje stranica. Ovo se radi na način da se u postavkama unutar varijable “INSTALLED_APPS” doda “django_browser_reload',”. Sadržaj za ovaj dio unutar postavki prikazan je na slici 7.

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    'main.apps.MainConfig',
    'crispy_forms',
    'django.contrib.sites',
    'allauth',
    'allauth.account',
    'allauth.socialaccount',
    'tailwind',
    'theme',
    'django_browser_reload',
]

TAILWIND_APP_NAME = 'theme'

INTERNAL_IPS = [
    "127.0.0.1",
]
```

Slika 7 Postavljanje postavki za TailWind

Zatim unutar varijable “MIDDLEWARE” dodajemo “”django_browser_reload.middleware.BrowserReloadMiddleware",” (slika 8) te u datoteci “AptBooker/urls.py” unutar urlpatterns dodajemo “path("__reload__/", include("django_browser_reload.urls")),” kao na slici 20. Pomoću ovog svaki put kad se spremi datoteka i kad se otvori web stranica, ona će biti automatski sama na novo učitana te se tako ubrzava rad oko dizajniranja. Nadalje, potrebno je u terminalu pokrenuti “./ manage.py tailwind install” te u vanjskom terminalu unutar projekta pokrenuti server za Tailwind komandu “./manage.py tailwind start”. Na ovaj način je sve spremno za korištenje Tailwind-a u svrhu lakšeg dizajniranja izgleda web stranica. Također, vrlo je bitno napomenuti da je potrebno imati i instaliran Node.js na računalu kako bi sve ispravno funkcioniralo.

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'django_browser_reload.middleware.BrowserReloadMiddleware',
]
```

Slika 8 Postavljanje middleware-a za automatsko učitavanje stranice

Potrebno je definirati i od kud će se učitavati statične datoteke kao što su css datoteke, slike koje developer doda kako bi se prikazale kao *cover* i slično. Iz tog razloga u postavkama treba dodati sadržaj kao na slici 9.

```
STATIC_URL = 'static/'

STATICFILES_DIRS = [
    os.path.join(BASE_DIR, "static"),
    '/var/www/static/',
]
```

Slika 9 Postavljanje postavki za statične datoteke

Zatim je unutar direktorija “AptBooker/main” potrebno kreirati direktorij “static” unutar kojeg se kasnije dodavaju statične datoteke. Tako npr. ako se želi koristiti prilagođeni css, unutar tog direktorija se kreira direktorij “css” te u njemu css datoteka ili se za slike kreira direktorij

"images" te se u njega stave slike. U ovom se projektu koristi Tailwind pa nije potrebno korištenje prilagođenog css-a.

2.2 Models.py

Ovaj projekt sadrži tri modela koji se odnose na apartman, recenzije i korisnike. No, model za korisnike se ne nalazi u "models.py" zato što je ovaj model predefiniran u Djangu, a u taj model se spremaju svi korisnici koji se prijave u aplikaciju. Ovaj model korišten je da se automatski postavi tko je vlasnik apartmana i recenzije kako korisnik ne bi morao sam dodavati da je on dodao apartman ili da je napisao recenziju.

Model "Apartman" sadrži varijable "apartman_naziv", "apartman_cijena", "apartman_vlasnik", "apartman_smijestaji", "apartman_lokacija", "apartman_odobren", "apartman_sadrzaj", "apartman_glavna_slika". Isto tako, ovaj model ima skrivenu varijablu "id" koja se zapravo sama doda u svakom modelu općenito, a u ovom projektu je korisna kada se u views.py uzima točno odabrani apartman. Varijabla "apartman_naziv" je tipa "CharField" te je najveća dužina postavljena na 100 znakova, a varijabla je korištena kako bi se u nju spremao naziv apartmana. Varijabla "apartman_cijena" također je tipa "FloatField", a ova varijabla predstavlja cijenu apartmana po jednoj noći. Varijabla "apartman_vlasnik" je varijabla tipa "CharField" te je najveća dužina postavljena na 100 znakova. Ova varijabla se koristi kako bi se u nju spremilo korisničko ime vlasnika tog apartmana, a korisnik i administrator nemaju pristup toj varijabli, tj. ne mogu postaviti ili uređivati tu varijablu već se ona automatski popunjava i ne prikazuje se kad se dodaje novi apartman. Njena vrijednost se prikazuje samo kad se ispisuju pojedinosti apartmana. Varijabla "apartman_smjestaj" koristi se kako bi se u nju spremio broj ležećih mjesta odnosno koliko ljudi može prenoćiti u apartmanu. Ova varijabla je tipa "IntegerField". Varijabla "apartman_lokacija" je tipa "CharField" te se u nju može spremiti najviše 100 znakova, a ona služi za pohranjivanje adrese apartmana. Varijabla "apartman_odobren" je tipa "BooleanField" te je neoznačena po zadanoj vrijednosti kako bi se apartman kad se doda prvo prikazao administratoru da ga provjeri te da ga odobri ako je sve kako treba. Ovoj varijabli korisnik koji nije administrator nema pristup. Varijabla "apartman_sadrzaj" koristi se kako bi u nju korisnik koji dodaje apartman mogao napisati sve što apartman sadrži te ono što bi moglo zanimati potencijalnog klijenta, a varijabla je tipa "TextField". Varijabla "apartman_glavna_slika" je tipa "ImageField", a parametri koji se koriste su "blank" i "null" koji su postavljeni na "True" kako bi se omogućilo dodavanje apartmana i bez da se postavi njegova slika, te "upload_to" koji služi za definiranje mjesta spremanja slike kad se doda apartman. Za spremanje slika koristi se dodatna funkcija "image_path" koja ima parametre "instance" koja predstavlja svaki pojedini objekt te "filename" koja predstavlja kako se slika zove. Ova funkcija služi kako bi se slike sortirale po identifikatoru pojedinog apartmana odnosno da svaki apartman ima svoju. Model za apartman te funkcija „image_path" prikazani su na slici 10.

```python
def image_path(instance, filename):
    return (str(instance.id)+"/"+filename)

class Apartman(models.Model):
    apartman_naziv = models.CharField(max_length=100)
    apartman_cijena = models.FloatField()
    apartman_vlasnik = models.CharField(max_length=100)
    apartman_smjestaji = models.IntegerField()
    apartman_lokacija = models.CharField(max_length=100)
    apartman_odobren = models.BooleanField(default=False)
    apartman_sadrzaj = models.TextField()
    apartman_glavna_slika = models.ImageField(upload_to=image_path, blank=True, null=True)
```

Slika 10 Model za apartman i pomoćna funkcija

Model "Recenzija" ima varijable "autor", "apartman", "sadrzaj", "ocjena". Varijabla "autor" kao i varijabla "apartman" same se popunjavaju prilikom popunjavanja forme te su one skrivene od korisnika. Varijabla "autor" je tipa "CharField" te podržava 100 znakova, a u nju se sprema korisničko ime prijavljenog korisnika kako bi se znalo tko je napisao recenziju. Varijabla "apartman" isto je tako tipa "CharField" te podržava 10 znakova što je dovoljno za spremanje identifikatora apartmana za koji se piše recenzija kako bi se kasnije mogli prikazivati samo recenzije tog točnog apartmana, te da se izbjegne prikazivanje svih recenzija za svaki apartman. Varijabla "sadrzaj" koristi se za spremanje teksta kojeg je korisnik napisao kao recenziju, a varijabla je tipa "TextField". Posljednja varijabla je varijabla "ocjena" koja je tipa "IntegerField" koja služi za spremanje ocjene koju bi korisnik dao tom određenom apartmanu.

```python
class Recenzija(models.Model):
    autor = models.CharField(max_length=100)
    apartman = models.CharField(max_length=10)
    sadrzaj = models.TextField()
    ocjena = models.IntegerField()
```

Slika 11 Model za recenziju

2.3 Admin.py

Unutar ove datoteke potrebno je definirati modele koji će se prikazivati na stranici koja se odnosi na administratora. No, ovaj dio nije toliko bitan ako se radi samo preko grafičkog sučelja kojeg se ručno napravi unutar projekta. Ovo je bitno samo kad se koristi Djangov sustav za administratora.

