Izrada web aplikacije korištenjem Djanga

Troha, Antonela

Sveučilište u Rijeci – Odjel za informatiku

Jednopredmetni preddiplomski studij informatike

Antonela Troha

Izrada web aplikacije u Djangu

Završni rad

Mentor: v. pred. dr. sc. Vedran Miletić

Rijeka, 24. rujna 2021.

Sažetak

U ovom završnom radu prikazana je web aplikacija u razvojnom okruženju Django, koje se koristi Python programskim jezikom. Najveća prednost ovog razvojnog okruženja je to što koristi model-obrazac-pogled metodu koja omogućuje jednostavan i brz razvoj web aplikacija i osigurava sigurnost stranica. U radu je prikazano što radi svaka od ovih komponenti i kako se pomoću Djangovog jezika za predloške može na jednostavan način manipulirati prikazom podataka iz baze. U završnom radu je također opisan i proces instalacije Django programskog okvira, te neki od obrazaca, funkcije i forme korištenih u izradi aplikacije.

Ključne riječi

završni rad, web aplikacije, programski okvir Django, Python

Sadržaj

1. Uvod

Tema ovog završnog rada je izrada web aplikacije za rezerviranje termina u salonu za uljepšavanje. Aplikacija je izrađena u web programskom okviru Django koji koristi Python kao programski jezik. Fokus završnog rada bio je naučiti se koristiti Django alatom za izradu back-end dijela aplikacije i proučiti mogućnosti koje ovaj okvir nudi.

U današnje vrijeme, Web je ono što nas okružuje i uvelike olakšava dugotrajne i teške procese. U salonima za uljepšavanje kao i na mnogim drugim radnim mjestima, zaposlene osobe gube previše vremena na razgovor s klijentima kako bi uopće našle termin i došle do datuma koji odgovara jednoj i drugoj strani. Zato je aplikacija u kojoj korisnica sama sebi može upisati termin nešto što bi olakšalo posao zaposlenicima salona, ali i omogućilo im da više vremena posvete zadovoljstvu klijenata i osobnom usavršavanju.

Na temelju same teme rada može se postaviti pitanje, zašto baš web aplikacija? Izrada web aplikacija puno je lakša i brža nego, u odnosu na primjerice izradu mobilnih aplikacija. Desktop aplikacije u današnje se vrijeme sve manje koriste, a pogotovo kada je u pitanju aplikacija koju korisnik može koristiti "S nogu". Web aplikacije se lako uz pomoć raznih alata mogu urediti da budu responzivne, s čime ne gube dizajn mijenjanjem veličine ekrana.

Osim toga, Web aplikacije ne zahtijevaju preuzimanje na uređaj pa tako štede memoriju, a i dalje im je lako pristupiti uz pomoć opcije "Dodaj na početni zaslon" koju ima gotovo svaki web preglednik.

Ovaj završni rad podijeljen je u tri dijela. U prvom dijelu proći prikazana je povijest razvoja web developmenta i spomenuti su detalji o nekim programskim jezicima. U drugom dijelu prikazane su pojedinosti o Djangu i načinu na koji je konstruiran, a u zadnjem dijelu prikazan je proces izrade web aplikacije i funkcionalnostima Djanga koje su korištene.

2. Povijest razvoja web developmenta

Kada je riječ o počecima World Wide Web-a, onda se radi o ideji koja je nastala u ožujku 1989. Te godine Tim Berners-Lee zaposlenik u CERN-u (Europska organizacija za nuklearna istraživanja) dobio je ideju kojom bi olakšao prijenos informacija među kolegama s raznih dijelova svijeta tako da se one prenose putem novog protokola (eng. HyperText Transfer protocol ili HTTP).

World Wide Web (WWW) Tim Berners-Lee predložio je kao hipertekstualni sustav na internetu koji je organiziran kao mreža međusobno povezanih HTML dokumenata.

U početku web se sastojao od statičkog teksta i veza na druge dokumente koje su nazvane hiperlinkovi. Takve web stranice nazivaju se statičkima, zbog čega do korisnika dolaze točno onakve kakve su napravljene. Korisnik može prelaziti na druge stranice preko već spomenutih hiperlinkova, čitati tekstove, gledati slike i ostalu multimediju koju stranica sadržava, ali na takvim stranicama nema interaktivnosti između sadržaja i korisnika.

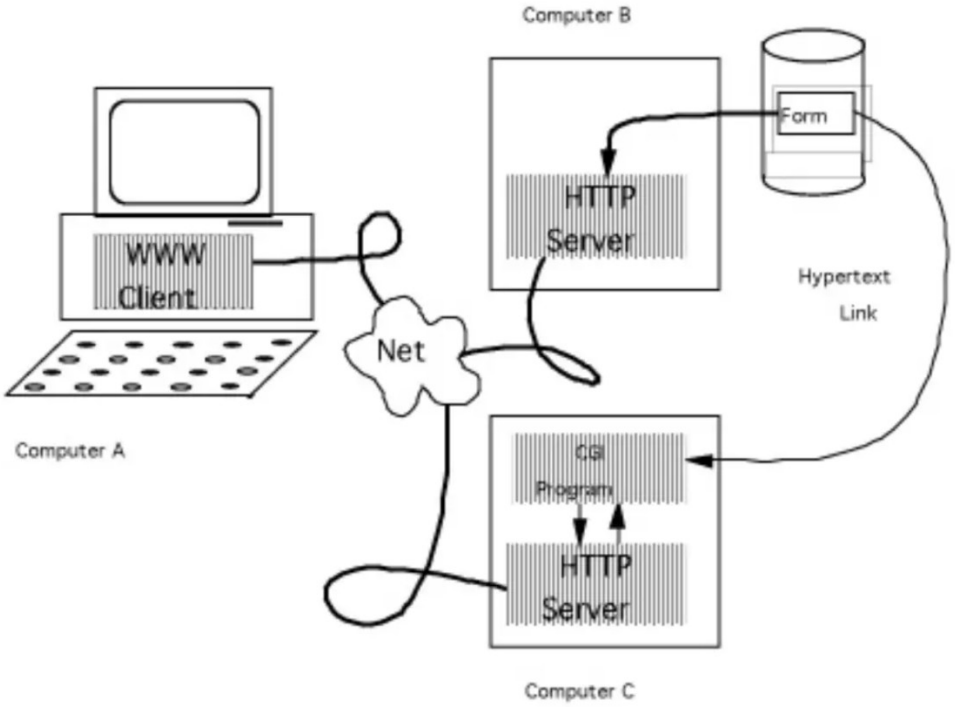
Razvojem weba dolazi do potrebe da se sadržaj stranica lakše i jeftinije prilagođava korisniku. Tako nastaju dinamičke web stranice kojima se sadržaj konstantno mijenja ovisno o korisnikovim potrebama.

