Web aplikacija servisa za računalo

Danilović, Kristijan

Undergraduate thesis / Završni rad

2025

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:467264

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-21**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Stručni studij računarstva

WEB APLIKACIJA SERVISA ZA RAČUNALO

Završni rad

Kristijan Danilović

Osijek, 2025.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1S: Obrazac za ocjenu završnog rada na stručnom prijediplomskom studiju

Ocjena završnog rada na stručnom prijediplomskom studiju

Ime i prezime pristupnika:	Kristijan Danilović
Studij, smjer:	Stručni prijediplomski studij Računarstvo
Mat. br. pristupnika, god.	AR4841, 27.07.2021.
JMBAG:	0165090278
Mentor:	Robert Šojo, univ. mag. ing. comp.
Sumentor:	
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	doc. dr. sc. Ivana Hartmann Tolić
Član Povjerenstva 1:	Robert Šojo, univ. mag. ing. comp.
Član Povjerenstva 2:	Marina Peko, dipl. ing. el.
Naslov završnog rada:	Web aplikacija servisa za računalo
Znanstvena grana završnog rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rada:	Opis teme: Web aplikacija na kojoj se mogu prijaviti neispravna računala za servis. Administrator postavlja savjete na stranici kako održavati računalo u ispravnom stanju te uvid u osnovnu analitiku. Gosti stranice mogu samo pregledavati sadržaj web stranice dok registrirani korisnici mogu prijaviti svoje računalo na servis s detaljnim objašnjenjem u čemu je problem. Registrirani korisnici imaju i svoje osobne stranice na kojoj mogu ostavljati informacije o svome osobnom računalu. Student mora napraviti funkcionalnu web stranicu primjenom različitih web tehnologija. Tema

Datum ocjene pismenog dijela završnog rada od strane mentora:	06.02.2025.
Ocjena pismenog dijela završnog rada od strane mentora:	Izvrstan (5)
Datum obrane završnog rada:	21.2.2025.
Ocjena usmenog dijela završnog rada (obrane):	Izvrstan (5)
Ukupna ocjena završnog rada:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio stručni prijediplomski studij:	21.02.2025.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O IZVORNOSTI RADA

Osijek, 21.02.2025.

Ime i prezime Pristupnika:	Kristijan Danilović
Studij:	Stručni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	AR4841, 27.07.2021.
Turnitin podudaranje [%]:	6

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija servisa za računalo**

izrađen pod vodstvom mentora Robert Šojo, univ. mag. ing. comp.

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis pristupnika:

SADRŽAJ

1. UVOD 1

1.1. Zadatak završnog rada 1

2. POSTOJEĆE WEB I ANDROID APLIKACIJE 3

2.1. Centralni servis računala 3

2.2. Computer Repair 4

2.3. Computer Repair Expert 5

2.4. MM-informatika 7

3. KORIŠTENE TEHNOLOGIJE PRI IZRADI WEB APLIKACIJE 9

3.1. HTML 9

3.2. CSS 10

3.3. Bootstrap 13

3.4. Django 15

3.5. JavaScript 17

3.6. Visual Studio Code 18

4. IZRADA I FUNKCIONALNOSTI APLIKACIJE 19

4.1. Struktura aplikacije 19

4.2. Struktura Django projekta 20

4.3. Struktura baze podataka 22

4.4. Neregistrirani i registrirani korisnici 27

4.4.1. Početna stranica aplikacije 27

4.4.2. Osnovna analitika servisa 30

4.4.3. Prikaz članaka 31

4.4.4. Prijava, registracija i odjava korisnika 32

4.5. Registrirani korisnici 35

4.5.1. Konfiguracija računala i postavke profila 35

4.5.2. Prijava kvara 36

4.5.3. Dijagnoza kvara 37
4.5.4. Instalacija operacijskog sustava 38
4.5.5. Čišćenje računala 39
4.5.6. Pregled narudžbi 41
4.6. Administratori 42
4.6.1. Dodavanje i brisanje članaka 43
4.6.2. Prikaz aktivnih soba za razgovor 44
4.6.3. Pregled i ažuriranje svih narudžbi 45
5. ZAKLJUČAK 48
LITERATURA 49
SAŽETAK 50
ABSTRACT 51
PRILOZI 52

1. UVOD

Porastom broja korisnika osobnih računala i laptopa, također je došlo do potrebe za sve većim brojem servisera računala. Računala se danas koriste u svakom segmentu industrije, obrazovanja i sličnim područjima. Primjerice, laptop koji koristi učiteljica razredne nastave potreban je svakodnevno, a u slučaju kvara laptopa, održavanje nastave može biti otežano. Kvar se mora otkloniti što prije, a ova web aplikacija za servis računala pomoći će da se na brz, jednostavan i interaktivan način opiše kvar, zatraži korisnička podrška, naruči na servis i dobije uvid u stanje popravka. Osim što se korisnicima pomaže, serviserima računala također se omogućuje jednostavna komunikacija s mušterijama u realnom vremenu, definiranje cijene usluge, dijeljenje savjeta o održavanju računala, bolja organizacija, lakše vođenje evidencije te omogućavanje lakšeg popravljanja računala i otklanjanja mogućih uzroka kvara.

U drugom poglavlju predstavljena su slična rješenja u obliku web i android aplikacija. Analizom tih aplikacija dobiven je uvid u to kako aplikacija treba izgledati, što izbjegavati i koje tehnologije koristiti. U trećem poglavlju objašnjene su tehnologije koje se koriste za izradu web aplikacije ovoga rada. U četvrtom poglavlju opisani su postupci i način izrade web aplikacije, kako aplikacija funkcionira i kako se koristi.

1.1. Zadatak završnog rada

Zadatak završnog rada je web aplikacija koja omogućava registriranim korisnicima usluge poput naručivanja računala ili laptopa na servis putem prijave kvara, dijagnoze kvara uz pomoć razgovora u realnom vremenu, naručivanja instalacije operacijskog sustava ili čišćenja računala na brz, interaktivan i jednostavan način. U postavkama svog profila registrirani korisnici mogu dodati svoju konfiguraciju računala, što ubrzava vrijeme popravka i dijagnoze kvara serviserima. Registriranim korisnicima omogućen je pristup pregledu svojih narudžbi. Također, svim registriranim korisnicima, kao i gostima, omogućen je pregled osnovne analitike servisa i članaka koji se tiču održavanja računala. Administratorima stranice – serviserima računala omogućeno je dodavanje i brisanje članaka, zatvaranje soba za razgovor u realnom vremenu i sudjelovanje u

razgovoru, pregledavanje, sortiranje, filtriranje i ažuriranje narudžbi te njihovo stanje. Web aplikacija treba biti izrađena uz pomoć tehnologija navedenih u trećem poglavlju.

2. POSTOJEĆE WEB I ANDROID APLIKACIJE

U ovom poglavlju prikazan je pregled sličnih web i android aplikacija za prikaz i prikupljanje podataka, analizu tehnologija i dobre prakse ili općenito uočavanje nedostataka kako bi iste greške bile izbjegnute.

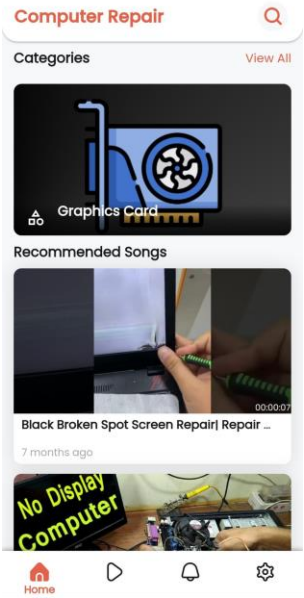
2.1. Centralni servis računala

Centralni servis računala je web stranica za servis računala koja također nudi i usluge servisa laptopa i printera, preventivne dijagnostike, spašavanja podataka, crypto miner servisa, backupa podataka, mrežnih instalacija, daljinske podrške te održavanja servera. Preko web stranice je moguće poslati upit za navedenu uslugu ili pomoć te se zatim dogovoriti o načinu servisa. Servis se može obaviti fizičkim posjetom klijenta u njihovu poslovnicu, mobilnim servisom – servis na Vašoj lokaciji, te dolazak servisera po Vaš uređaj i opremu. Osim dogovaranja usluge, na web stranici moguće je pregledavati i novosti iz svijeta servisa računala i ostalih uređaja. Slika 2.1. prikazuje početnu stranicu centralnog servisa računala [1].

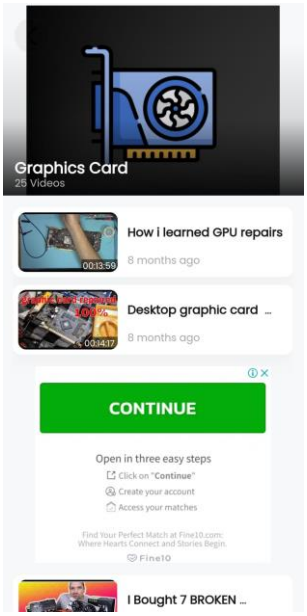
Sl. 2.1. *Servis.com.hr.*

2.2. Computer Repair

Computer Repair je jednostavna aplikacija za android uređaje koja prikazuje videozapise vezane za kvarove kako bi korisnik mogao samostalno popraviti. Aplikacija se može preuzeti s Googleovog servisa Trgovina Play. Na slici 2.2. prikazana je početna stranica aplikacije u kojoj je moguće pretraživati određene kvarove preko tražilice ili odabirom određene kategorije poput grafičke kartice, procesora, zamjene ventilatora i tako dalje. Ako se odabere kategoriju primjerice grafička kartica, otvara se drugi ekran aplikacije koji je prikazan na slici 2.3. te on prikazuje različite videozapise koji su vezani za rješavanje kvara u odabranoj kategoriji [2].



Sl. 2.2. *Početna stranica aplikacije Computer Repair.*



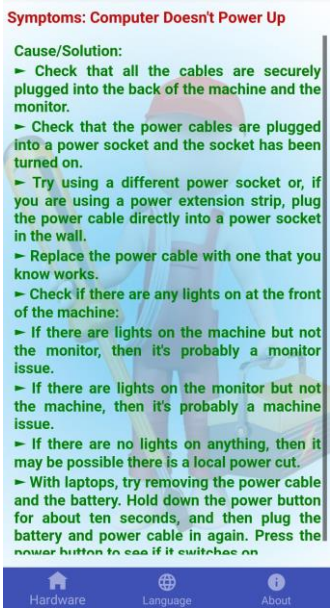
Sl. 2.3. *Prikaz određene kategorije hardvera unutar aplikacije.*

2.3. Computer Repair Expert

Computer Repair Expert je aplikacija za android uređaje u kojoj se na interaktivan i jednostavan način nudi dijagnoza hardverskih kvarova. Nakon što je dio hardvera izabran, postavljaju se pitanja na koja se odgovara s da ili ne. Na temelju odgovora rješenje i/ili uzrok kvara su dati korisniku. Također se dijagnosticiraju poruke o pogrešci kao i zvučni signali. Dijelovi hardvera koji mogu biti dijagnosticirati preko aplikacije su: tipkovnica, miš, računalo, ekran, USB priključak, printer, radna memorija i tvrdi disk. Na slici 2.4. je prikazana početna stranica aplikacije u kojoj se može odabrati dio hardvera, a na slici 2.5. je prikazan način na koji su pitanja o simptomima kvara postavljena. Na slici 2.6. je prikazano kako je uzrok pretpostavljen i rješenje kvara ponuđeno [3].

Sl. 2.4. *Početna stranica aplikacije Computer Repair Expert.*

Sl. 2.5. *Ispitivanje mogućih simptoma kvara.*

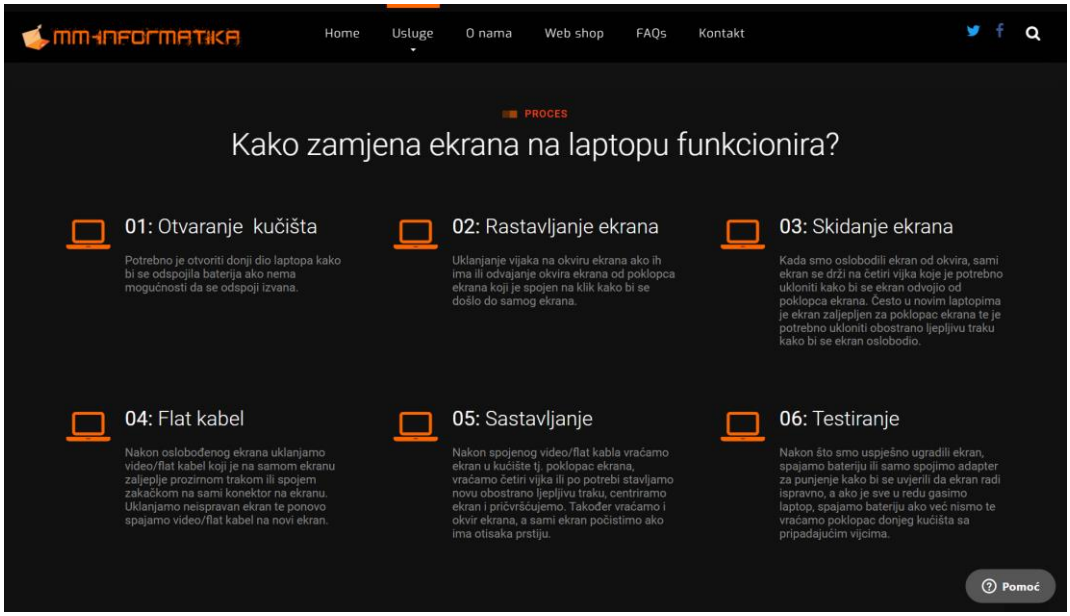


Sl. 2.6. *Uzrok i/ili rješenje kvara.*

2.4. MM-informatika

MM-informatika je interaktivna, dinamična i responzivna web stranica za servis računala i laptopa. Glavna ideja je da se odabere grupa uređaja koja treba popravak, zatim se nazove za detaljan dogovor u vezi popravka i/ili preuzimanja uređaja, te se očekuje popravljeni uređaj na korisnikovoj adresi. Osim toga, na stranici se može pristupiti njihovom web shopu putem hiperveze na drugoj web stranici ili pregledavati često postavljena pitanja i odgovore te kontaktirati njihove servisere. Na slici 2.7. je prikazana početna stranica sa svim bitnim informacijama vezanim uz servis. Kada se izabere usluga poput zamjena LCD (engl. *Liquid Crystal Display*) ekrana na laptopu (slika 2.8.), prikazuju se opće informacije o LCD ekranima, njihov postupak zamjene, načini na koji LCD ekran može biti oštećen ili neispravan, za koje se proizvođače usluga zamjene ekrana nudi, cijena ekrana te mogućnost kontaktiranja [4].

Sl. 2.7. *mm-informatika.com.*



Sl. 2.8. *mm-informatika usluga zamjene LCD ekrana na laptopu.*

3. KORIŠTENE TEHNOLOGIJE PRI IZRADI WEB APLIKACIJE

U nastavku ovog poglavlja bit će ukratko objašnjene tehnologije koje će nam pomoći pri izradi web aplikacije.

3.1. HTML

HTML (engl. *HyperText Markup Language*) je prva stepenica za izradu web stranica i nije programski jezik već daje opis hipertekstualnih dokumenata web pregledniku, a zatim web preglednik daje prikaz te stranice. Strukturu i značenje web stranice definira se HTML-om. Kao što se vidi iz samog imena, HTML povezuje više hipertekstualnih dokumenata – poveznica koje spajaju web stranice. Drugi dio naziva – „Markup“ je način označavanja sadržaja na web stranici kako bi ga web preglednik mogao prikazati.

Sadržaj na stranici se označava pomoću elemenata koji se sastoje od oznaka (eng. *tags*), atributa i samog sadržaja elementa. Ako se kaže da je HTML prva stepenica za izradu stranice, HTML elementi su još manje stepenice koje čine tu jednu veliku stepenicu [5].

