

# Izrada web aplikacije za digitalnu kuharicu

---

**Fabac, Ema**

**Undergraduate thesis / Završni rad**

**2024**

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

*Permanent link / Trajna poveznica:* <https://um.nsk.hr/um:nbn:hr:195:439690>

*Rights / Prava:* [In copyright](#)/[Zaštićeno autorskim pravom.](#)

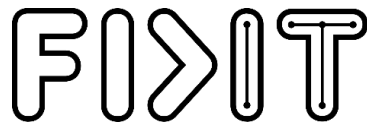
*Download date / Datum preuzimanja:* **2025-03-14**



*Repository / Repozitorij:*

[Repository of the University of Rijeka, Faculty of Informatics and Digital Technologies - INFORI Repository](#)





Sveučilište u Rijeci

**Fakultet informatike  
i digitalnih tehnologija**

Sveučilišni prijediplomski studij Informatika

Ema Fabac

# Izrada web aplikacije za digitalnu kuharicu

Završni rad

Mentor: doc. dr. sc. Martina Ašenbrener Katić

Rijeka, srpanj 2024.

Rijeka, 03. svibnja 2024.

## Zadatak za završni rad

Pristupnica: Ema Fabac

Naziv završnog rada: Izrada web aplikacije za digitalnu kuharicu

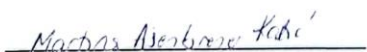
Naziv završnog rada na engleskom jeziku: Development of a web application for a digital cookbook

### Sadržaj zadatka:

Zadatak završnog rada je opisati postupak planiranja i stvaranja aplikacije za digitalnu kuharicu. Aplikacija će imati funkcionalnosti kao što su pretraživanje recepata po sastojcima, prijavu korisnika u sustav, objavljivanje/brisanje vlastitih recepata i ostavljanje recenzija za prijavljene korisnike, te kategorizacija recepata. Procesu izrade aplikacije prethodit će analiza sustava, definiranje korisničkih zahtjeva, oblikovanje modela podataka i izrada persona. Potrebno je aplikaciju izraditi koristeći razvojni okvir Django te opisati postupak izrade aplikacije.

Mentorica  
Doc. dr. sc. Martina Ašenbrener Katić

Voditelj za završne radove  
Izv. prof. dr. sc. Miran Pobar

  
\_\_\_\_\_

  
\_\_\_\_\_

Zadatak preuzet: 03.05.2024.

  
\_\_\_\_\_  
(potpis pristupnice)

## **Sažetak**

U završnom radu opisan je postupak izrade aplikacije za pronalazak recepata. Aplikacija je izrađena u Django web okviru zasnovanom na Pythonu. Prije same izrade aplikacije opisan je postupak korištenja web aplikacije te po opisu izrađen je model podataka na kojim se temelji web aplikaciji. Osnovna funkcionalnost aplikacije je pronalazak recepata odabirom željenih sastojaka. U radu je detaljno opisan model podataka, izrađene su 3 persone, definirani korisnički zahtjev i opisan proces izrade aplikacije.

Ključne riječi: Django, recept, model, sastojak, filter

# SADRŽAJ

1. Uvod .....	1
2. Persone.....	2
3. Korisničke priče .....	4
4. Model podataka .....	5
5. Izrada aplikacije.....	7
5.1. Stvaranje projekta.....	7
5.2. Postavljanje aplikacije.....	8
5.3. Modeli .....	9
5.4. Spajanje na bazu pomoću admin portala.....	13
5.5. Navigacija .....	14
5.6. Prijava i registracija .....	17
5.7. Početna .....	24
5.8. Detalji o receptu .....	34
5.9. Profil.....	49
5.10. Unos recepta.....	56
6. Zaključak .....	67
7. Literatura.....	68
8. Popis slika .....	69

# 1. Uvod

Zadatak ovog završnog rada je izrada web aplikacije za recepte u Django i opisati postupak izrade aplikacije. Django je Python web framework visoke razine koji omogućuje brz razvoj sigurnih web aplikacija koje se mogu održavati. Django je izvorno razvijen između 2003. i 2005. godine od strane web tim koji je bio odgovoran za izradu i održavanje novinskih web stranica. Nakon izrade niz web-mjesta, tim je počeo izdvajati i ponovno koristiti mnoge uobičajene kodove i obrasce dizajna. Ovaj zajednički kod razvio se u generički okvir za web razvoj, koji je bio otvoren kao projekt "Django" u srpnju 2005.godine, te je nastavio rasti i poboljšavati se [1]. Django upravlja kodom koristeći Model-View-Template (MVT) arhitekturu koja je alternativa MVC arhitekture. MVT arhitektura radi tako da modeli djeluju kao sučelje između baze podataka i poslužiteljskog koda, odnosno preslikava svaki model u jednu tablicu baze podataka, potom pogledi obrađuju zahtjev korištenjem modela, odnosno uzimaju zahtjev korisnika i vraćaju mu odgovor, te predložak upravlja prezentacijom web stranice u pregledniku [2].

Iako postoji mnogo web aplikacija s receptima, često se susrećemo s ograničenjima. Primjerice, ako imamo samo nekoliko sastojaka kod kuće, teško je pronaći odgovarajuće recepte. Također, mnoge mrežne stranice nemaju kategorizirane recepte, što otežava pretraživanje. Nadalje, mnoge mrežne stranice ne pružaju mogućnost korisnicima da objave vlastite recepte. Ove probleme želi se riješiti izradom web aplikacije u Django. Takva web aplikacija imala bi bazu recepata podijeljenu u kategorije, te bi korisnici mogli filtrirati recepte prema dostupnim sastojcima. Dodatno, korisnici bi se mogli prijaviti i ostaviti recenzije na recepte, te dodati vlastite recepte. Time bi se omogućilo korisnicima da podijele svoje recepte s drugima.