2.1. Common Gateway Interface

Početak interaktivnosti na webu se može povezati s izumom CGI (eng. Common Gateway Interface) metode, a izumljena je 1993. godine u Nacionalnom centru za superračunalne aplikacije (NCSA). CGI skripte funkcionirale su tako da bi se pokretale na vanjskom programu i onda se slale natrag na originalnu web stranicu i to u obliku HTML koda.

U početku CGI se fokusirao na spajanje aplikacije s bazom podataka tako da su se uz pomoć takvih skripti najčešće izrađivale tražilice, kontakt obrasci i slično.

Primjer korištenja CGI skripte prikazana je Shemom 1.



Shema 1: Primjer korištenja CGI skripte

Kao što biblioteke funkcioniraju danas, tako je to bilo i s CGI skriptama. Ako je netko izrađivao stranicu na kojoj je bilo potrebno ugraditi obrazac za kontakt, ta osoba nije trebala gotovo nikakvo znanje o CGI skriptama nego je jednostavno na internetu mogla pronaći CGI skriptu koja je to omogućavala.

CGI skripte mogle su se pisati u bilo kojem programskom jeziku, a neki od njih bili su C, C++, Fortran, Perl, Visual Basic i tako dalje. Na početku je bilo uobičajeno vidjeti CGI programe koji su bili pisani u programskom jeziku C jer je to bio jedan od bržih programskih jezika. Upravo zbog toga što ih se često koristilo, C i C++ postali su baza za mnogo drugih programskih jezika. Neki od njih su JavaSript, Perl i PHP.

2.2. Perl

Iako se za CGI u početku više koristio C, zapravo je Perl jezik koji je imao veliki utjecaj na razvoj CGI skripti, ali i dinamičkog weba općenito.

Kao što je već spomenuto, Perl se koristio za pisanje CGI skripti koje su se spremale u arhive na internetu i mogle slobodno koristiti. Perl je programski jezik koji ne samo da ima vrlo kvalitetnu arhivu gotovih programskih rješenja nego ima i veliku zajednicu programera i korisnika. Autor Perla je Larry Wall koji za ovaj programski jezik kaže da je poput ljudskog, a svakako se smatra

jezikom kojeg nije teško naučiti.

Jedna od vrlo zanimljivih i poučnih priča je primjer srednjoškolca Matta Wrighta koji je 1995. godine otvorio svoju arhivu gotovih skripti. Mattova najpoznatija skripta zapravo je bila obrazac po imenu FormMail . Taj obrazac analizirao je poslane podatke i slao ih korisniku koji je bio naveden kao primatelj. Nažalost, taj obrazac nije imao dobro postavljene zaštite pa su ga brzo počeli zloupotrebljavati za slanje nepoželjnih mailova. Ovaj slučaj bio je inspiracija za ostatak Perl zajednice da stvori kvalitetnu arhivu po imenu No Matt's Scripts.

Ipak, jedan od slogana Perlove zajednice je "There's more than one way to do it", a to zapravo može dovesti do toga da se više istih stvari radi na puno različitih načina, zbog čega može doći do nečitljivosti koda.

2.3. Personal Home Page Tools

PHP (eng. Personal Home Page Tools) skriptni jezik bio je također korišten za pisanje CGI skripti, a prvi koji ga je koristio bio je Ramsus Lerdorf 1994. godine.

Prve verzije PHP-a bile su zapravo skup Perl skripti, a kasnije zbog potrebe za većim brojem funkcija razvila se nova verzija u C programskom jeziku.

PHP je bio jedan od prvih jezika na strani poslužitelja koji se mogao pisati direktno u HTML dokumentu, a neke od web aplikacija koje koristimo i danas pisane su baš ovim programskih jezikom (Wordpress, veliki dio Facebook-a i tako dalje…)

Čak i danas puno neiskusnih programera počinju upravo s učenjem PHP-a i upravo zbog toga velika količina ljudi smatra da jezik nije kvalitetan.

2.4. Python

1980. godine Guido van Rossum stvorio je programski jezik po imenu Python. Ovaj objektno-orijentirani jezik omogućava brz razvoj aplikacija. Ipak, bez odgovarajućih web okvira (eng. Framework) Python nije bio dobro prihvaćen jezik i nije se puno koristio.

2003. Django se pojavljuje, ali javnosti postaje dostupan tek 2005. godine. Osim Djanga,jedan od najpopularnijih mikro-okvira koji koriste Python je Flask.

Posebna pažnja u daljnjem nastavku rada posvećena je Djangu, u trećem poglavlju.

2.5. Ruby

Baš kao i Python, Ruby ima čistu sintaksu i objektno-orijentirani dizajn. Ovaj programski jezik, prvi put je objavljen 1995. godine i nije bio pretjerano poznat dok se nije pojavio web programski okvir Ruby on Rails. Ruby on Rails je, baš kao i Django, model-pogled-upravitelj framework i brzo ga prihvaćaju mnogi programeri. Jedna od najpoznatijih platformi napravljena u Ruby-ju je Twitter. Ipak, kako se Twitter razvijao, a Ruby je imao problem s performansama, Twitter je morao neke svoje dijelove prebaciti u Java programski jezik i slične.

2.6. JavaScript

Programski jezik JavaScript izašao je 1995. godine i brzo je dobio na popularnosti, ali nije bio korišten za neke veće projekte, a prvo se pojavio na Netscape Navigatoru. U početku, ovaj se skriptni programski jezik koristio samo za manje dijelove nekih projekata i to zbog svoje sporije brzine. S vremenom, naravno da je i sam jezik napredovao, ali je postao i iznimno popularan pojavom Node.js programskog okvira. Danas, većina web stranica koristi se JavaScriptom i već se dugo vremena JS smatra najpopularnijim programskim jezikom na webu.

