IZRADA WEB APLIKACIJE ZA PRAĆENJE RADA AMBULANTE

Varnica, Ines

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijska tehnologija

INES VARNICA

Z A V R Š N I R A D

IZRADA WEB APLIKACIJE ZA PRAĆENJE RADA AMBULANTE

Split, rujan 2022.

SVEUČILIŠTE U SPLITU
SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijska tehnologija

Predmet: Informacijski sustavi

Z A V R Š N I R A D

Kandidat: Ines Varnica

Naslov rada: Izrada aplikacije za praćenje rada ambulante

Mentor: dr. sc. Igor Nazor, profesor v.š.

Split, rujan 2022.

Sadržaj

Sažetak

Cilj ovog završnog rada je ukazati na važnost razvijanja aplikacija koje se koriste u medicinske svrhe koje imaju za zadatak omogućiti što bržu i jednostavniju obradu pacijenta. Aplikacija se sastoji od tri djela: administrativnoga, liječničkog te sestrinskog.

Poslužiteljski dio aplikacije (engl. *backend*) izrađen je korištenjem razvojnog okvira Django zasnovanog na programskom jeziku Python. Za pohranu podataka je korištena baza podataka SQLite koja je sastavni dio Python Django Framework. Za pregled baze podataka je korišten DB browser, a za dizajn i izgled aplikacije (engl. *frontend*) je korišten Bootstrap, CSS i HTML. Kao uređivač kôda korišten je razvojni alat Visual Studio Code.
Ključne riječi: *Python, Django, SQLite, Bootstrap, CSS*

Summary

Developing of the web application for tracking a clinical workflow in hospitals

The main goal of this thesis is to point out the importance of developing medical applications that enable the fastest and easiest way to process patients. The application consists of three parts meant to be used by the administrative staff, doctors, and nurses.

The backend of this web application was built with Python Django Framework, which enabled the implementation of its functionality. SQLite database, an integral part of Python Django REST Framework, was used to store data, while DB browser was used to show the created database. The frontend of the app, its design and visual appearance, was developed using Bootstrap, CSS, and HTML. Visual Studio Code was used as a code editor.
Keywords: *Python, Django, SQLite, Bootstrap, CSS*

1. Uvod

Napretkom tehnologije dolazi do ubrzavanja procesa rada u svim gospodarskim granama. Različite vrste aplikacija su dovedene gotovo do savršenstva, a razni poslovni procesi su ubrzani do te mjere da više ne postoji „prazni hod“. Zdravstveni sektor u Hrvatskoj kaska sa modernizacijom.

Glavni motiv za izradu ove aplikacije je omogućiti medicinskim sestrama, tehničarima i liječnicima što brži i jednostavniji rad kako bi imali više vremena za posvetiti se pacijentima te kako bi se smanjilo vrijeme čekanja pacijenata u čekaonicama i na šalterima. Cilj je napraviti aplikaciju koja će medicinskom osoblju omogućiti obradu pacijenta u samo nekoliko koraka.

U prvom djelu završnog rada opisane su korištene tehnologije. Poslužiteljski dio aplikacije je razvijen korištenjem razvojnog okvira Django. Za dizajn aplikacije korišteni su: HTML (engl. *Hypertext Markup Language),* CSS (engl. *Cascading Style Sheets*) te popularni razvojni okvir Bootstrap koji nam omogućava vrlo jednostavnu i brzu izradu modernog izgleda.

2. Pregled područja

Bolnički informacijski sustavi veliki su i kompleksni sustavi koji se sastoje od velikog broja pod-aplikacija. U ovom završnom radu opisuje se jedna od pod-aplikacija, a to je pod-aplikacija za praćenje rada ambulante, u ovom slučaju ambulante koja je dio neke bolnice.

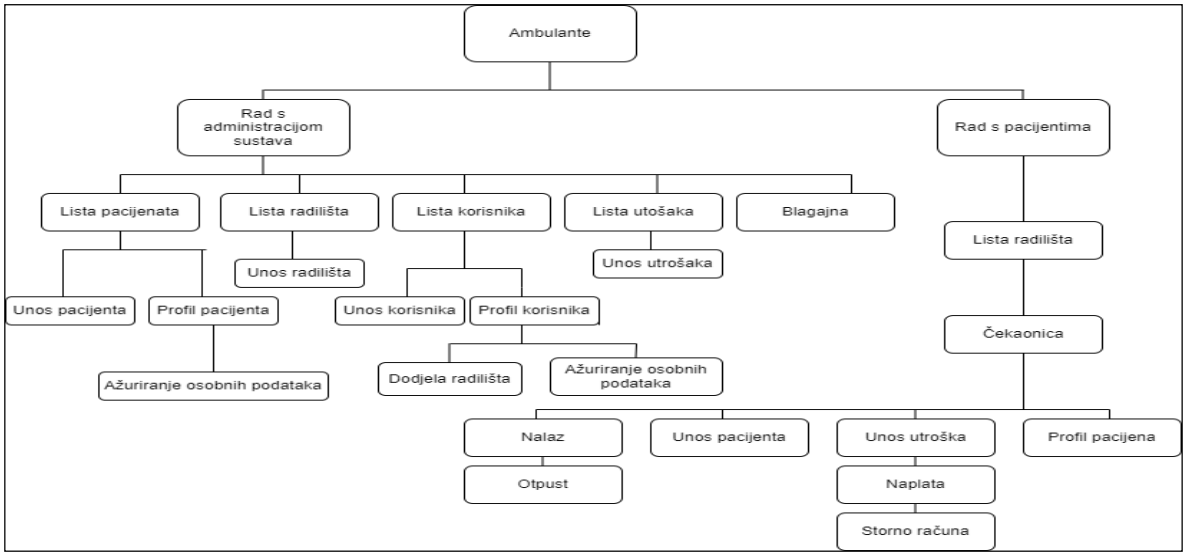
Trenutne aplikacije za praćenje rada ambulante ne omogućavaju korisnicima brzu i jednostavnu obradu pacijenata. Dizajn aplikacija je poprilično zastario i nije intuitivan, što otežava korisnicima da lako dođu do potrebnih informacija. Aplikacije zahtijevaju veliki broj koraka koje je potrebno napraviti kako bi pacijent bio ubačen u čekaonicu ili bio pregledan. Ovakve aplikacije najčešće ne sadrže navigaciju koja će korisnicima omogućiti lako kretanje po aplikaciji te najčešće sadrže veliki broj informacija na jednoj stranici, a to samo povećava konfuznost korisnika.

Cilj ovog završnog rada je pokazati kako napraviti aplikaciju koja će korisnicima biti intuitivna i jednostavna za korištenje. Stranice aplikacije su pomno osmišljene te se na njima mogu pronaći samo najvažnije informacije tako da korisnik ne bi bio opterećen prevelikom količinom informacija. Aplikacija naravno sadrži i navigaciju koja omogućava korisnicima da u svakom trenutku mogu doći do potrebnih informacija. Aplikacija slijedno vodi korisnika kroz upis pacijenata u čekaonicu, a uz nekoliko koraka korisnik može doći do svih podataka o pacijentu. Prilikom izrade je korišten responzivni dizajn koji omogućava da se aplikacija prikazuje i na mobilnim uređajima, što s do sada korištenim aplikacijama za praćenje rada ambulante nije bilo moguće. Čitljiv prikaz na mobilnim uređajima omogućava liječnicima pregledavanje nalaza svojih pacijenata, u bilo kojem trenutku, bez korištenja stolnog računala. Sigurnost, jedan od važnih segmenata je postignuta korištenjem modernih tehnologija poput različitih razvojnih okvira, kao što je u ovom slučaju Django, koji ima ugrađene određene module sigurnosti. Dakle, nije potrebno koristiti dodatnu zaštitu protiv upada treće strane.

3. Specifikacija aplikacije za praćenje rada ambulante

U prvom koraku su specificirane funkcionalnosti aplikacije, koje su raspoređene u tri skupine: administracijski, sestrinski i doktorski.