Na slici 3.1. prikazan je primjer kako bi bio definiran element za glavni naslov. Element počinje početnom oznakom <h1> i završava završnom oznakom </h1>. Tekst „Sadržaj naslova“ predstavlja sadržaj samog elementa. Atributi unutar elementa pružaju dodatne informacije o elementu, u ovom slučaju „ lang=“hr“ “ definira jezik sadržaja unutar elementa. Nazive elemenata koji su unaprijed definirani, pišu se i velikim i malim slovima, no praksa je da se svaki element piše malim slovima.

```
<h1 lang="hr"> Sadržaj naslova </h1>
```

Sl. 3.1. *Primjer elementa glavnog naslova.*

Svaki osnovni HTML kod se sastoji od zaglavlja (engl. *head*) i tijela (engl. *body*). Primjer osnovnog HTML koda je prikazan na slici 3.2., i on se sastoji od zaglavlja u kojem je naslov (engl. *Title*) „Document“ koji se prikazuje samo naslovima kartica unutar web preglednika, te od tijela u kojem se nalazi glavni naslov i paragraf označen s „<p>“. Svaki HTML kod počinje s „<html>“ i

završava s „</html>“ oznakom. Na slici 3.3. se vidi kako će web preglednik prikazati prethodno napisani osnovni kod [6].

```
1   <!DOCTYPE html>
2   <head>
3       <title>Document</title>
4   </head>
5   <body>
6
7       <h1 lang="hr"> Sadržaj naslova </h1>
8       <p> Paragraf </p>
9
10  </body>
11  </html>
```

Sl. 3.2. *Struktura Osnovnog HTML koda.*

Sl. 3.3. *Prikaz osnovnog HTML koda na web pregledniku.*

3.2. CSS

CSS (engl. *Cascading Style Sheets*) je stilski jezik koji se koristi za definiranje stila i dizajna HTML ili XML (engl. *Extensible Markup Language*) web stranica. Potreba za CSS-om je nastala kada su inženjeri počeli dodavati oznake poput „<font“> i atribute za određenu boju u HTML 3.2 specifikaciju, što je znatno zakompliciralo i produljilo razvoj velikih web stranica. Kao odgovor na to, CSS je stvoren od strane organizacije *World Wide Web Consortium*. Dakle, CSS se koristi kako bi se dodali razni stilovi kao što su boje, fontovi, razmaci i pozicije te kako bi se odvojila prezentacija i dizajn podataka od strukture podataka. Korištenjem CSS-a također se štedi vrijeme jer omogućuje promjenu dizajna više web stranica odjednom ili prilagodbu različitih stilova na jednoj web stranici.

CSS možemo uključiti u HTML dokument na tri različita načina:

- Unutar linije (engl. *inline*)
- Interni ili unutarnji (engl. *internal*) način
- Eksterni ili vanjski (engl. *external*) način

Ako se želi koristiti način unutar linije, onda se unutar HTML elementa dodaje „style“ atribut kao što je prikazano na slici 3.4. U internom načinu, CSS se uključuje koristeći element „<style>“ unutar zaglavlja (<head>) kao što je prikazano na slici 3.5. U eksternom načinu koristi se element poveznice „<link>“ kako bi se povezala vanjska CSS datoteka, kao što je prikazano na slici 3.6. Najbolja praksa je da se koristi eksterni način uključivanja CSS-a unutar HTML-a zbog preglednosti i lakšeg implementiranja na ostale web stranice.

```
<h1 lang="hr" style="color: red"> Sadržaj naslova </h1>
```

Sl. 3.4. *Uključivanje CSS-a na način unutar linije*

```
<head>
    <title>Document</title>
    <style>

        p{
            color: blue;
            font-weight: bold;
        }
    </style>


</head>
```

Sl. 3.5. *Uključivanje CSS-a na interni način.*

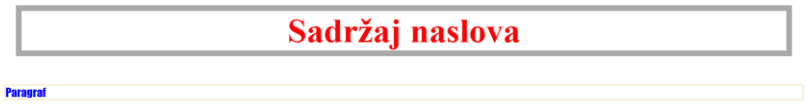
```
<head>
    <title>Document</title>
    <link rel="stylesheet" href="styles.css">
</head>
```

Sl. 3.6. *Uključivanje CSS-a na eksterni način.*

Na slici 3.7. je prikazana vanjska CSS datoteka, te su prikazana neka od osnovnih i najčešće korištenih CSS svojstva poput boje, fonta, veličine fonta, granica, margina i razmaka. Na slici 3.8. je prikazano kako stranica izgleda na web pregledniku kada je HTML dokumentu uključena CSS datoteka [7].

```
index.html ×   # styles.css ×
C: > Users > kdani > Desktop > # styles.css > h1
1  p{
2      color: blue;
3      font-family:Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;
4      font-size:20px;
5      border: 2px solid wheat;
6      margin: 30px;
7
8  }
9
10 h1{
11     color: red;
12     font-weight: bold;
13     font-size:60px;
14     border: 10px solid darkgray;
15     margin: 50px;
16     text-align:center;
17 }
```

Sl. 3.7. *Prikaz vanjske CSS datoteke.*



Sl. 3.8. *Izgled web stranice nakon uključivanja prethodno dodane CSS datoteke.*

U kontekstu sintakse, na prethodnoj slici **Pogreška! Izvor reference nije pronađen.** prikazano je da se svaki stilski obrazac sastoji od selektora i deklaracije. Dizajn se primjenjuje na točno određeni element putem selektora, dok deklaracija određuje kako će taj element biti opisan i dizajniran CSS-om. Primjerice, crvena boja je dodana za naslov „h1“ putem deklaracije „color: red;“ te je vidljiv uzorak „selektor {deklaracija Naziv elementa se najčešće koristi za selektiranje, no mogu se definirati i vlastiti ID i klasa selektori. ID selektor se koristi za definiranje stila samo jednog HTML elementa, dok se klasa koristi za definiranje stila za više HTML elemenata, čime se ubrzava proces razvoja. Kod sva tri tipa selektora, deklaracija se piše na isti način, uz dodavanje ljestvi „#“ na početak ID selektora i točke na početak klasa selektora [8].

3.3. Bootstrap

Bootstrap se smatra jednim od od najpopularnijih HTML, CSS i JavaScript front-end razvojnih okvira za brzu, stabilnu i laku izradu i dizajniranje web stranica. Ovaj okvir temelji se na „mobile-first“ responzivnom pristupu, pri čemu se dizajn i razvoj prvo provode za mobilne uređaje, a zatim se lako prilagođavaju većim zaslonima. Trenutno je najnovija verzija Bootstrap 5, koja se razlikuje od prijašnjih verzija po novim komponentama i većoj responzivnosti, te koristi JavaScript umjesto jQuery programskog jezika. Bootstrap je kompatibilan sa svim modernim web preglednicima.

Bootstrap 5 se može vrlo jednostavno instalirati na dva načina:

- Putem upravitelja paketa: Naredbom unutar konzole: „npm install bootstrap@5.3.3“
- Putem CDN-a (engl. *Content Delivery Network*) korištenjem poveznice na vanjsku CSS datoteku i JavaScript skriptu, kao što je prikazano na slici 3.9.

Oba načina su besplatna i mogu se preuzeti sa njihove službene Bootstrap stranice na poveznici https://getbootstrap.com/ [9].