3. Web programski okvir Django

Django je jedan od najpoznatijih programskih okvira koji koristi Python kao programski jezik. Simon Willison i Adrian Holovaty 2003. godine su odustali od PHP-a i odlučili razviti Django zbog toga što je tada aktualna novinarska organizacija World Online zahtijevala da se aplikacije izrade u jako kratkim rokovima. Willison i Holovaty počeli su koristiti Python za izradu web stranica i tako su korak po korak kroz dvije godine došli do toga da su razvili web programski okvir Django. World Online organizacija odlučila je da će ovaj web programski okvir objaviti kao otvoreni softver (eng. Open source).

Ime Django inspirirano je poznatim jazz gitaristom Djangom Reinhardtom koji se i dan danas smatra najboljim gitaristom svih vremena.

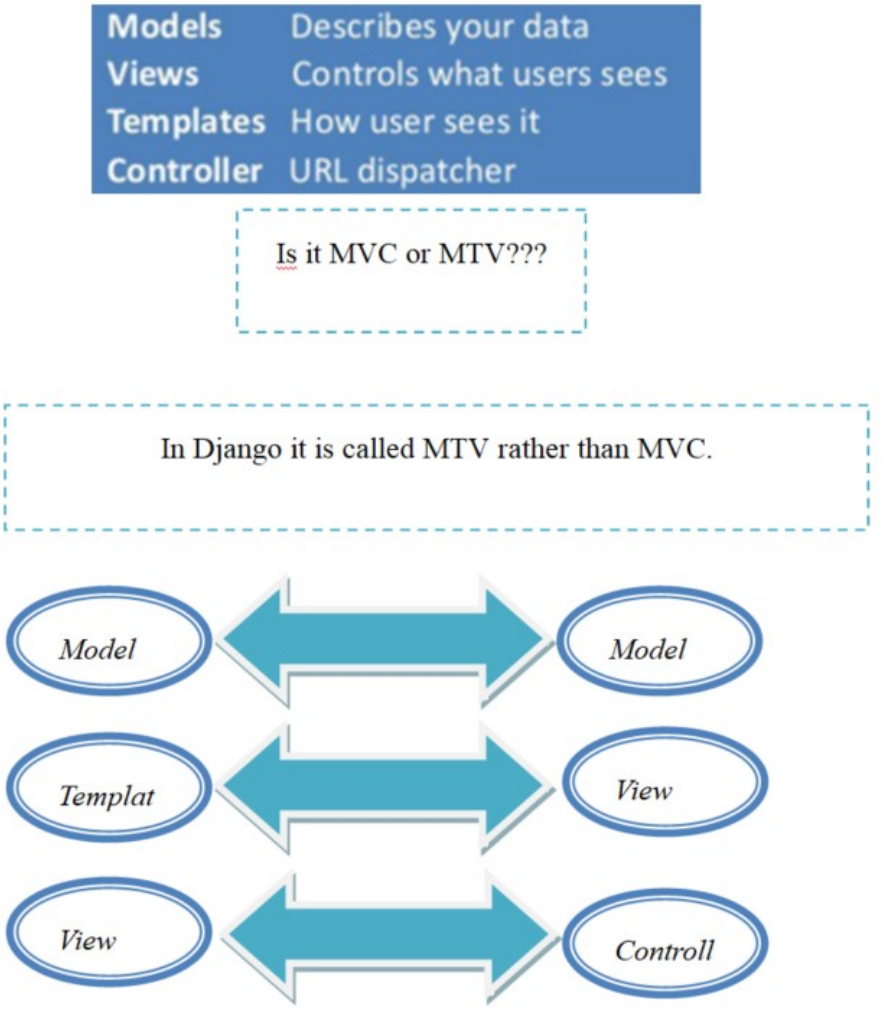
Neke od poznatih kompanija koje koriste Django za razvoj svojih aplikacija su Instagram, Pinterest, Udemy, Spotify. Za Instagram još uvijek nije sigurno koristi Django, ali se svakako koristi Python. Treba se istaknuti da je sigurno da je Instagram koristio Django i tada je već prešao broj od milijun korisnika svoje aplikacije.

Na Statista web stranici, Django je deveti na listi najkorištenijih web okvira na svijetu ove godine, a ono što svakako moramo spomenuti je to nam omogućava tri jako bitne stvari kod izrade dinamičkih web aplikacija, što uključuje:

- brz razvoj aplikacije,
- sigurnost,
- skalabilnost.

Django je Model-Pogled-Upravitelj (eng. Model-View-Controller, MVC) web programski okvir, a to znači da su dijelovi aplikacije odvojeni u komponente ovisno o njihovim funkcijama.

Django Model-Template-Controller prikazan je Shemom 2.



Shema 2: Django Model-Template-Controller

3.1. Model (eng. Model)

Modeli u Djangu definiraju se kao klase (eng. Class), a predstavljaju ono što će se nalaziti u bazi podataka. Svaka klasa predstavlja tablicu u bazi podataka, a varijable predstavljaju kolumne u tablicama i pomoću njih se definiraju veze.

3.2. Pogled (eng. View)

U Djangu datoteci view se definiraju funkcije pomoću kojih se dohvaćaju podaci iz baze podataka, te se nakon toga prosljeđuju u obrasce koji će se prikazati korisniku. Za svaki obrazac potrebno je napisati funkciju.

3.3. Obrazac (eng. Template)

Pogled zapravo predstavlja obrazac (eng. Template). U aplikaciji se stvara mapa pod nazivom Template i u njoj se kreiraju HTML dokumenti. Na svakoj od stranica se prikazuju podaci ovisno o funkcijama koje smo definirali u view datoteci i zahtjevima koje je poslao korisnik. U obrascima ne može koristiti standardno upisivanje naredbi u Pythonu, jer su to ipak HTML datoteke, ali zato Django ima svoj jezik predložaka. Ovo omogućuje lakše određivanje koje će podatke i na koji način prikazati. Obrazac je prikazan Tablicom 1.