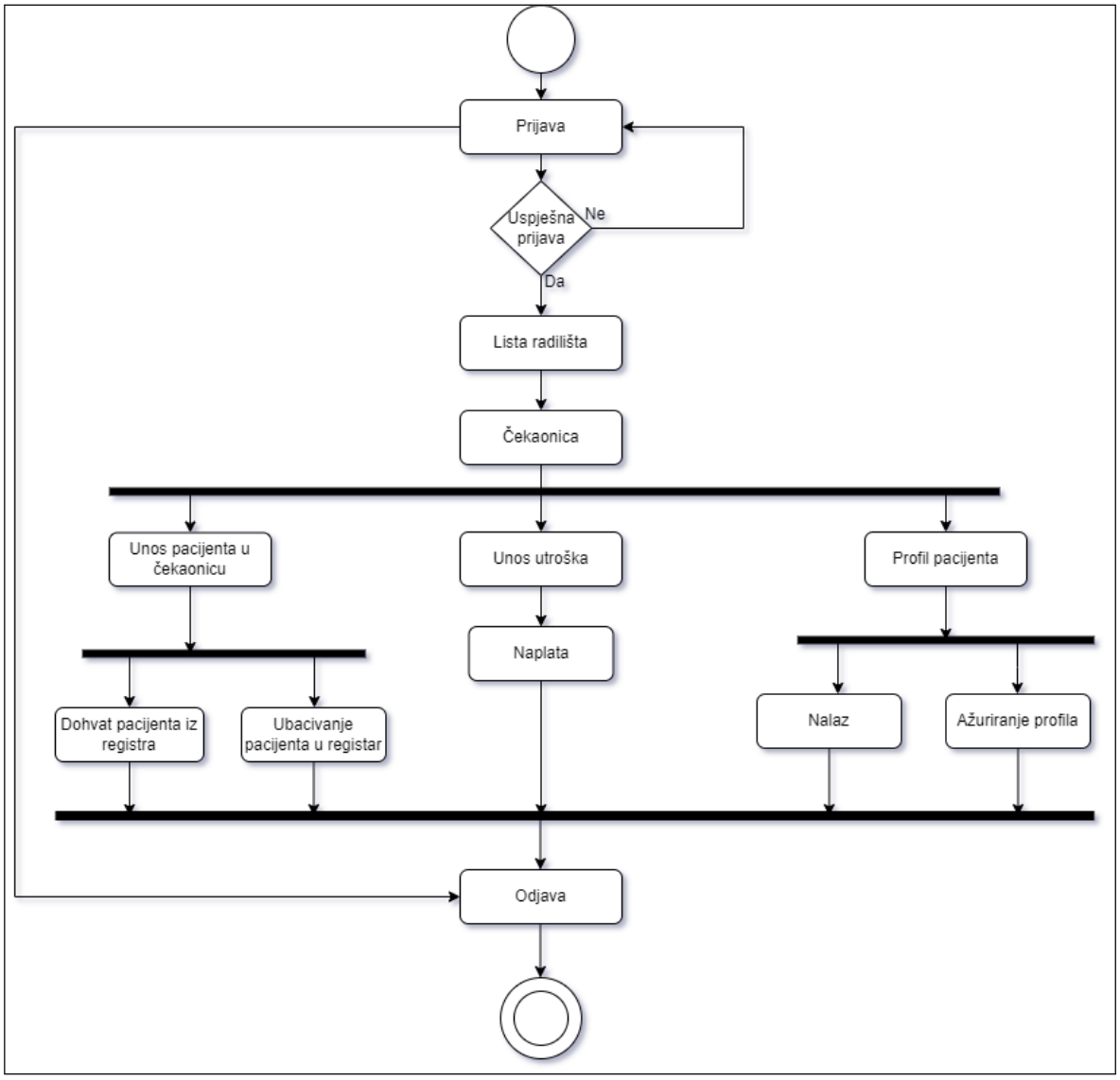
Administratori mogu vidjeti listu korisnika, pacijenata koji su upisani u registar, utrošaka i radilišta. Na stranici *blagajna* se prikazuju sve naplate koje su napravljane od strane medicinskih sestra i tehničara. Opcija za unos novih podataka pojavljuje se prilikom ulaska u neku od lista. Također, postoji mogućnost otvaranja profila pacijenta ili korisnika prilikom ulaska u liste pacijenta ili korisnika. Dekompozicija administrativnog djela aplikacije je prikaza na slici 1.



Slika 1: Dekompozicija funkcija aplikacije

Nakon uspješne prijave u sustav medicinske sestre i tehničari na početnim stranicama mogu vidjeti radilišta koja im je dodijelio administrator. Prilikom klika na neko radilište otvara se čekaonica radilišta. U čekaonici su vidljivi pacijenti koji su uneseni u čekaonicu taj dan. Iz čekaonice je moguće unijeti nove pacijente ili postojećim pacijentima unijeti utrošak. Utrošci predstavljaju materijale i usluge koji su korišteni za liječenje pacijenta. Svim pacijentima koji nemaju zdravstveno osiguranje je potrebno naplatiti pregled. Naplata se izvršava nakon unošenja utroška. U slučaju da je odrađena naplata, a medicinska sestra ili tehničar nije unio sve utroške, potrebno je prvo stornirati račun pa zatim obaviti unos preostalih utrošaka. Medicinske sestre i tehničari odabirom nekog pacijenta mogu vidjeti profil odabranog pacijenta. Na profilu pacijenta se nalaze osnovni podaci o pacijentima te

arhiva nalaza pacijenta. UML dijagram aktivnosti koji opisuje rad medicinskih sestra i tehničara u aplikacije je prikazan na slici 2.



Slika 2: UML dijagram aktivnosti sestrinskog dijela aplikacije

Liječnici kao i medicinske sestre i tehničari nakon prijave imaju prikaz radilišta na kojima su zaposleni. Prilikom odabira nekog od radilišta otvara se čekaonica koja je podijeljena u dva dijela. U jednom djelu čekaonice se nalaze pacijenti koji nemaju nalaz, a u drugom oni koji imaju. Na taj način liječnici mogu pratiti koje su pacijente pregledali, a koje ne. Liječnici imaju mogućnost pisanja nalaza pacijentima, pregled arhive pacijenta, pregled osnovnih informacija o pacijentu i otpust pacijenta.

Svi korisnici da bi započeli s radom se moraju prvo prijaviti te nakon završetka rada odjaviti iz aplikacije.

4. Korištene tehnologije

4.1. Django

Za izradu serverskog dijela aplikacije koristi se razvojni okvir Django

Django je razvojni okvir otvorenog kôda koji je besplatan, a napisan je u Pythonu koji radi na mnogim poslužiteljskim platformama, što znači da Django nije vezan ni za jednu određenu poslužiteljsku platformu. Django potiče stvaranje kôda koji se može održavati i ponovno koristiti, konkretno koristi „*Nemoj te se ponavljati*“ princip pisanja tako da nema nepotrebnog dupliciranja kôda. Također Django promiče grupiranje povezanih funkcionalnosti u višekratno upotrebljive aplikacije na nižoj razini, grupira povezani kôd u modele po uzoru na MVT (engl. *Model-View-Template)* arhitekturu.

Za kreiranje Django projekta prvo je potrebno kreirati virtualno okruženje pomoću Python paketa `pip` te nakon toga unutar virtualnog okruženja instalirati Django pomoću naredbe `pip install Django`. Naredbe za instalaciju virtualnog okruženja i instalaciju Djanga unutar virtualnog okruženja su prikazane na slici 3.

```
PS C:\Zavrsni\Zavrsni rad1> python -m venv venv
PS C:\Zavrsni\Zavrsni rad1> cd venv
PS C:\Zavrsni\Zavrsni rad1\venv> ./Scripts/activate
(venv) PS C:\Zavrsni\Zavrsni rad1\venv> cd ..
(venv) PS C:\Zavrsni\Zavrsni rad1> pip install django
Collecting django
  Using cached Django-4.1-py3-none-any.whl (8.1 MB)
Collecting sqlparse>=0.2.2
  Using cached sqlparse-0.4.2-py3-none-any.whl (42 kB)
Collecting asgiref<4,>=3.5.2
  Using cached asgiref-3.5.2-py3-none-any.whl (22 kB)
Collecting tzdata
  Using cached tzdata-2022.1-py2.py3-none-any.whl (339 kB)
Installing collected packages: tzdata, sqlparse, asgiref, django
Successfully installed asgiref-3.5.2 django-4.1 sqlparse-0.4.2 tzdata-2022.1

[notice] A new release of pip available: 22.2.1 -> 22.2.2
[notice] To update, run: python.exe -m pip install --upgrade pip
(venv) PS C:\Zavrsni\Zavrsni rad1>
```

Slika 3: Kreiranje virtualnog okruženja i instalacija Djanga

Nakon kreiranja virtualnog okruženja i instalacije Djanga, unutar virtualnog okruženja, moguće je kreirati Django projekt. Django projekt se kreira pomoći Python naredbe koja se pokreće u komandnoj liniji. Kada se naredba izvrši, kreirat će se mapa s pripadnim skriptama koja nosi naziv projekta. Svaki Django projekt može sadržavati jednu

ili više aplikacija. Aplikacija se također kreira iz komande linije pomoću Python naredbe. Izrada Django projekta i aplikacije je prikazana na ispisu 1.

```
$ django-admin startproject Ambulante

$ python manage.py startapp App
```

Ispis 1: Naredbe za kreiranje Django projekta i aplikacije

Dango razvojni okvir dolazi s nizom već ugrađenih biblioteka i aplikacija koje olakšavaju rad korisnicima. Lista ugrađenih aplikacija se može pronaći u skripti *settings.py*. Svaka sljedeća aplikacija koju korisnik kreira mora se registrirati unutar polja INSTALLED_APPS; u ovom primjeru to je bila aplikacija *App*. U slučaju da aplikacija nije registrirana, prilikom pozivanja naredbe za pokretanje poslužitelja aplikacija ne bi bila vidljiva. Svaka aplikacija mora imati jedinstveni naziv. Lista ugrađenih aplikacija prikazana je na slici 4.

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

Slika 4: Lista ugrađenih aplikacija

Django koristi arhitekturu MVT koja se može smatrati alternativom arhitekturi MVC (engl. *Model-View-Controler*).