U našem slučaju, u ovoj datoteci dovoljno je u varijablu "model_list" navesti sve modele, a to su "Apartman" i "Recenzija". Još je potrebno podesiti da se navedeni modeli prikazuju kad

je korisnik prijavljen kao administrator, a to je moguće pomoću komande "admin.site.register(model_list)" (slika 12).

```
1  from django.contrib import admin
2  from .models import *
3
4  # Register your models here.
5
6  model_list = [Apartman, Recenzija]
7  admin.site.register(model_list)
```

Slika 12 Sadržaj admin.py datoteke

2.4 Forms.py

Kako bi forme funkcionirale, potrebno ih je definirati u "forms.py". Ova datoteka ne postoji tako da ju treba dodati u mapi "main". U njoj za svaki model za koji je potrebna forma definiramo formu za njega. U ovom projektu postoje dvije forme, a to su "ApartmanForm" i "RecenzijaForm".

Npr. za model vezan za apartmane kreiramo klasu "ApartmanForm" te kao parametar stavimo "ModelForm". Unutar te klase dodamo klasu "Meta" te u njoj varijable "model" koja označava za koji model je vezana ova forma te varijablu "fields" koja označava koja sve polja će imati ova forma, odnosno koji podaci modela će biti u toj formi. Za formu vezanu za apartmane model postavimo "Apartman", a varijablu "fields" postavimo na "__all__" kako bi svi podaci modela bili u formi. Ako nekim slučajem za koji model nisu potrebni svi podaci, dovoljno je samo navesti koje podatke treba sadržavati forma. No, u ovom projektu su svi podaci u obje forme. Sadržaj datoteke "forms.py" prikazan je na slici 13.

```python
from attr import fields
from django import forms
from django.forms import ModelForm
from .models import *


class ApartmanForm(ModelForm):
    class Meta:
        model = Apartman
        fields = '__all__'

class RecenzijaForm(ModelForm):
    class Meta:
        model = Recenzija
        fields = '__all__'
```

Slika 13 Sadržaj forms.py datoteke

2.5 Views.py

"Views.py" koristi se za definiranje načina na koji će se određena stranica učitavati. Točnije u "views.py" definira se ono što će se učitati na stranici te se u "urls.py" odredi za koji *link* će se učitati ta funkcija iz "views.py". Ako se radi o statičnoj stranici, odnosno onoj koja je uvijek ista, potrebno je samo uz "render" funkciju dati parametar "request" te naziv određene html datoteke. Ako se radi o dinamičkoj stranici, odnosno onoj koja je uvijek drugačija, potrebno je dodati i parametar "context" u koju se postavljaju podaci koji će se učitati prilikom učitavanja stranice. Varijablu "context" možemo definirati tako da je jednaka nekoj našoj varijabli u kojoj su definirani podaci ili možemo samo postaviti tu varijablu kao parametar. Ovaj projekt ima nekoliko različitih načina definiranja, no neki su jako slični odnosno samo html datoteka koju učitavaju nije ista.

Stranica za administratora i stranica "O nama" su statične te je dovoljno samo u "render" funkciju dati parametre "request" te ime html datoteke kao na slici 14.

```python
def admin_panel(request):
    return render(request, 'admin/admin_panel.html')

def o_nama(request):
    return render(request, 'o_nama.html')
```

Slika 14 Definiranje pogleda za statične stranice

Prilikom učitavanja naslovne stranice učitaju se svi apartmani te se nasumično odaberu do četiri apartmana koja će se prikazati. Pritom se provjerava postoje li četiri apartmana ili više te se tada odaberu samo četiri apartmana, a u slučaju da je manje od 4 apartmana, onda se prikaže onoliko koliko ih ima. Oni se spremaju u varijablu "random_apartmani" pomoću koje se u "homepage.html" ispisuju. Ovo se definira u varijabli "podaci" koja se prosljeđuje kao parametar za "context". Na tom principu samo bez opcije za nasumično odobravanje su definirane i funkcije "odobravanje_apartmana" koja se koristi kako bi se administratoru prikazali samo oni apartmani koji nisu odobreni te "moji_apartmani" koja se koristi za prikaz svih apartmana koje posjeduje prijavljeni korisnik. Navedeno je prikazano na slici 15.

```
def homepage(request):
    apartmani = list(Apartman.objects.all())
    if len(apartmani) >= 4:
        apartmani = random.sample(apartmani, 4)
    else:
        apartmani = random.sample(apartmani, len(apartmani))
    podaci = {
        'random_apartmani': apartmani
    }
    return render(request, 'homepage.html', podaci)

def odobravanje_apartmana(request):
    svi_apartmani = Apartman.objects.all()
    podaci = {
        'svi_apartmani': svi_apartmani
    }
    return render(request, 'admin/apartman_list.html', podaci)

def moji_apartmani(request):
    svi_apartmani = Apartman.objects.all()
    podaci = {
        'moji_apartmani': svi_apartmani
    }
    return render(request, 'main/moji_apartmani.html', podaci)
```

Slika 15 Definiranje pogleda korištenjem modela

Kako bi se prikazala lista svih apartmana koji postoje koristi se klasa "ApartmanList" koja je tipa "ListView" te se u njoj definira koji model se koristi za taj pogled (slika 16). Za ovaj način nije potrebno definirati povratnu funkciju u kojoj će se odrediti koja html datoteka se koristi, već je potrebno u mapi "templates" dodati "apartman_list.html" te ga urediti po potrebi. Važno je sve imenovati na taj način iz razloga što se tako preskače povratna funkcija jer je u "ListView" definirano da se učitava točno tu html datoteku koja je naziva kao model te se doda "_list.html".

Na isti način je definirana i klasa “ApartmanDetailView” koja je tipa “DetailView” pomoću koje se Django sam brine o tome da se prikažu podaci apartmana kojeg je korisnik odabrao. Točnije pri izradi stranice nije se potrebno brinuti o identifikatoru apartmana te definirati da se od svih apartmana odabere samo taj odabrani apartman. Za to je potrebno kreirati html datoteku “apartman_detail.html” te ju urediti po potrebi. No, ovaj način se u ovom projektu ne koristi jer se kod svakog pojedinog apartmana prikazuju i recenzije te je zbog toga definirana funkcija “pojedini_apartman”..

```
from django.views.generic import ListView, DetailView

class ApartmanList(ListView):
    model = Apartman

class ApartmanLDetailView(DetailView):
    model = Apartman
```

Slika 16 Definiranje pogleda korištenjem “ListView” i “DetailView”

Funkcija “pojedini_apartman” kao parametar prima i parametar “pk” koji se učitava iz “urls.py” datoteke te taj parametar predstavlja identifikator odabranog apartmana. Taj se parametar koristi kako bi se dobio točno odabrani apartman jer se sad ne učitavaju svi već se pronalazi točno određeni apartman pomoću funkcije “get” u koju se postavlja da je identifikator “id” jednak parametru “pk”. Isto tako, ovaj parametar koristi se i za filtriranje recenzija kako bi se odabrale samo one recenzije koje se odnose na taj određeni apartman, a za ovo se koristi funkcija “filter”. Nakon tog se u varijablu koja će se proslijediti na “context” spremaju svi ti podaci te se odredi povratna funkcija “render” koja ima parametre “request”, naziv html datoteke te “context”. Na sličan način je definirano i “brisanje_apartmana” samo što se ovdje još gleda je li “request” definiran kao “POST” metoda, te ako je, onda se apartman briše i preusmjerava na drugi link. Navedeno je prikazano na slici 17.

```python
def pojedini_apartman(request, pk):
    odredjeni_apartman= Apartman.objects.get(id=pk)
    sve_recenzije = Recenzija.objects.filter(apartman=pk)
    podaci = {
        'apartman': odredjeni_apartman,
        'recenzije': sve_recenzije,
        'pk': pk
    }
    return render(request, 'main/apartman_odredjeni.html', context=podaci)

def brisanje_apartmana(request, pk):
    odabrani_apartman = Apartman.objects.get(id=pk)
    if request.method == "POST":
        odabrani_apartman.delete()
        return redirect("/apartmani")

    context = {"apartman": odabrani_apartman}
    return render(request, "main/apartman_delete.html", context)
```