Završni rad je podijeljen u četiri glavna poglavlja. U prvom poglavlju opisani su osnovni koncepti modela podataka te prikazan je model podataka za web aplikaciju za pretraživanje recepata te su opisani svi entiteti modela kao i njihove veze. U drugom poglavlju opisana je funkcija persona te su priložene slike tri persone. U trećem poglavlju opisana je funkcija korisničkih priča te je navedeno sedam korisničkih priča, koje prikazuju scenarije korištenja aplikacije. U četvrtom poglavlju opisuje se cjelokupna izrada aplikacije, koja je podijeljena na deset potpoglavlja.

## 2. Persone

Persona je fiktivna osoba koja predstavlja različite tipove korisnika koji će koristiti određenu uslugu, proizvod, stranicu ili brend na sličan način. Kreiranje persona olakšava proizvođaču usluge da shvati potrebe, iskustva, ponašanja i ciljeve njegovih korisnika te da dobiju uvid da različite osobe imaju različite potrebe i očekivanja. Dobro definirane persone trebale bi predstavljati glavne značajke skupine korisnika, opisuju osobe sa stvarnim vrijednostima i ciljevima, izražavaju glavne potrebe i očekivanja korisnika, daju jasnu sliku očekivanja korisnika i kakvu vrtu interakcije očekuju te pružaju pomoć pri definiranju glavnih značajki i funkcionalnosti usluge [3].

Svaka persona trebala bi se sastojati od demografskih pokazatelja kao što su fiktivno ime, prezime, fotografija, zanimanje, prebivalište, starost i spol. Potrebno je imati i neku tipičnu izjavu te biografiju da saznamo kako persona provodi slobodno vrijeme, koji su njezini hobiji i osobnosti. Sljedeće što treba definirati su ciljevi koji mogu biti neki generalni ciljevi persone ili ciljevi vezani uz sami proizvod. Potrebno je definirati i frustracije kako bi se saznalo što smeta personi u svakodnevnom životu i stvara joj frustracije. Također definiraju se i scenariji kako bi se saznalo što persona želi moći učiniti u proizvodu ili koje informacije bi htjela dobiti. Te posljednja stvar koja se definira su važnosti kod proizvoda za osobu, odnosno je li mu važniji lijepi dizajn ili da brzo i efikasno pronađe informacije koje želi i sl.

Za potrebe izrade aplikacije 'Home Chef' koja se ponaša kao baza recepata koju je dodatno moguće pretraživati po odabiru sastojaka ili kategorija i daje mogućnost unosa vlastitog recepta definirane su tri persone. Svaka od persona predstavlja različitog korisnika i zahtjeva različite potrebe od aplikacije. Na Slici 1., Slici 2. i Slici 3. nalazi se prikaz tri definiranih persona sa svim njezinim pripadajućim informacijama kako bi se izradila kvalitetna aplikacija.



Slika 1. Prikaz prve persone



### Emili Habek

**Dob:** 60 godine  
**Spol:** Ženski  
**Lokacija:** Rijeka, Hrvatska  
**Zanimanje:** Doktor

“

Zdravlje započinje u kuhinji!

#### Biografija

Emili je osoba kojoj je prioritet dobra prehrana i održavanje uravnoteženog načina života. Aktivno traži recepte koji odgovaraju njezinim prehranbenim preferencijama. Emili cijeni mogućnost lakog pronalaska receptata sa zdravim sastojcima. Emili planira svoje tjedne obroke i želi u svoju prehranu uključiti više biljaka. Često posjećuje web stranice s receptima kako bi pronašla nova i uzbudljiva jela koja su u skladu s njezinim zdravstvenim ciljevima.

#### Ciljevi

- Pronaći recepte koji su u skladu s njezinom prehranbenim preferencijama i prehranbenim ciljevima
- Veliki izbor receptata
- Uredno kategorizirane recepte u različite kategorije

#### Frustracije

- Nemogućnost pronalazak receptata koji zadovoljavaju njezine specifične prehranbene potrebe
- Nedovoljno receptata koji daju prioritet zdravoj prehrani
- Nemogućnost pronalazak zdravih receptata u hrpi ostalih

#### Scenariji korištenja web stranice

- Želi vidjeti sastojke receptata
- Želi pronaći recepte po zdravim sastojcima koje je odabrala
- Želi vidjeti postupak pripreme jela

#### Važnosti kod stranice

Brzo i efikasno pronalaženja informacije



Raznovrsnos sadržaja



Dizajn



Filteri sadržaja



Slika 2. Prikaz druge persone



### Ana Moč

**Dob:** 28 godina  
**Spol:** Ženski  
**Lokacija:** Ogulin, Hrvatska

“

Život je prekratak za dosadne oborke.

#### Biografija

Ana je strastvena kuharica koja voli eksperimentirati s novim receptima i sastojcima. Svoje slobodno vrijeme provodi pregledavajući web stranice s receptima, isprobavajući različite kuhinje. Također voli i sama isprobavati nove recepte i dijeliti ih sa svojom obitelji i prijateljima. Ana želi imati cijeli široku izbor receptata nadohvat ruke i uredno kategorizirane. Dodatno mogućnost pronalaženja receptata po sastojcima otvorile bi joj se nove mogućnosti isprobavanja novih sastojaka, i pružala novu inspiraciju za kuhanjem.