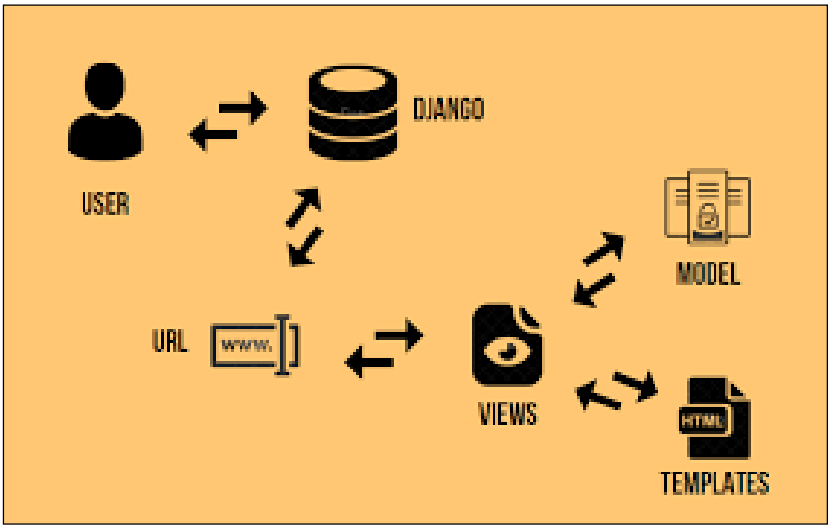
Model u MVT arhitekturi je isto što i model u MVC arhitekturi. Modeli u obje arhitekture predstavljaju sučelje za izradu tablica i pohranjivanje podataka u bazu. Svaki model u skripti *models.py* preslikava jednu tablicu baze podataka.

Predložak (engl. *templates*) u MVT arhitekturi je isto što i pogled (engl. *view*) u MVC arhitekturi Predložak, odnosno pogled služi za prikazivanje podataka modela te preuzimanje informacija od korisnika. Predlošci su način za dinamičko generiranje HTML-a. Svaki predložak ima ekstenziju .html i kombinaciju statičkog i dinamičkog sadržaja.

Pogled u MVT arhitekturi je isto što i kontroler u MVC arhitekturi. Pogled i kontroler su odgovorni za svu logiku web stranice. Oni predstavljaju sponu između baze podataka i predložaka koji prikazuju podatke krajnjem korisniku. Pogled prima zahtjev i vraća odgovor. Odgovor može biti sadržaj web stranice, preusmjeravanje, pogreška ili slika.

Rad aplikacije koja je izrađena korištenjem arhitekture MVT se odvija u sljedećim koracima (Slika 5):

1. Korisnik Djangu šalje URL zahtjev za resurs.
2. Razvojni okvir Django traži putanju resursa.
3. Ako je putanja pronađena i povezana s pogledom tada se poziva taj pogled.
4. Pogled stupa u interakciju s modelom i dohvaća odgovarajuće podatke iz baze podataka.
5. Pogled vraća korisniku odgovarajući predložak s dohvaćenim podacima.



Slika 5: Princip rada MVT arhitekture

Skripta *urls.py* koja se nalazi unutar mape projekta sadrži sve poveznice (engl. *urls*) koje predstavljaju vezu između predložaka i pogleda. Svaka poveznica je vezana za jednu funkciju iz pogleda koja vraća ili upisuje podatke u bazu.

Svaka aplikacija može imati svoju skriptu *urls.py*. U ovom slučaju da aplikacija nema svoju skriptu *urls.py* već koristi unaprijed definiranu skriptu koja se nalazi unutar projekta i kreirana je prilikom kreiranja projekta. Lista kreiranih poveznica iz skripte *urls.py* je prikazana na ispisu 2. Poveznice se sastoje od naziva (poveznica koja je prikazana u pretraživaču), funkcije koja se poziva prilikom pozivanja poveznice te naziva koji se koristi

u predlošcima ili pogledu. Npr. klikom na poveznicu `"{% url 'unos_pacijenata' radilište_id %}` iz predloška *pretraži_pacijenta.html* dolazi do pozivanja poveznice `path('unos_pacijenata/<int:radilište_id>', views.unos_pacijenata, name='unos_pacijenata'),` i funkcije *unos_pacijenta* iz skripte *views.py*. Funkcija prima *request* zahtjev i *id radilišta* koji je prenesen poveznicom. *Id radilišta* u ovom slučaju predstavlja čekaonicu radilišta u koju će biti upisan pacijent.

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('login/', LoginView.as_view(template_name='login.html'),
name='login'),
    path('index/', views.index, name='index'),
    path('lista_korisnika/', views.lista_korisnika,
name='lista_korisnika'),
    path('unos_korisnika/', views.unos_korisnika,
name='unos_korisnika'),
    path('lista_radilista/', views.lista_radilista,
name='lista_radilista'),
    path('unos_radilista/', views.unos_radilista,
name='unos_radilista'),
    path('lista_utrosaka/', views.lista_utrosaka,
name='lista_utrosaka'),
```

Ispis 2: Skripta *urls.py*

U skripti *urls.py* nalazi automatski definirana poveznica za administratorsko sučelje. Automatsko administratorsko sučelje je jedan od najmoćnijih dijelova Djanga čija je uloga da čita meta podatke iz modela kako bi pružilo sučelje usmjereno na modele gdje pouzdani korisnici mogu upravljati sadržajem web aplikacije. Administratorskom sučelju može pristupiti samo korisnik s ulogom *Super korisnik* (engl. *super user*). Super korisnik se kreira automatski kod kreiranja projekta. Način kreiranja super korisnika prikazan je na slici 6.

```
(venv) PS C:\Zavrsni\Zavrsni rad1\mysite> python manage.py createsuperuser
Username (leave blank to use 'inesvarnica'): ines
Email address: ines.tg1@gmail.com
Password:
Password (again):
Superuser created successfully.
(venv) PS C:\Zavrsni\Zavrsni rad1\mysite>
```

Slika 6: Kreiranje super korisnika

Modele koji se prikazuju unutar automatskog administrativnog sučelja potrebno je registrirati u skripti *admin.py*. Dva modela, grupe i korisnici, će biti vidljivi bez prethodne registracije. Ukoliko se neki od ta dva modela proširuje, potrebno je u skripti *admin.py* obaviti registraciju proširenih polja modela kako bi se i ona prikazala u administratorskom sučelju.

Ovlasti koje ima super korisnik unutar administratorskog sučelja su ograničena. Preporučljivo je administratorsko sučelje ne koristiti kao izgled aplikacije jer najčešće ne može pružiti sve potrebne funkcionalnosti.

4.2. Python

Python je programski jezik visoke razine koji dopušta objektno orijentirani, strukturni i aspektno orijentirani stil programiranja. Njegova sintaksa i fleksibilnost omogućava jednostavno pisanje kôda koji su lako čitljivi, što ga čini jednim od najpopularnijih programskih jezika. Prvu inačicu Pythona, kao nasljednika programskog jezika ABC, je 1990. godine stvorio Guido van Rossum. Programski jezik je dobio ime prema seriji Monty Python's Flying Circus. Zadnja dostupna verzija Pythona je 3.10 i može se koristiti na svim operacijskim sustavima.

Python je interpreterski jezik. U usporedbi s ostalim kompajlerskim jezicima, kao što su C i C++, programi napisani u Pythonu nešto se sporije izvršavaju, što ne utječe na njegovu popularnost. Za razliku od programskog jezika Java, koji se često primjenjuje za mobilne aplikacije, Python se više koristi za aplikacije koje se izvode na osobnom računalu.

4.3. SQLite

Razvojni okvir Django dolazi s ugrađenom bazom podataka SQLite. Python skripta *manage.py* sadrži funkcije za kreiranje tablica i migriranje podataka u SQLite bazu podataka. Kreiranje tablica moguće je obaviti bez da se unutar skripte *manage.py* kreira ijedna klasa s obzrom da baza dolazi s nizom već ugrađenih tablica kao npr. tablice za korisnike, grupe, dozvole itd.

Tablice se kreiraju u skripti *models.py* definiranjem modela. Svaki se model preslikava u jednu tablicu unutar baze koja mora imati jedinstveno ime. Svaki atribut modela predstavlja jedno polje unutar tablice. Moguće je koristiti već postojeće modele unutar Djanga te ih proširiti s novim atributima. U ovom primjeru korišten je već ugrađen model za

korisnike koji je proširen atributom *role*. Primjer modela *Korisnici* s proširenim atributom prikazan je na ispisu 3.

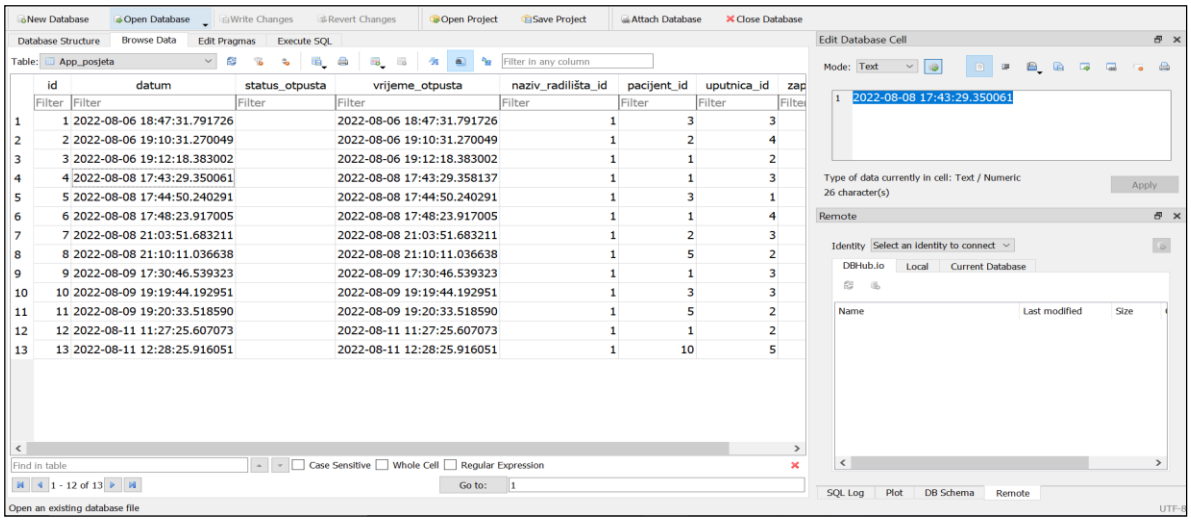
Prošireni model Korisnici u skripti *settings.py* postavljen kao model na kojem se vrši autorizacija i autentifikacija pomoći naredbe `AUTH_USER_MODEL = 'App.Korisnik'`