Slika 17 Definiranje pogleda za točno odabrani apartman

Kako bi se mogli dodavati apartmani, potrebno je kreirati formu, a za njih ćemo koristiti Crispy Forms. Za ovo se koristi funkcija “dodavanje_apartmana”. Na samom početku se učita forma “ApartmanForm” koja je definirana u “forms.py” kako bi se prikazala sva polja. U slučaju da korisnik popuni formu te stisne gumb da spremi formu, “request” metoda postane “POST” te se u ovoj definiciji ulazi u if petlju. Ponovno se definira forma, no ovaj put bude popunjena s podacima kojima je korisnik unio te se provjeri je li sve kako treba. Ako je sve u redu, forma se sprema te se korisnika preusmjerava na stranicu gdje su svi apartmani. Unutar varijable “context” sprema se forma zato što je ona dinamička obzirom da je popunjava korisnik. Na isti način je definirana i funkcija “pisanje_recenzije” samo što se tamo koristi forma “RecenzijaForm” te se korisni i parametar “pk” kako bi se automatski odredilo za koji apartman se piše recenzija. Također funkcija “pisanje_recenzije” je na isti način definirana kao i funkcija “dodavanje_apartmana” samo što je forma druga. Na sličan način su definirane funkcija “uredivanje_apartmana” te funkcija “odobravanje_odredjenog_apartmana” samo što se kod njih koristi parametar “pk” kako bi se odredilo o kojem se točno apartmanu radi. Prilikom učitavanja forme, kao instanca se postavi taj apartman koji je pronađen pomoću parametra “pk” kako bi se forma popunila s njegovim podacima. Navedeno je prikazano na slici 18.

```python
def dodavanje_apartmana(request):
    form = ApartmanForm()
    if request.method == "POST":
        form = ApartmanForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect("/apartmani")

    context = {"form": form}
    return render(request, 'main/apartman_create.html', context=context)

def uredivanje_apartmana(request, pk):
    odabrani_apartman = Apartman.objects.get(id=pk)
    form = ApartmanForm(instance=odabrani_apartman)
    if request.method == "POST":
        form = ApartmanForm(request.POST, instance=odabrani_apartman)
        if form.is_valid():
            form.save()
            return redirect("/apartmani/"+str(pk))

    context = {"form": form}
    return render(request, 'main/apartman_create.html', context=context)

def odobravanje_odredjenog_apartmana(request, pk):
    odredjeni_apartman = Apartman.objects.get(id=pk)
    form = ApartmanForm(instance=odredjeni_apartman)
    if request.method == "POST":
        form = ApartmanForm(request.POST, instance=odredjeni_apartman)
        if form.is_valid():
            form.save()
            return redirect("/adminpanel/OdobravanjeApartmana")

    context = {"form": form, "pk":pk}
    return render(request, 'admin/apartman_odobravanje.html', context=context)

def pisanje_recenzije(request, pk):
    form = RecenzijaForm()
    if request.method == "POST":
        form = RecenzijaForm(request.POST)
        if form.is_valid():
            form.save()
            return redirect("/apartmani/"+str(pk))

    context = {"form": form, "pk":pk}
    return render(request, 'main/pisanje_recenzije.html', context=context)
```

Slika 18 Definiranje pogleda korištenjem formi i identifikatora

2.6 Main/urls.py

Ova datoteka se koristi kako bi se definiralo za koju putanju, odnosno URL će se učitati koji pogled iz datoteke “views.py”. Ovaj projekt ima par različitih načina definiranja putanja. Definiranje putanja izvodi se u varijabli “urlpatterns” unutar uglatih zagrada.

Prvi način je taj da u funkciji “path” definiramo parametar za putanju koju želimo te parametar za *view* koji se odnosi na tu putanju. Primjerice za naslovnu stranicu za putanju su navedeni prazni navodnici što označava da će putanja biti samo ime hosta. Nadalje, definiran je parametar za *view*. Potrebno je definirati koju funkciju će se uzeti iz datoteke “views.py” kao na primjer “views.homepage” koja uzima funkciju “homepage”. Također, definiran je i parametar “name” u kojeg možemo postaviti naziv putanje, no to nije obavezno.

Drugi način za definiranje putanja je kad imamo definiranu klasu za listu nekog modela kao što u ovom projektu postoji “ApartmanList” koja ispisuje sve apartmane koji su odobreni.

Ovaj način se radi kao u ranije opisanom primjeru samo što za parametar vezan za *view* nije postavljen parametar "views.ime_definicije" već "Ime_modelaList.as_view()".

Treći način je također sličan prvom samo što su u putanjama definirane i varijable. U ovom projektu koristi se na način da se u putanji definira identifikator apartmana pomoću kojeg se u "views.py" datoteci odabere samo taj apartman s navedenim identifikatorom. Definiranje varijable radi se na način primjerice "<int:pk>" što znači da je definirana varijabla "pk" koja je tipa *integer*. Ovakva definicija varijable se koristi u projektu u više putanja. Primjerice u putanji "apartmani/<int:pk>" pomoću koje se u "views.py" pronađe određeni apartman kako bi se prikazali detalji tog apartmana.

Na kraju, definirane su i putanje za slike koje korisnik objavi kad dodaje novi apartman. Za ovo je dovoljno nakon zatvorenih uglatih zagrada varijable "urlpatterns" dodati "+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)". Klikom na sliku ona će se pokazat u originalnoj veličini.

```python
from django.urls import path
from . import views
from main.views import *
from django.conf import settings
from django.conf.urls.static import static

app_name = 'main'  # here for namespacing of urls.

urlpatterns = [
    path('', views.homepage, name='homepage'),
    path('onama', views.o_nama,),

    path('adminpanel/', views.admin_panel),
    path('adminpanel/OdobravanjeApartmana', views.odobravanje_apartmana,),

    path('apartmani/', ApartmanList.as_view()),
    path('apartmani/MojiApartmani', views.moji_apartmani),
    path('apartmani/DodavanjeApartmana', views.dodavanje_apartmana),
    path('apartmani/<int:pk>', views.pojedini_apartman, name='id'),
    path('apartmani/<int:pk>/Uredi', views.uredivanje_apartmana),
    path('apartmani/<int:pk>/Obrisi', views.brisanje_apartmana),
    path('apartmani/<int:pk>/NapisiRecenziju', views.pisanje_recenzije),
    path('apartmani/<int:pk>/OdobravanjeOdredjenogApartmana', views.odobravanje_odredjenog_apartmana),
]+ static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

Slika 19 Sadržaj Main/urls.py datoteke

2.7 AptBooker/urls.py

Unutar ove datoteke potrebno je definirati poveznice na sve aplikacije unutar glavne aplikacije. Pa je tako potrebno dodati poveznicu na "main" aplikaciju, na sustav autentifikacije te dodati poveznicu pomoću koje se automatski web aplikacija osvježi svaki put kad se promjene spreme. Sadržaj ove datoteke prikazan je na slici 20.

```python
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', include('main.urls')),
    path('accounts/', include('allauth.urls')),
    path("__reload__/", include("django_browser_reload.urls")),
]
```

Slika 20 Sadržaj AptBooker/urls.py datoteke

2.8 HTML datoteke

Sve HTML datoteke nalaze se unutar direktorija "template" u kojem su organizirane. Dijele se na četiri vrste: općenite HTML datoteke, HTML datoteke vezane za apartmane i recenzije, HTML datoteke vezane za administratora te one HTML datoteke vezane za autentifikaciju.

Općenite HTML datoteke su one koje se ne odnose na unutarnje sustave aplikacije. Ovdje se nalaze HTML datoteke za naslovnu stranicu, stranicu o nama te datoteka "base_generic.html". Unutar zaglavlja nalazi se logotip, navigacija te sustav za prijavu korisnika. Ovisno o tome je li korisnik neprijavljen ili prijavljen, odnosno je li korisnik administrator, navigacija se proširuje. Točnije, ovisno o vrsti korisnika koji pregledava aplikaciju, navigacija prikazuje dodatne poveznice. Kako se na svakoj stranici ne bi trebalo posebno definirati zaglavlje onda se koristi ovaj način proširenja stranice. Opisano se postiže dodavanjem sadržaja prikazanog na slici 21.