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6
7       <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet"
8       integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
9
10      <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
11      integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz" crossorigin="anonymous"></script>
12
13
14      <title>Document</title>
15  </head>
16  <body>
17
18  </body>
19  </html>
20  </html>
```

Sl. 3.9. *Bootstrap instalacija.*

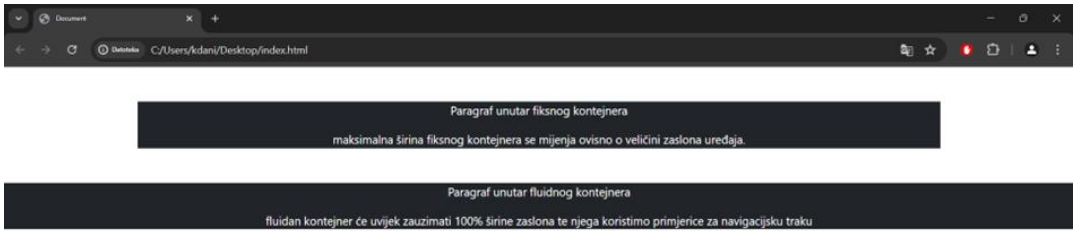
Također, važno je da se napiše sljedeća linija koda prikazana na slici 3.10., odnosno da se doda meta oznaka unutar zaglavlja koja omogućuje normalno funkcioniranje Bootstrap mobilne responzivnosti. Dio „width=device-width“ postavlja širinu stranice da prati širinu zaslona uređaja, dok dio „initial-scale=1“ postavlja početnu razinu zumiranja kada se stranica prvi put učita. Osim

navedene linije koda za mobilnu responzivnost, važno je koristiti i HTML5 tip dokumenta, kao i odgovarajući atribut jezika i ispravni skup znakova.

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

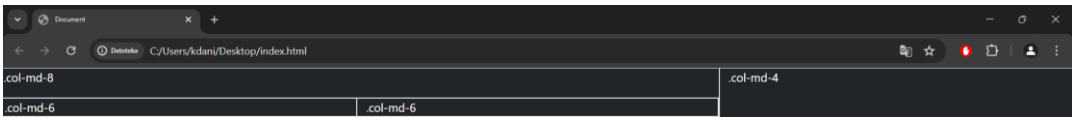
Sl. 3.10. *Meta oznaka za mobilnu responzivnost.*

Kontejneri su jedni od najosnovnijih elemenata unutar Bootstrapa i koriste se za smještanje sadržaja stranice unutar njih. Postoje dvije vrste kontejnera u Bootstrapu: fiksni kontejner i fluidni kontejner. Fiksni kontejner je responzivan, a njegova maksimalna širina mijenja se ovisno o veličini zaslona uređaja. Fiksni kontejner dobiva se pomoću klase „container“. S druge strane, fluidni kontejner uvijek zauzima 100% širine i koristi se, primjerice, za navigacijsku traku kada je potrebno da sadržaj uvijek bude preko cijele širine zaslona. Fluidni kontejner dobiva se pomoću klase „container-fluid“. Primjer kontejnera je prikazan na slici 3.11.



Sl. 3.21. *Primjer Bootstrap kontejnera.*

Osim kontejnera, vrlo važan sustav unutar Bootstrapa je sustav rešetki (engl. *grid system*). Sustav rešetki koristi se za raspoređivanje sadržaja elemenata na web stranici. Sustav je izrađen pomoću niza redova (engl. *rows*) i stupaca (engl. *columns*) koji su potpuno responzivni i mogu se kombinirati po želji. Omogućeno je stvaranje do 12 kolona na jednoj stranici, no nije obavezno koristiti svih 12, te se može kreirati vlastiti raspored stupaca i kolona. Sustav rešetki uključuje šest klasa: *sm*, *md*, *lg*, *xl* i *xxl*, koje se koriste za kreiranje fleksibilnog rasporeda ovisno o širini ekrana. Na slici 3.12. prikazan je primjer jednostavnog sustava rešetki.



Sl. 3.12. *Primjer Bootstrap rešetki.*

Korištenjem unaprijed definiranih Bootstrap klasa može se uređivati tipografija, slike i tablice, oblikovati forme, slike i tablice te dodavati ikone, upozorenja i druge elemente. Na slici 3.13. prikazan je primjer korištenja različitih Bootstrap klasa [10].

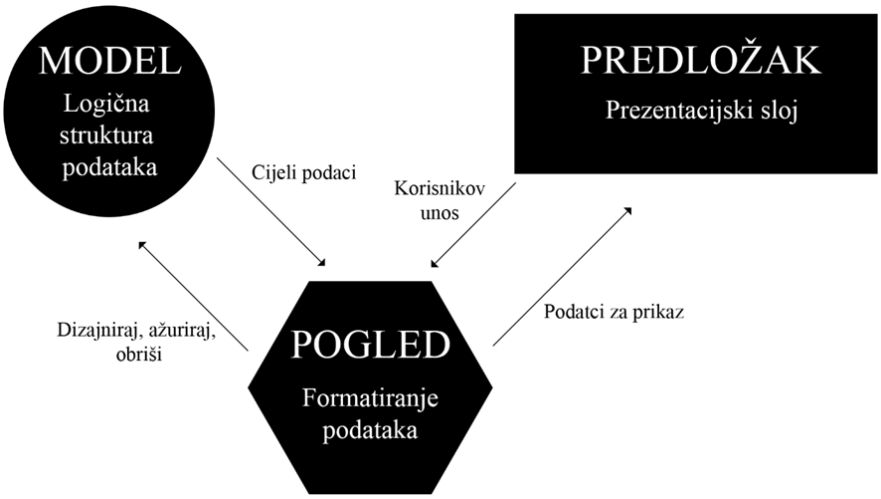
Sl. 3.33. *Primjer korištenja Bootstrap klasa.*

3.4. Django

Django je razvojni okvir visoke razine za back-end web programiranje u Python programskom jeziku. Besplatan je i otvorenog je koda, što omogućava svakome da prouči izvorni kod. Razvijen je od strane iskusnih programera, što omogućava brzi razvoj web aplikacija s čistim kodom i pragmatičnim dizajnom. Dizajniran je s namjerom da se programeru omogući relativno brz prelazak od koncepta do stvarnog proizvoda. Najveće prednosti nalaze se u području backenda, uključujući jednostavnu bazu podataka i administraciju te rješavanje problema autentifikacije korisnika. S ugrađenom korisničkom autentifikacijom sprječavaju se sigurnosni upadi poput SQL injectiona i cross-site falsificiranja zahtjeva. Svestranost mu omogućava primjenu u raznim tvrtkama, organizacijama, pa čak i vladinim organizacijama za razvoj web stranica [11].

Django je baziran na MVT (engl. *Model View Template*) arhitekturi. Modeli predstavljaju strukturu podataka i njihove odnose iz baze podataka. Django ne koristi komplicirane SQL upite, već upravlja modelima s pomoću Python objekata koristeći ORM (engl. *Object-Relational Mapping*). Modeli se nalaze u datoteci „models.py“. Pogledi (engl. *view*) su funkcije ili metode

koje uzimaju HTTP zahtjeve, obrađuju ih, uvoze potrebne modele i pronalaze koje podatke treba poslati u predložak, vraćajući rezultat kao HTTP odgovor. Ukratko pogledi prikazuju podatke korisnicima, te se nalaze u „views.py“ datoteci. Predložak (engl. *template)* je HTML tekstualna datoteka koja prezentira kako će web stranica i podatci na njoj biti prikazani. Django koristi vlastiti mehanizam za izradu pogleda tako da koristi Django oznake i time omogućava unos dinamičkih podataka. Predlošci se nalaze u „templates“ mapi. Na slici 3.14. je prikazana struktura Django MVT arhitekture.



Sl. 3.44. *Struktura Django MVT arhitekture.*

Osim MVT arhitekture, Django se također oslanja na URL-ove koji se koriste za navigaciju između različitih stranica unutar web stranice. URL-ovi su povezani s pogledima, tako da Django određuje koji pogled će biti pozvan kada se zatraži određeni URL. Konfiguracija URL-ova se obavlja unutar datoteke „urls.py“.

Ukratko, Django funkcionira na sljedeći način:

1. URL se unosi od strane korisnika u preglednik i šalje HTTP zahtjev Django aplikaciji. HTTP zahtjev se prima od strane Django-a, provjerava se datoteka „urls.py“, te se poziva odgovarajući pogled.
2. Pogled, koji se nalazi u datoteci „views.py“, obrađuje zahtjev i pristupa odgovarajućim modelima iz „models.py“.
3. Zatim se pogled pronalazi odgovarajući predložak unutar mape predložaka te šalje podatke predlošku.
4. Generira se HTML s Django oznakama i šalje se korisniku kao HTTP odgovor [12].

3.5. JavaScript

JavaScript je jedan od najvažnijih programskih jezika današnjice. Teško je zamisliti kako bi današnje web stranice izgledale bez JavaScripta. Svaki web programer se mora naučiti tri jezika, to su: HTML za definiranje strukture web stranice, CSS za dizajn i definiranje rasporeda stranice te sam JavaScript za programiranje ponašanja web stranice. JavaScript je tipizirani i dinamički programski jezik koji omogućuje stvaranje interaktivnih elemenata na web stranici kao što su validacija formi i DOM (engl. *Document Object Model*) manipulaciju.

HTML sadržaj može biti mijenjan putem JavaScripta korištenjem metoda poput „getElementById()“ koja zahvaća element unutar HTML dokumenta po njegovoj ID oznaci i omogućuje njegovu manipulaciju. JavaScript može mijenjati vrijednosti HTML atributa, što omogućuje primjerice promjenu izvora slike na HTML elementu. JavaScript također može mijenjati stil HTML elementa, sakriti ili pokazati HTML element i još puno toga [13].

Važna tehnika u razvoju web stranica je AJAX (engl. *Asynchronous JavaScript and XML*). AJAX nije programski jezik, već koristi kombinaciju *XMLHttpRequest* objekta koji je već ugrađen u sami web preglednik, kako bi zatražio podatke od servera i JavaScripta i HTML DOM za prikaz ili manipulaciju podataka. Glavna mu je odlika to da omogućuje dinamičko prikazivanje i slanje podataka u pozadini, bez potrebe za ponovnim učitavanjem i osvježavanjem stranice.

Ajax funkcionira na sljedeći način:

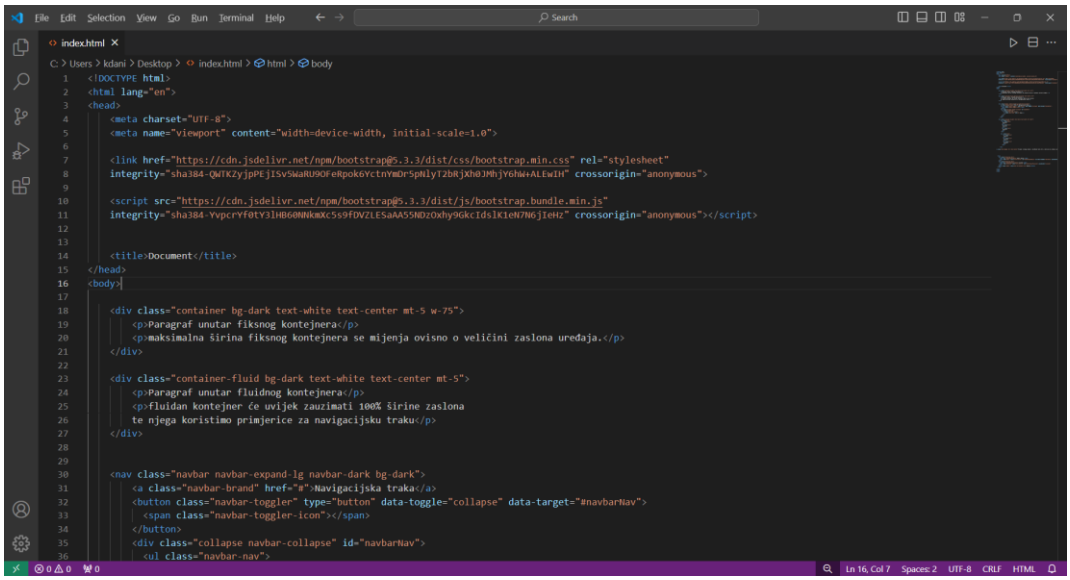
1. Nekakav događaj poput klika na gumb se pojavi.
2. JavaScript kreira *XMLHttpRequest* objekt.
3. Taj kreirani *XMLHttpRequest* objekt šalje zahtjev serveru.
4. Server obrađuje zahtjev i šalje odgovor natrag na web stranicu.
5. JavaScript pročita odgovor.
6. JavaScript napravi određenu akciju poput ažuriranja sadržaja stranice.

AJAX omogućuje bržu, dinamičniju i responzivniju web aplikaciju, ne guši mrežu jer prenosi samo potreban promet i pruža bolje korisničko iskustvo. Najveća mana mu je sigurnost, zato što je podložan napadima poput cross-site skripting napada [14].

3.6. Visual Studio Code

Visual Studio Code je besplatni softver za uređivanje koda napravljen od strane Microsofta, te je baziran na otvorenom kodu. Dostupan je za većinu operacijskih sustava poput macOS, Windows i Linux, te se može preuzeti s njihove službene stranice, s pomoću naredbe u konzoli ili preko različitih sustava za upravljanje paketima. Ima vrlo jednostavno korisničko sučelje i vrlo brzo ga mogu naučiti programeri početnici. Vrlo je popularan zbog svoje brzine i mogućnosti uređivanja korisničkog sučelja na razne načine poput dodavanja tema i ostalih ekstenzija. Ima odličnu integraciju s različitim sustavima za kontrolu verzija poput Gita, dolazi s ugrađenim terminalom i podržava velik broj programskih jezika poput C, C++, C#, JavaScript, Python, Java i tako dalje. VS Code dolazi s IntelliSense sustavom za automatsko dovršavanje koda, dokumentacija i uputa. Pruža mogućnost debugiranja koda izravno iz samog uređivača koristeći prekidne točke (engl. *break points*) [15].

Na slici 3.15. je prikazano korisničko sučelje VS Codea na primjeru jednostavnog projekta u HTML-u.



Sl. 3.55. *Korisničko sučelje Visual Studio Codea.*

4. IZRADA I FUNKCIONALNOSTI APLIKACIJE

U ovom poglavlju prikazane su funkcionalnosti i struktura web aplikacije, prikazuje se izrada web aplikacije te je opisana struktura baze podataka i dohvaćanje podataka iz baze podataka.

4.1. Struktura aplikacije

Izrađena aplikacija omogućuje da se definiraju četiri tipa korisnika: administrator, koji je ujedno i zaposleni serviser računala, registrirani korisnik, neregistrirani korisnik ili gost te superkorisnik (engl. *superuser*), koji predstavlja vlasnika servisa.

Administratori se postavljaju od strane superkorisnika – vlasnika servisa, koji jedini ima pristup bazi podataka i sva dopuštenja kao i administrator, a ujedno ima najveću ulogu. Po hijerarhiji uloga, administrator dolazi sljedeći te mu se dodjeljuju sljedeća dopuštenja: dodavanje i brisanje članaka vezanih uz održavanje računala, prikaz svih postojećih aktivnih soba za razgovor svakog korisnika, sudjelovanje kao glavni odgovorni sugovornik u sobama za razgovor te pregled i ažuriranje svih postojećih narudžbi.

Registrirani korisnik definira se kao korisnik koji je izradio svoj profil putem registracije na stranici i trenutačno je prijavljen na stranici. Registriranim korisnicima omogućuje se pregledavanje članaka o održavanju računala, uvid u osnovnu analitiku servisa na početnoj stranici te naručivanje na servis prijavom kvara na računalu ili laptopu, izradom narudžbe za čišćenje računala ili instalaciju operacijskog sustava. Omogućuje im se i dijagnoza kvara uz pomoć soba za razgovor u realnom vremenu, gdje se u suradnji s administratorom pokušava otkloniti kvar ili dogovoriti naručivanje na servis. Registriranim korisnicima pruža se stalan uvid u njihove aktivne sobe za razgovor, uz mogućnost povratka na prošlu aktivnu sobu ili zatvaranja sobe kada je problem riješen. Također, omogućuje im se pregled novih aktivnih i prošlih završnih narudžbi.

Neregistrirani korisnici, ili samo gosti, imaju uvid u osnovnu analitiku servisa te mogu čitati članke o održavanju računala. Međutim, neregistriranim korisnicima nije omogućeno naručivanje na servis, čišćenje računala, instalacija operacijskog sustava, niti sudjelovanje u sobama za razgovor.

U nastavku ovog rada detaljno će se objasniti svaka od funkcionalnosti aplikacije.

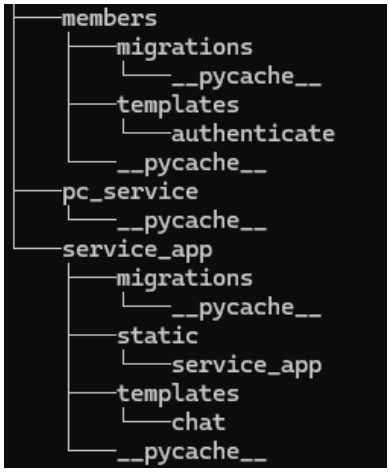
4.2. Struktura Django projekta

Prilikom pokretanja aplikacije, potrebno je prvo instalirati i postaviti Python virtualno okruženje u direktorij projekta koji je prethodno preuzet s GitLab repozitorija. Bash naredba za kreiranje virtualnog okruženja je sljedeća: „python -m venv env“. Nakon toga, virtualno okruženje se aktivira koristeći bash naredbu: „source env/bin/activate“.

Idući korak je instalirati potrebne pakete za rad aplikacije, poput Djanga, koji su navedeni u *requirements.txt* datoteci, s pomoću bash naredbe: „pip install -r requirements.txt“.

Korištenjem terminala, trenutni direktorij se mijenja u projekt aplikacije *pc_service* s pomoću naredbe: „cd pc_service“. Nakon toga, razvojni poslužitelj se pokreće naredbom: „py manage.py runserver“. Kada je poslužitelj pokrenut, aplikaciji se može pristupiti putem preglednika upisivanjem adrese: „http://localhost:8000“.

Unutar korijenskog direktorija Django projekta nalazi se skripta *manage.py* koja služi za pokretanje servera, upravljanje migracijama i korištenje konzole, te tri poddirektorija: *members*, *pc_service* i *service_app*. Na slici 4.1.prikazana je osnovna struktura Django projekta.



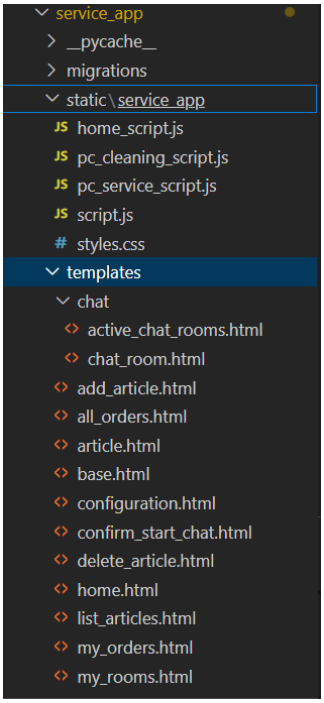
Sl. 4.1. *Osnovna struktura Django projekta.*

Glavni direktorij projekta je *pc_service* u kojem se nalaze datoteke *settings.py*, *urls.py*, *wsgi.py* i *asgi.py*. Datoteka *settings.py* koristi se za definiranje postavki projekta, kao što su instalirane aplikacije, vremenska zona projekta i tako dalje. Datoteka *urls.py* unutar ovog direktorija koristi se za definiranje glavnih ruta web aplikacije i spajanje s preostale dvije aplikacije unutar projekta.

Datoteka *tests.py* koristi se za pisanje testova. Datoteke *wsgi.py* i *asgi.py* služe za pokretanje servera u produkciji.

Preostala dva poddirektorija su aplikacije *members* i *service_app*. Aplikacija *members* koristi se za autorizaciju korisnika, postavljanje konfiguracije računala i promjenu postavki profila. Aplikacija *service_app* sadrži sve ostale funkcionalnosti projekta, poput početne stranice, navigacijske trake, naručivanja na servis i ostale mogućnosti.

Svaka aplikacija u Django projektu ima istu standardnu strukturu, pa tako i naše dvije aplikacije *members* i *service_app*. Struktura se sastoji od sljedećih datoteka: *models.py* za definiranje baze podataka i modela, *views.py* za funkcije i metode koje uzimaju http zahtjeve, *urls.py* za rute svake aplikacije posebno, *admin.py* za konfiguraciju administracijskog sučelja, poput prikaza modela na adresi: „http://localhost:8000/admin“, te *migrations.py* za migracije baze podataka koje su generirane naredbom. Također, obadvije aplikacije sadrže i *templates* i *static* poddirektorije u kojima su smještene HTML, CSS, JavaScript i slične datoteke, kao što je prikazano na slici 4.2.



Sl. 4.2. *Static i templates poddirektoriji aplikacije service_app.*

Unutar navedenih dviju aplikacija naknadno je dodana datoteka *forms.py* za definiranje formi, poput tekstualnih polja, lozinki, email adresa i slično, koje služe za prikaz na web aplikaciji i

povezivanje s bazom podataka. Primjerice, u aplikaciji *members* postoji datoteka *forms.py* koja sadrži obrasce za registraciju i prijavu korisnika.

Unutar direktorija aplikacije *members* dodane su i datoteke: *context_processors.py*, *apps.py* i *signals.py*. Datoteka *context_processors.py* služi za kreiranje kontekstualnih procesora, odnosno za kreiranje funkcije koja se može koristiti u svim HTML šablonama. Unutar te datoteke postoji funkcija koja provjerava je li trenutačno prijavljeni korisnik administrator, koju zatim možemo proslijediti drugim HTML šablonama.

Datoteka *apps.py* koristi se za upravljanje i organizaciju aplikacije. U slučaju ove aplikacije, postavlja se naziv aplikacije *members* i zadani tip polja za automatske identifikatore u modelima na velike cijele brojeve, zbog toga što se radi o korisničkim računima. Datoteka *signals.py* sadrži signale koji izvršavaju funkciju kada se dogode događaji poput spremanja modela. U slučaju ove aplikacije, postoji signal koji se aktivira nakon što je model spremljen; kada se kreira ili sprema novi korisnik, automatski se sprema odgovarajući profil korisnika.

4.3. Struktura baze podataka

Iako Django podržava više baza podataka poput PostgreSQL, MariaDB, MySQL i slično, za potrebe ove aplikacije korištena je zadana baza podataka za Django – SQLite. SQLite je baza podataka koja je jednostavna za korištenje, ne koristi server, već sprema podatke u lokalnu datoteku, što ju čini idealnom za kreiranje, testiranje i razvoj malih web aplikacija.

Odabir baze podataka i konfiguracija postavki baze podataka odrađuje se unutar datoteke *settings.py* koja se nalazi u glavnom direktoriju projekta.

Django koristi ORM (engl. *Object-Relational Mapping*) za rad s bazama podataka, čime se eliminira potreba za SQL upitima i olakšava rad s bazom podataka. Kada se kreira Django projekt, dobiva se prazna SQLite baza podataka koja se nalazi u korijenskom direktoriju projekta pod nazivom „db.sqlite3“. Svi modeli koji se kreiraju bit će spremljeni kao tablice unutar te datoteke.

Modeli su podaci baze podataka koji su spremljeni kao objekti u Djangu i čine strukturu baze podataka. Svaki model je tablica u bazi podataka, a atributi modela su stupci.

Na slici 4.3. prikazan je model za članak na web aplikaciji koji se nalazi u datoteci *models.py* aplikacije *service_app*. Model je klasa koja nasljeđuje *models.Model*, a u ovom slučaju prikazan je model *Article* koji ima pet polja koja odgovaraju različitim tipovima podataka u bazi podataka.