```
class Korisnik(AbstractUser):

    ROLES = (('dr', 'liječnik'), ('spec', 'specijalist'),('admin',
'administrator'),('ms', 'medicinska sestra'),('vms', 'viša medicinska
sestra'))
    role = models.CharField(max_length=50, choices=ROLES)


    def __str__(self):
        return '%s %s %s ' % (self.username, self.email , self.role )
```

Ispis 3: Model Korisnik

4.4. DB browser

DB browser je besplatan i jednostavan alat za upravljanje bazama podataka. DB browser je najmanjem samo za SQLite baze podataka te je upravo zbog toga pogodan za Djangovu bazu podataka. U nekoliko koraka je moguće pregledati tablice, atribute i podatke u njima. DB browser omogućava kreiranje novih i brisanje postojećih tablica, a sadrži i konzolu u kojoj je moguće izvršavati različite upite (engl. *querry*). DB browser omogućava i jednostavno dodavanje, brisanje, izmjenu i filtriranje podataka bez da je potrebno pisati upit. Sve podatke iz tablica kao i sve podatke koji su dobiveni iz upita je moguće ispisati. Sučelje DB browsera je prikazan na slici 7.



	id	datum	status_otpusta	vrijeme_otpusta	naziv_radilišta_id	pacijent_id	uputnica_id
1	1	2022-08-06 18:47:31.791726		2022-08-06 18:47:31.791726	1	3	3
2	2	2022-08-06 19:10:31.270049		2022-08-06 19:10:31.270049	1	2	4
3	3	2022-08-06 19:12:18.383002		2022-08-06 19:12:18.383002	1	1	2
4	4	2022-08-08 17:43:29.350061		2022-08-08 17:43:29.358137	1	1	3
5	5	2022-08-08 17:44:50.240291		2022-08-08 17:44:50.240291	1	3	1
6	6	2022-08-08 17:48:23.917005		2022-08-08 17:48:23.917005	1	1	4
7	7	2022-08-08 21:03:51.683211		2022-08-08 21:03:51.683211	1	2	3
8	8	2022-08-08 21:10:11.036638		2022-08-08 21:10:11.036638	1	5	2
9	9	2022-08-09 17:30:46.539323		2022-08-09 17:30:46.539323	1	1	3
10	10	2022-08-09 19:19:44.192951		2022-08-09 19:19:44.192951	1	3	3
11	11	2022-08-09 19:20:33.518590		2022-08-09 19:20:33.518590	1	5	2
12	12	2022-08-11 11:27:25.607073		2022-08-11 11:27:25.607073	1	1	2
13	13	2022-08-11 12:28:25.916051		2022-08-11 12:28:25.916051	1	10	5

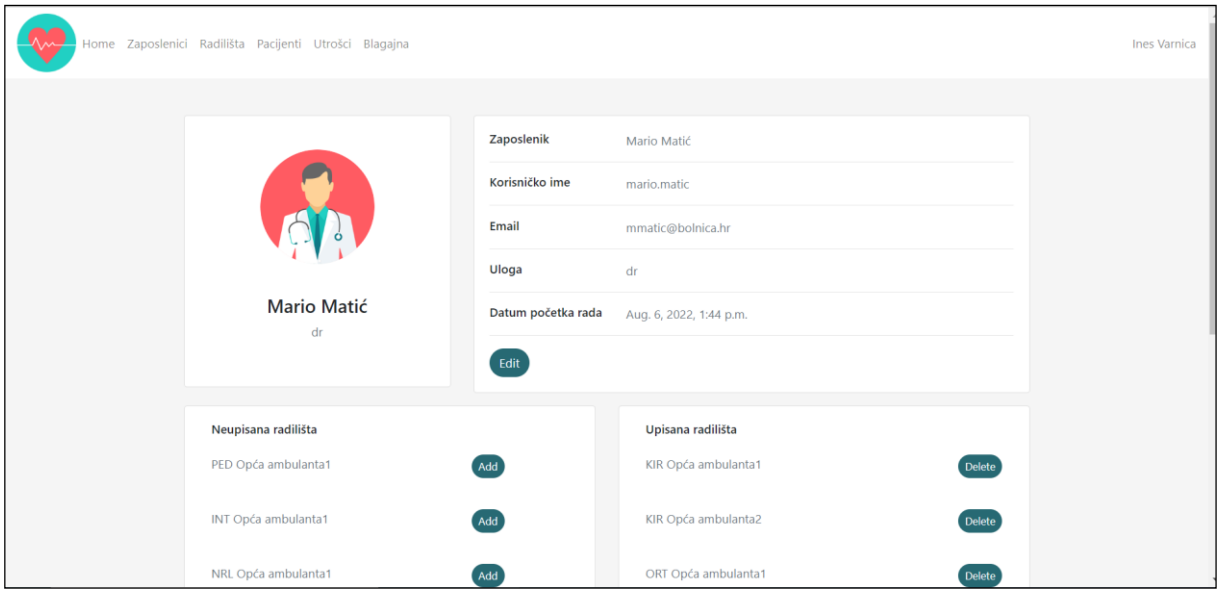
Slika 7*:* Sučelje DB browsera

4.5. Bootstrap

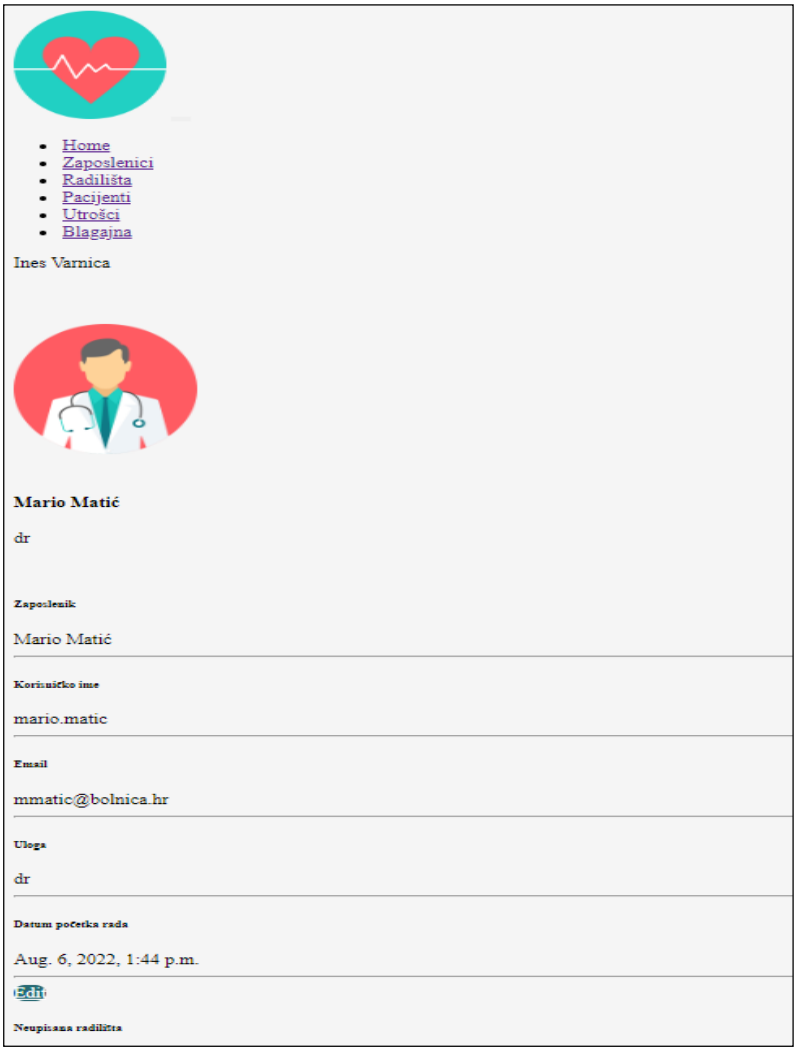
Bootstrap je besplatni razvojni okvir otvorenog kôda koji je namijenjen za izradu izgleda web stranica i web aplikacija. Bootstrap pruža zbirku predložaka dizajna temeljenih na HTML-u i CSS-u. Također, uključuje osnove responzivnog web razvoja tako da programeri samo trebaju umetnuti kôd u unaprijed definirani mrežni sustav. Bootstrap omogućuje web stranici ili aplikaciji da otkrije veličinu i orijentaciju zaslona posjetitelja i automatski prilagodi prikaz u skladu s tim. Okvir Boostrapa izgrađen je u HTML, CSS i JavaScriptu. Bootstrap omogućava programerima bržu izradu web stranica u odnosu na tradicionalnu izradu izgleda korištenjem HTML-a i CSS-a jer nije potrebno trošiti vrijeme na izradu osnovnih funkcija i naredbi.

Mark Otto i Jacob Thornton razvili su Bootstrap na Twitteru kako bi poboljšali dosljednost alata koji se koristi i smanjili vrijeme koje troše na održavanje stranice. Bootstrap je prije bio poznat pod nazivom Twitter Blueprint, a ponekad se naziva i Twitter Bootstrap.

Zadnja dostupna verzija Bootstrapa je 5 i korištena je u ovoj alikaciji. Na slikama 8 i 9 se može vidjeti primjer stranice sa i bez Bootstrapa. Na slici koja prikazuje profil pacijenta bez korištenja Bootstrapa može se vidjeti da su elementi stranice poredani jedan ispod drugog i da se cijeli sadržaj nalazi na jednoj kartici. Na slici koja pokazuje profil pacijenta s Bootstrapom se mogu vidjeti različite kartice i navigacija koje su poredane po stranici nekim redoslijedom.



Slika 8*:* Profil pacijenta izrađen korištenjem Bootstrapa



Slika 9: Profil pacijenta izrađen bez korištenja Bootstrapa

4.6. CSS

CSS je stilski jezik koji se koristi za dizajn dokumenta koji je napisan u HTML-u. CSS se koristi da se uredi izgled i raspored na stranici. Drugim riječima CSS odlučuje kako će se prikazati HTML elementi. Samo pisanje i učenje CSS-a nije zahtjevno i nije potrebno nikakvo prethodno znanje. Dobra praksa je prvo naučiti HTML te samo proširiti znanje HTML-a CSS-om. CSS omogućava da stranice postaju zanimljivije, uređenije i dinamičnije. Prilikom korištenja HTML-a i CSS-a potrebno je pridržavati se određenih pravila kako bi stranice ispravno radile.

Prva verzija CSS-a je izašla u prosincu, 1996. godine. Zadnja verzija CSS-a izašla je u prosincu 2020. godine i zove se CSS 4.15. CSS je temeljna tehnologija WWW-a (engl. *World Wide Web*) kao i HTML te JavaScript.

4.7. HTML

HTML je prezentacijski jezik koji se koristi za izradu web stranicu. HTML je temeljna tehnologija WWW-a kao i CSS i JavaScript. HTML dokumenti se stvaraju koristeći HTML jezik. HTML jezik omogućava oblikovanje sadržaja stranice i stvaranje hiperveza hipertext dokumenata. HTML jezik je vrlo rasprostranjen i svoju popularnost može zahvaliti tome što je od početka zamišljen kao besplatan jezik koji se lako uči. Osnovna zadaća HTML jezika je uputiti web pregledik kako prikazati hipertext dokument, a da pri tome nastoji da taj dokument izgleda jednako u svim web preglednicima. HTML nije programski jezik i ne može izvršiti nikakvu zadaću, čak niti zbrajanje brojeva. On se koristi samo za opis hipertekstualnih dokumenata. HTML datoteke imaju ekstenziju .html i u ovoj aplikaciji svi predlošci su napisani u HTML