Ime elementa	**Sintaksa**	**Primjer**
Varijabla	{{ varijabla }}	{{ ime }} – daje vrijednost ime
Filteri	{{ varijabla|filter }}	{{ ime|lower }} – vraća vrijednost imena malim slovima
Oznaka	{% oznaka %}	{% for n in narudzba %} <p>Ispisi<p> {% endfor %} -za svaku narudžbu u listi narudžbi ispiši paragraf

Tablica 1: Obrazac

4. Izrada web aplikacije u Djangu

4.1. Instalacija

Ovu sam aplikaciju izrađivala u Visual Studio Code editoru, ali inače se može koristiti i bilo koji drugi programski editor. Kako koristim Windows, najbolja opcija bila je spajanje na WSL i instaliranje Pythona i Djanga u takvom okruženju. Inače, Python je programski jezik koji se koristi na Linuxu pa ga tako i u ovom slučaju treba instalirati na bilo kakvom Ubuntu sustavu.

Naredbu za instalaciju Pythona:

```
$ sudo apt install python3 python3-pip python-is-python3 pylint
```

Naredba za instalaciju Djanga:

```
$ pip3 install Django
```

Nakon što su Python i Django instalirani, možemo započeti izradu aplikacije.

4.2. Stvaranje projekta i aplikacije

Da bismo započeli s izradom aplikacije, trebamo prvo kreirati projekt. Naredbu možemo poslati preko terminala u Visual Studio Code-u. Projekt ćemo u ovom slučaju nazvati "mishka":

```
$ django-admin startproject mishka
```

Ovom naredbom kreirali smo novi direktorij u kojemu su se kreirale datoteke manage.py i mishka. Mishka je naš središnji direktorij čije se datoteke tretiraju kao Python package, a manage.py je datoteka koju koristimo u radu s terminalom, na primjer kod pokretanja servera, izvršavanja migracija kao i kod kreiranja aplikacije.

Naredbu za kreiranje aplikacije koju ćemo u ovom slučaju nazvati "main" šaljemo tako da se pozicioniramo unutar projekta i pošaljemo naredbu:

```
$ ./manage.py startapp main
```

4.3. Postavke aplikacije

Prva stvar koju trebamo napraviti nakon kreiranja aplikacije je dodati ju na listu instaliranih aplikaciju u settings.py koji se nalazi u našem središnjem direktoriju – mishka.

Postavke aplikacije prikazane su Slikom 1.

```
mishka > settings.py > ...
28  ALLOWED_HOSTS = []
29
30
31  # Application definition
32
33  INSTALLED_APPS = [
34      'django.contrib.admin',
35      'django.contrib.auth',
36      'django.contrib.contenttypes',
37      'django.contrib.sessions',
38      'django.contrib.messages',
39      'django.contrib.staticfiles',
40      #my
41      'main.apps.MainConfig',
42      'main.funkcije',
43      #django_allauth
44      'django.contrib.sites',
45      'allauth',
46      'allauth.account',
47      'allauth.socialaccount',
48
49  ]
50  LOGIN_REDIRECT_URL = '/narucise'
51  SITE_ID = 1
```

Slika 1: Postavke aplikacije

Na primjeru u kodu možemo vidjeti sve instalirane aplikacije koje sam koristila u izradi svoje aplikacije. Za početak, bilo je dovoljno na listu dodati MainConfig aplikaciju, a kasnije s korištenjem nekih dodatnih funkcija i paketa dodavala sam na listu aplikacije koje su mi bile potrebne.

Elementi u ovoj listi pišu se pod navodnicima jer su tipa string i odvajaju se zarezom.

4.4. Modeli

Kao što smo već napomenuti u samom radu, modeli u Djangu definiraju se kao klase unutar kojih se nalaze atributi odnosno polja (eng. field). Svako polje mora imati definirani tip podatka, a neki od njih zahtijevaju i dodatne argumente.

Django za svaku definirano klasu, automatski dodaje polje id i postavlja ga kao primarni ključ tako da to ne moramo raditi samostalno. Ipak, ako odlučimo da ćemo primarni ključ definirati sami, definiramo ga tako da za polje koje smo odabrali, dodamo argument primary_key=True. Ako to napravimo, Django neće sam definirati polje id.

U klasama se definiraju i veze među tablicama, a one se dijele na:

- Many to One
- Many to Many
- One to One

Za izradu ove aplikacije, ja sam morala definirati dvije klase. Klasa Usluga sadržavala je informacije o uslugama koje salon nudi kao i cijenama istih, a klasa Narudzba informacije o narudžbama.

Na Slici 2. vidljivo je kako su definirani ovi modeli i koje sam tipove podataka koristila.

main > models.py

```python
from django.utils.timezone import datetime
from django.db import models
from django.contrib.auth.models import User
from django.forms.fields import DateField
from django.utils import timezone
from datetime import timedelta
from django.core.exceptions import ValidationError


# Create your models here.

class Usluga(models.Model):
    ime_usluge = models.CharField(max_length=30)
    cijena = models.DecimalField(max_digits=5, decimal_places=2)

    def __str__(self):
        return self.ime_usluge


class Narudzba(models.Model):
    user = models.ForeignKey(User, on_delete=models.CASCADE)
    za_uslugu = models.ForeignKey(Usluga, on_delete=models.CASCADE)
    datum = models.DateField (default=datetime.today())
    pocetak = models.TimeField (default=datetime.now())
    kraj = models.TimeField (default=datetime.now())
    cijena_nar = models.DecimalField(
        max_digits=5, decimal_places=2, default=00.00)

    def __str__(self):
        return f'{self.user} naručuje {self.za_uslugu} od {self.pocetak} do {self.kraj}'

```

Slika 2: Modeli

Kao što vidimo, ova dva modela povezana su preko klase Narudžba u kojoj atribut za_uslugu predstavlja vanjski ključ koji je povezan s primarnim ključem usluge. Kod definiranja vanjskih ključeva moramo definirati što se događa kada se element iz glavne tablice obriše. U ovom slučaju

to sam definirala kao CASCADE što znači da kada bi se obrisala određena usluga, obrisale bi se i sve narudžbe koje su bile vezane na tu uslugu.

Vidimo da je u svakoj klasi definirana i funkcija __str__ u kojoj definiramo što će se biti izlazni podatak ako pozovemo ovaj model.