```
<body>
    <div class="content m-0">
        {% block content %}
        {% endblock %}
    </div>
</body>
```

Slika 21 Prikaz načina na koji se proširenje stranicu

Nakon toga je unutar svake HTML datoteke potrebno definirati da se ta HTML datoteka povezuje na datoteku "base_generic.html" te je unutar bloka potrebno dodati sadržaj za tu stranicu kao što je prikazano na slici 22.

```
{% extends "base_generic.html" %}
{% block content %}
    ...
    Sadržaj produžene stranice
    ...
{% endblock %}
```

Slika 22 Prikaz načina kako povezati HTML datoteku na drugu

Unutar direktorija “template/main” nalaze se sve HTML datoteke vezane za apartmane i recenzije kao što su dodavanje, brisanje, uređivanje apartmana i slično. Tako se ovdje također nalazi HTML datoteka pomoću koje se korisniku prikazuju svi odobreni apartmani. Na samom početku prikaže se *cover* slika te nakon nje slijede apartmani. Prikaz apartmana radi na način da se vrti petlja sve dok se ne prođe kroz sve apartmane. Za svaki apartman, koji je odobren, prikaže se slika tog apartmana te naziv, cijena i lokacija apartmana. Sadržaj ove HTML datoteke nalazi se na slici 23.

```
{% extends "base_generic.html" %}
{% block content %}
<div class="relative bg-gray-900" style="margin-bottom: 3rem;">
    <img class="w-full opacity-75 object-cover" src="/static/images/dubrovnik.jpg" style="height: 35rem;">
    <h1 class="absolute p-6 text-white text-6xl top-1/2 left-1/2 -translate-x-1/2 -translate-y-1/2"> Apartmani </h1>
</div>
{% for apartmani in object_list %}
    {% if apartmani.apartman_odobren == True %}
    <a href="/apartmani/{{ apartmani.id }}">
        <div class="border border-black rounded-2xl flex items-center" style="margin-left: 7rem; margin-right: 7rem; margin-bottom: 3rem;">
            <div>
                <img style="width: 700px; height: 400px; border-top-left-radius: 1rem; border-bottom-left-radius: 1rem;" src="/media/{{apartmani.apartman_glavna_slika}}">
            </div>
            <div style="margin-left: 4rem; border-left: 1px solid black; height: 300px;" class="flex items-center">
                <div style="margin-left: 4rem;">
                    <h4 style="padding-bottom: 4rem;" class="text-4xl text-blue-900 font-bold">{{ apartmani.apartman_naziv }}</h4>
                    <h5 style="padding-bottom: 1rem;" class="text-xl">Cijena: {{ apartmani.apartman_cijena }} kn / noć</h5>
                    <p class="text-xl">Lokacija: {{ apartmani.apartman_lokacija }}</p>
                </div>
            </div>
        </div>
    </a>
    {% endif %}
{% endfor %}
{% endblock %}
```

Slika 23 Sadržaj HTML datoteke za prikaz svih apartmana koji su odobreni

Direktorij “template/admin” sadrži HTML datoteke koje su vezane za administratora kao što su panel za administratora, HTML za prikaz samo onih apartmana koji nisu odobreni te HTML za odobravanje pojedinog apartmana. Kako bi se neki apartman odobrio, potrebno je promijeniti formu tog apartmana. Zbog tog je u HTML-u za odobravanje pojedinog apartmana potrebno koristiti formu tog apartmana koju je prvo potrebno učitati. Nakon naslova slijedi forma koja sadrži sve varijable tog modela. Ispod se nalaze dva gumba od kojih je jedan za potvrdu, a drugi za brisanje apartmana u slučaju da administrator ne želi odobriti ombjavljivanje tog apartman. Sadržaj ove HTML datoteke je prikazan na slici 24.