#### Ciljevi

- Veliki izbor receptata
- Mogućnost traženja receptata na temelju specifičnih sastojka
- Pronalazak novih i uzbudljivih receptata
- Podjela svojih receptata s drugima

#### Frustracije

- Nedovoljno opcija receptata
- Poteškoće pronalaženja receptata koji odgovaraju njezinom ukusu
- Nedostatak inspiracije za kuhanjem nova jela
- Nemogućnost podijele receptata s drugim ljudima

#### Scenariji korištenja web stranice

- Želi vidjeti slike receptata
- Želi saznati postupak pripreme jela
- Želi otkriti nove recepte birajući sastojke
- Želi ostaviti komentar
- Želi objaviti vlastiti recept

#### Važnosti kod stranice

Brzo i efikasno pronalaženja informacije



Raznovrsnos sadržaja



Dizajn



Filteri sadržaja



Slika 3. Prikaz treće persone



### 3. Korisničke priče

Korisnička priča je kratak i jednostavan opis ispričan iz perspektive osobe koja želi novu mogućnost, obično korisnika ili kupca sustava. Obično slijede jednostavan predložak: Kao <tip korisnika>, želim <neki cilj> tako da <neki razlog> [4]. Korisnička priča napisana je na lako dostupnom jeziku kako bi pružila jasnu sliku o tome što korisnik treba. Napisana je jezikom koji je razumljiv kako krajnjem korisniku tako i onome koji treba napraviti proizvod. Svrha korisničkih priča je postaviti korisnika u središte razgovora o tome što treba dodati ili promijeniti na proizvodu. One daju razvojnom timu kontekst i razlog zašto stvaraju određeni proizvod [5].

Za potrebe izrade aplikacije sveukupno je definirano sedam korisničkih priča, koje pokrivaju sve funkcionalnosti aplikacije. Napisane su po navedenom predlošku. Definirane su sljedeće korisničke priče:

1. Kao zaposlena majka, želim brzo pronaći recepte za zdrava i ukusna jela koja mogu pripremiti od sastojaka koje već imam u kuhinji, kako bih uštedjela vrijeme i novac.
2. Kao student, želim pregledavati recepte filtrirane po sastojcima kako bih pronašao jela koja mogu pripremiti s namirnicama koje već imam, kako bih smanjio bacanje hrane.
3. Kao osoba na dijeti, želim pregledavati recepte koje sadržavaju zdrave namirnice kako bih pronašao zdrave opcije koje odgovaraju mojim prehrambenim potrebama.
4. Kao domaćin zabave, želim pregledavati recepte za pića kako bih pronašao nove i zanimljive koktele koje mogu poslužiti gostima.
5. Kao početnik u kuhanju, želim pregledavati detaljne upute pripreme recepte kako bih naučio pripremati nova jela.
6. Kao osoba koja voli isprobavati nove recepte, želim moći dodavati vlastite recepte na web aplikaciju i dijeliti ih s drugima.
7. Kao osoba koja voli kuhati, želim moći ostaviti recenzije na recepte kako bih podijelio svoje iskustvo s drugim korisnicima.