```
class Article(models.Model):

    author = models.ForeignKey(User, on_delete=models.CASCADE, null=True)
    title = models.CharField(max_length=150)
    description = models.TextField()
    image = models.CharField(max_length=600)
    pub_date = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.title
```

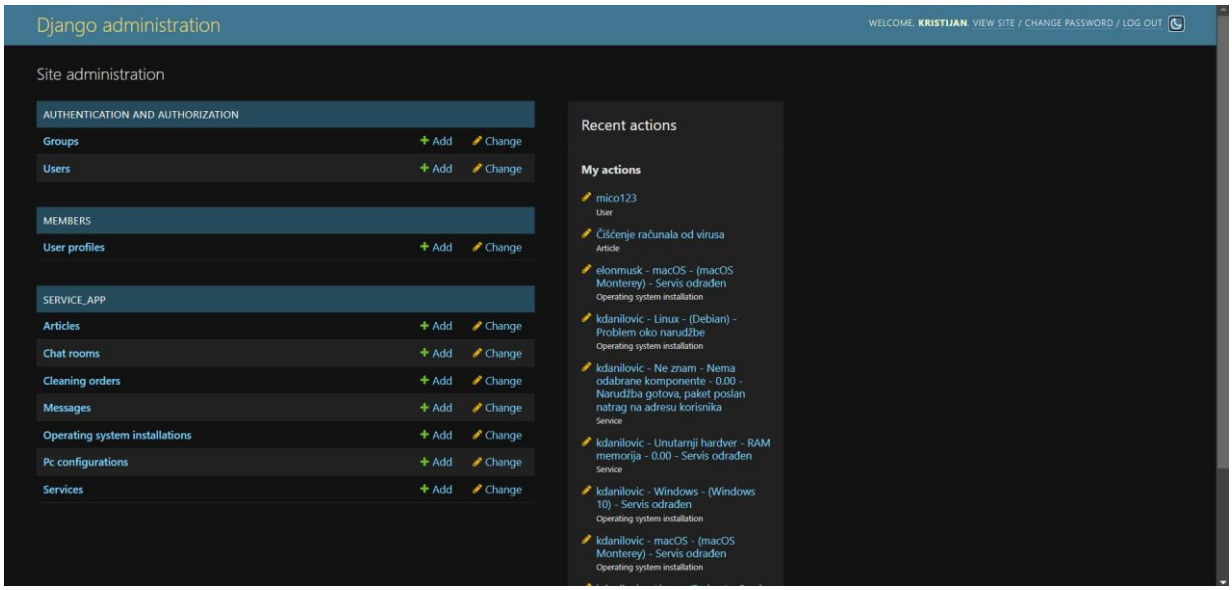
Sl. 4.3. *Model za članak na web aplikaciji.*

Polje *author* sadrži vezu između modela *Article* i modela *User*, koji predstavlja korisnika, odnosno svaki članak može imati jednog autora koji je korisnik iz modela *User*. Opcija *on_delete=models.CASCADE* u slučaju brisanja korisnika „kaskadno“ briše sve povezane članke s tim autorom. Opcija *null=True* omogućuje da polje *author* bude prazno. Naslov je pohranjen kao tekst u tipu *CharField*, ograničen na 150 znakova. Opis se pohranjuje u tipu *TextField*, bez ograničenja dužine. Slika je predstavljena URL-om ili putanjom pohranjenom u *CharField* tipu, s ograničenjem od 600 znakova. Datum objave pohranjuje se u *DateTimeField* tipu, gdje je automatsko postavljanje datuma i vremena omogućeno opcijom *auto_now_add=True*.

Za potrebe razumljivosti i jasnoće prikaza objekta koristi se „___str___“ metoda koja u navedenom slučaju vraća naslov članka umjesto npr. „<Article object(55)>“ . Ovo je korisno za ispis objekta ili za prikaz objekta u administrativnom sučelju.

Svaki model, nakon definiranja, potrebno je spremiti u bazu podataka. S pomoću migracija, koristeći naredbu „py manage.py makemigrations“, datoteka će biti kreirana i spremljena u *migrations* direktorij. Nakon toga, potrebno je izvršiti naredbu „py manage.py migrate“ kako bi SQL upit, baziran na opisu nove datoteke, bio izvršen od strane Djanga.

Za lakše upravljanje podacima koristi se *Django Admin* alat, koji pruža korisničko sučelje prikazano na slici 4.4. i kojem se pristupa na adresi „http://localhost:8000/admin/“. *Django Admin* alat automatski se generira za svaki model koji se registrira u *admin.py* datoteci, omogućujući jednostavno upravljanje, filtriranje i pretraživanje podataka, te eliminirajući potrebu za klasičnim SQL upitima.



Sl. 4.4. *Korisničko sučelje Django Admina.*

Na slici 4.5. prikazan je ER dijagram baze podataka aplikacije servis računala. Prijavljeni korisnik može biti administrator ili običan korisnik, a uloga je definirana u tablici *Groups*, u kojoj su sadržani ID i ime uloge. Model *Groups* je dio *Django auth* modela. Veza više na više između korisnika i tablice *Groups* ostvarena je putem ID-a korisnika i ID-a grupe.

Korisnik ima atribute ime, prezime, email adresu, datum registracije, korisničko ime, lozinku, datum kada je zadnji put prijavljen te boolean atribute koje odgovaraju na pitanje je li korisnik aktivan, je li korisnik super korisnik ili član osoblja.

Glavna tablica korisnika izvedena je nasljeđivanjem modela *AbstractUser* iz Django modela, čime je dodatno stvorena i tablica *UserProfile* za opisivanje korisnika. *UserProfile* povezana je vezom jedan na jedan preko ID-a korisnika, omogućujući da svaki korisnik ima samo jedan profil. Osim veze jedan na jedan, tablica *UserProfile* proširena je atributima poput samostalnog identifikacijskog broja, adrese, grada i države korisnika. Važnost te tablice očituje se u osiguravanju ispravne dostave paketa.

Konfiguracija računala svakom je korisniku dodijeljena, a unutar tablice *PCConfiguration* veza je ostvarena stranim ključem koji se odnosi na ID korisnika. Svaka konfiguracija dodatno je opisana atributima poput procesora, vrste uređaja, grafičke kartice, operacijskog sustava, RAM memorije, veličine i vrste pohrane.

Za objavljivanje članka na stranici, korisniku mora biti dodijeljena administratorska uloga. Funkcija za provjeru ove uloge definirana je u pogledima, pri čemu se kao rezultat vraća True ako je korisnik administrator, odnosno False ako nije. Pristup stranici za dodavanje članaka onemogućen je korisnicima koji nisu administratori. Članak je definiran tablicom *Article,* a svaki članak povezan je s autorom preko stranog ključa, što predstavlja vezu jedan na više. Jedan autor može objaviti više članaka. Atributi članka uključuju tekst članka ili opis, sliku, datum objave i naslov.

Razgovor u realnom vremenu omogućuje se korisniku preko sobe za razgovor, koja je definirana u tablici *ChatRoom*. Prilikom kreiranja sobe, automatski joj se dodjeljuje jedinstveni identifikacijski broj. Veza sobe za razgovor s korisnikom ostvarena je preko stranog ključa *user*, koji pokazuje na ID korisnika, čime je omogućeno da jedan korisnik ima više soba za razgovor. Sobi za razgovor dodijeljen je i status – otvorena ili zatvorena, te datum kada je kreirana.

Poruke nastale unutar sobe za razgovor definirane su u tablici *Message*. Svakoj poruci automatski se dodjeljuje identifikacijski broj, kao i broj sobe (strani ključ na ID sobe za razgovor). Pošiljatelj poruke definiran je kao korisnik koji je trenutačno prijavljen, a njegov ID dodijeljen je kao strani ključ. Poruci su također dodijeljeni atributi teksta poruke i datuma kada je napisana.

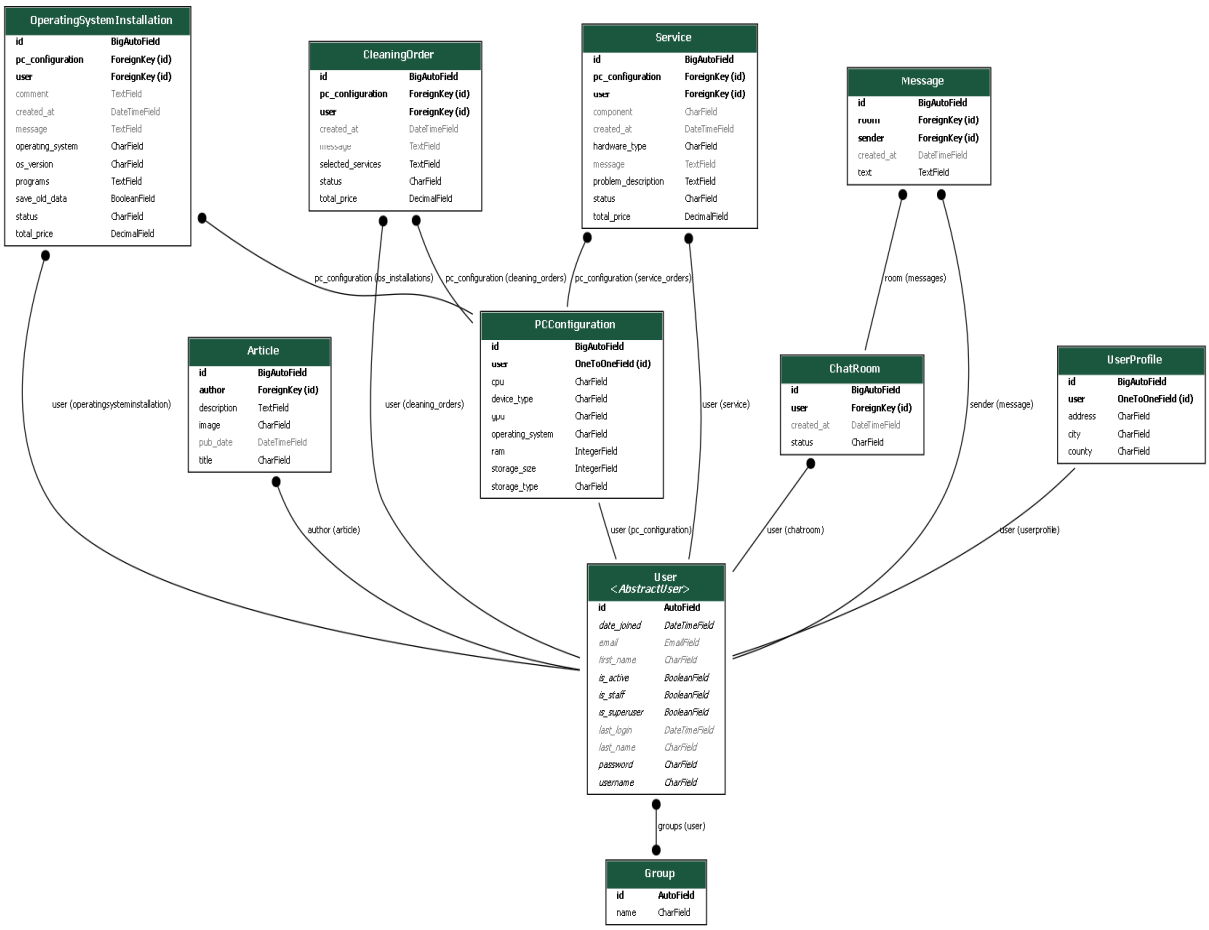
Narudžba za instalaciju operacijskog sustava može biti postavljena od strane korisnika i definirana je u tablici *OperatingSystemInstallation*. Svakoj narudžbi automatski se dodjeljuje jedinstveni identifikacijski broj. Veza između narudžbe i korisnika ostvarena je preko stranih ključeva: *pc_configuration,* koji pokazuje na konfiguraciju korisnika, i *user,* koji pokazuje na ID korisnika. U narudžbi za instalaciju operacijskog sustava nalaze se podaci o operacijskom sustavu, verziji operacijskog sustava, programima, statusu narudžbe, cijeni, poruci servisera, komentaru korisnika te odluci korisnika o čuvanju starih podataka.

Narudžba za čišćenje računala može biti postavljena od strane korisnika i definirana je u tablici *CleaningOrder*. Svakoj narudžbi automatski se dodjeljuje jedinstveni identifikacijski broj. Veza između narudžbe za čišćenje računala i korisnika ostvarena je na isti način kao i kod narudžbe za instalaciju operacijskog sustava. U narudžbi za čišćenje računala nalaze se podaci o odabranim

uslugama čišćenja, statusu narudžbe, poruci servisera, datumu kreiranja narudžbe i konačnoj cijeni.

Narudžba za servis računala može biti postavljena od strane korisnika i definirana je u tablici *Service*. Kao i u prethodna dva slučaja narudžbi, veza s korisnikom također je ostvarena na isti način. U narudžbi za servis računala navedeni su podaci o komponentama za popravak, tipu hardvera, poruci servisera, opisu problema, statusu narudžbe, konačnoj cijeni i datumu kreiranja narudžbe.

Tablice i veze između njih osmišljene su kako bi omogućile pravilno funkcioniranje aplikacije u skladu sa zahtjevima.



Sl. 4.5. ER dijagram *baze podataka aplikacije Servis računala* .

4.4. Neregistrirani i registrirani korisnici

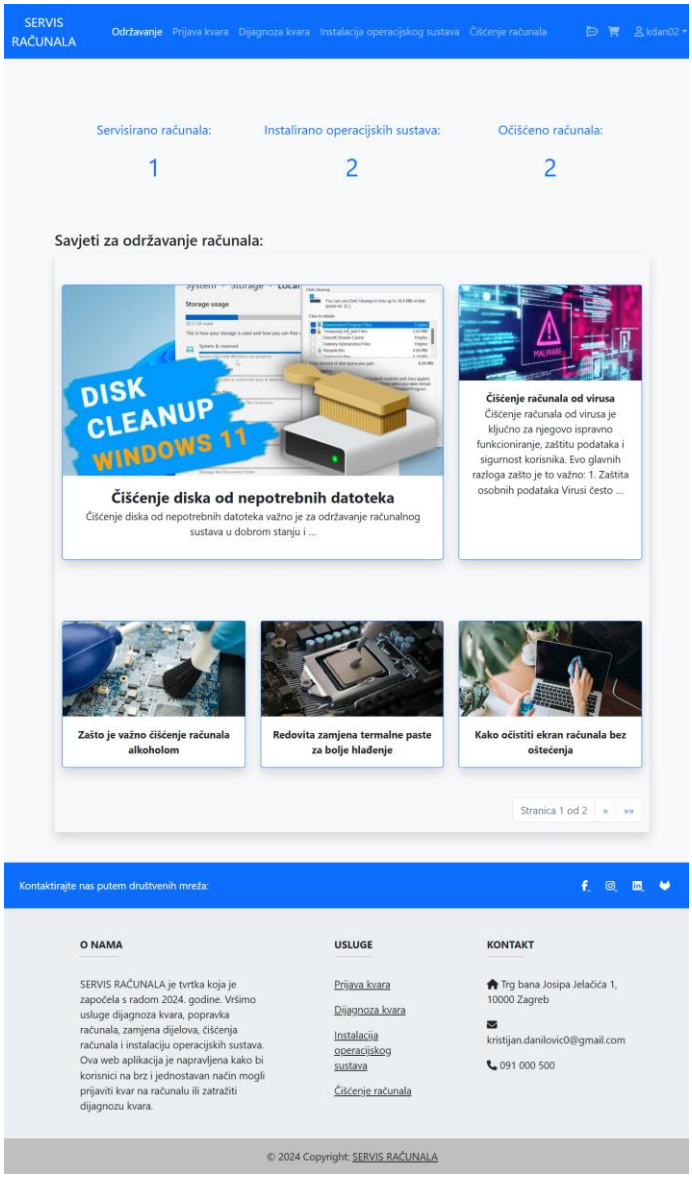
Način rada, izrada funkcionalnosti te prikaz aplikacije za registrirane i neregistrirane korisnike opisani su u ovom poglavlju.

4.4.1. Početna stranica aplikacije

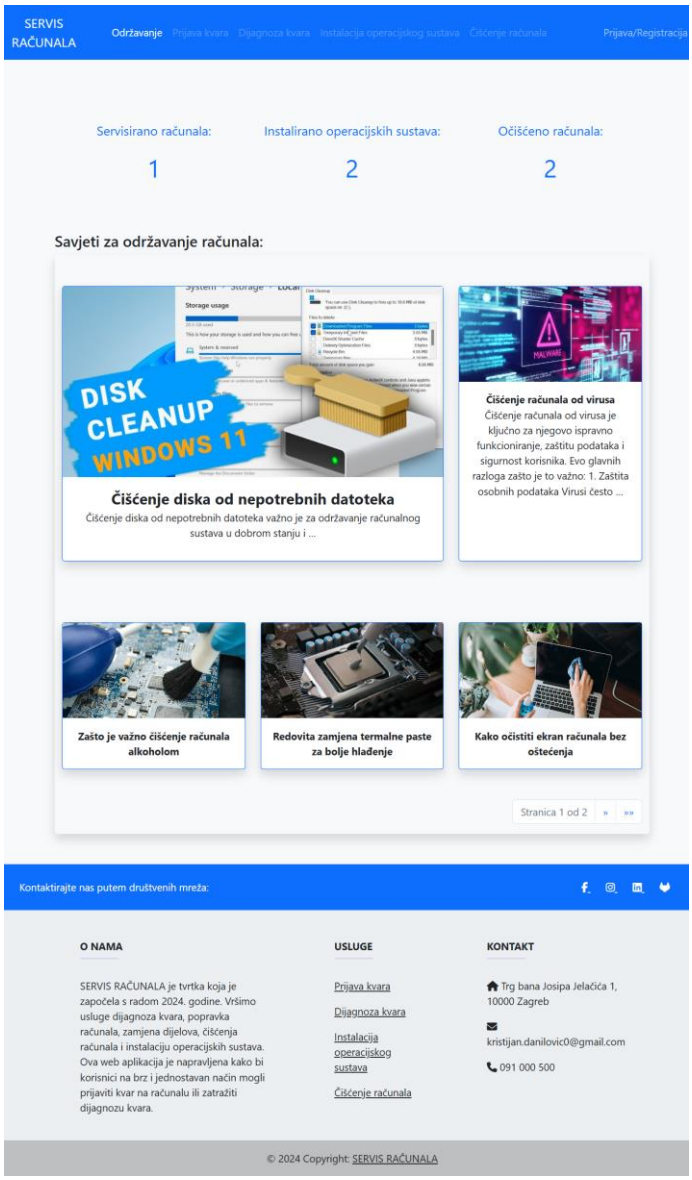
Na početnoj stranici aplikacije nalazi se navigacijska traka na kojoj su prikazani naziv aplikacije te poveznice na sljedeće usluge: održavanje, prijava kvara, dijagnoza kvara, instalacija operacijskog sustava, čišćenje računala, prijava ili registracija. U slučaju da je korisnik trenutno prijavljen kao što je prikazano na slici 4.6., ponuđen mu je pregled narudžbi i pregled aktivnih soba za razgovor te padajući izbornik s mogućnostima uređivanja konfiguracije računala, postavki profila i odjava.

U slučaju neregistriranog korisnika kao što je prikazano na slici 4.7., poveznice na održavanje, prijavu kvara, dijagnozu kvara, instalaciju operacijskog sustava i čišćenje računala su vidljive, ali im se ne može pristupiti, osim poveznici za održavanje, odnosno početnoj stranici. Poveznica na registraciju i prijavu također je dostupna.

Detaljan opis poveznica bit će opisan u nastavku rada.



Sl. 4.6. *Početna stranica aplikacije za registriranog korisnika.*



Sl. 4.7. *Početna stranica aplikacije za neregistriranog korisnika.*

Na stranici je prikazana osnovna analitika servisa koja pokazuje koliko je računala uspješno servisirano, očišćeno ili koliko je operacijskih sustava instalirano. Analitika je dostupna za pregled registriranim i neregistriranim korisnicima te je prikazana u dinamičkom obliku s animacijom.

Ispod osnovne analitike servisa prikazani su savjeti za održavanje računala kroz kartice sa slikama, naslovom i kratkim opisom. Klikom na karticu korisnik je preusmjeren na detaljan članak.

Na dnu stranice nalazi se podnožje s opisom servisa, poveznicama dostupnim registriranim korisnicima te kontakt podacima (adresa, telefon, e-mail). Prikazane su i ikone društvenih mreža povezane s profilima servisa.

Početna stranica dizajnirana je s pomoću Bootstrap 5 razvojnog okvira koji je ranije opisan u radu i CSS-a. Animacije i dohvaćanje podataka za osnovnu analitiku servisa omogućeni su s pomoću JQuery JavaScript biblioteke. Bootstrap 5 je instaliran kao paket s pomoću sljedeće naredbe u bashu: „pip install django-bootstrap5“, a zatim je dodan u instalirane aplikacije unutar *settings.py* datoteke. CSS i ostale JavaScript skripte smještene su u direktorij *static*.

Navigacijska traka izrađena je u posebnoj datoteci *base.html* unutar *templates* direktorija glavne aplikacije *pc_service* i povezuje se sa svakom stranicom u aplikaciji. Navigacijska traka povezuje se sljedećom linijom koda na početku svake HTML datoteke: „{% extends "base.html" %}“.

4.4.2. Osnovna analitika servisa

Osnovna analitika servisa, koja prikazuje broj servisiranih i očišćenih računala te instaliranih operacijskih sustava, smještena je na početnoj stranici u datoteci *home.html*. Brojači za prikaz analitike definirani su HTML kodom označeni su klasom *.counter*, koja omogućuje animaciju brojeva s pomoću JavaScript koda. Podaci za brojače dohvaćaju se iz baze podataka kroz *Django template* sintaksu i prikazuju s pomoću atributa *data-target*, koji sadrži ciljnu vrijednost.

Filtriranjem objekata sa statusima poput "servis odrađen" i "završena narudžba" generiraju se vrijednosti za analitiku, koje se zatim vraćaju funkcijom *render*. JavaScript kod prikazan na slici 4.8. animira brojače tako da vrijednosti postupno rastu od nule do cilja, koristeći funkciju *requestAnimationFrame* za glatku promjenu.

service_app > static > service_app > JS home_script.js > ...