```
{% extends "base_generic.html" %}
{% block content %}
{% load crispy_forms_tags %}
<div class="bg-blue-900">
    <h1 class="text-5xl text-white p-6 text-center"> Odobravanje apartmana </h1>
</div>
<form action="" method="POST" class="p-6">
    {% csrf_token %}
    <div style="display: none;">
        {{ form.apartman_naziv|as_crispy_field }}
    </div>
    <div style="display: none;">
        {{ form.apartman_cijena|as_crispy_field }}
    </div>
    <div style="display: none;">
        {{ form.apartman_vlasnik|as_crispy_field }}
    </div>
    <div style="display: none;">
        {{ form.apartman_smjestaji|as_crispy_field }}
    </div>
    <div style="display: none;">
        {{ form.apartman_lokacija|as_crispy_field }}
    </div>
    <div style="display: none;">
        {{ form.apartman_sadrzaj|as_crispy_field }}
    </div>
    <div style="display: none;">
        {{ form.apartman_glavna_slika|as_crispy_field }}
    </div>
    <div style="margin-top: 1rem;">
        {{ form.apartman_odobren|as_crispy_field }}
    </div>
    <div class="flex items-center" style="margin-top: 2rem;">
        <p><input type="submit" name="Potvrdi" value="Potvrdi" class="text-white bg-blue-500 border border-blue-900 py-2 px-4"></p>
        <button class="border border-blue-900 py-2 px-4 ml-6"><a href="/apartmani/{{ pk }}/Obrisi">Obriši</a></button>
    </div>
</form>
{% endblock %}
```

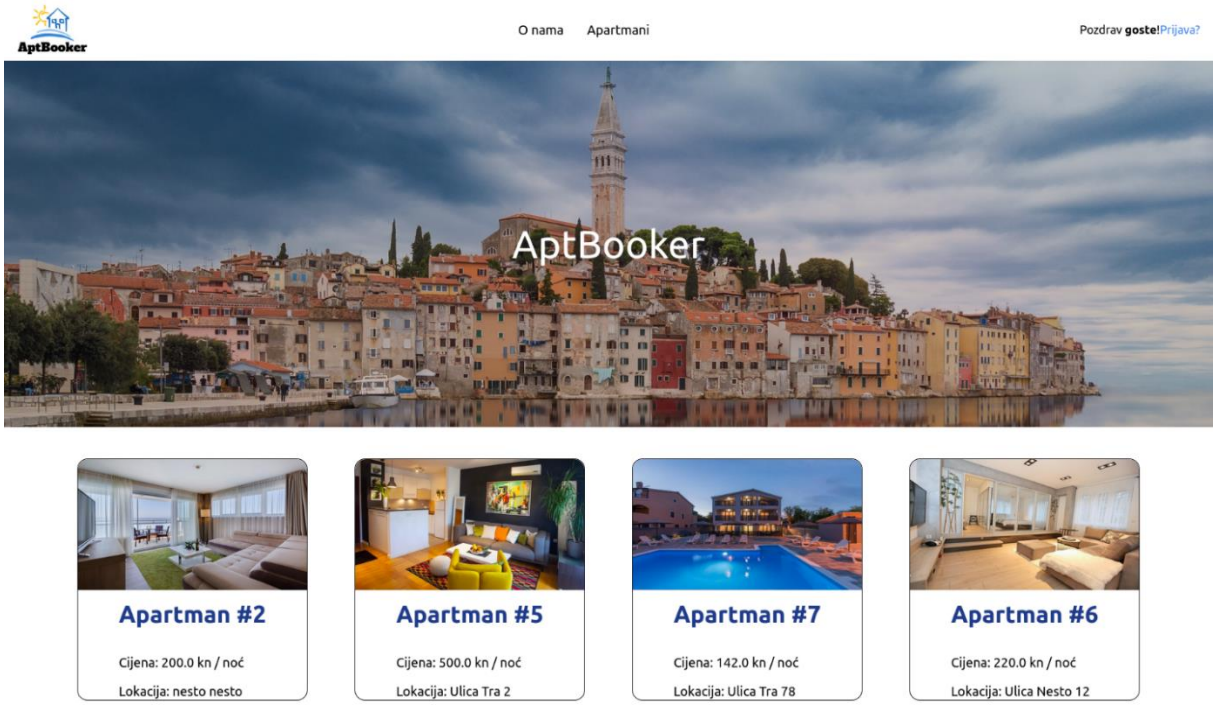
Slika 24 Sadržaj HTML datoteke za odobravanje apartmana

Unutar direktorija “template/account” nalaze se HTML datoteke koje su vezane za autentifikaciju. Vrlo je važno tako nazvati mapu zato što se na taj način nadjačaju postojeće HTML datoteke koje su vezane za taj dio. Što znači da će se na primjer za prijavu korisnika učitati HTML datoteka vezana za taj dio, a ne ona koja je napravljena od strane Allauth-a. Originalne HTML datoteke pronađene su na GitHub-u na poveznici https://github.com/pennersr/django-allauth/tree/master/allauth/templates/account.

3. Prikaz kreirane aplikacije

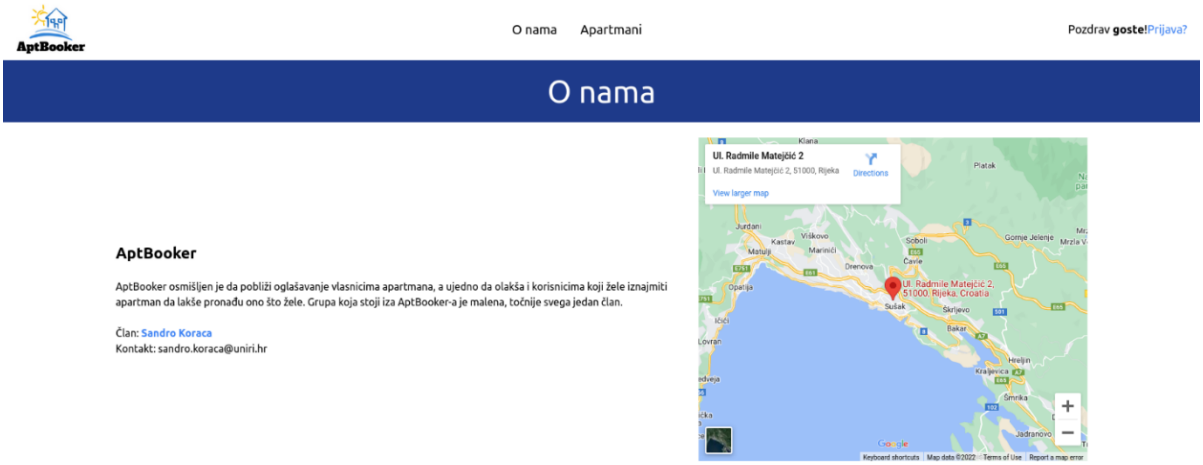
3.1 Neprijavljeni korisnik

Na slici 25 prikazana je naslovna stranica na kojoj se nalazi *cover* slika te 4 nasumično odabrana apartmana.



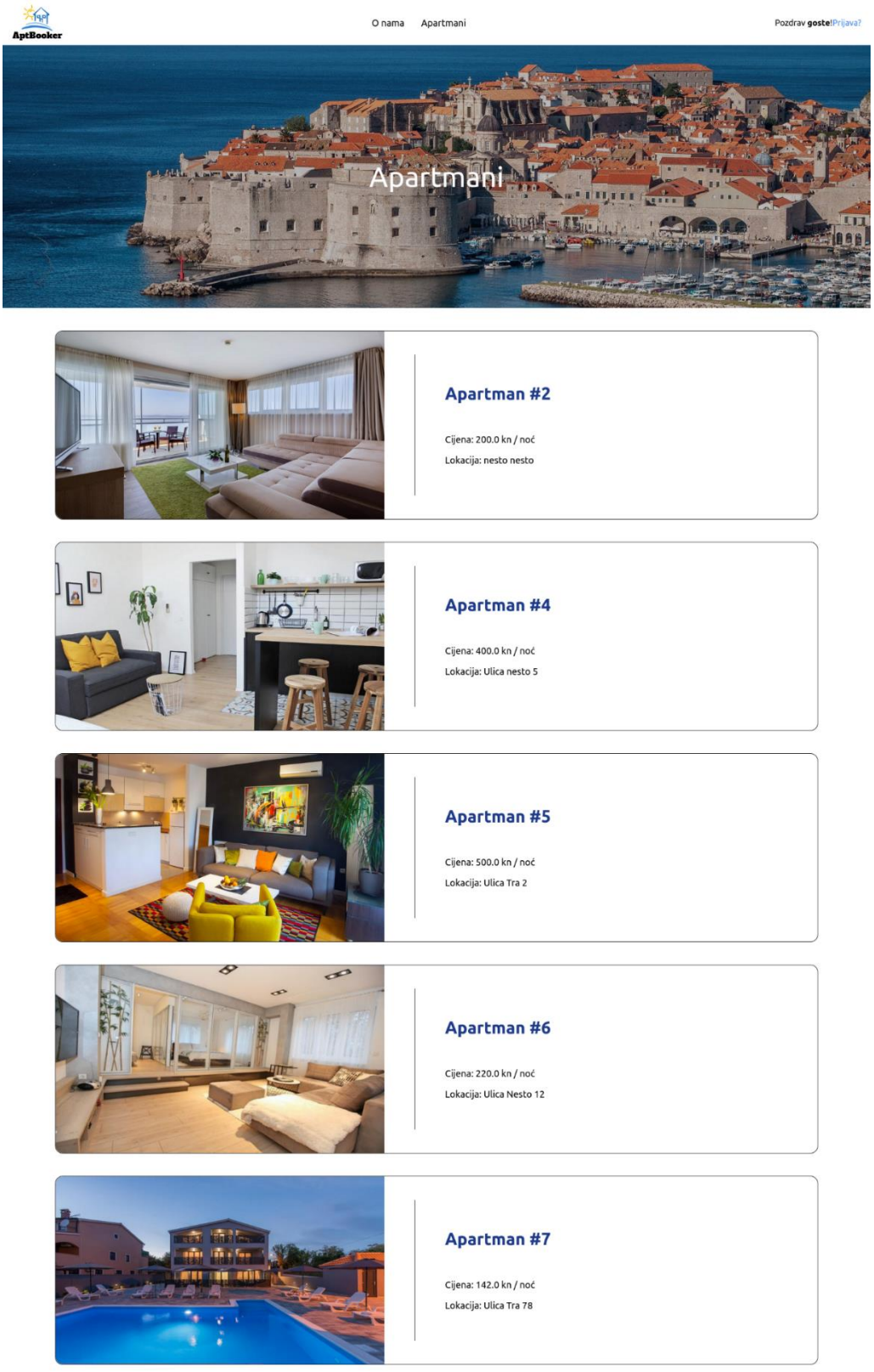
Slika 25 Prikaz naslovne stranice

Slika 26 prikazuje stranicu “O nama” na kojoj je maleni tekst o aplikaciji „AptBooker“ te lokacija sjedišta aplikacije.



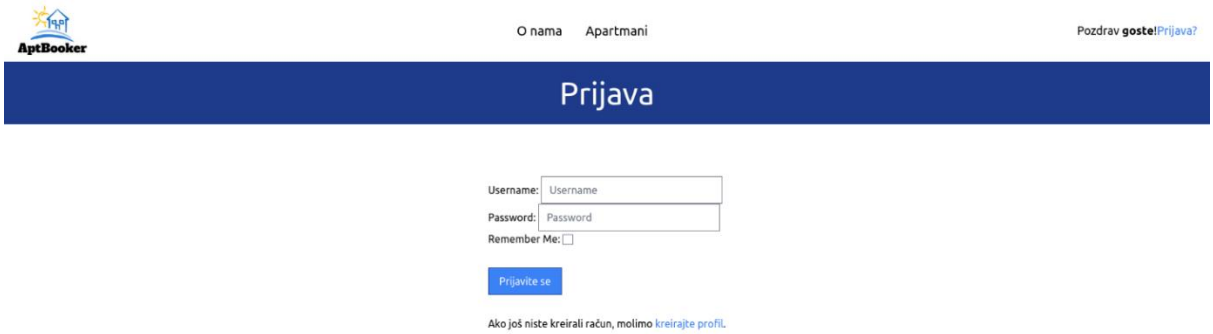
Slika 26 Prikaz stranice "O nama"

Na stranici “Apartmani” prikazani su svi apartmani koje je administrator odobrio. Za svaki apartman prikaže se slika, naziv, cijena za jednu noć te lokacija apartmana. Ova stranica prikazana je na slici 27.


AptBooker
O nama
Apartmani
Pozdrav goste!Prijava?
Apartmani
Apartman #2
Cijena: 200.0 kn / noć
Lokacija: nesto nesto
Apartman #4
Cijena: 400.0 kn / noć
Lokacija: Ulica nesto 5