HTML je nastao 1991. godine i prvi ga spominje Tim Berners - Lee. Prva verzija HTML-a je objavljena 1993. godine. Zadnja verzija HTML-a je HTML5 koja je izašla 2014. godine i korištena je u ovoj aplikaciji.

4.8. Visual studio code

Visual studio code koji se naziva još i skraćenim nazivom VS code je uređivač kôda. VS Code je najpopularniji uređivač kôda na tržištu. Objavljen je 2015. godine. Izvorni kôd je otvoren i besplatan te su kompajlirani binarni dokumenti dostupni za privatnu i komercijalnu upotrebu. Razvio ga je Microsoft i podržan je na Windows, Linux i Mac OS

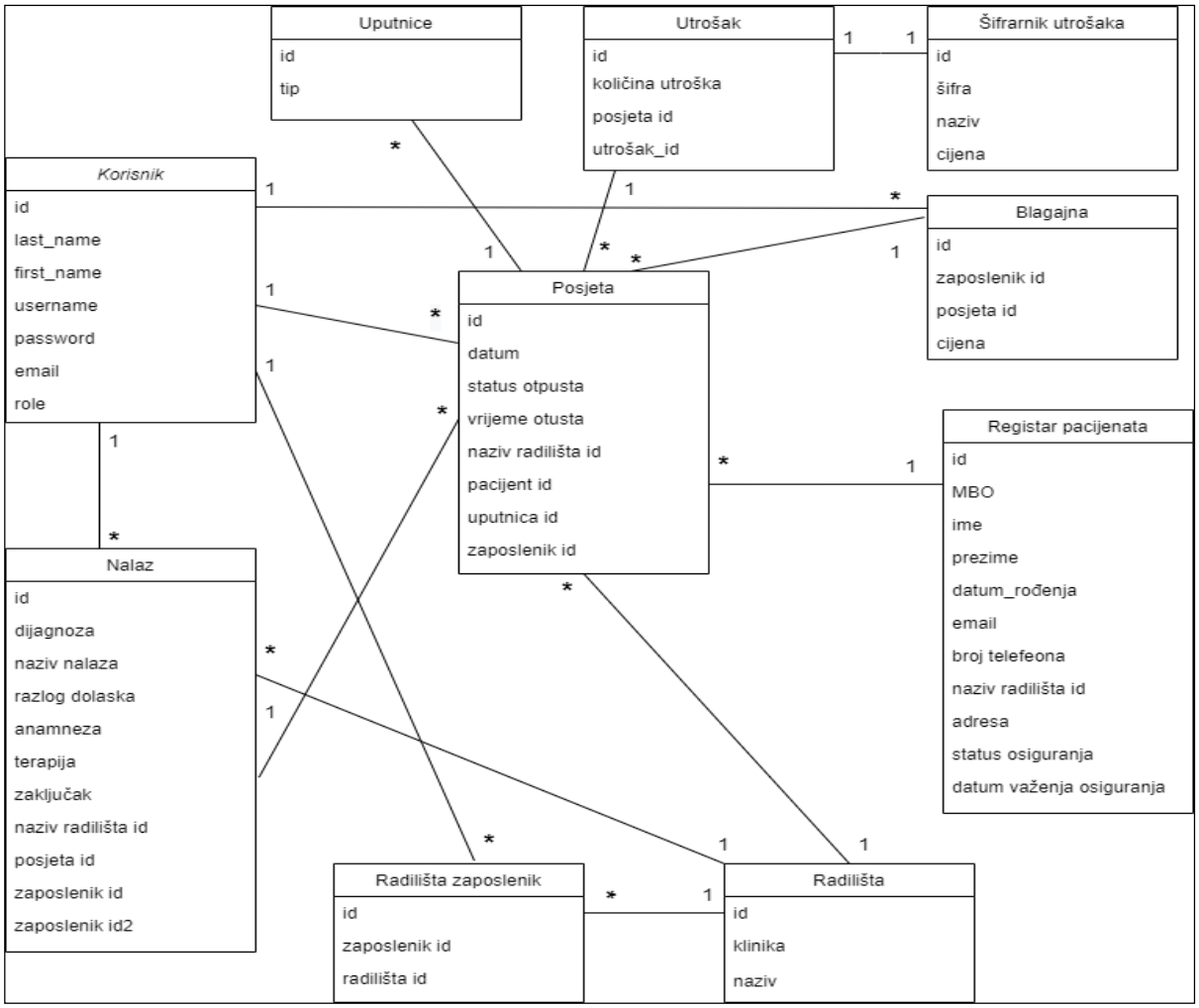
operacijskim sustavima. VS code se može koristiti za razne programske jezike kao što su Python, Java, JavaScript, C++, Node.js.

VS code ima ugrađenu git kontrolu i GitHub. Također VS code omogućava isticanje dijelova sintakse, inteligentno dovršavanje kôda i otklanjanje pogrešaka. Još jedna od prednosti VS coda je ta što mu je sučelje prilagodljivo pa je korisnicima omogućeno da mijenjaju temu, prečace na tipkovnici te laku instalaciju dodatnih funkcionalnosti. VS code omogućava korisniku da otvori više direktorija istovremeno, a to znači da korisnik može raditi na više aplikacija istovremeno. Samim time, više prozora može biti sačuvano kao radni prostor za ponovnu upotrebu.

Zadnja verzija VS codea je 1.70 i izašla je u srpnju 2022. godine, ali ona nije korištena za izradu ove aplikacije, već je korištena verzija 1.68 koja je izašla u svibnju 2022. godine.

5. Princip rada baze podataka

Nakon definiranja funkcionalnosti koje bi trebala sadržavati aplikacija potrebno je pomno isplanirati bazu podataka, odnosno koje bi tablice trebala sadržavati te kakva bi veza trebala biti među tablicama. Kada je riječ planiranju izrade baze podataka možemo koristiti tri principa, a to su *code first, database first i model first.* Ova baza je definirana po principu *code first. Code first* je princip planiranja izrade baze koji je popularan među MVC i MVT arhitekturama. Django koristi arhitekturu MVT. Kod principa razvoja *code first* najprije se definiraju entiteti, atributi i veze među njima. Ova aplikacija je razvijena korištenjem uređivača kôda VS code. Tablice u bazi i veze te veze među njima se generiraju na temelju klasa pokretanjem skripti `python manage.py makemigration` i `python manage.py migrate`, koje se izvode u komandnoj liniji. Tablice je moguće pregledavati pomoću preglednika DB SQLite. U Djangu će se osim tablica koje su definirane klasama kreirati i sistemske tablice, koje su sastavni dio razvojnog okvira. Prikaz tablica i veza među njima se može vidjeti na slici 10.



Slika 10: Relacijski dijagram baze podataka

6. Opis rada aplikacije za praćenje rada ambulante

Glavni zadatak ove aplikacije je omogućiti svim korisnicima brzo i jednostavno obavljanje svojih radnih zadataka. Cilj ove aplikacije je smanjiti vrijeme čekanja pacijenata u čekaonicama i na šalterima tako što je medicinskim sestrama omogućen unos pacijenata u čekaonicu te naplata odrađenih usluga u samo nekoliko koraka. Liječnicima je omogućeno brzo pisanje nalaza i pregled medicinske dokumentacije pacijenta, tako da liječnici mogu imati više vremena za pregled pacijenta, a ne gubiti vrijeme na administrativne poslove.

Korisnici koji koriste aplikaciju mogu imati 3 različite uloge, a to su:

- Medicinske sestre/tehničari
- Liječnici
- Administratori

Sama aplikacija svojim funkcionalnostima i sučeljem prilagođena je tim korisnicima.

6.1. Administrativni korisnici

Administrativni korisnici se kreiraju kreiranjem super korisnika u komandnoj liniji kao što je prikazano na slici 6. u poglavlju 3. Administrativnom korisniku se nakon prijave prikaže pet kartica te pripadna navigacija koja sadrži logo bolnice, ime i prezime trenutno prijavljenog korisnica. Omogućena je i odjava korisnika. Administrativni korisnik ima mogućnost odabira prikaza liste radilišta, liste korisnika, liste pacijenata, liste utrošaka te blagajne. Primjer početne stranice administratora je prikazana na slici 11.

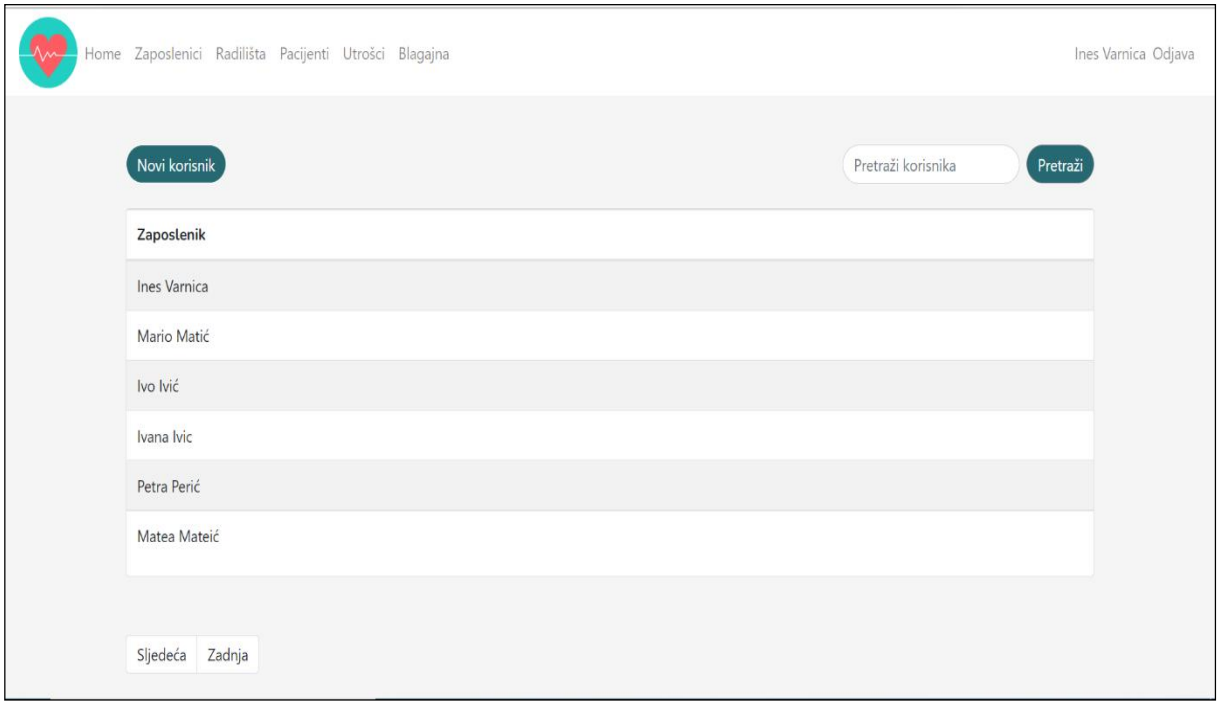
Slika 11: Početna stranica administratora

Prilikom klika na bilo koju opciju *otvori* kreira se *get* zahtjev te se prikaže neka od odabranih lista u obliku tablice. Opcija otvori je u stvari poveznica koja vodi na odabranu stranicu. U *urls.py* se pronađe poveznica s pripadnom funkcijom. Na ispisu 4 se nalazi primjer funkcije *lista_korisnika.* U ovom konkretnom primjeru prilikom pozivanja funkcije iz baze podataka iz tablice *Korisnici* se dohvaćaju svi korisnici te se prikazuju u podlošku *lista_korisnika.html* (slika 12).