Za klasu usluga definirala sam da će izlazni podatak biti samo naziv usluge, a za klasu narudžba prikazati će se rečenica. Klasu korisnika nisam definirala samostalno nego sam umjesto toga iz django.contrib.auth modela učitala Djangov model Korisnika (eng. User).

Bilo kakva promjena na modelima zahtjeva izvršavanje migracija koje služe da se promjene na modelima pohrane u sustav.

Naredbe:

```
$ ./manage.py makemigrations
$ ./manage.py migrate
```

4.5. Admin portal

Django ima svoj Admin portal (eng. Admin site) koji olakšava popunjavanje baze podataka korisnicima kojima je ovdje omogućen pristup. Admin portal pomoću modela koje smo definirali može lako stvoriti bazu podataka kao i raditi s istom.

Kako bismo pristupili Admin portalu, moramo kreirati superusera naredbom u terminalu:

```
$ python manage.py createsuperuser
```

Ovom naredbom otvaraju se polja koja moramo ispuniti u terminalu. Prvo upišemo korisničko ime (eng. Username), zatim Email (nije obavezno polje i može ostati prazno) i na kraju upisujemo lozinku i potvrdu lozinke.

Kako bismo na admin portalu mogli vidjeti modele koje smo kreirali pa čak i upisati nešto u bazu, moramo ih uključiti u datoteku admin.py koja se nalazi unutar main aplikacije. Admin portal prikazan je Slikom 3.

```
main > admin.py > ...
1  from django.contrib import admin
2  from .models import *
3
4  # Register your models here.
5  model_list = [Usluga, Narudzba]
6  admin.site.register(model_list)
```

Slika 3: Admin portal

4.6. Registracija i prijava

Za registraciju i prijavu korisnika, instalirala sam django_allauth paket. Ovaj paket jedan je od najpoznatijih i najkorištenijih Django paketa. Njegova instalacija i implementacija je vrlo jednostavna, a upute se lako mogu pronaći u dokumentaciji.

Osim same prijave i registracije sadrži i obrasce za promjenu lozinke kao i mnogo drugih mogućnosti koje ubrzavaju proces izrade aplikacije.

Registracija je prikazana Slikom 4.

Slika 4: Registracija

4.7. Pogledi i povezivanje URL

Nakon upisanih nekoliko usluga i narudžbi u bazu podataka preko admin portala, moj sljedeći korak bio je napraviti prikaz iz baze podataka. To sam napravila tako da sam učitala ListView i definirala klase UslugaList i NarudzbaList. Na ovaj način mogla sam lako pozvati kreirane modele i prikazati podatke iz baze bez pisanja puno koda. ListView je klasa koja izumljena baš iz razloga kako bi programerima olakšala prikaz liste objekata i vrlo je jednostavna za korištenje.

Na sljedećoj slici možemo vidjeti kako u kodu izgledaju ove klase.

U klasi NarudzbaList osim samog poziva modela definirana je i funkcija koja sprema određene objekte u listu, ovisno o tome je li prijavljeni korisnik zaposlen u salonu ili je samo klijent.

ListViews prikazan je Slikom 5.

main > views.py > NarudzbaView > form_valid

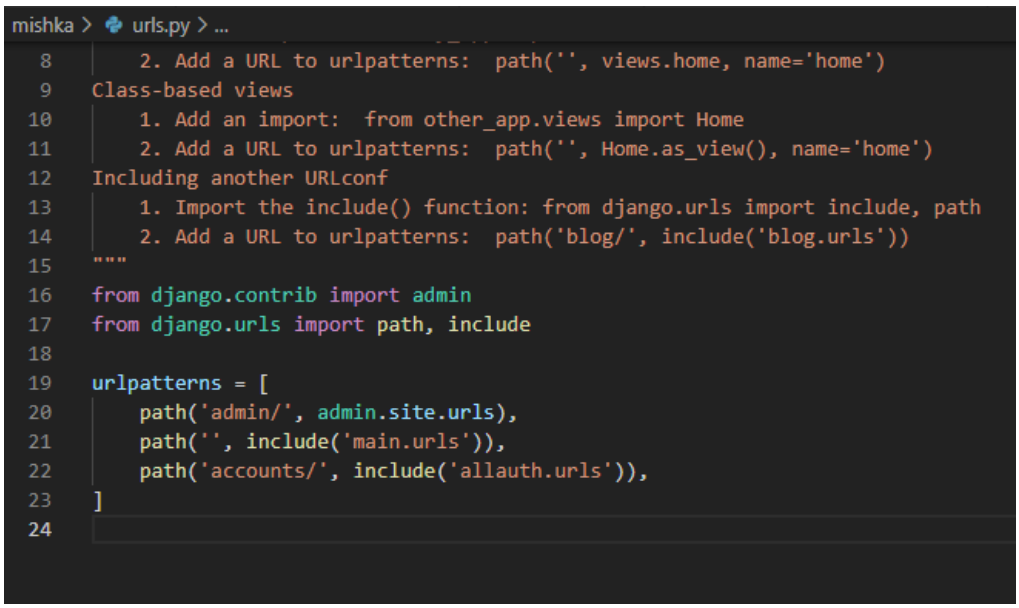
```python
from django.contrib.auth.models import AnonymousUser
from django.shortcuts import redirect, render, HttpResponse
from django.views.generic import ListView, FormView
from .models import *
from .forms import SlanjeNarudzbeForm, PisanjeNarudzbeForm, DodajUsluguForm
from urllib import request
from main.funkcije.provjeri_termin import provjeri_termin
import datetime
epoch = datetime.datetime(1970, 1, 1)
# Create your views here.


class UslugaList(ListView):
    model = Usluga


class NarudzbaList(ListView):
    model = Narudzba

    def get_queryset(self, *args, **kwargs):
        """[Funkcija koja određuje što će se prikazati na listi narudzbi]

        Returns:
            [list]: [U listu se spremaju narudzbe ovisno o tome je li korisnik zaposlenik ili nije]
        """
        if self.request.user.is_staff:
            narudzba_list = Narudzba.objects.all().order_by('datum')
            return narudzba_list
        else:
            narudzba_list = Narudzba.objects.filter(
                user=self.request.user).order_by('datum')
            #narudzba_cijena = 0
            # for narudzba in narudzba_list:
            #    narudzba_cijena += narudzba.cijena_nar
            # print(narudzba_cijena)
            return narudzba_list

```

Slika 5: ListViews

Sada napokon možemo kreirati mapu templates u kojoj ćemo napraviti HTML datoteke za prikaz podataka. Ovu mapu također kreiramo unutar main aplikacije.