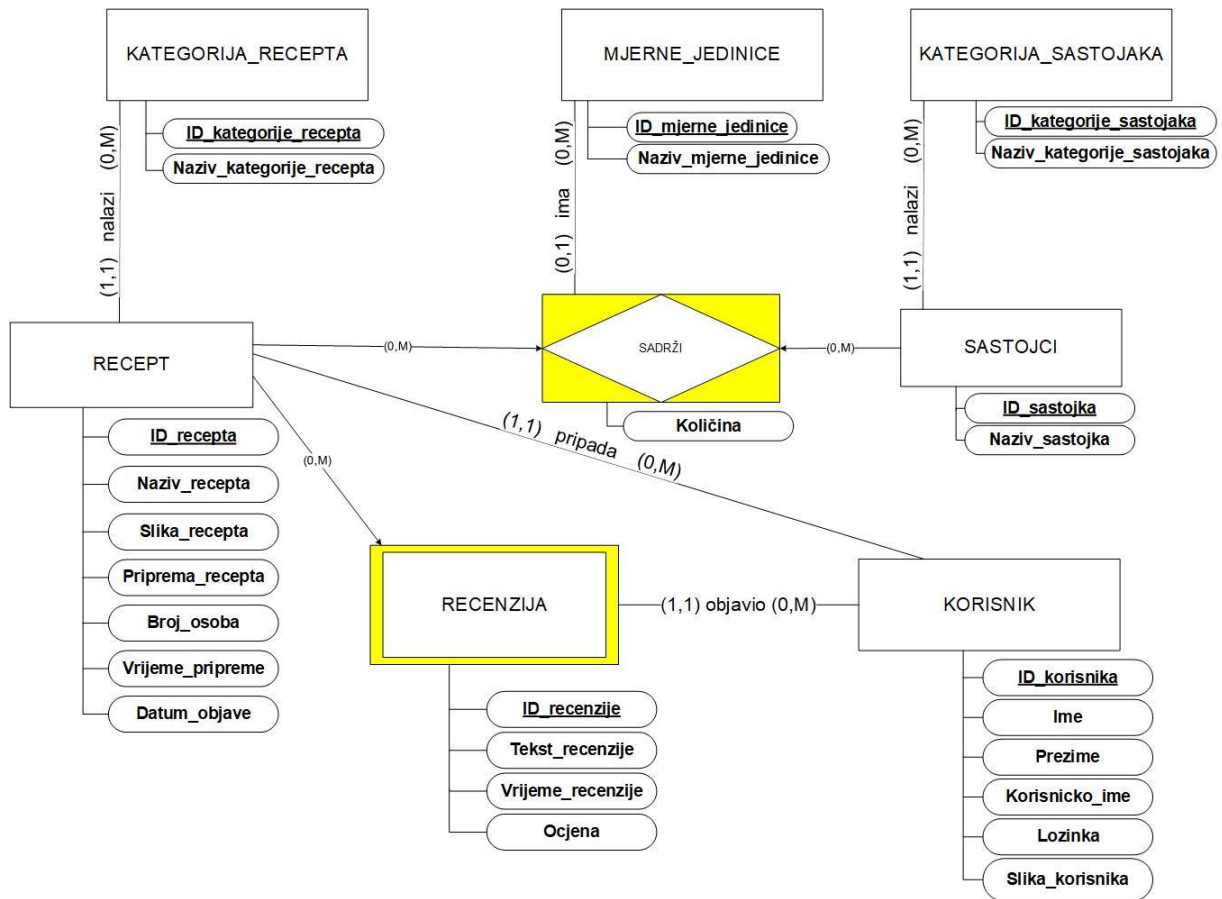
## 4. Model podataka

Definiranje modela podataka ključan je korak prije razvoja svake aplikacije. Model podataka opisuje veze između entiteta i njihovih atributa, te njegovi osnovni koncepti su entiteti, tipovi entiteta, veze, tipovi veze, atributi i brojnosti. Entitet je stvar koja ima stvarno ili pojedinačno postojanje u stvarnosti ili mislima, npr. osoba, pojedini kupac i sl. Slični entiteti koji imaju zajednička svojstva se klasificiraju u tipove entiteta. Atribut je karakteristika nekog entiteta, npr. ime, prezime i sl. Veza predstavlja odnos koji postoji među entitetima, dok je tip veze skup veza između istih tipova entiteta. Brojnost tipa veze je broj koji pokazuje koliko entiteta pojedinog tipa entiteta sudjeluje u tipu veze s entitetom drugog tipa entiteta [6].

Prvi tip entiteta je *KATEGORIJA\_RECEPATA* kojoj je primaran ključ *ID\_kategorije\_recepata* te sadrži atribut *Naziv\_kategorije\_recepata*. Tip entiteta *KATEGORIJA\_RECEPATA* sadrži jedan ili mnogo recepata iz tipa entiteta *RECEPT*. Dok jedan recept iz tipa entiteta *RECEPT* nalazi se u jednoj i samo jednoj kategoriji iz tipa entiteta *KATEGORIJA\_RECEPATA*. Tip entiteta *RECEPT* primaran je ključ *ID\_recepta* te sadrži attribute *Naziv\_recepta*, *Slika\_recepta*, *Priprema\_recepta*, *Broj\_osoba*, *Vrijeme\_pripreme* i *Datum\_objave*. Tip entiteta *RECEPT* povezan je s tipom entiteta *KORISNIK* tako da jedan recept pripada jednom i samo jednom korisniku dok jedan račun može imati nula ili mnogo recepata. Također tip entiteta *RECEPT* je povezan na tip entiteta *RECENZIJA* tako da jedan recept može imati nula ili mnogo recenzija dok jedna recenzija pripada jednom i samo jednom receptu. *RECENZIJA* je slabi entitet od entiteta *RECEPT* te ima primaran ključ *ID\_RECENZIJE* dok su njegovi atributi *Tekst\_recenzije*, *Vrijeme\_recenzije*, *Ocjena*. Tip entiteta *RECENZIJA* povezan je s tipom entiteta *KORISNIK* tako da jedna recenzija pripada jednom i samo jednom korisniku, dok korisnik može imati nula ili mnogo recenzija. Tip entiteta *KORISNIK* ima primaran ključ *ID\_korisnika* dok su njegovi atributi *Ime*, *Prezime*, *Korisnicko\_ime*, *Lozinka* i *Slika\_korisnika*.

Model podataka sadrži agregaciju *SADRŽI* koja ima atribut *Količina*. Agregacija *SADRŽI* je između tipa entiteta *RECEPT* i *SASTOJCI*, tako da jedan recept sadrži nula ili mnogo sastojka dok jedan sastojak pripada nula ili mnogo recepata. Na agregaciju dodatno povezan je i tip entiteta *MJERNE\_JEDINICE*, tako da mjerna jedinica ima nula ili mnogo opisa sastojaka dok jedan opis sastojaka u receptu ima nula ili samo jednu mjernu jedinicu. Tip entiteta *MJERNE\_JEDINICE* sadrži primaran ključ *ID\_mjerene\_jedinice* i atribut *Naziv\_mjerne\_jedinice*.

Tip entiteta *SASTOJCI* ima primaran ključ *ID\_sastojka*, dok sadrži atribut *Naziv\_sastojka*. Tip entiteta *SASTOJAK* nalazi se u jednoj i samo jednoj kategoriji iz tipa entiteta *KATEGORIJA\_SASTOJKA*, dok jedna kategorija iz tipa entiteta *KATEGORIJA\_SASTOJKA* može sadržavati nula ili mnogo sastojka iz tipa entiteta *SASTOJAK*. Tip entiteta *KATEGORIJA\_SASTOJKA* ima primaran ključ *ID\_kategorije\_sastojaka* i atribut *Naziv\_kategorije\_sastojaka*. Na Slici 4. nalazi se prikaz modela podataka.