Apartman #5
Cijena: 500.0 kn / noć
Lokacija: Ulica Tra 2
Apartman #6
Cijena: 220.0 kn / noć
Lokacija: Ulica Nesto 12
Apartman #7
Cijena: 142.0 kn / noć
Lokacija: Ulica Tra 78

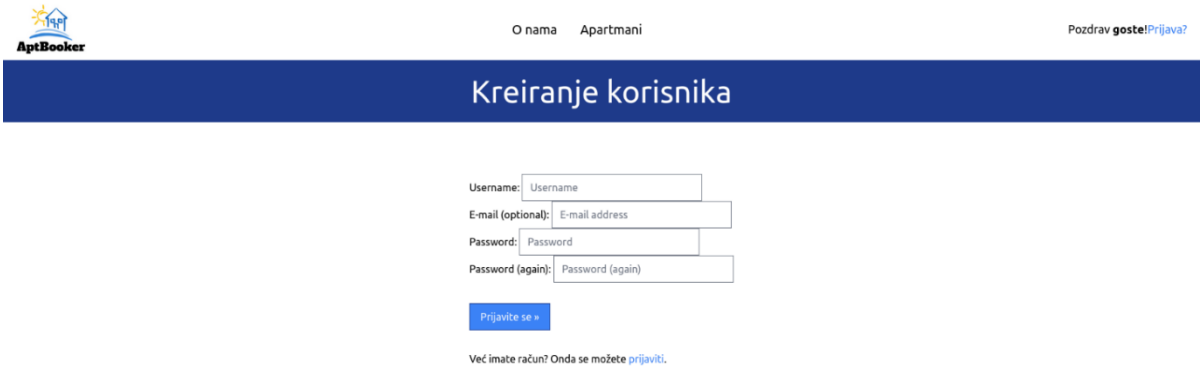
Slika 27 Prikaz stranice za apartmane koji su odobreni te ih korisnik može vidjeti

Na slici 28 može se vidjeti kako izgleda stranica za prijavu korisnika.



Slika 28 Prikaz stranice za prijavu korisnika

Ako korisnik nema profil može ga kreirati, a izgled stranice za kreiranje profila može se vidjeti na slici 29.



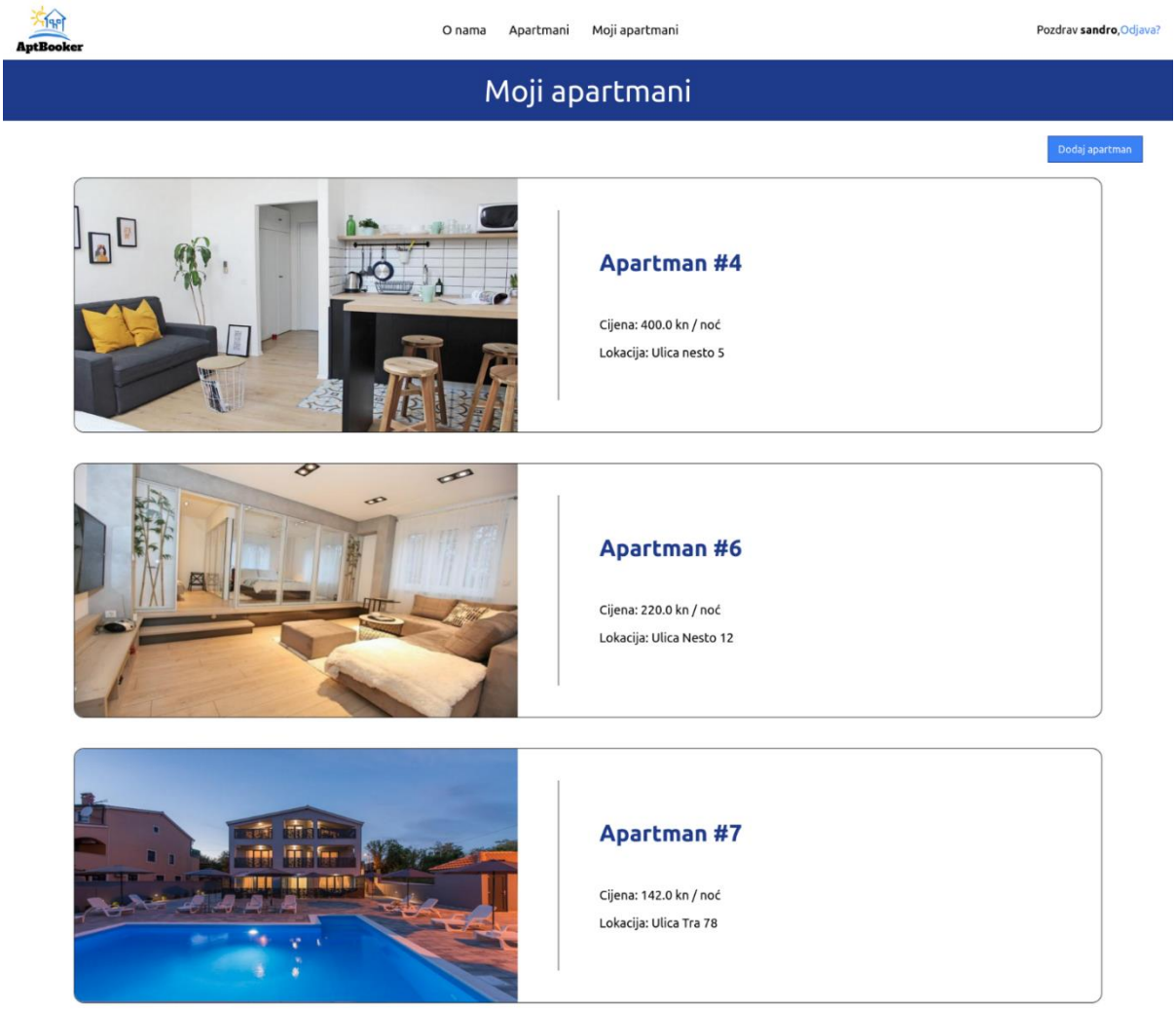
Slika 29 Prikaz stranice za kreiranje korisnika

Slika 30 prikazuje izgled stranice za odjavu korisnika.

Slika 30 Prikaz stranice za odjavu korisnika

3.2 Prijavljeni korisnik

Kad je korisnik prijavljen pojavljuje se drugačija navigacija te korisnik može pregledavati apartmane koje je dodao. Slika 31 prikazuje stranicu za pregled vlastitih apartmana od strane prijavljenog korisnika.

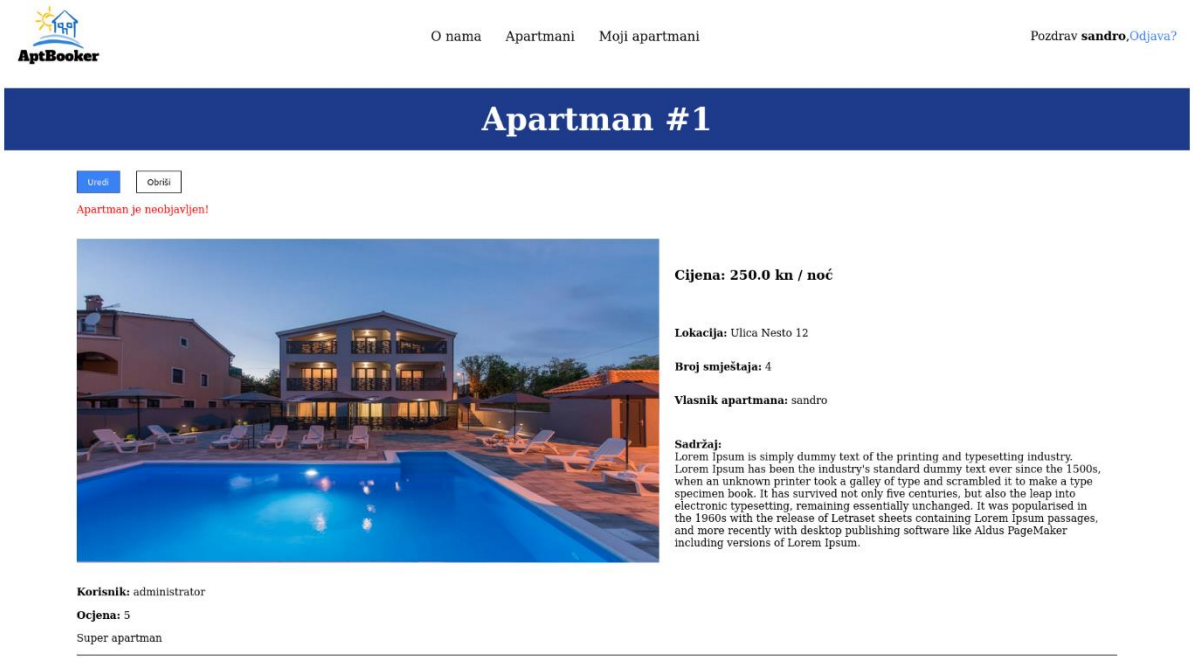


Slika 31 Prikaz stranice s apartmanima koje korisnik posjeduje

Također, prijavljeni korisnik može dodavati apartmane, a kako bi to učinio, popunjava formu i potvrđuje unos. Slika 32 prikazuje kako izgleda stranica za popunjavanje forme za dodavanje apartmana.

Slika 32 Prikaz stranice za dodavanje apartmana

Kada korisnik doda apartman, on je neobjavljen te je za objavljivanje potrebno odobrenje administratora. No, bez obzira je li apartman objavljen ili ne, korisnik ga može uređivati ili obrisati. Također, ako je apartman objavljen te korisnik odluči urediti apartman, on opet postaje neobjavljen te opet mora biti odobren od strane administratora. Kad korisnik uređuje podatke o apartmanu, onda stranica izgleda kao što je prikazano na slici 32. Na slici 33 prikazano je kako izgleda stranica poslije dodavanja ili uređivanja apartmana.

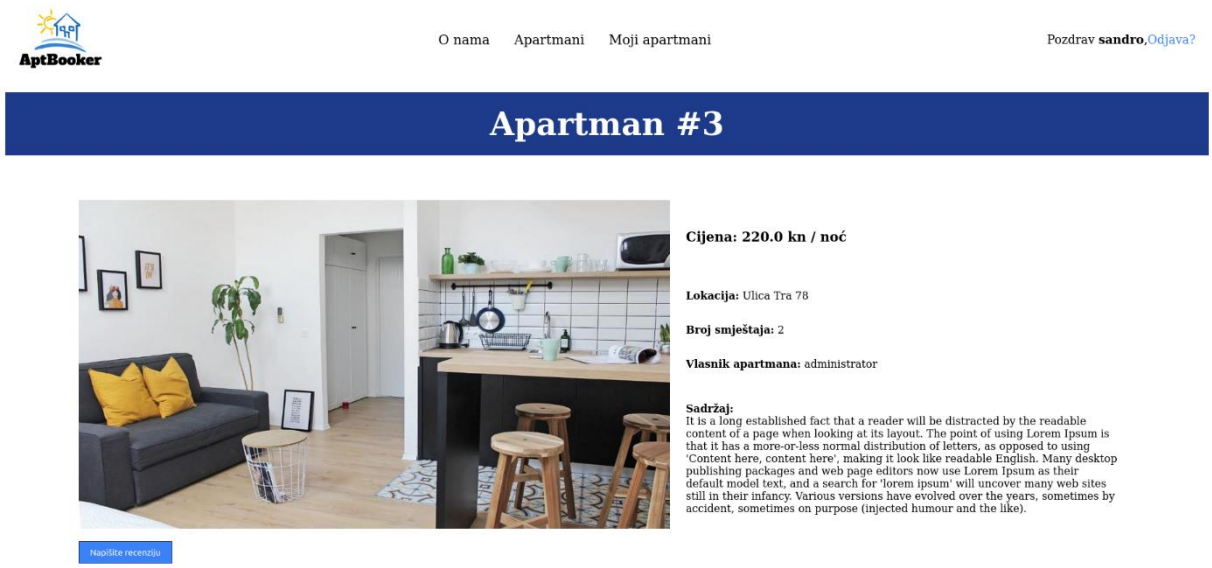


Slika 33 Prikaz stranice za apartman koji je u vlasništvu prijavljenog korisnika te apartman nije objavljen

Ako korisnik odluči obrisati apartman, klikom na gumb “Obriši” prikazuje se stranica kao na slici 34. Korisnik može odabrati potvrđivanje brisanja ili može odustati od brisanja.

Slika 34 Prikaz stranice za brisanje apartmana

Također, ako prijavljeni korisnik pregledava apartmane koje su objavili drugi korisnici, on ih može recenzirati. Slika 35 prikazuje kako izgleda stranica kad se pregledava apartman od drugog korisnika.



Slika 35 Prikaz stranice apartmana koji nije u vlasništvu prijavljenog korisnika

Ako korisnik odluči napisati recenziju za apartman klikom na gumb “Napiši recenziju” prikazuje se stranica (slika 36) za pisanje recenzije.

Slika 36 Prikaz stranice za pisanje recenzija

3.3 Korisnik prijavljen kao administrator

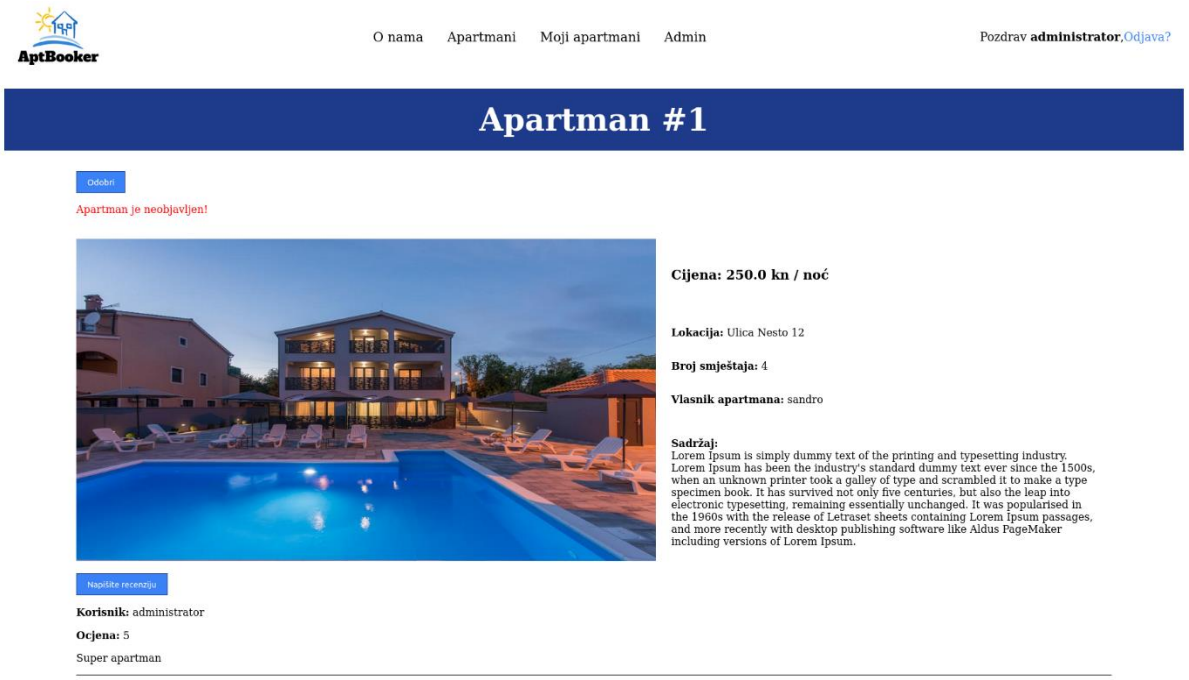