Prije nego pređemo na kreiranje obrazaca za prikaz podataka, prisjetit ćemo se da u views.py definiramo sve funkcije koje će na naše zahtjeve dohvaćati i prikazivati obrasce. U sljedećim poglavljima dotaknut ćemo se ostalih funkcija i klasa koje se nalaze u views.py, ali prvo ćemo prikazati kako povezati URL-ove s ovim definiranim funkcijama.

Slikom 6. prikazan je urls.py.



Slika 6: urls.py

U mishka direktoriju već imamo urls.py u kojem smo dodali putanju (eng. path) za django_allauth, a moramo ju usmjeriti i na urls.py naše main aplikacije kako bismo dohvatili sve ono što se nalazi u views.py.

Urls.py datoteku u main aplikaciji moramo kreirati sami, a ona će naposljetku, kada budu uključeni svi putevi, što je prikazano Slikom 7.

Slika 7: Urls.py datoteku u main aplikaciji

Urlpatterns je lista tako da svi elementi moraju biti odvojeni zarezom. Path je funkcija koja usmjerava url-ove na funkcije definirane u pogledu. Name možemo, a i ne moramo pisati.

4.8. Upis u bazu podataka

Kako bi korisnik mogao rezervirati termin u aplikaciji moramo mu napraviti formu za slanje narudžbe. Sve forme za upise u bazu definiraju se u forms.py kojeg kreiramo u main-u.

Forme koje sam definirala u ovoj aplikaciji možemo pogledati na Slici 8.

main > forms.py > SlanjeNarudzbeForm

```python
from datetime import timedelta
from django.utils import timezone
from django import forms
from django.forms.fields import DateTimeField
from main.models import Narudzba, Usluga, User

class SlanjeNarudzbeForm(forms.Form):
    services = Usluga.objects.all().values_list('id', 'ime_usluge')
    za_uslugu = forms.ChoiceField(choices=services)
    datum = forms.DateField(
        required=True, input_formats=["%Y-%m-%d", ])
    pocetak = forms.TimeField(
        required=True, input_formats=["%H:%M", ])
    kraj = forms.TimeField(
        required=True, input_formats=["%H:%M", ])


class PisanjeNarudzbeForm(forms.Form):
    users = User.objects.all().values_list('id', 'username')
    services = Usluga.objects.all().values_list('id', 'ime_usluge')
    user = forms.ChoiceField(choices=users)
    za_uslugu = forms.ChoiceField(choices=services)
    datum = forms.DateField(
        required=True, input_formats=["%Y-%m-%d", ])
    pocetak = forms.TimeField(
        required=True, input_formats=["%H:%M", ])
    kraj = forms.TimeField(
        required=True, input_formats=["%H:%M", ])

class DodajUsluguForm(forms.Form):
    ime_usluge = forms.CharField(max_length=30)
    cijena = forms.DecimalField(max_digits=5, decimal_places=2)
```

Slika 8: forms.py

Forma SlanjeNarudzbeForm je forma koju ispunjavaju klijentice i ona od klijentice traži da odabere uslugu za koju će se naručiti, datum, početak i kraj.

Forma PisanjeNarudzeForm zapravo je vrlo slična prošloj formi. Jedina razlika je u tome da je ova namijenjena za zaposlenike salona pa tako osim već spomenutih podataka, ovdje zaposlenik odabire i jednog od korisnika za kojeg kreira narudžbu.

Treća na listi je DodajUsluguForm koja omogućava zaposlenicima salona da dodaju nove usluge i njihove cijene.

Osim formi, za upis narudžbe u bazu, definirana je funkcija koja provjerava je li zatraženi termin već zauzet. Funkcija se zove provjera_termina i prikazana je Slikom 9.

main > funkcije > provjeri_termin.py > ...

```python
from main.models import Usluga, Narudzba
import datetime


def provjeri_termin(datum, pocetak, kraj):
    """[Funkcija provjerava je li zatraženi termin već zauzet.]

    Args:
        datum ([date]): [Datum termina]
        pocetak ([Time]): [Pocetak termina]
        kraj ([Time]): [Kraj termina]

    Returns:
        [bool]: [Funkcija all vratit će nam False ako se zatraženi termin nije upisao na listu - već je bio zauzet]
    """
    lista_termina = []
    lista_narudzbi = Narudzba.objects.all()
    #print(lista_narudzbi, pocetak, kraj)
    for narudzba in lista_narudzbi:
        #print("\n", narudzba, "\n   ")
        if datum==narudzba.datum:
            if kraj < narudzba.pocetak or pocetak > narudzba.kraj:
                lista_termina.append(True)
            else:
                lista_termina.append(False)
        else:
            lista_termina.append(True)
    #print(lista_termina)
    return all(lista_termina)

```

Slika 9: Funkcija provjeri_termin

Funkcija provjeri_temin u listu lista_termina prvo sprema sve narudžbe. Zatim za svaku narudžbu izvodi usporedbu datuma postojeće narudžbe s datumom kojeg je korisnik poslao preko forme. Ako takav datum već postoji, izvodi se sljedeća provjera koja uspoređuje odabrano vrijeme početka i kraja s početkom i krajem te narudžbe. Ako se zatraženi datum ili vrijeme poklapa s onim koje već postoji u bazi u listu lista_termina sprema se vrijednost False, a ako ne sprema se True. Funkcija vraća boolean ovisno o tome što se nalazi u listi. Ako se zatraženi termin poklopio i sa samo jednom narudžbom i postoji jedan False u listi, funkcija all vratit će vrijednost False što znači da termin nije slobodan. Ako su sve vrijednosti u listi True, funkcija će vratiti True vrijednost.