```
def lista_korisnika(request):

    p = Paginator(Korisnik.objects.all(), 6)
    page = request.GET.get('page')
    korisnici = p.get_page(page)
    return render(request,'lista_korisnika.html',{"data":korisnici})
```

Ispis 4: Funkcija lista korisnika

Također dolazi do promjena navigacije koja sad za razliku od navigacije na početnoj stranici sadrži poveznice koje vode na listu zaposlenika, listu radilišta, listu pacijenata, listu utrošaka te blagajnu. Na vrhu svake tablice se nalazi opcija za unos podataka u tablicu.



Slika 12: Predložak lista_korisnika.html

Podaci se unose u bazu podataka putem formi. Slanjem zahtjeva vrste *get* od strane klijenta, server šalje odgovarajuću formu. Na ispisu 5 prikazan je primjer registracije novih zaposlenika. U ovom primjeru se nakon kreiranja *get* zahtjeva poziva forma *AddUserForm.* Forma je napisana u skripti *forms.py* unutar aplikacije. Za *AddUserForm* se koristila ugrađena Djangova forma *UserCeartionForm.* Ova forma je ugrađeni modul naslijeđen od klase ModelForm. *UserCeartionForm* ima tri polja: *korisničko ime, lozinka1 i lozinka2* (koja se koristi za potvrdu lozinke) Moguće je i nadodati druga polja koja se koriste u modelu *Korisnik*, a u ovoj aplikaciji nadodana su sva polja. Da bi se koristila forma *UserCreationForm* potebno ju je uvesti iz django.contrib.auth.forms. Prilikom pritiska na opciju *unesi korisnika* dolazi do kreiranja *post* zahtjeva te se dohvaćaju podaci koji su uneseni unutar forme. Prvi korak je provjera validnosti unesenih podataka pozivanjem Djangove funkcije *.is_valid().* Zahvaljujući *UserCreationForm* nije potrebno provjeravati jesu li unesene lozinke jednake te je li korisničko ime jedinstveno jer *UserCreationForm* to radi za nas. U ovom primjeru nije dozvoljeno ponavljanje emaila, ali provjera jedinstvenosti emaila je u ovom slučaju obavljena u samoj funkciji. Iz baze podataka se dohvaća korisnik koji ima isti email kao što je email koji je unesen u formi. U slučaju da takav korisnik postoji pojavit će se upozoravajuća poruka te će doći do preusmjeravanja na istu stranicu odnosno na ponovni unos. U slučaju da je sve ispravno napravljeno podaci se spremaju pozivom funkcije *save*, korisniku se prikazuje poruka o uspješnom spremanju podataka te dolazi do preusmjeravanja na istu stranicu.