```javascript
window.onload = function() {
    document.querySelectorAll('.counter').forEach(counter => {
        const target = +counter.getAttribute('data-target');
        let count = 0;

        function updateCounter() {
            const increment = target / 50;
            count += increment;
            if (count > target) count = target;

            counter.textContent = Math.floor(count);

            if (count < target) {
                requestAnimationFrame(updateCounter);
            }
        }

        updateCounter();
    });
};
```

Sl. 4.8. *JavaScript kod za animaciju brojača.*

4.4.3. Prikaz članaka

Na početnoj stranici aplikacije prikazani su članci koji sadrže savjete o održavanju računala u obliku kartica sa slikom, tekstom i kratkim opisom. Članci mogu biti listani na sljedeće stranice, a svaka stranica sadrži 5 članaka, od čega je najnoviji članak najveće veličine, drugi najnoviji malo manji, a ostala tri članka su iste veličine. Članci na početnoj stranici dizajnirani su pomoću Bootstrap 5 razvojnog okvira u responzivnom *grid* sustavu. Svaki članak smješten je unutar kartice koja sadrži sliku i tekst, a veličina članka definirana je veličinom stupca, pa tako najveći članak ima veličinu stupca „col-md8“, dok ostali imaju veličinu stupca „col-md4“. Slike su prikazane unutar *div* elementa i prilagođavaju se dimenzijama kartice pomoću stila „object-fit: cover“.

Paginacija je implementirana pomoću Bootstrapove paginacije, koja prikazuje brojeve stranica u donjem desnom kutu, zajedno s gumbima za prelazak na sljedeću ili prethodnu stranicu. Brojevi stranica dohvaćeni su kroz Django koristeći *Django paginator* klasu, koja automatski obrađuje paginaciju, a template prikazuje te brojeve kroz *page_obj* varijablu, kao što je prikazano na slici 4.9.