U views.py definirani su pogledi koji pozivaju ove forme, i šalju podatke na html obrasce. Na sljedećoj slici možemo vidjeti kako u views.py izgleda pozivanje forme, funkcije te na koji se način uneseni podaci šalju, provjeravaju i spremaju u bazu. Provjera forme za upisivanje narudžbe prikazana je Slikom 10.

```python
class NarudzbaView(FormView):
    form_class = SlanjeNarudzbeForm
    template_name = 'slanje_narudzbe_form.html'


    def form_valid(self, form):
        data = form.cleaned_data
        slobodni_termin = provjeri_termin(data['datum'],data['pocetak'], data['kraj'])
        if slobodni_termin:
            cijenan = Usluga.objects.get(id=data['za_uslugu'])
            cijena_nar = cijenan.cijena
            datum_p=datetime.date(1,1,1)
            pocetak_p = datetime.datetime.combine(datum_p, data["pocetak"])
            kraj_p = datetime.datetime.combine(datum_p, data["kraj"])
            razlika = kraj_p-pocetak_p
            if razlika.total_seconds()>=7200:
                narudzba = Narudzba.objects.create(
                    user=self.request.user,
                    za_uslugu=Usluga.objects.get(id=data['za_uslugu']),
                    datum=data['datum'],
                    pocetak=data['pocetak'],
                    kraj=data['kraj'],
                    cijena_nar=cijena_nar
                )
                narudzba.save()
                return redirect('/narudzbe')
            else:
                return HttpResponse('Kraj mora biti minimalno 2 sata nakon početka.', status=400)
        else:
            return HttpResponse('Termin koji ste odabrali je zauzet.', status=400)

```

Slika 10: Provjera forme za upisivanje narudžbe

U slobodni_termin sprema se boolean vrijednost koju je vratila funkcija provjeri_termin.

Ako je funkcija vratila True, i slobodni_termin ima vrijednost True, ovdje provodimo provjeru o tome kolika je razlika između vremena koje je odabrano za početak i za kraj te korisniku dajemo mogućnost da upiše samo termine koji nisu kraći od dva sata. Ako je i ova provjera zadovoljena kreira se nova narudžba u koju se spremaju svi podaci poslani preko forme. Nakon kreiranja, narudzbu spremamo i korisnika preusmjeravamo na obrazac koji prikazuje listu narudzbi. Ako neka od provjera nije dobro prošla, korisniku vraćamo HTTP odgovor s odgovarajućom porukom i statusom requesta.

Slanje narudžbe klijentice prikazano je Slikom 11.

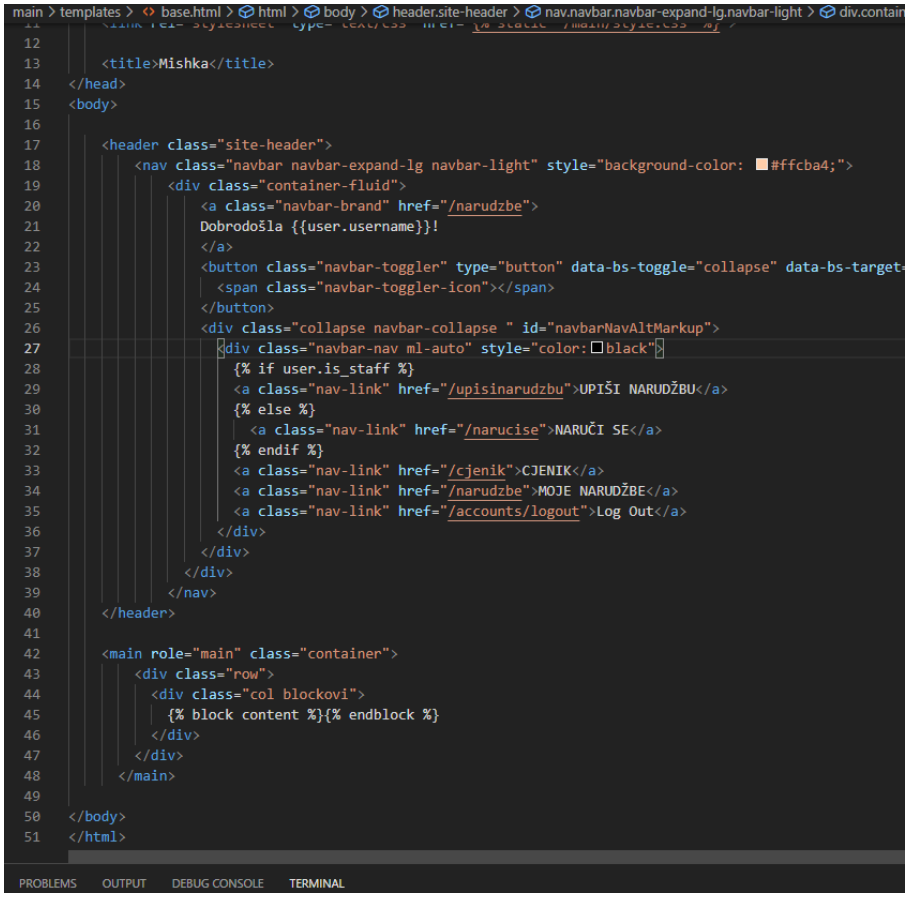
Slika 11: Slanje narudžbe klijentice

4.9. Izrada obrazaca

Pravilan način za izradu obrazaca je taj da imamo barem jedan bazni html u kojem definiramo što će se prikazivati na svim stranicama. U ovoj aplikaciji, ja sam napravila dva bazna HTML dokumenta. Jedan je služio za prikaz stranice dok korisnik još nije prijavljen, a drugi za prijavljene korisnike.

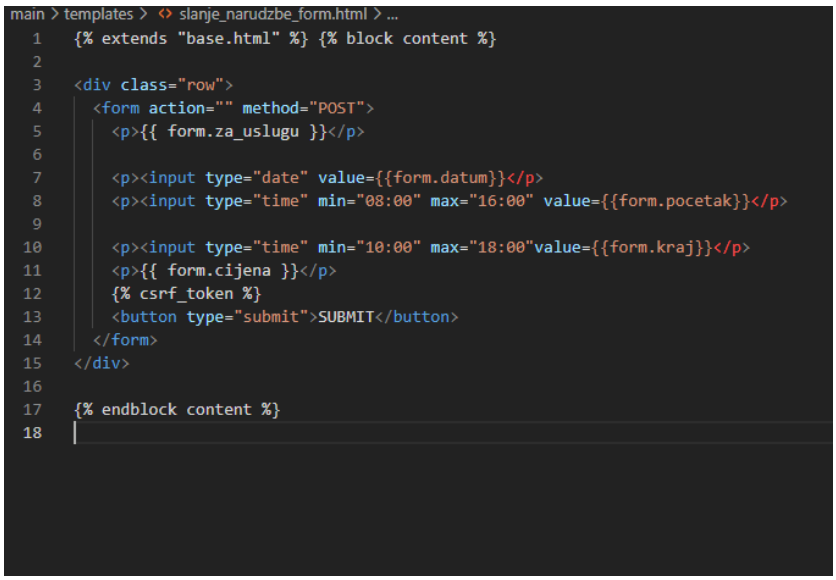
U baznom HTML dokumentu obično izradimo navigaciju i dodamo dizajn, a onda u ostalim obrascima radimo samo razlike ovisno o tome što želimo prikazati korisniku.