```
def unos_korisnika(request):
    if request.method == 'GET':
        UserForm = AddUserForm()
        return render(request,
'unos_korisnika.html',{'form':UserForm})
    else:
        UserForm = AddUserForm(request.POST)
        if UserForm.is_valid():
            uneseni_email = UserForm.cleaned_data.get('email')
            username = UserForm.cleaned_data.get('username')
            korisnik_email=Korisnik.objects.filter(email=uneseni_emai
l)
            korisnik_username=Korisnik.objects.filter(username=userna
me)
            if(korisnik_email ):
                messages.error(request,'Email je zauzet')
                return redirect('unos_korisnika')
            else:
                UserForm.save()
                messages.error(request,'Korisnik uspješno dodan')
                return redirect('unos_korisnika')
        else:
            messages.error(request,'Niste ispravno unijeli podatke')
            return redirect('unos_korisnika')
```

Ispis 5: Funkcija za unos novih korisnika

Administratoru je osim unosa novih korisnika omogućeno i ažuriranje korisnikovih podataka. Prilikom klika na nekog korisnika u tablici korisnici otvara se profil korisnika. Na profilu se nalaze osnovne informacije o korisniku kao i radilišta na koja korisnik ima pravo. Također, omogućeno je i dodavanje novih radilišta korisniku te brisanje postojećih. Informacija o tome čiji se profil treba otvoriti prilikom klika na korisnika se prosljeđuje putem poveznice. Unutar poveznice osim što se prosljeđuje ime sljedeće stranice koju je potrebno otvoriti, prosljeđuje se *id* korisnika koji se prenosi sve do forme za ažuriranje korisnika. Ažuriranje korisnika se obavlja na način da se iz baze podataka iz tablice *Korisnik* dohvate podaci korisnika čiji je *id* proslijeđen. Podaci su prikazani u *editUserForm* koja se

prikazuje u predlošku nakon slanja zahtjeva *get*. Metodom *post* se dohvaćaju novi uneseni podaci se i ponovno spremaju u bazu podataka. *EditUserForm* koristi Djangovu ugrađenu formu *ChangeUserForm* koja provjerava jedinstvenost korisničkog imena. *SetPasswordForm* je forma koja se koristi za promjenu lozinke korisnika. Primjer forme za ažuriranje korisnika prikazan je na slici 13.

Slika 13: Forma za ažuriranje korisnika

Dodjela prava korisnicima se obavlja u predlošku *korisnik_radiliste.html* te ju može obavljati samo administrator. Unutar predloška, osim osnovnih informacija o pacijentu, nalazi se i lista upisanih i neupisanih radilišta. Dodavanje i brisanje radilišta se obavlja odabirom odgovarajućih opcija. Svaka opcija sadrži parametar *id_radilista* koji je potrebno dodati ili izbrisati. Akcije su primjerice *submit*, što znači da se prilikom pokretanja stvara *post* zahtjev. Nakon slanja zahtjeva *post* poziva se funkcija *korisnik_radiliste* i ponovo se učitava ista stranica. Ako u *post* zahtjevu postoji naziv nedodani, dohvatit će se *id_radilista* iz *post* zahtjeva te će se preko *id_radilista* u bazi podataka pronaći radilište koje je potrebno upisati. Podaci o zaposlenicima i pripadajućim radilištima spremaju se u tablicu *Radiliste_korisnik*. Ta tablica ima polja *id_korinsika* i *id_radilista*. Spremanje podataka o korisnicima i radilištima na koja korisnici imaju pravo obavlja se pozivom funkcije *save*, a brisanje veze između korisnika i radilišta pozivom funkcije *delete* (Ispis 6).

```
zaposlenik_u=Korisnik.objects.get(id=korisnik_id)
if 'nedodani' in request.POST:
    q=(request.POST)
    radilište_u=Radilišta.objects.get(pk=q['nedodani'])
    dodana_radilišta=Radilište_korisnik(radilište=radilište_u,
zaposlenik=zaposlenik_u)
    dodana_radilišta.save()
if 'dodana' in request.POST:
    q=(request.POST)
    #upisano_radilište=Radilište_korisnik.objects.filter(zaposlenik=k
orisnik_id)
    upisano_radilište=Radilište_korisnik.objects.filter(zaposlenik=ko
risnik_id).get(radilište=q['dodana'])
    upisano_radilište.delete()
```

Ispis 6: Kôd za dodavanje i brisanje radilišta korisnika

6.2. Medicinske sestre/tehničari

Administratorsko i medicinsko sučelje se razlikuju. Prilikom prijave korisnika u aplikaciju se provjerava njegova uloga. U slučaju da je ulogirani korisnik medicinski djelatnik dolazi do preusmjeravanja korisnika na stranicu *lista korisnikovih radilišta.* Korisniku se prikazuju samo ona radilišta koja su mu dodana od strane administratora. Klikom na neko radilište kreira se *get* zahtjev i otvara se čekaonica. Naziv radilišta je poveznica unutar koje se nalazi naziv stranice koju je potrebno otvoriti i *id radilišta*.

Čekaonica je podijeljena u dva dijela: na pacijente koji imaju napisan nalaz i na pacijente koji nemaju. U čekaonici su prikazani samo oni pacijenti koji upisani na taj dan. U čekaonicu je moguće upisati pacijenta klikom na opciju *ubaci pacijenta.* Opcija predstavlja poveznicu pomoću koje se otvara forma za pretraživanje pacijenata. Unutar poveznice se prenosi i *id radilišta.* Kada se odabere pacijent kojeg je potrebno ubaciti u čekaonicu, njegov se *id* zajedno s *id-om radilišta* prenosi sve do forme za posjetu. Unutar forme se odabire liječnik i tip uputnice te se svi podaci zajedno s *id radilišta i id pacijenta* spremaju u bazu podataka u tablicu *Posjeta*. Kao vodećeg liječnika moguće je odabrati samo liječnika koji ima ulogu specijalist. Vrijeme posjete se generira automatski te je jednako trenutnom vremenu. Funkcija koja se poziva prilikom kreiranja posjete prikazana je na ispisu 7.

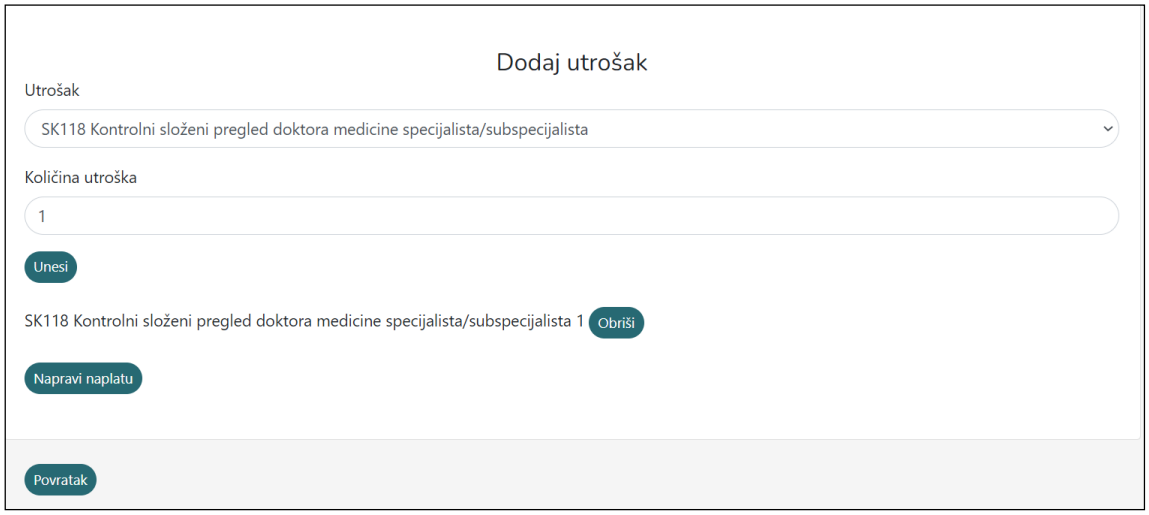
```
def posjeta(request,radilište_id, pacijent_id):
    if request.method == 'GET':
        Form = PosjetaForm()
        return render(request, 'posjeta.html',{'form':Form ,
"radilište_id":radilište_id})
    else:
        radilište=Radilišta.objects.get(pk=radilište_id)
        print(radilište)
        pacijent_u=RegistarPacijenata.objects.get(pk=pacijent_id)
        print(pacijent_u)
        Form = PosjetaForm(request.POST)
        if Form.is_valid():
            zaposlenik_u = Form.cleaned_data.get('zaposlenik')
            uputnica_u = Form.cleaned_data.get('uputnica')
            posjeta =
Posjeta(pacijent=pacijent_u,naziv_radilišta=radilište,zaposlenik=zapo
slenik_u,uputnica=uputnica_u)
            posjeta.save()
            return(redirect('sestrinska_čekaonica' , radilište_id =
radilište_id))
        else:
            messages.error(request,'Nešto je pošlo po zlu')
            return
HttpResponseRedirect(request.META.get('HTTP_REFERER'))
```

Ispis 7: Funkcija za kreiranje posjete

Posjetu je moguće i izbrisati klikom na opciju *obriši*. Klikom na opciju *posjeta* otvara se forma za unos usluga i materijala koji su potrošeni za liječene nekog pacijenta. Svakom pacijentu je potrebno obračunati utrošak, bez obzira ima li pacijent osiguranje ili nema, te samo medicinske sestre imaju mogućnost obračuna utroška. Unutar forme *PosjetaUtrošakForm* se prikazuju samo utrošci koji su uneseni i tablicu *ŠifrarnikUtrošaka* koju može popunjavati samo administrator. Prilikom odabira utroška potrebno je odabrati i količinu utroška. Dozvoljen je unos jednog utroška više puta, te će se njegova količina automatski povećati. Postoji i mogućnost brisanja utroška. Svaki put kada se klikne opcija *unesi* ili *obriši* kreira se *post* zahtjev. Nakon čega dolazi do brisanja ili dodavanja podataka u bazu u tablicu *Utrošak_posjeta.* Tablica *Utrošak_posjeta* sastoji se od dva strana ključa,

utrošak koji je vezan za tablicu *Šifrarnik_utrošaka,* te *posjetu* koji je vezan za tablicu *Posjeta.* Polje količina pripada samo tablici *Utrošak_posjeta.*

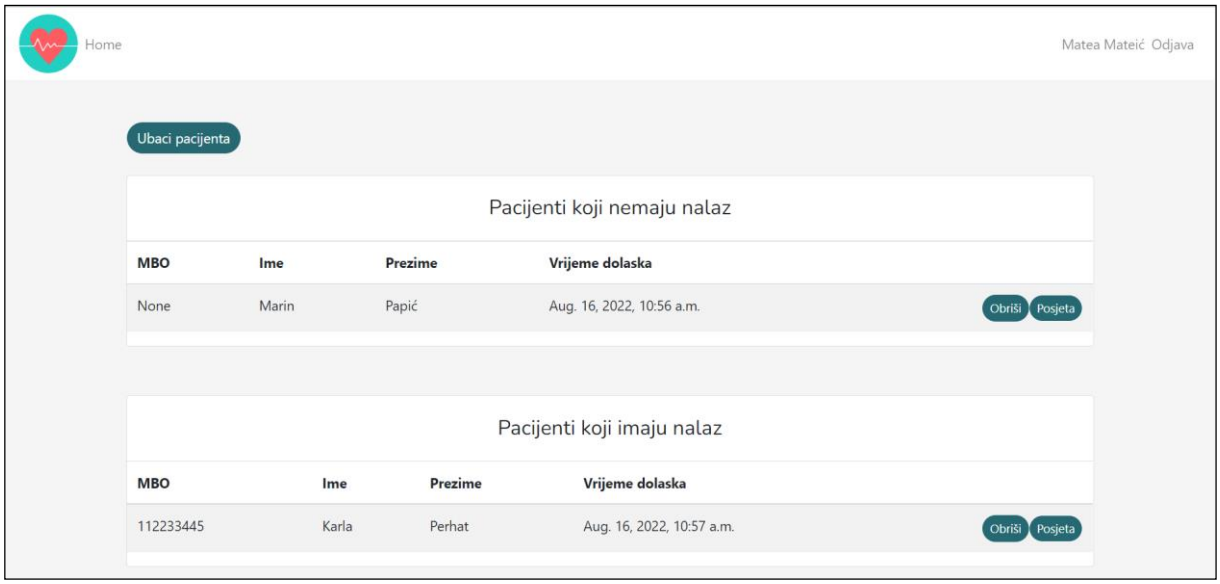
Ukoliko pacijent nema osiguranje, potrebno mu je naplatiti usluge koje su rađene tijekom njegovog liječena. Prilikom unosa utrošaka u formu *PosjetaUtrošakForm* ukoliko pacijent nema osiguranje pojaviti će se opcija *Napravi naplatu.* Kada se napravi naplata nije više moguće unositi utrošak sve dok se ta naplata ne stornira. Sve naplate koje su napravljene bit će vidljive administratoru u predlošku *blagajna* te ukupan iznos svih naplata. Primjer forme za unos utroška prikazan je na slici 14.



Slika 14*:* Forma za unos utroška

6.3. Liječnici

Nakon uspješne prijave korisnika u sustav dolazi do provjere uloge korisnika. U slučaju da je prijavljeni korisnik medicinski djelatnik dolazi do preusmjeravanja korisnika na stranicu koja prikazuje sva radilišta na koja prijavljeni korisnik ima prava. Početno sučelje za sve medicinske djelatnike izgleda isto, samo se razlikuju po prikazanim ambulantama. Odabirom nekog od radilišta dolazi do otvaranja čekaonice radilišta kao što je i slučaj kôd medicinskih sestara. Čekaonica je podijeljena u dva djela. U prvom dijelu čekaonice se nalaze pacijenti koji nemaju napisan nalaz, a u drugom dijelu čekaonice oni pacijenti koji imaju napisan nalaz tako da liječnici mogu lakše pratiti koje pacijente su obradili. Liječnicima je omogućeno i ubacivanje pacijenata u čekaonicu te također brisanja pacijenata iz čekaonice. Prilikom brisanja pacijenta dolazi do brisanja samo te određene posjete, a ne i čitavog profila pacijenta. U čekaonici su prikazani pacijenti koji imaju kreiranu posjetu na današnji dan. Primjer čekaonice je prikazan na slici 15.



Slika 15: Čekaonica

Klikom na opciju *nalazi* otvara se forma *nalazForm.* Kada se gradi aplikacija koja se temelji na bazi podataka velike su šanse da će se koristiti forme koje se blisko preslikavaju Django modelima. Takve se forme u Djangu nazivaju *ModelForm* i *nalazForm* koja je jedna od njih. U ovom slučaju je napravljen obrazac za pisanje nalaza te podaci koji su uneseni u obrazac se spremaju u tablicu *nalazi.* Tablica u kojoj će se spremati nalazi je definirana prilikom pisanja same forme. Prednost korištenja *ModelForm* je ta što nije potrebno definirati vrstu polja jer je već definirana u svoj modelu. Prilikom prikaza forme u predlošku nije potrebno prikazati sva polja iz tablice. Polja koja će biti prikazati u formi se definiraju unutar same forme. Polja koja predstavljaju strane ključeve u tablici se u formama prikazuju na malo drugačiji način. Takva polja će biti prikazana kao polja izbora, a izbori dolaze iz polja modela. Izbori obično uključuju i prazan izbor, a ako je polje obavezno tada je korisnik prisiljen napraviti izbor. Ukoliko je unutar polja modela definirano da polje ne smije biti prazno *(blank=False)* tada se prazan izbor neće prikazivati. U nalazForm (ispis 8) takva polja su polja *zaposlenik* te *zaposlenik2*. Također unutar formi je mguće koristiti i filtere. U ovom slučaju se u polju *zaposlenik* prikazuju samo zaposlenici koji imaju ulogu *'spec'* dok se u polju *zaposenik2* prikazuju samo zaposlenici koji imaju ulogu *'dr'*.