```
12  def home(request):
13      articles = Article.objects.all().order_by('-pub_date')
14      paginator = Paginator(articles, 5)
15
16      page_number = request.GET.get('page')
17      page_obj = paginator.get_page(page_number)
18
19      total_services = Service.objects.filter(
20          status__in=['service_done', 'completed']
21      ).count()
22
23
24      total_os_installs = OperatingSystemInstallation.objects.filter(
25          status__in=['service_done', 'completed']
26      ).count()
27
28
29      total_cleanings = CleaningOrder.objects.filter(
30          status__in=['service_done', 'completed']
31      ).count()
32
33
34      return render(request, 'home.html', {
35          'page_obj': page_obj,
36          'total_services': total_services,
37          'total_os_installs': total_os_installs,
38          'total_cleanings': total_cleanings,
39          })
```

Sl. 4.9. *Pogled koji se koristi prikaz članaka i početne stranice.*

Struktura članka kao objekta u bazi podataka detaljno je opisana u poglavlju o strukturi baze podataka.

Klikom na određeni članak na početnoj stranici otvara se stranica s detaljnim tekstom i opisom članka.

Jedinstveni URL za svaki članak definiran je u *urls.py* datoteci: „path('article/int:article_id/', views.article, name='article')“. Podaci se dohvaćaju kroz Django pogled prikazan na slici 4.10., koji vraća objekt članka čiji ID odgovara ID-u iz URL-a i koji se potom može koristiti unutar HTML datoteke. Ako takav članak ne postoji, pokreće se iznimka *Http404*.

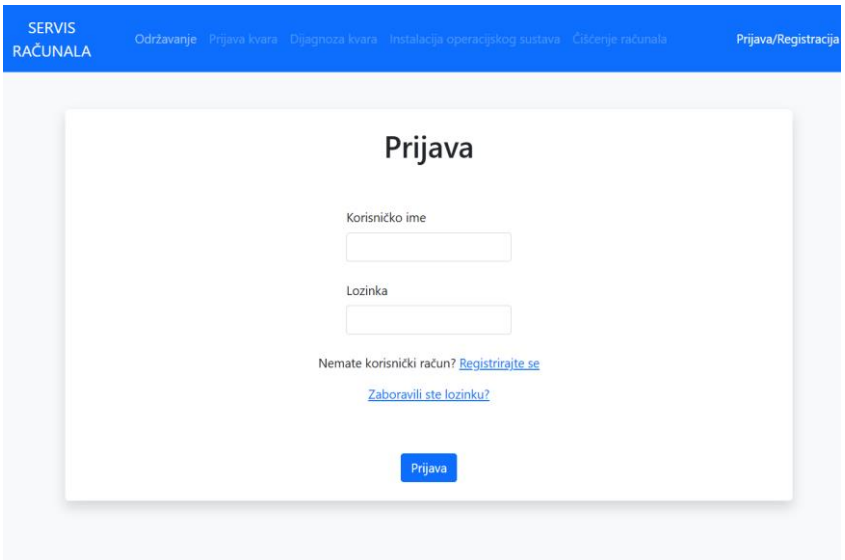
```
79  def article(request, article_id):
80      article = Article.objects.get(pk=article_id)
81      if article is not None:
82          return render(request, 'article.html', {'article':article})
83
84      else:
85          raise Http404('Ova vijest ne postoji!')
```

Sl. 4.10. *Pogled za prikaz određenog članka*

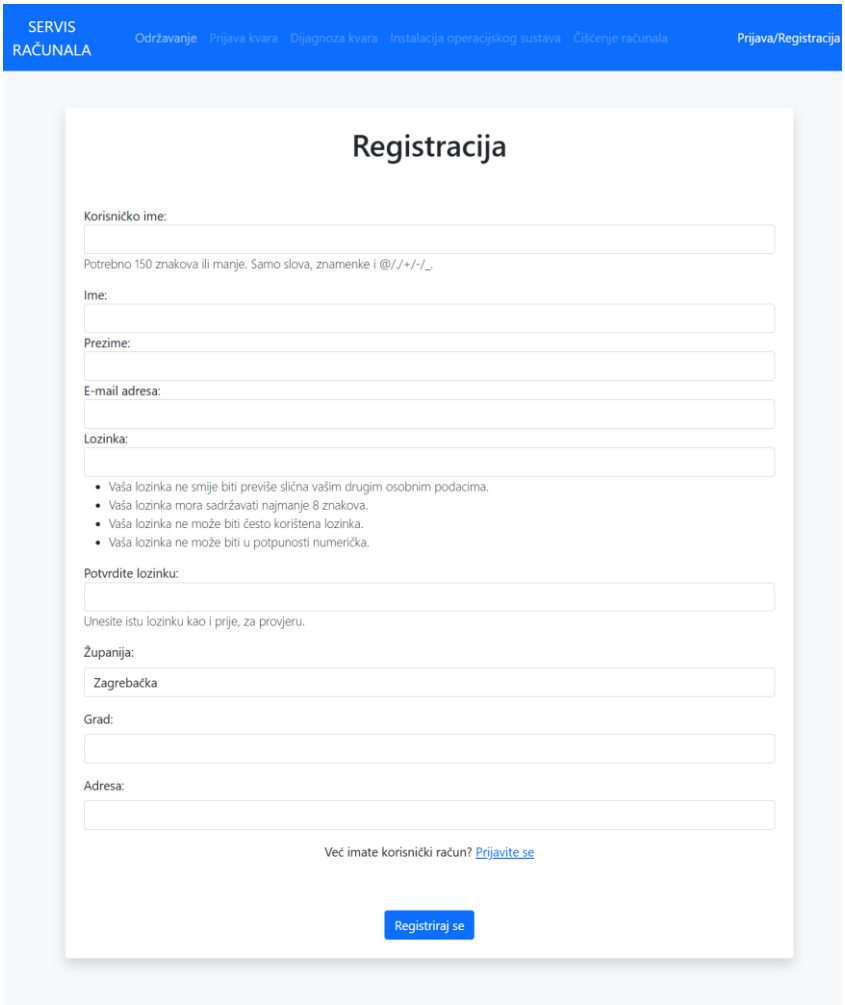
4.4.4. Prijava, registracija i odjava korisnika

Za prijavu korisnika potrebno je kliknuti na poveznicu prijava/registracija na navigacijskoj traci. Stranica za prijavu korisnika prikazana je na slici 4.11., a za prijavu su potrebni korisničko ime i lozinka. U slučaju da je lozinka zaboravljena, može se ponovno postaviti putem email adrese klikom na poveznicu: „Zaboravili ste lozinku?“. U slučaju da korisnik nema korisnički profil, može se registrirati klikom na poveznicu: „Registrirajte se“. Odjava korisnika vrši se putem padajućeg izbornika u kojem se nalazi korisničko ime, a odabirom poveznice „Odjava“.

Na slici 4.12. prikazana je stranica za registraciju korisnika.



Sl. 4.11. *Prijava korisnika.*



Sl. 4.12. *Registracija korisnika.*

Kako bi registracija korisnika bila uspješno obavljena, prilikom ispunjavanja forme za registraciju moraju biti ispunjeni uvjeti navedeni na obrascu za registraciju.

Prijava, registracija, odjava korisnika, resetiranje zaboravljene lozinke te upravljanje sesijama i pristupima omogućeni su s pomoću ugrađenih funkcija Django autentifikacije. Model korisnika (*User*) unaprijed je definiran u modulu *django.contrib.auth.models*. Za proširenje korisničkih podataka koristi se dodatna tablica *UserProfile*, umjesto prilagođavanja modela *AbstractUser*.

Autentifikacija je implementirana kroz aplikaciju *members*, dok je u datoteci *settings.py* uključena aplikacija *django.contrib.auth*. Registracija korisnika omogućena je s pomoću HTML predloška, funkcije u *views.py*, obrasca u *forms.py* temeljenog na *UserCreationForm*, te definiranog URL-a u *urls.py*. U obrascu za registraciju dodana su polja poput županije, grada i adrese, a korisnici županiju biraju iz unaprijed definiranog popisa. Nakon uspješne registracije, korisnički podaci spremaju se u modele *User* i *UserProfile*. Obrazac je stiliziran s pomoću Bootstrap klasa radi responzivnosti.

U datoteci *signals.py* koriste se signali *create_user_profile* i *save_user_profile*, koji automatski stvaraju i ažuriraju profil korisnika nakon kreiranja ili izmjene korisnika. Funkcija *register_user* u *views.py* obrađuje registraciju. Ako je zahtjev POST i obrazac ispravan, korisnički račun sprema se u bazu, autentificira i automatski prijavljuje. Nakon registracije korisnik se preusmjerava na stranicu za opis konfiguracije računala. Kod GET zahtjeva prikazuje se prazan obrazac za registraciju.

Prijava korisnika funkcionira na sličan način kao registracija. Za upravljanje sesijama koristi se ugrađena funkcija *login*. Kod GET zahtjeva prikazuje se obrazac za prijavu, a POST zahtjev koristi se za autentifikaciju. Prijava je uspješna ako su vjerodajnice ispravne: „user is not None“. Nakon uspješne prijave, korisnik se preusmjerava na početnu stranicu, a povratna informacija korisniku se daje putem *message.success* poruke koja se prikazuje na početnoj stranici. Ako prijava nije bila uspješna, korisnik se preusmjerava ponovno na stranicu za prijavu. Odjava korisnika odvija se s pomoću funkcije *logout_user*, koja automatski zatvara sesiju korisnika, preusmjerava ga na početnu stranicu i daje mu povratnu informaciju.

Kako bi se osiguralo da samo prijavljeni korisnik ima pristup određenim uslugama na stranici, koristi se *@login_required* dekorator, koji se stavlja iznad funkcije unutar pogleda. Ako korisnik nije prijavljen, a želi pristupiti nekoj usluzi putem URL-a, bit će usmjeren na stranicu za prijavu.

U slučaju zaboravljene lozinke, proces resetiranja lozinke je sljedeći:

1. Na putanji *reset_password/* unosi se e-mail adresa. E-mail s poveznicom za resetiranje lozinke šalje se putem *PasswordResetView*.
2. Nakon što je zahtjev poslan, prikazuje se stranica *reset_password_sent* koja potvrđuje da je e-mail poslan (putem *PasswordResetDoneView*).
3. Putanja *password-reset-confirm/<uidb64>/<token>/* predstavlja poveznicu iz e-maila koja se koristi za potvrdu identiteta. Za unos nove lozinke koristi se *PasswordResetConfirmView*.
4. Ako je proces resetiranja lozinke uspješan, prikazuje se stranica *reset_password_complete*, koja informira korisnika da je lozinka uspješno promijenjena.

Za slanje e-mailova putem interneta konfigurirane su SMTP (engl. *Simple Mail Transfer Protocol*) postavke unutar *settings.py* datoteke.

4.5. Registrirani korisnici

U ovom poglavlju opisane su funkcionalnosti aplikacije kada je registrirani korisnik trenutačno prijavljen u aplikaciji. Sve funkcije namijenjene registriranim korisnicima zaštićene su dekoratorom *@login_required*, što omogućava pristup samo prijavljenim korisnicima.

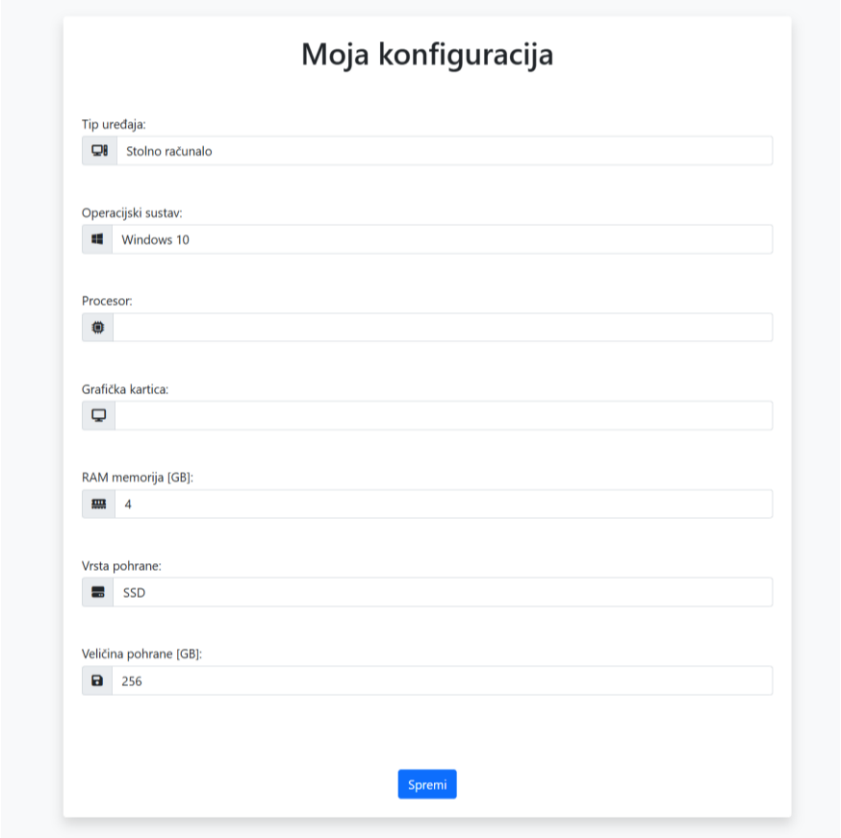
4.5.1. Konfiguracija računala i postavke profila

Prvi put kada se korisnik registrira, preusmjerava se na stranicu prikazanoj na slici 4.13., na kojoj se mogu ispuniti informacije o konfiguraciji računala. Također, prijavljeni korisnik može u bilo kojem trenutku promijeniti svoju konfiguraciju klikom na poveznicu "Moja konfiguracija" unutar padajućeg izbornika. Konfiguracija računala olakšava serviserima otkrivanje kvara prije nego što računalo dođe na servis.

Funkcija *add_configuration* omogućava dodavanje ili ažuriranje konfiguracije, a promjene se pohranjuju nakon validacije forme. Funkcija *view_configuration* prikazuje trenutačnu konfiguraciju korisnika ako ona postoji.

U aplikaciji je omogućeno mijenjanje korisničkih postavki profila definiranih u modelu *UserProfile*, kao i izmjena imena, prezimena, lozinke i e-mail adrese iz modela *User*. Ova funkcionalnost dostupna je unutar padajućeg izbornika.

Funkcija *profile_settings* omogućava promjenu postavki profila, obrađuje podatke iz obrasca i, ako su valjani, sprema ih u bazu podataka te korisniku daje povratnu informaciju. Ako je zahtjev GET, prikazuju se obrasci za uređivanje podataka.

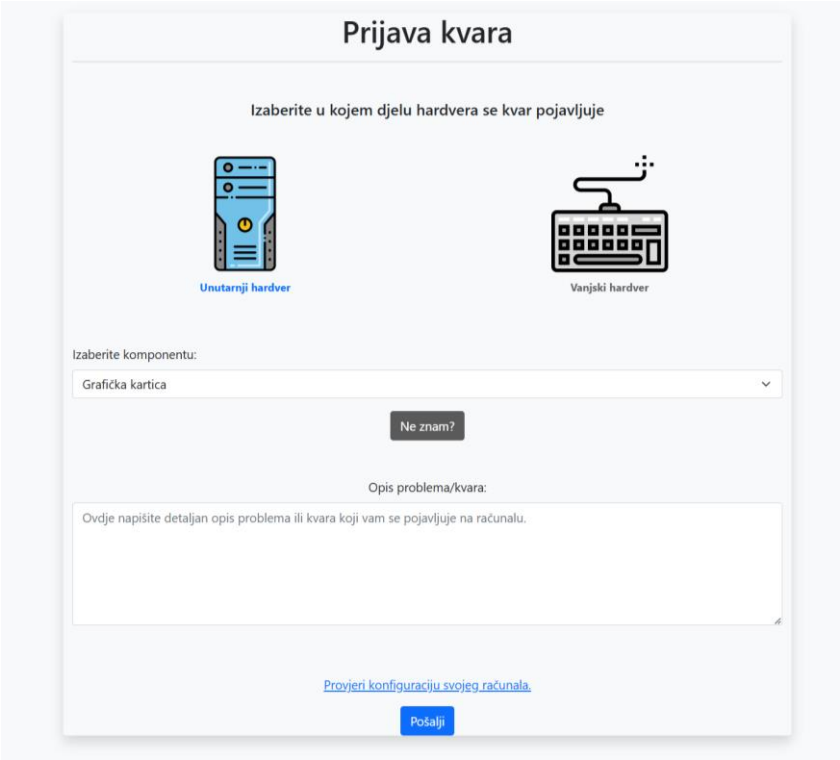


Sl. 4.13. *Moja konfiguracija.*

4.5.2. Prijava kvara

Korisniku je omogućeno prijavljivanje kvara putem poveznice "Prijava kvara" unutar navigacijske trake. Na stranici za prijavu kvara prikazanoj na slici 4.14., korisniku je omogućeno interaktivno odabiranje tipa hardvera i komponente za koju se smatra da je u kvaru, te unos detaljnog opisa problema. U slučaju nepoznavanja konkretne komponente koja uzrokuje problem, korisniku je omogućeno klikom na gumb "Ne znam?" da unese samo opis problema. Korisniku je također omogućeno provjeriti konfiguraciju svog računala putem poveznice "Provjeri konfiguraciju svog računala". Kada se ispune svi potrebni podaci za prijavu kvara, korisnik se preusmjerava na početnu stranicu i prima povratnu poruku.

Funkcija *pc_service* omogućuje prijavu kvara na računalo. Prvo se provjerava postoji li konfiguracija računala za prijavljenog korisnika. Ako je metoda zahtjeva POST, prikupljaju se podaci o tipu hardvera, komponenti i opisu problema. Ako su svi podaci uneseni ispravno, prijava kvara se sprema u bazu podataka, a korisniku se prikazuje poruka o uspjehu. Ako podaci nisu ispravno uneseni, prikazuje se poruka o pogrešci.