Slika 4. Prikaz modela podataka

## 5. Izrada aplikacije

Kako bi se moglo započeti s izradom aplikacije najprije je potrebno kreirati novi direktorij koji je u ovom slučaju nazvan *Završni\_rad\_Ema\_Fabac*. Ovaj direktoriji može se otvoriti preko Visual Studio Codea, preko kojeg će se izvoditi sve naredbe.

### 5.1. Stvaranje projekta

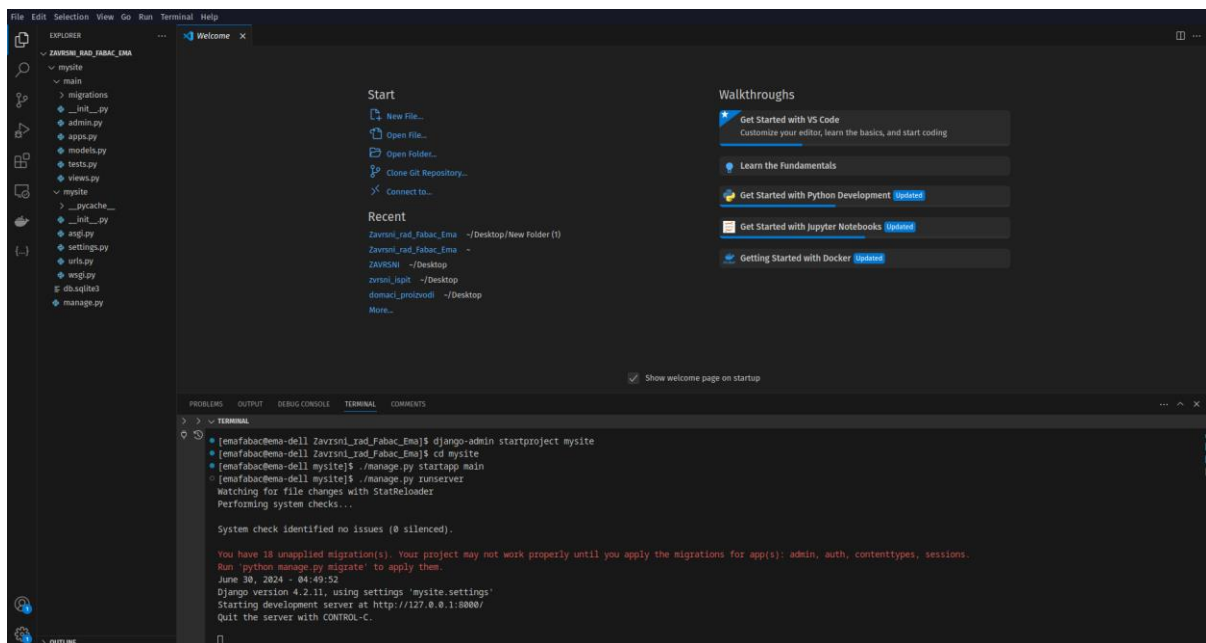
Prvo što je potrebno kreirati je novi Django projekt u ovom slučaju naziva *mysite* sljedećom naredbom:

```
django-admin startproject mysite
```

*Mysite* direktorij je središnji direktorij web aplikacije, i njegova glavna uloga je preusmjeravanje Djanga na ostale aplikacije, te s pomoću novonastale datoteke *manage.py* izvršavaju se naredbe unutar terminala. Aplikacija se izrađuje unutar *mysite* direktorija koja se u ovom slučaju naziva *main*, pomoću naredbe:

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

Na Slici 5. prikazan je cjelokupni kreirani projekt u Visual Studio Codeu, sa svim datotekama koje su instalirane za postavljanje okruženja i terminalom u kojem su izvršene prijašnje navedene naredbe.

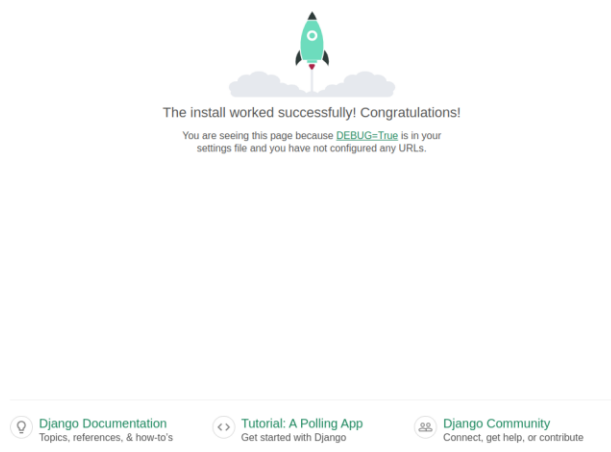


Slika 5. Prikaz kreiranog projekta *mysite* i aplikacije *main* u Visual Studio Codeu

Kako bi se moglo pristupiti web stranici aplikacije pokreće se lokalni poslužitelj naredbom:

```
./manage.py ruserver
```

Aplikacija se nalazi u web pregledniku na adresi <http://127.0.0.1:8000/>, gdje se za sad prikazuje Django welcome stranica koja se nalazi na Slici 6. kako još ništa nije definirano u samo kreiranoj aplikaciji i projektu.