```
class NalazForm(ModelForm):
    zaposlenik = CustomModelFilter(queryset   = Korisnik.objects
.filter( role='spec').filter())
    zaposlenik2 = CustomModelFilter(queryset=Korisnik.objects.
filter(role='dr').filter())
    class Meta:
        model = Nalaz
       fields  = ['dijagnoza', 'razlog_dolaska',
'anamneza','terapija', 'zaključak','zaposlenik','zaposlenik2']
```

Ispis 8: Forma nalaz

Napisani nalaz je moguće ažurirati ili izbrisati. Mogućnost brisanja ili ažuriranja nalaza imaju samo oni djelatnici koji su potpisani na nalaz, odnosno u praksi oni djelatnici koji su sudjelovali u pisanju nalaza. Zabrana brisanja ili ažuriranja nalaza je napravljena na način da aplikacija provjerava je li se ulogirani djelatnik nalazi u potpisu nalaza. U slučaju da provjera nije prošla, opcije za ažuriranje i brisanje nalaza neće biti vidljivi.

Liječnicima je omogućeno pregledavanje arhive nalaza pacijenata. Prilikom ulaska u nalaz dolazi do promjene navigacije koja od tog trenutka sadrži poveznicu za arhivu nalaza pacijenta te za stranicu na kojoj se vrši otpust pacijenta. Arhiva nalaza pacijenta je prikazana u oblike tablice koja sadrži listu nalaza. Odabirom nekog nalaza na listi otvara se forma *nalazForm* s već popunjenim poljima.

Prilikom otpusta kreira se status i vrijeme otpusta. Podaci o statusu i vremenu otpusta se upisuju u tablicu *Posjeta.*

7. Zaključak

Aplikacija je isplanirana da korisniku koji je koristi olakša procese rada. Odnosno omogućava medicinskim sestrama upis pacijenata u čekaonicu u samo nekoliko koraka te unos usluga i materijala koji su potrošeni na liječene bolesnika. Također liječnicima je omogućeno lako i jednostavno praćenje bolesnika koji su pregledani te onih koji nisu. Pisanje nalaza te pregled dosadašnje medicinske dokumentacije je moguć u samo nekoliko koraka. Cilj ove aplikacije je smanjiti redove čekanja unutar čekaonica i na šalterima radilišta. Također je jedan od ciljeva omogućiti liječnicima da što manje vremena provode pišući medicinsku dokumentacija te da više vremena provode pregledavajući pacijenta.

Django ima ugrađenu zaštitu od većine CSRF napada pod uvjetom da je zaštita omogućena tamo gdje je to prikladno. CSRF zaštita radi tako da provjerava tajnu u svakom *post* zahtjevu. Ovo omogućava da zlonamjerni korisnik ne može ponovno reproducirati obrazac *post* i natjerati drugog korisnika da nesvjesno podnese taj obrazac. Zlonamjerni korisnik mora znati tajnu specifičnu za korisnika (upotrebom kolačića). Također, Djangovi upiti su zaštićeni od SQL ubacivanja budući da su upiti konstruirani korištenjem parametrizacije upita. Upit SQL se definira odvojeno od parametra upita.

Bootstrap je fleksibilan i jednostavan za rad. Glavna prednost Bootstrapa je što nudi responzivan dizajn i održava široku kompatibilnost s preglednicima. Također, Bootstrap se može koristiti s bilo kojim IDE ili uređivačem te bilo kojim jezikom na strani poslužitelja.

Korištenjem navedenih alata zadovoljene su sve potrebe napravljene aplikacije. Ideja je da se korištenjem moćnih i jednostavnih alata stvori što bolja i praktičnija aplikacija, što je ovdje i postignuto.

8. Literatura

[1] Django: The web framework for perfectionists with deadlines, Django introduction - Learn web development | MDN, Django Project MVT Structure – GeeksforGeeks, Django MVT – Javatpoint (posjećeno 11.08.2022.)

[2] Welcome to Python.org, What is Python? Executive Summary (posjećeno 11.08.2022.)

[3] Bootstrap dokumentacija, A Simple Bootstrap Tutorial | Toptal (posjećeno 11.08.2022.)

[4] DB Browser for SQLite (posjećeno 12.08.2022.)

[5] Visual Studio Code - Code Editing. Redefined (posjećeno 13.08.2022.)

[6] CSS Tutorial - W3Schools (posjećeno 13.08.2022.)

[7] HTML Tutorial - W3Schools (posjećeno 14.08.2022.)

[8] Code First vs. Database First vs. Model First Approach (posjećeno 15.08.2022.)

9. Popis slika