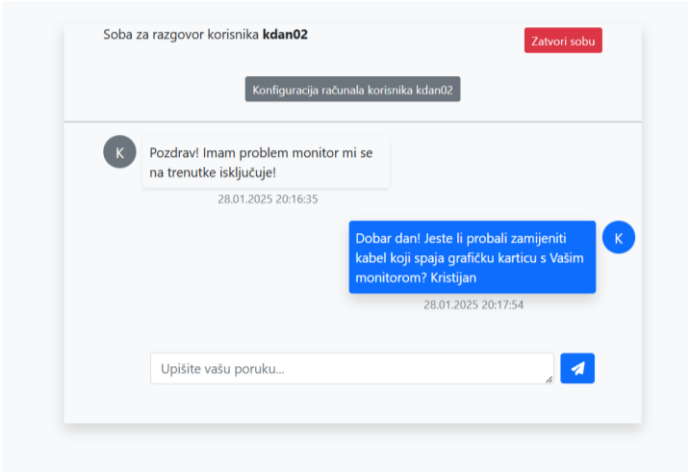
Sl. 4.14. *Prijava kvara.*

4.5.3. Dijagnoza kvara

Dijagnoza kvara može se pristupiti s pomoću poveznice Dijagnoza kvara koja se nalazi na navigacijskoj traci. Na toj poveznici postoji izbor hoće li se pokrenuti razgovor uživo (engl. *Live chat*) ili se želi vratiti na početnu stranicu. Ako se razgovor uživo pokrene, nova soba za razgovor bit će otvorena s bilo kojim trenutačno dostupnim serviserom. Unutar sobe za razgovor kao što je prikazano na slici 4.15. mogu se razmjenjivati poruke sa serviserom, pregledavati konfiguracija računala ili zatvoriti soba. Administratori, odnosno serviseri, također imaju pristup konfiguraciji računala korisnika s kojim razgovaraju. Nakon što je problem riješen, soba može biti zatvorena, a

ako problem nije riješen, korisnik se uvijek može vratiti na svoje postojeće otvorene sobe klikom na ikonu u obliku balončića s porukom na navigacijskoj traci.

Funkcionalnost slanja poruka temelji se na AJAX-u, što omogućuje dinamičko slanje i dohvaćanje poruka bez ponovnog učitavanja stranice. Sve poruke pohranjuju se u bazu podataka. Funkcija *start_chat* kreira novu chat sobu povezanu s trenutačno prijavljenim korisnikom, postavlja status na aktivan i korisnik se preusmjerava na tu sobu. Svaka soba ima svoj ID, te funkcija *chat_room* dohvaća sobu za razgovor prema ID-u i provjerava postoji li konfiguracija računala korisnika. Funkcija *send_message* omogućuje slanje poruka unutar sobe za razgovor koje se pohranjuju u bazu podataka i vraća se odgovor u JSON formatu. Funkcija *fetch_messages* dohvaća sve poruke iz sobe za razgovor i sortira ih prema vremenu nastanka. Funkcija *close_chat_room* zatvara sobu za razgovor i korisnik se preusmjerava na početnu stranicu.



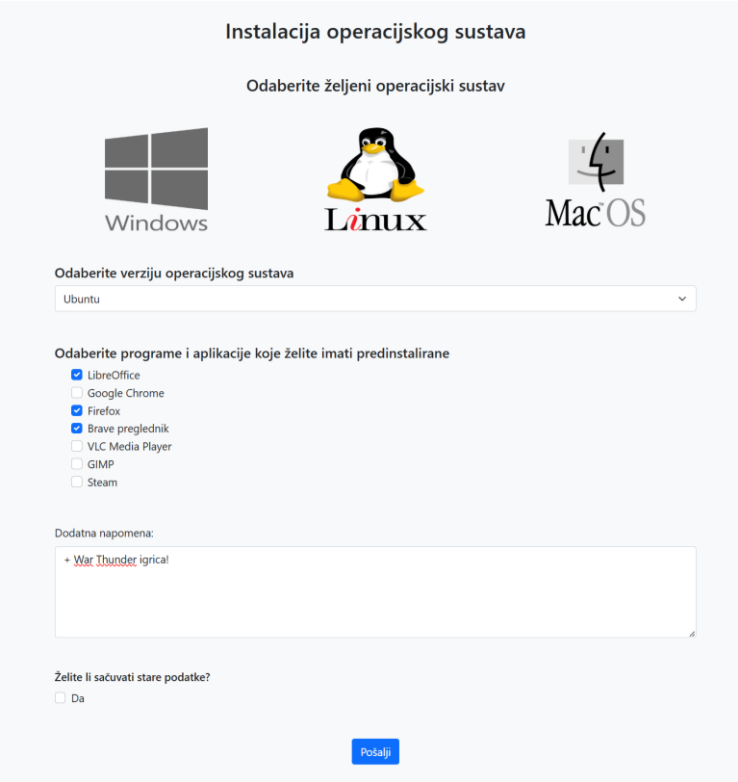
Sl. 4.15. *Primjer sobe za razgovor.*

4.5.4. Instalacija operacijskog sustava

Naručivanje za instalaciju operacijskog sustava može se obaviti klikom na poveznicu Instalacija operacijskog sustava unutar navigacijske trake. Odabir jednog od tri operacijska sustava: *Windows*, *Linux* ili *macOS*, omogućuje se korisniku na interaktivan način. Kada se odabere operacijski sustav, odabire se i verzija operacijskog sustava, primjerice ako je odabran *Linux*, može se odabrati verzija poput *Ubuntu*, *Debian*, *Mint* i tako dalje. Nakon odabira željene verzije operacijskog sustava, nudi se mogućnost odabira programa koji će biti predinstalirani, a koji odgovaraju odabranom operacijskom sustavu. Dodatna napomena vezana uz instalaciju

operacijskog sustava može biti ostavljena. Na kraju, korisniku se nudi mogućnost odlučivanja želi li sačuvati stare podatke i aplikacije ili želi čistu instalaciju bez starih podataka. Nakon što se obrazac ispuni i pošalje narudžba, korisnik se preusmjerava na početnu stranicu te prima povratnu poruku. Na slici 4.16. prikazana je usluga naručivanja za instalaciju operacijskog sustava.

Funkcija koja omogućuje naručivanje instalacije operacijskog sustava je *os_installation*. Ako je metoda zahtjeva POST, odabrani podaci iz obrasca će biti dohvaćeni, te će, ako su svi podaci ispravni, biti stvoren zapis za narudžbu koji će biti poslan u bazu podataka. Korisniku će biti prikazana uspješna poruka i bit će preusmjeren na početnu stranicu. Ako zahtjev nije POST, obrazac za narudžbu instalacije operacijskog sustava bit će prikazan korisniku.



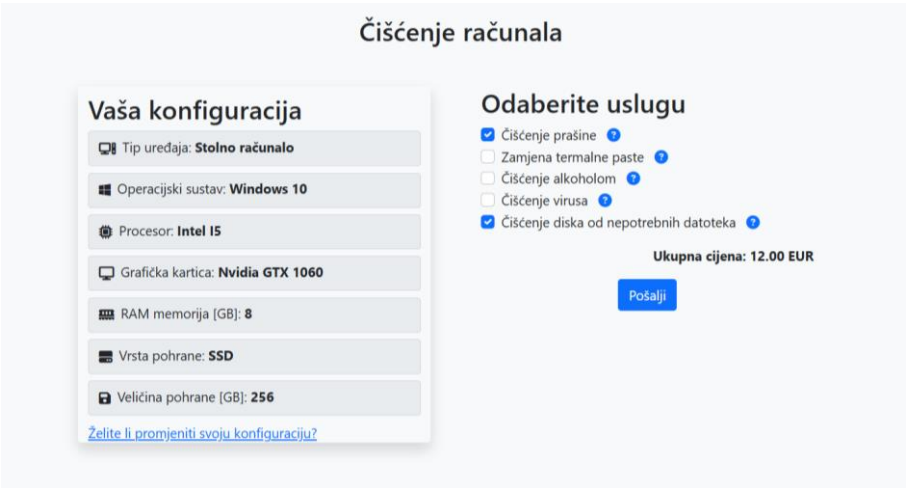
Sl. 4.16. *Naručivanje za instalaciju operacijskog sustava.*

4.5.5. Čišćenje računala

Korisnik se može naručiti za čišćenje računala putem poveznice Čišćenje računala koja se nalazi na navigacijskoj traci. Korisniku se nudi prikaz konfiguracije njegovog računala, poveznica

na promjenu konfiguracije u slučaju da nešto fali ili nije u redu, te odabir usluge čišćenja s dinamičkom promjenom ukupne cijene. Također, svaka usluga čišćenja ima poveznicu u obliku upitnika u krugu koja korisnika preusmjerava na članak o održavanju računala povezan s tom uslugom. Ovo je jedina usluga u aplikaciji koja korisniku odmah prikazuje ukupan iznos novca koji mora platiti za uslugu. Svim ostalim uslugama cijenu servisera određuje kasnije i ažurira ih u narudžbama, što će biti opisano kasnije u radu. Na slici 4.17. prikazan je primjer naručivanja za čišćenje računala.

Funkcija *pc_cleaning* omogućuje korisnicima stvaranje narudžbi za čišćenje računala. Kada korisnik pošalje narudžbu, funkcija dohvaća odabrane usluge iz obrasca i ukupnu cijenu, te kreira narudžbu za čišćenje računala koja se sprema u bazu podataka. Cijena se dinamički izračunava na temelju odabranih usluga te se ažurira u HTML-u i šalje kao vrijednost polja *total_price* u POST zahtjevu. JavaScript skripta dinamički ažurira cijenu te postavlja tu vrijednost u skriveno polje. Unutar JavaScript funkcije *calculateTotalPrice*, cijena se dohvaća iz atributa *data-price* svakog označenog checkboxa i pretvara u float s pomoću funkcije *parseFloat(checkbox.dataset.price)*. Ako je narudžba uspješna, korisnik se preusmjerava na početnu stranicu i prikazuje mu se poruka o uspjehu. Ako korisnik ne odabere uslugu, prikazuje mu se poruka o grešci.



Sl. 4.17. *Naručivanje za čišćenje računala.*

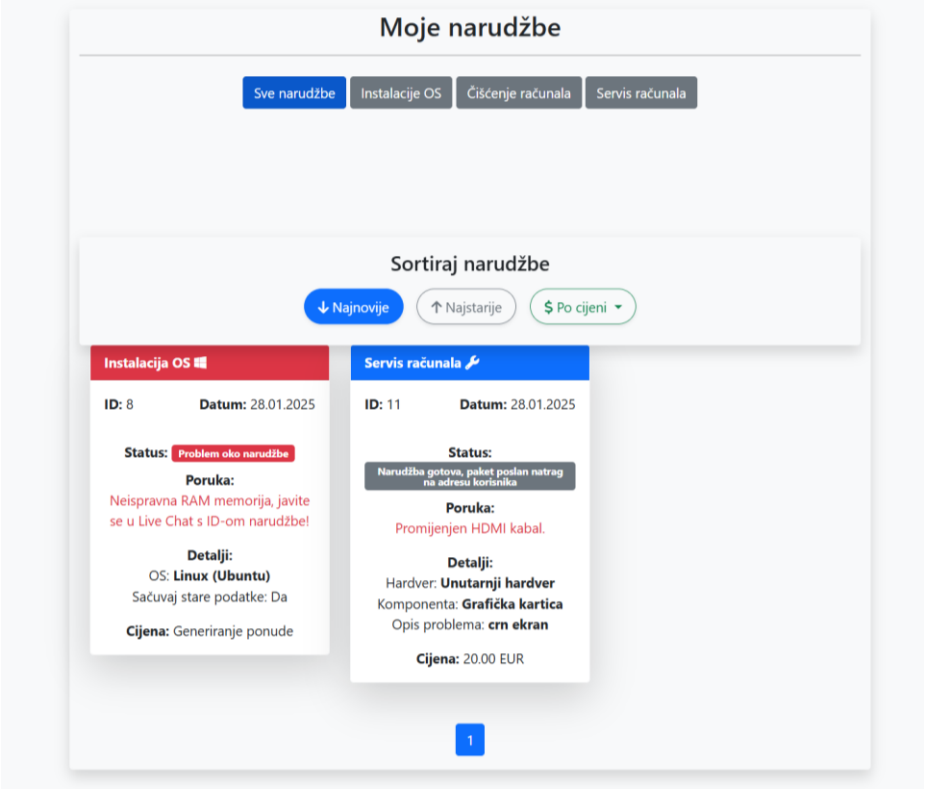
4.5.6. Pregled narudžbi

Narudžbe za određene usluge mogu se vidjeti klikom na ikonicu košarice za kupovinu. Klikom na tu ikonicu otvara se stranica Moje narudžbe na kojoj je prikazan kronološki pregled svih narudžbi napravljenih do sada, koje se mogu listati po stranicama. Narudžbe se mogu kategorizirati prema svim uslugama ili prema svakoj usluzi posebno. Također, narudžbe se mogu sortirati vremenski, od najnovijih do najstarijih, ili po cijeni, od najmanje do najveće. Klikom na ikonicu s upitnikom, otvara se modalni prozor s detaljnim uputama o tome kako poslati paket servisu i s objašnjenjem statusa narudžbe. Svaka narudžba prikazana je u obliku kartice koja sadrži naziv usluge u naslovu, ID narudžbe, datum kada je narudžba napravljena, status narudžbe, pojedinačne detalje određene usluge te cijenu. Na slici 4.17. prikazan je primjer svih narudžba korisnika.

Postoje 5 mogućih statusa, svaki status je simboliziran bojom:

- Žuta boja: pošaljite paket.
- Svijetloplava boja: Narudžba zaprimljena na servis.
- Tamnoplava boja: Paket je došao na servis.
- Zelena boja: Servis odrađen.
- Siva boja: Narudžba gotova, paket poslan natrag na adresu korisnika.
- Crvena boja: Problem oko narudžbe.

Funkcijom *my_orders* omogućuje se prikaz korisnikovih narudžbi uz mogućnost filtriranja prema vrsti usluge s pomoću parametra *filter* iz URL-a (GET zahtjev) i sortiranja po datumu ili cijeni s pomoću parametra *sort* iz URL-a. Narudžbe se grupiraju po 10 na svakoj stranici, a za navigaciju između stranica koristi se parametar *page*.



Sl. 4.17. *Pregled korisnikovih narudžbi.*

4.6. Administratori

Način rada, izrada funkcionalnosti te prikaz aplikacije za administratore opisani su u nastavku ovog poglavlja. Kako bi bilo omogućeno da administratorskim pravima, poput dodavanja i brisanja članaka, pregleda i ažuriranja svih narudžbi te prikaza aktivnih soba za razgovor, mogu pristupiti samo administratori, implementirana je funkcija *is_admin*.

Funkcija *is_admin* koristi se kao dekorator *@user_passes_test(is_admin)* iznad svih funkcija koje su povezane s administratorskim pravima. Vraća se True ako je korisnik prijavljen i ako mu je dodijeljena uloga unutar grupe *Admins*.

Također, kako bi prikaz administratorskih opcija u predlošcima, primjerice u navigacijskoj traci, bio omogućen, kreirana je funkcija *is_user_admin* unutar datoteke *context_processors.py*, a njezinom primjenom varijabla *is_user_admin* dodaje se u sve predloške.

4.6.1. Dodavanje i brisanje članaka

Dodavanje članka moguće je putem poveznice Dodaj članak, koja se nalazi na navigacijskoj traci unutar padajućeg izbornika Administratorska prava. Članak se može objaviti s naslovom, tekstom i slikom, koja je predočena URL-om. Na slici 4.18. prikazano je kako se članak dodaje.

Dodavanje članaka omogućeno je funkcijom *add_article*. Ako je zahtjev tipa POST, podaci iz obrasca *ArticleForm* obrađuju se, trenutačni administrator navodi se kao autor, članak se sprema u bazu podataka, a administrator biva preusmjeren na početnu stranicu, gdje će novonastali članak odmah biti vidljiv. Ako je zahtjev tipa GET, prazan obrazac se kreira i prikazuje na stranici za dodavanje članka.

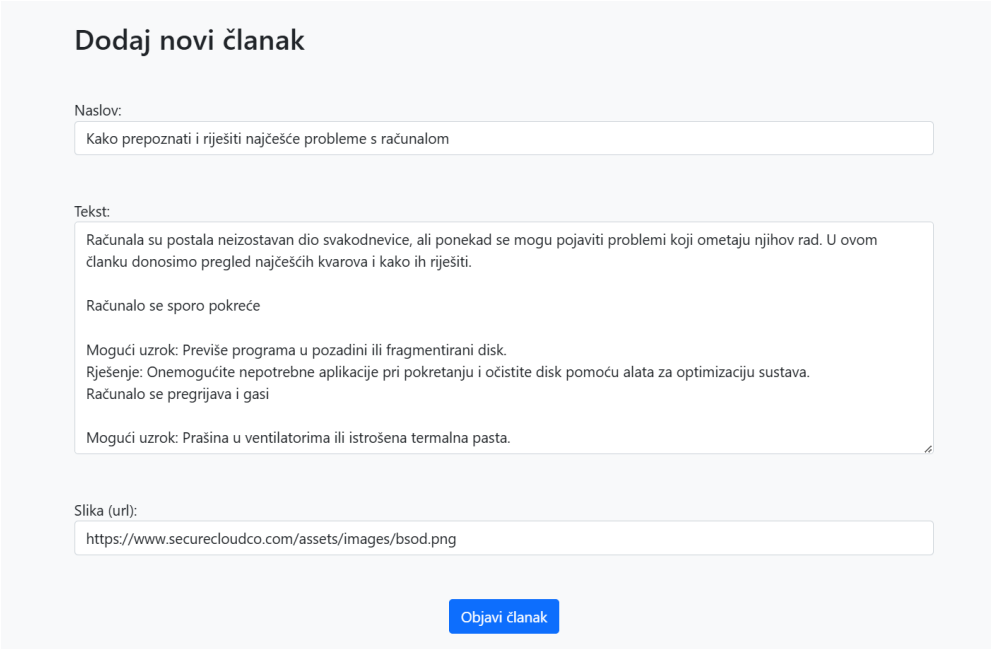
Brisanje članka moguće je putem poveznice Obriši članak, koja se nalazi na navigacijskoj traci unutar padajućeg izbornika Administratorska prava. Prikazuje se popis članaka u tabličnom obliku, po 10 članaka po stranici, poredanih kronološki od najnovijeg prema najstarijem. Svaki naslov članka postavljen je kao poveznica koja vodi na taj članak.

Ako se odluči obrisati članak, korisnik biva preusmjeren na dodatnu stranicu gdje može potvrditi ili odustati od brisanja. U slučaju potvrde, korisnik se preusmjerava na početnu stranicu, dok se u suprotnom vraća na popis članaka. Na slici 4.19. prikazan je primjer brisanja članka.