Slika 6. Prikaz Django welcome stranice

## 5.2. Postavljanje aplikacije

Nakon kreiranja projekta i aplikacije, potrebno je povezati aplikaciju na projekt. Prva stvar koju je potrebno napraviti je dodati kreiranu aplikaciju na listu instaliranih aplikacija u *settings.py* koji se nalazi unutra projekta *mysite*.

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

U kodu se mogu vidjeti sve aplikacije i direktoriji koji su korišteni za izradu projekta. Najprije je bilo dovoljno dodati samo *MainConfig*, no kroz izradu projekta bilo je potrebno dodati i direktorij *templatetags*.

Potom je potrebno projekt povezati s aplikacijom. Unutar *mysite/mysite/urls.py* datoteke već je definiran jedan URL koji je vezan za stranicu administracije, te je potrebno definirati novi URL *path("", include('main.urls'))* koji će usmjeriti glavnu aplikaciju na novostvorenu *main* aplikaciju.

```
from django.contrib import admin  
from django.urls import path, include  
urlpatterns = [  
    path('', include('main.urls'))  
]
```

```

    path('', include('main.urls')),
    path('admin/', admin.site.urls),
]

```

Nakon toga potrebno je najprije kreirati *mysite/main/urls.py* datoteku unutar koje će biti upisani svi linkovi za potrebu rada aplikacije te kod sada izgleda ovako:

```

from django.urls import path
from . import views
app_name = 'main'
urlpatterns = [
]

```

Sada kada je aplikacija povezana na Django projekt unutar *urlpatterns* možemo definirati likove stranice i usmjeriti aplikaciju da pokrene određenu funkciju iz *views.py* datoteke.

### 5.3. Modeli

Modeli se u Django definiraju u datoteci *main/models.py*, te se definiraju kao klase unutar kojih se nalaze atributi. Za svaki atribut potrebno je definirati tip podataka, te pojedini imaju definirane i neke dodatne argumente. Svaka definirana klasa ponaša se kao tablica u bazi podataka, unutar koje je svaki atribut jedan stupac tablice. Django za svaku definiranu klasu će automatski kreirati primarni ključ ako on nije definiran kao poseban atribut, ako je definiran kao atribut potrebno mu je dodati argument *primary\_key=True*. U klasama je potrebno definirati i veze među njima. U Django je moguće kreirati *Many to One*, *Many to Many* i *One to One* veze. Za potrebe ove aplikacije veze između klasa definirane su s pomoću *ForeignKey()* što označuje da jedna instanca jedne tablice može pripadati ili imati samo jednu instancu tablice na koju ju povezana. Dok je jedino između klase *User* i *Korisnik* definirana *One to One* veza, što znači da jednoj *User* instanci pripada samo jedan *Korisnik* i obrnuto. Također za svaku tablicu definiran je i *\_\_str\_\_(self)* koji definira kako će se objekt pojedine klase prikazati kao string u bazi podataka na web pregledniku.

Prije samog kreiranja klasa potrebno je uvesti sve potrebne funkcije za njihov rad. Osim njih ovdje se poziva i *User* model koji je ugrađeni Django model i predstavlja korisnika u sustavu. Ovaj *User* model ima predefinirana polja, od koji se za potrebe ove aplikacije koriste *first\_name*, *last\_name*, *email* i *password*. Inicijalizirane funkcije prikazane su sljedećim kodom:

```

from django.db import models
from django.contrib.auth.models import User
from django.db.models.signals import post_save
from django.dispatch import receiver

```

Prva klasa koja je definirana je *KategorijaRecepta* koja se sastoji od atributa *id\_kategorije\_recepta* i *naziva\_kategorije\_recepta*. *id\_kategorije\_recepta* je primarni ključ te je time definiran i kao *AutoField* kako bi se automatski generirali podaci. Dok je *naziv\_kategorije\_recepta* definiran kao *CharField*, te postavljeno ograničenje na maksimalno 255 znakova. Ova klasa koristi se kako bi sadržavala popis kategorija u koje će se svrstati svi

recepti, te kako bi se moglo u završnoj aplikaciji filtrirati recepte po kategorijama. Za stvaranje navedene klase koristi se sljedeći kod:

```
class KategorijaRecepta(models.Model):
    id_kategorije_recepta = models.AutoField(primary_key=True)
    naziv_kategorije_recepta = models.CharField(max_length=255)
    def __str__(self):
        return self.naziv_kategorije_recepta
```