Ako je korisnik prijavljen kao administrator, on može sve što i prijavljen korisnik samo što ima još dodatnu mogućnost odobravanja apartmana. Unutar navigacije prikazuje mu se poveznica za administratora. Slika 37 prikazuje kontrolni centar za administratora koji trenutno ima samo opciju za odobravanje apartmana. No, unapređivanjem aplikacije može se proširiti s više funkcija.

Slika 37 Prikaz stranice kontrolnog centra za administratora

Klikom na gumb "Odobravanje apartmana" prikazuje se stranica kao na slici 38. Na toj stranici prikazuju se svi apartmani koji nisu odobreni.

Slika 38 Prikaz stranice za apartmane koji nisu objavljeni

Nadalje, kad administrator odabere koji apartman želi pregledati, prikazuje se stranica kao na slici 39.



Slika 39 Prikaz stranice za pojedini apartman kojeg administrator mora odobriti

Kad administrator pregleda apartman mora kliknuti na gumb “Odobri” unutar kojeg može odobriti ili obrisati apartman te se prikazuje stranica kao na slici 40. Ako želi odobriti apartman, mora označiti da želi odobriti te potvrditi odabir, a ako ne želi odobriti onda ga može obrisati te se prikazuje stranica na kojem mora potvrditi brisanje.

Slika 40 Prikaz stranice za potvrdu odobravanja ili brisanje apartmana

4. Zaključak

Rezultat ovog završnog rada je aplikacija za prikaz apartmana za iznajmljivanje. Kroz rad je detaljno opisan postupak izrade aplikacije, dodavanje autentifikacije i različitih razina dozvola za različite skupine korisnika. Također, kroz prikaz izgleda stranica koje aplikacija sadrži, objašnjene su sve njezine funkcionalnosti i mogućnosti koje nudi.

Što se tiče alata koji su se koristili za izradu aplikacije, Django modul koji se koristio kao osnovni alat jednostavan je za korištenje, a pozitivna je i činjenica da ima svoju bazu podataka te je besplatan za korištenje. Dapače, Django se još razvija te je vjerojatno da će s vremenom postati još bolji. Iz tih je razloga jedan od atraktivnijih alata za izradu web aplikacija. Dovoljno je reći to da Instagram, Pinterest, National Geographic kao i mnogi drugi koriste upravo Django. Pozitivno je i to što postoji puno dodataka koji se mogu koristiti uz Django pa su tako u ovom radu korišteni Crispy forms, Allauth i Tailwind dodaci kako bi upotpunili i olakšali izradu projekta.

Moje iskustvo u radu s Djangom je također pozitivno jer sam se tijekom studija već upoznao s tim alatom te nije kompliciranо njegovo korištenje. Tijekom ovog projekta prvi put sam se susreo s dodatkom Tailwind te sam jako zadovoljan lakoćom njegova funkcioniranja. Zbog poteškoća oko prebacivanja aplikacije na server donekle sam morao izbjeći korištenje predefiniranih klasa i ručno definirati pojedine klase. No, usprkos problemima s kojim sam se susreo, zadovoljan sam postupkom izrade aplikacije i iskustvom koje sam stekao tijekom izrade „AptBooker“ aplikacije te njezinim sadašnjim izgledom i funkcionalnostima. Naravno, ima još puno mjesta za njezino poboljšanje i dodavanje funkcionalnosti u budućem radu.