Na Slici 12. prikazan je izgled dijela base.html-a.



Slika 12: base.html

U prvom dijelu napravljena je navigacija (korišten Bootstrap v5.0). Ovaj se obrazac nikada ne prikazuje takav kakav je i služi samo kao baza za prikaz ostalih pogleda. To radimo tako da na mjesto gdje želimo da se prikazuje sadržaj ostalih stranica napravimo prostor za sadržaj., ali moramo i definirati kraj tog prostora. Base.html proširujemo načinom prikazanim Slikom 13.



Slika 13: Slanje narudzbe.html

4.10. Ostali obrasci i korištene funkcije

Sada ćemo proći ostale funkcije definirane u views.py. Prva funkcija je funkcija home u kojoj se provjerava je li korisnik prijavljen kada dolazi na početnu stranicu. Ako korisnik nije prijavljen, prikazuje se stranica koje mu daje mogućnost da se registrira ili prijavi. Za prijavljene korisnike vrši se provjera je li korisnik zaposlenik ili klijent salona, a ovisno o tome prikazuje mu se forma za slanje ili upisivanje narudžbe. Slikom 14. prikazana je home funkcija.

```
def home(request):
    """[Funkcija koja definira što će biti početna stranica.]

    Returns:
        [html]: []
    """
    print(request.user)
    print(type(request))
    if request.user.is_authenticated:
        if request.user.is_staff:
            return render(request, 'pisanje_narudzbe_form.html')
        else:
            return render(request, 'slanje_narudzbe_form.html')
    else:
        return render(request, 'main/pocetna.html')
```

Slika 14: Home funkcija

Početna stranica prikazana je Slikom 15.

Slika 15: Početna stranica

Još jedna od funkcija koje korisnik ima je brisanje narudžbe iz baze. Slikom 16. je prikazano kako zapravo na stranici izgleda lista narudžbi.

Slika 16: Lista narudžbi

Korisnica antonela.troha ima status zaposlenika pa u listi vidi narudžbe svih klijentica i ima mogućnost brisanja svake narudžbe. Klijentice vide samo svoje narudžbe pa tako samo njih mogu i brisati. Funkcija za brisanje prikazana je Slikom 17.

```
def otkazivanje(request, narudzba_id=None):
    """[brisanje narudzbi]

    Args:
        narudzba_id ([int], optional): [u narudzba_id prosljeđuje se id narudzbe koju brisemo]. Defaults to None.

    Returns:
        [html]: [vraćamo se na listu narudzbi]
    """

    object = Narudzba.objects.get(id=narudzba_id)
    object.delete()
    return render(request, 'main/narudzba_list.html')
```

Slika 17: Otkazivanje narudžbe

5. Zaključak

U današnje vrijeme tehnologija je posvuda, postala je gotovo nezamisliva u svakodnevici svakog čovjeka, a zbog užurbanog načina života i nedostatka vremena konstantno se razvija potreba da nam tehnologija pomogne u bržem i lakšem rješavanju svakodnevnih problema. Web aplikacije postaju sve popularnije jer im se lako pristupa preko web preglednika. Zbog svega navedenoga, javlja se potrebe da se takve aplikacije razvijaju brže i jednostavnije uz što je manje moguće pisanja koda.

Iako je Python jezik za koji se kaže da ima spor interpreter to nije bitno jer je to jezik koji je lak za naučiti, besplatan je i otvorenog koda što znači da je dostupan svima.

Django programski okvir jedan je od popularnijih upravo zbog brzog i sigurnog načina izrade web aplikacija. Kroz izradu web aplikacije za rezerviranje termina u salonu za uljepšavanje uvjerila sam se da je Django stvarno razvojno okruženje koje ima mnogo već implementiranih gotovih rješenja kao i mogućnost instalacije dodatnih paketa kako bi razvoj web aplikacija bio brži i jednostavniji.

Literatura

[1] https://doriantaylor.com/a-brief-and-fuzzy-history-of-web-application-development, preuzeto 14.7.2021.

[2] https://dzone.com/articles/5-facts-to-know-about-nodejs-development, preuzeto 14.7.2021.

[3] https://en.wikipedia.org/wiki/Perl, preuzeto 28.7.2021.

[4] https://group.miletic.net/hr/nastava/kolegiji/DWA2/, preuzeto 6.8.2021.

[5] https://home.cern/science/computing/birth-web/short-history-web, preuzeto 6.8.2021.

[6] https://hotframeworks.com/, preuzeto 28.7.2021.

[7] https://hr.birmiss.com/sto-je-ruby-programski-jezik-ruby/, preuzeto 6.8.2021.

[8] https://hr.wikipedia.org/wiki/Perl, preuzeto 6.8.2021.

[9] https://medium.com/shecodeafrica/understanding-the-mvc-pattern-in-django-edda05b9f43f, preuzeto 6.8.2021.

[10] https://opensource.com/life/16/11/perl-and-birth-dynamic-web, preuzeto 6.8.2021.

[11] https://webdevelopmenthistory.com/1993-cgi-scripts-and-early-server-side-web-programming/, preuzeto 28.7.2021.

[12] https://webdevelopmenthistory.com/1995-php-quietly-launches-as-a-cgi-scripts-toolset/. preuzeto 28.8.2021.

[13] https://www.djangoproject.com/, preuzeto 28.8.2021.

[14] https://www.djangoproject.com/start/overview/, preuzeto 28.7.2021.

Popis slika

Popis tablica