Druga klasa koja je definirana je *Recept* koja se sastoji od atributa *id\_recepta*, *naziv\_recepta*, *slika\_recepta*, *priprema\_recepta*, *vrijeme\_pripreme*, *broj\_osoba*, *datum\_objave*, *kategorije\_recepta* i *id\_korisnika*. *Id\_recepta* postavljen je kao primaran ključ, te definiran kao *AutoField* kako bi se automatski kreirali podaci za ovu klasu. Zatim su definirani neki atributi koji opisuju recept, gdje *naziv\_recepta* predstavlja sam naziv recepta, te je postavljeno ograničenje od najviše 255 znakova, *slika\_recepta* predstavlja sliku recepta te će se one spremati unutra *images/recepti* direktorija, *priprema\_recepta* predstavlja postupak pripreme pojedinog recepta te je definiran kao *TextField* koji može sadržavati više znakova, *broj\_osoba* definira za koliko je osoba navedena količina sastojka recepta, *vrijeme\_pripreme* predstavlja koliko je vremena potrebno da se pripremi navedeni recept, *datum\_objave* predstavlja datum kada je recept objavljen te sadržava i funkciju *auto\_now\_add=True*, što znači da će se prilikom objave recepta automatski spremati trenutno vrijeme i datum. Definirane su i dvije *ForeignKey* veze, prva je *kategorija\_recepta* koja povezuje recept s klasom *KategorijaRecepta*, što znači da se jedan recept može nalaziti samo u jednoj kategoriji, druga je *id\_korisnika*, koja povezuje recept s korisnikom koji ga je objavio, i također vrijedi da recept pripada samo jednom korisniku koji ga je obavio. Također za oba atributa definiran je i *on\_delete = models.CASCADE* što znači da ako dođe do brisanje neke instance u natklasi, izbrisat će se i sve instance u klasi recept koje su pripadale toj natklasi. Kod za definiranje modela *Recept* je sljedeći:

```
class Recept(models.Model):
    id_recepta = models.AutoField(primary_key=True)
    naziv_recepta = models.CharField(max_length=255)
    slika_recepta = models.ImageField(upload_to='images/recepti/')
    priprema_recepta = models.TextField()
    broj_osoba = models.IntegerField()
    vrijeme_pripreme = models.IntegerField()
    datum_objave = models.DateTimeField(auto_now_add=True)
    kategorija_recepta = models.ForeignKey(KategorijaRecepta,
    on_delete=models.CASCADE)
    id_korisnika = models.ForeignKey('Korisnik', on_delete=models.CASCADE)
    def __str__(self):
        return self.naziv_recepta
```

Treća klasa koja je definirana je *MjerneJedinice* koja se sastoji od atributa *id\_mjerne\_jedinice* i *naziv\_mjerne\_jedinice*. *Id\_mjerne\_jedinice* predstavlja primaran ključ, dok

*naziv\_mjerne\_jedinice* predstavlja skraćenice naziv mjerne jedinice kao što su L, kg, g i slično. Kod za definiranje klase *MjerneJedinice* je sljedeći:

```
class MjerneJedinice(models.Model):
    id_mjerne_jedinice = models.AutoField(primary_key=True)
    naziv_mjerne_jedinice = models.CharField(max_length=255)
    def __str__(self):
        return self.naziv_mjerne_jedinice
```

Četvrta klasa koja je definirana je *Sastojci* i ona se sastoji od atributa *id\_sastojka*, *naziv\_sastojka* i *kategorija\_sastojka*. *id\_sastojka* kao i kod prethodnih klasa predstavlja primarni ključ, dok *naziv\_sastojka* predstavlja naziv sastojka kao što su na primjer brašno, mlijeko, jaja i slično. *Kategorija\_sastojka* je *ForeignKey* veza na klasu *KategorijaSastojka* te označuje da jedan sastojak može se nalaziti samo u jednoj kategoriji sastojaka. *on\_delete=models.CASCADE* označuje da ako dođe do brisanja kategorije sastojaka izbrisat će se i sve instance u modelu sastojka koje se nalaze u toj kategoriji. Za definiranje modela *Sastojci* koristi se sljedeći kod:

```
class Sastojci(models.Model):
    id_sastojka = models.AutoField(primary_key=True)
    naziv_sastojka = models.CharField(max_length=255)
    kategorija_sastojaka = models.ForeignKey('KategorijaSastojka',
on_delete=models.CASCADE)
    def __str__(self):
        return self.naziv_sastojka
```

Peta klasa koja je definirana je *KategorijaSastojaka* koja sadržava attribute *id\_kategorije\_sastojaka* i *naziv\_kategorije\_sastojaka*. *id\_kategorije\_sastojaka* kao i u prethodnim klasama predstavlja primarni ključ dok *naziv\_kategorije\_sastojaka* predstavlja nazive kategorija i postavljeno je ograničenje na najviše 255 znakova.

```
class KategorijaSastojaka(models.Model):
    id_kategorije_sastojaka = models.AutoField(primary_key=True)
    naziv_kategorije_sastojaka = models.CharField(max_length=255)
    def __str__(self):
        return self.naziv_kategorije_sastojaka
```

Šesta klasa koja je definirana je *Korisnik* i sadržava attribute *user* i *slika\_korisnika*. U atribut *slika\_korisnika* sprema se profilna slika korisnika te definirana je lokacija na */images/korisnici* i dodatno su definirani i uvjeti *null=True* i *blank=True* što označuje da ovaj atribut ne treba nužno biti ispunjen. Atribut *user* je *OneToOneField* veza na već ugrađeni model *User* i *Korisnik*, te označava da jednom korisniku pripada samo jedan *user* i obrnuto. Za definiranje klase *Korisnik* koristi se sljedeći kod:

```
class Korisnik(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    slika_korisnika = models.ImageField(upload_to='images/korisnici/', null=True,
blank=True)
```



```
def __str__(self):
    return f"{self.user.first_name} {self.user.last_name}"
```