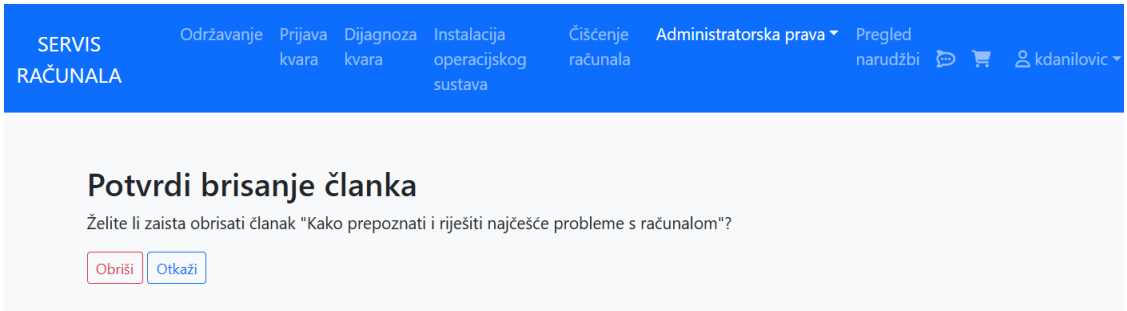
Tablica je realizirana s pomoću Bootstrapa. Za brisanje članaka koriste se dvije funkcije: *list_articles*, kojom se prikazuje popis svih članaka, te *delete_article*, kojom je omogućeno njihovo brisanje.

Funkcija *delete_article* u slučaju POST zahtjeva dohvaća članak prema ID-ju, briše ga i preusmjerava korisnika na početnu stranicu. Ako zahtjev nije POST, prikazuje se stranica za potvrdu brisanja.

Funkcija *list_articles* koristi se za prikaz popisa svih članaka, sortirajući ih prema datumu objave, a uz pomoć *Paginator-a* osigurava se prikaz do 10 članaka po stranici.



Sl. 4.18. *Primjer dodavanja članka.*



Sl. 4.19. *Primjer brisanja članka.*

4.6.2. Prikaz aktivnih soba za razgovor

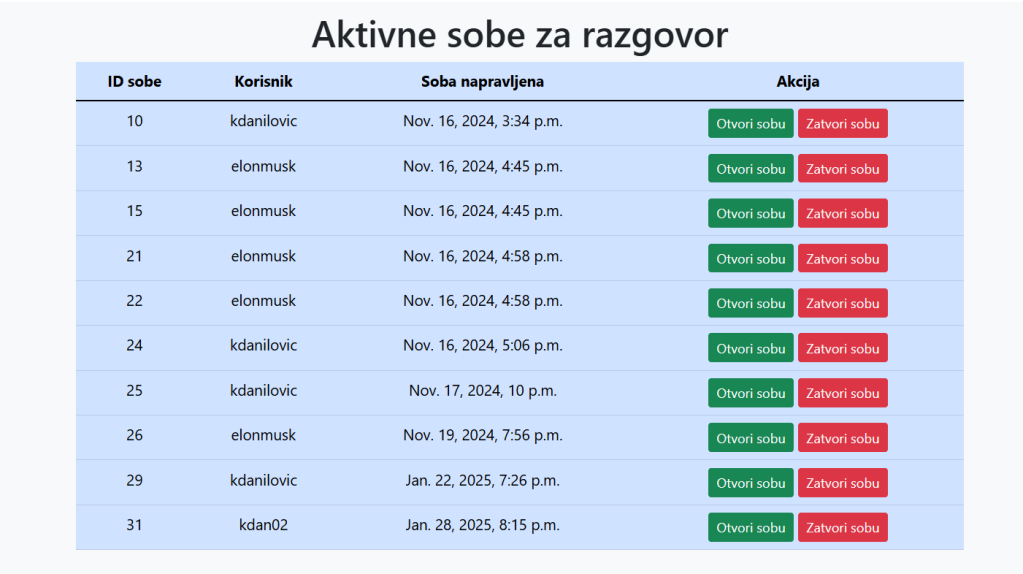
Administratori mogu vidjeti sve aktivne sobe za razgovor, sudjelovati u razgovoru uživo otvaranjem sobe ili zatvoriti sobu.

Pristup svim aktivnim sobama za razgovor dostupan je putem poveznice *Live chat sobe*, koja se nalazi unutar padajućeg izbornika Administratorska prava na navigacijskoj traci.

Popis aktivnih soba prikazan je u tabličnom obliku, a tablica se sastoji od stupaca: ID sobe, korisnika koji je kreirao sobu, datuma stvaranja sobe te akcija, koje omogućuju otvaranje ili zatvaranje sobe. Redoslijed prikaza soba u tablici postavljen je od najstarije do najnovije, kako bi

se osiguralo da serviseri ne prelaze na rješavanje novog problema prije nego što je prethodni problem riješen. Popis svih aktivnih soba prikazan je na slici 4.20.

Funkcija *active_chat_rooms* zadužena je za prikaz svih aktivnih soba za razgovor. Sobe koje imaju aktivan status dohvaćaju se iz baze podataka i pohranjuju u varijablu *rooms* koja se šalje u predložak.



Aktivne sobe za razgovor

ID sobe	Korisnik	Soba napravljena	Akcija
10	kdanilovic	Nov. 16, 2024, 3:34 p.m.	Otvori sobu Zatvori sobu
13	elonmusk	Nov. 16, 2024, 4:45 p.m.	Otvori sobu Zatvori sobu
15	elonmusk	Nov. 16, 2024, 4:45 p.m.	Otvori sobu Zatvori sobu
21	elonmusk	Nov. 16, 2024, 4:58 p.m.	Otvori sobu Zatvori sobu
22	elonmusk	Nov. 16, 2024, 4:58 p.m.	Otvori sobu Zatvori sobu
24	kdanilovic	Nov. 16, 2024, 5:06 p.m.	Otvori sobu Zatvori sobu
25	kdanilovic	Nov. 17, 2024, 10 p.m.	Otvori sobu Zatvori sobu
26	elonmusk	Nov. 19, 2024, 7:56 p.m.	Otvori sobu Zatvori sobu
29	kdanilovic	Jan. 22, 2025, 7:26 p.m.	Otvori sobu Zatvori sobu
31	kdan02	Jan. 28, 2025, 8:15 p.m.	Otvori sobu Zatvori sobu

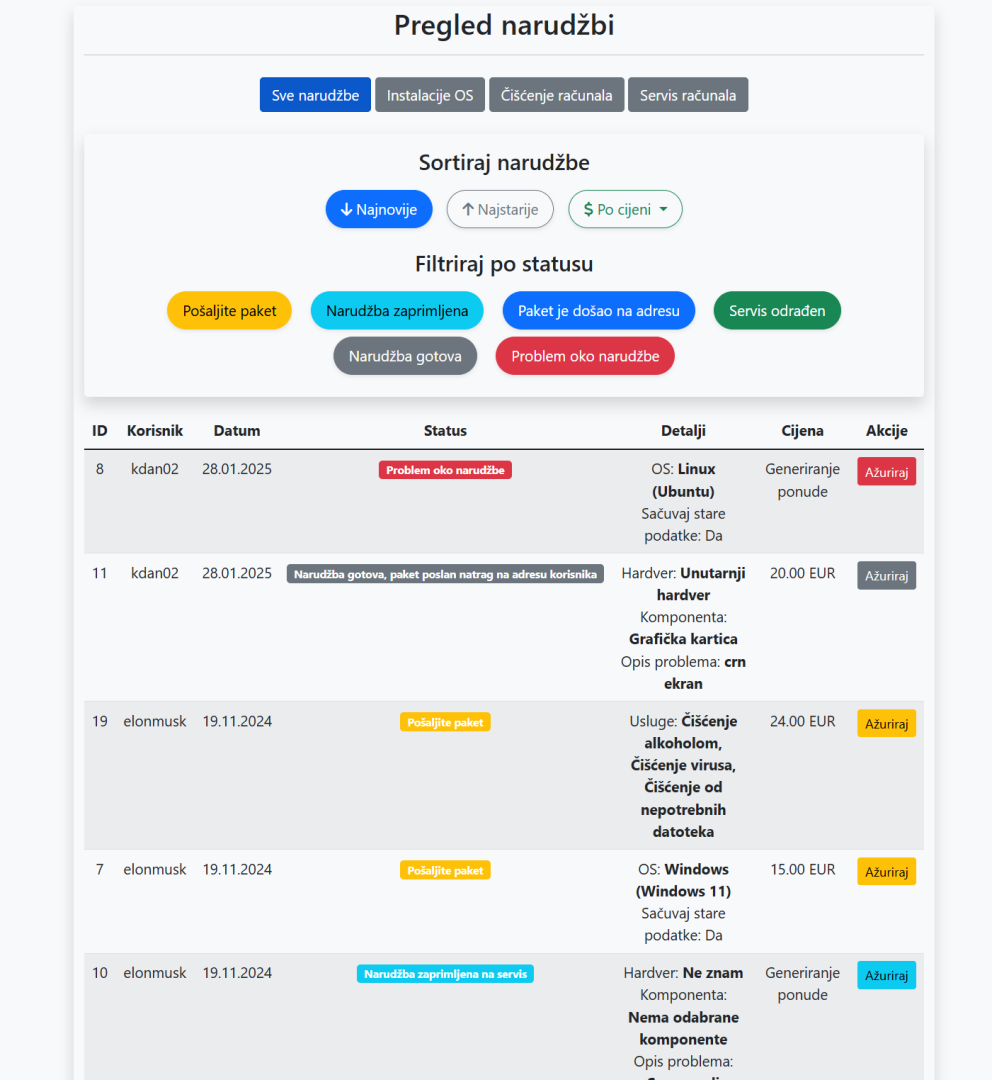
Sl. 4.20. *Prikaz svih aktivnih soba za razgovor.*

4.6.3. Pregled i ažuriranje svih narudžbi

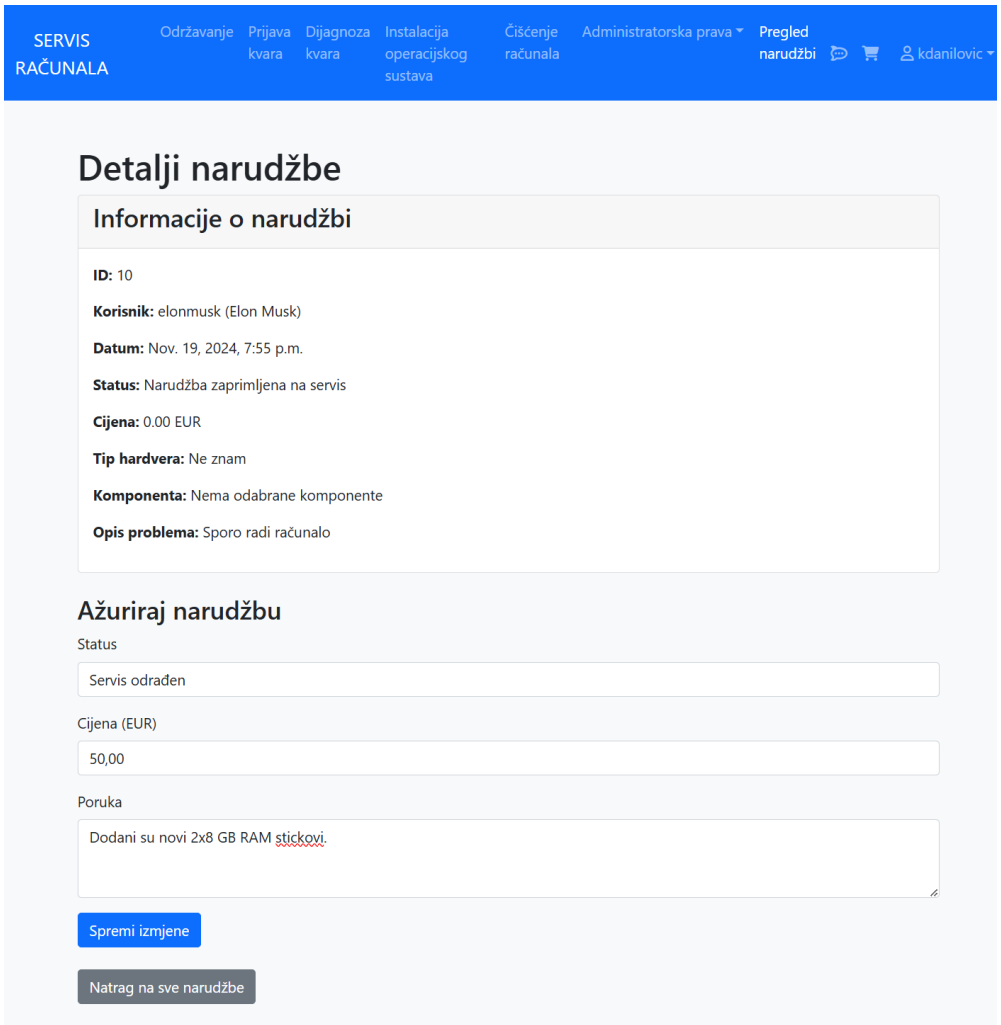
Poveznica Pregled narudžbi, koja se nalazi na navigacijskoj traci, omogućuje administratoru prikaz svih narudžbi. Na toj stranici prikazan je kronološki pregled svih dosad napravljenih narudžbi, koje se mogu listati po stranicama. Narudžbe se mogu kategorizirati prema svim uslugama ili prema svakoj usluzi zasebno. Također, omogućeno je vremensko sortiranje narudžbi, od najnovijih do najstarijih, ili prema cijeni, od najmanje do najveće.

Za razliku od korisničkog pregleda narudžbi, ovdje su prikazane narudžbe svih korisnika, pri čemu se svaka narudžba prikazuje kao red unutar tablice, umjesto u obliku kartice. Omogućeno je filtriranje prema statusu narudžbe, kao i ažuriranje detalja, poput cijene i statusa svake narudžbe.

Tablica se sastoji od stupaca: ID narudžbe, korisnika koji je napravio narudžbu, datuma kreiranja, statusa narudžbe, detalja, cijene i akcije, odnosno gumba za ažuriranje pojedine narudžbe. Na slici 4.21. prikazana je stranica za pregled narudžbi, dok je na slici 4.22. prikazan primjer ažuriranja narudžbi.



Sl. 4.21. *Administratorski pregled svih narudžbi.*



Sl. 4.22. *Primjer ažuriranja narudžbe.*

Funkcija *all_orders* omogućuje administratoru filtriranje i sortiranje narudžbi prema vrsti usluge, vremenu nastanka ili cijeni. Sve narudžbe iz modela usluga se dohvaćaju, a zatim se primjenjuju filteri za vrstu usluge, sortiranje po datumu ili cijeni te filtriranje prema statusu narudžbe. Nakon filtriranja i sortiranja, narudžbe se paginiraju po 10 narudžbi na 1 stranicu, a predložak sa svim narudžbama se renderira.

Funkcija *order_details* omogućuje administratoru detaljniji uvid na narudžbu i ažuriranje njezinih podataka. Odgovarajuća narudžba se dohvaća korištenjem *get_object_or_404*. Ako je zahtjev tipa POST, podaci narudžbe se ažuriraju. Ako je cijena neispravno unesena, prikazuje se poruka o grešci. Nakon uspješnog ažuriranja, narudžba se sprema u bazu podataka.

5. ZAKLJUČAK

Web aplikacija servisa za računala predstavlja aplikaciju koja registriranim korisnicima omogućuje jednostavno i interaktivno naručivanje za servis ili čišćenje računala te instalaciju operacijskog sustava. Registrirani korisnici mogu jednostavno pratiti svoje narudžbe i njihovo stanje. Aplikacija također pomaže serviserima u organizaciji rada. Prikazana su četiri slična rješenja, odnosno Android i web aplikacije s tematikom sličnom ovoj. Za izradu stranice HTML je korišten kao „kostur“ web aplikacije. Za dinamičko upravljanje interakcijama na stranici, poput komuniciranja u realnom vremenu, kao i za interaktivne funkcionalnosti poput osnovne analitike servisa, odabira komponenti i slično, korišten je JavaScript. Za upravljanje backendom aplikacije, poput obrade zahtjeva, autentifikacije korisnika i upravljanja bazom podataka, korišten je Python Django razvojni okvir. Za brzo pokretanje servera, instaliranje potrebnih paketa i postavljanje razvojnih okruženja korišten je Bash. Kako bi stranica imala dosljedan i minimalistički dizajn te mobilnu responzivnost, korišten je Bootstrap 5 razvojni okvir, kao i CSS. Funkcionalnosti aplikacije i korištene tehnologije detaljno su opisane. Aplikacija se može poboljšati implementacijom mogućnosti za dodavanje slika u sobama za razgovor, povezivanjem s vanjskim sustavima za naplatu, dodavanjem mogućnosti za ostavljanje ocjena odrađenim uslugama servisa, poboljšanjem dizajna cjelokupne aplikacije, boljim prikazom članaka te proširenjem ponude s još više usluga servisa. Izradom aplikacije za servisiranje računala može se zaključiti da se veliki broj procesa, poput obrade narudžbi, praćenja statusa servisa i komunikacije s korisnicima, može automatizirati, što rezultira bržim i učinkovitijim radom servisera.

LITERATURA

[1] https://servis.com.hr/ 25.05.2024.

[2] https://play.google.com/store/apps/datasafety?id=nz.co.appli.computer.repair 25.05.2024.

[3] https://play.google.com/store/apps/details?id=alsamman.hwexpert_en 25.05.2024.

[4] https://www.mm-informatika.com/ 26.05.2024.

[5] https://tesla.carnet.hr/mod/book/view.php?id=5430&chapterid=885 27.05.2024.

[6] https://www.w3schools.com/whatis/whatis_html.asp 27.05.2024.

[7] https://www.w3schools.com/css/css_intro.asp 27.05.2024.

[8] https://www.webtech.com.hr/css.php 27.05.2024.

[9] https://getbootstrap.com/ 28.05.2024.

[10] https://www.w3schools.com/bootstrap/bootstrap_get_started.asp 28.05.2024.

[11] https://www.djangoproject.com/start/overview/ 28.05.2024.

[12] https://www.w3schools.com/django/django_intro.php 28.05.2024.

[13] https://www.w3schools.com/js/js_intro.asp 29.05.2024.

[14] https://www.w3schools.com/js/js_ajax_intro.asp 29.05.2024.

[15] https://code.visualstudio.com/ 29.05.2024.

SAŽETAK

Web aplikacija pod nazivom "Servis računala" izrađena je uz pomoć tehnologija poput HTML-a, CSS-a, Bootstrap 5, Python Django, JavaScripta i SQLitea. Registrirani korisnici mogu se naručivati servis računala, čišćenje ili instalaciju operacijskog sustava te sudjelovati u sobama za razgovor i čitati članke o održavanju računala. Također, imaju svoje osobne stranice na kojima mogu ostavljati informacije o svom računalu. Gosti stranice mogu samo pregledavati sadržaj i čitati članke. Administratori postavljaju savjete o održavanju računala, upravljaju narudžbama i sobama za razgovor.

Ključne riječi: baza podataka, narudžbe, održavanje računala, servis računala, web aplikacija

ABSTRACT

Title: Web application for computer service

The web application called "Computer Service" was developed using technologies such as HTML, CSS, Bootstrap 5, Python Django, JavaScript, and SQLite. Computer service, cleaning, or operating system installation, can be ordered by registered users, who also participate in chat rooms, and read articles on computer maintenance. They also have personal pages where the information about their computers can be left. Guests can only browse the content and read articles. Computer maintenance tips are provided by administrators, who manage orders and oversee chat rooms.

Keywords: computer maintenance, computer service, database, orders, web application

PRILOZI

Projektna mapa s izvornim kodom nalazi se na GitLab repozitoriju na poveznici:

https://gitlab.com/kdanilovic/pc-service