5. Popis slika

Slika 1 Početne datoteke 8
Slika 2 Datoteke nakon dodavanja "main" aplikacije 9
Slika 3 Postavljanje direktorija za templates 10
Slika 4 Postavljanje "media" direktorija 10
Slika 5 Postavljanje instaliranih aplikacija 11
Slika 6 Postavljanje poveznica za autentifikaciju 11
Slika 7 Postavljanje postavki za TailWind 12
Slika 8 Postavljanje middleware za automatsko učitavanje stranice 13
Slika 9 Postavljanje postavki za statične datoteke 13
Slika 10 Model za apartman i pomoćna funkcija 16
Slika 11 Model za recenziju 16
Slika 12 Sadržaj admin.py datoteke 17
Slika 13 Sadržaj forms.py datoteke 18
Slika 14 Definiranje pogleda za statične stranice 18
Slika 15 Definiranje pogleda korištenjem modela 19
Slika 16 Definiranje pogleda korištenjem “ListView” i “DetailView” 20
Slika 17 Definiranje pogleda za točno odabrani apartman 21
Slika 18 Definiranje pogleda korištenjem formi i identifikatora 22
Slika 19 Sadržaj Main/urls.py datoteke 23
Slika 20 Sadržaj AptBooker/urls.py datoteke 24
Slika 21 Prikaz načina na koji se proširenje stranicu 24
Slika 22 Prikaz načina kako povezati HTML datoteku na drugu 25
Slika 23 Sadržaj HTML datoteke za prikaz svih apartmana koji su odobreni 25
Slika 24 Sadržaj HTML datoteke za odobravanje apartmana 26
Slika 25 Prikaz naslovne stranice 27
Slika 26 Prikaz stranice "O nama" 28
Slika 27 Prikaz stranice za apartmane koji su odobreni te ih korisnik može vidjeti 30
Slika 28 Prikaz stranice za prijavu korisnika 30
Slika 29 Prikaz stranice za kreiranje korisnika 31
Slika 30 Prikaz stranice za odjavu korisnika 31
Slika 31 Prikaz stranice s apartmanima koje korisnik posjeduje 32
Slika 32 Prikaz stranice za dodavanje apartmana 33
Slika 33 Prikaz stranice za apartman koji je u vlasništvu prijavljenog korisnika te apartman nije objavljen 34
Slika 34 Prikaz stranice za brisanje apartmana 34
Slika 35 Prikaz stranice apartmana koji nije u vlasništvu prijavljenog korisnika 35
Slika 36 Prikaz stranice za pisanje recenzija 36
Slika 37 Prikaz stranice kontrolnog centra za administratora 37
Slika 38 Prikaz stranice za apartmane koji nisu objavljeni 37
Slika 39 Prikaz stranice za pojedini apartman kojeg administrator mora odobriti 38
Slika 40 Prikaz stranice za potvrdu odobravanja apartmana 39

6. Literatura

Django Software Foundation. (2022). Dohvaćeno iz Django project: https://www.djangoproject.com/foundation/

Heroku. (26. kolovoz 2022). Dohvaćeno iz Wikipedia: https://en.wikipedia.org/wiki/Heroku

Petrović, M. (2022). *Python: Django*. Dohvaćeno iz gaseri: https://gaseri.org/hr/nastava/materijali/python-modul-django/

Tailwind CSS. (28. kolovoz 2022). Dohvaćeno iz Wikipedia: https://en.wikipedia.org/wiki/Tailwind_CSS

7. Popis priloga

Web aplikacija: https://aptbooker.herokuapp.com/

Kod web aplikacije: https://github.com/SandroKoraca/AptBooker