Sedma klasa je *Recenzija* i sadržava atribute *id\_recenzije*, *tekst\_recenzije*, *vrijeme\_recenzije*, *ocjena*, *recept* i *korisnik*. *id\_recenzije* kao i u prethodnim klasa označava primaran ključ, dok *tekst\_recenzije* predstavlja tekst koji korisnik upisuje u recenziju, *vrijeme\_recenzije* predstavlja vrijeme kada je recenzija objavljena te postavljen je uvjet *auto\_now\_add* što znači da će se automatski ispuniti s trenutnim vremenom i datumom, *ocjena\_recenzije* označava koliko je zvjezdica korisnik dao nekom receptu i nalazi se u rang od 1 do 5. *recept* i *korisnik* atributi definirani su kao *ForeignKey* veza na klasu *Recept* i *Korisnik* što označava da jedna recenzija pripada jednom receptu i objavio ju je jedan korisnik. Za definiranje klase *Recenzija* koristi se sljedeći kod:

```
class Recenzija(models.Model):
    id_recenzije = models.AutoField(primary_key=True)
    tekst_recenzije = models.TextField()
    vrijeme_recenzije = models.DateTimeField(auto_now_add=True)
    ocjena = models.IntegerField()
    recept = models.ForeignKey(Recept, on_delete=models.CASCADE)
    korisnik = models.ForeignKey(Korisnik, on_delete=models.CASCADE)
    def __str__(self):
        return f"{self.korisnik.user.first_name} {self.korisnik.user.last_name} - {self.id_recenzije}: {self.recept.naziv_recepta}"
```

Osmo klasa je *Sadrzi* koja predstavlja u modelu podataka agregaciju između sastojka i recepta. U njoj se nalazi popis svih sastojaka koji se nalaze u određenom receptu kao kombinacija naziva sastojka, mjerne jedinice i količine sastojka koji je potreban za određeni recept. Ovaj model sastoji se od tri *ForeignKey* veze, čime se povezuje na klase *Recept*, *Sastojci* i *MjerneJedinice*, te dodatno je za mjerne jedinice određen i uvjet *null=True*, *blank=True* što označuje da može polje i ostati prazno. Sadržava i atribut *kolicina* koji ujedno može ostati prazan i ima ograničenu veličinu od 5 znakova. Ovaj model definiran je sljedećim kodom:

```
class Sadrzi(models.Model):
    id_recepta = models.ForeignKey(Recept, on_delete=models.CASCADE)
    id_sastojka = models.ForeignKey(Sastojci, on_delete=models.CASCADE)
    id_mjerne_jedinice = models.ForeignKey(MjerneJedinice,
on_delete=models.SET_NULL, null=True, blank=True)
    kolicina = models.CharField(max_length=5, null=True, blank=True)
    def __str__(self):
        if self.id_mjerne_jedinice and self.kolicina:
            return f"{self.id_sastojka.naziv_sastojka} - {self.kolicina} {self.id_mjerne_jedinice.naziv_mjerne_jedinice} : {self.id_recepta.naziv_recepta}"
        elif self.kolicina:
            return f"{self.id_sastojka.naziv_sastojka} - {self.kolicina} : {self.id_recepta.naziv_recepta}"
        elif self.id_mjerne_jedinice:
            return f"{self.id_sastojka.naziv_sastojka} - {self.id_mjerne_jedinice.naziv_mjerne_jedinice} : {self.id_recepta.naziv_recepta}"
```

```

else:
    return f"{self.id_sastojka.naziv_sastojka} :
{self.id_recepta.naziv_recepta}"

```

Osim klasa za bazu podataka u *model.py* definirana je i funkcija *created\_korisnik*, ova funkcija poziva se kada se kreira novi korisnik u *User* tablici, te automatski kreira novu instancu modela *Korisnik* i povezuje je s novostvorenom instancom modela *User*. Kod koji prikazuje ovu funkciju je sljedeći:

```

@receiver(post_save, sender=User)
def create_korisnik(sender, instance, created, **kwargs):
    if created:
        Korisnik.objects.create(user=instance)

```

Nakon kreiranja svih potrebnih klasa za funkcioniranje aplikacije u *models.py* datoteci potrebno je dodati sve kreirane klase u *main/admin.py* kako bi one postale vidljive u admin panelu. Kod koji se nalazi u *main/admin.py* je sljedeći:

```

from django.contrib import admin
from .models import *
model_list = [KategorijaRecepta, Recept, MjerneJedinice, Sastojci,
KategorijaSastojaka, Korisnik, Recenzija, Sadrzi ]
admin.site.register(model_list)

```

Nakon što se sve definiralo potrebno je primijeniti pripremu i nakon toga migraciju s naredbama. Ujedno ako se promjene bilo kakve promjene nad modelima zahtjeva se izvršavanje migracija kako bi se promjene nad modelima pohranile u sustav. Priprema i migracije izvršavaju se sljedećim naredbama u terminalu:

```

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

```

## 5.4. Spajanje na bazu pomoću admin portala

Također Django ima svoj Admin portal koji olakšava rad s bazom korisnicima kojima je dan pristup. U admin portalu moguće je vidjeti sve modele koje smo kreirali te lako ih popunjavati i uređivati. Kako bi se imao pristup Admin portalu, mora se kreirati superuser naredbom:

```

$ ./manage.py createsuperuser

```

Ovom naredbom otvaraju nam se polja za unos korisničkog imena, email (neobavezno polje) te lozinke i potvrdu lozinke. Nadalje nakon ponovnog povezivanja na Django web preglednik na lokaciji <http://127.0.0.1:8000/admin/> potrebno se ulogirati s podacima koji su prethodno upisani kao superuser i time pristupiti bazi podataka koju možemo vidjeti na Slici 7.

## Django administration

### Site administration

AUTHENTICATION AND AUTHORIZATION	
Groups	<a href="#">+ Add</a> <a href="#">Change</a>
Users	<a href="#">+ Add</a> <a href="#">Change</a>

MAIN	
Kategorija receptas	<a href="#">+ Add</a> <a href="#">Change</a>
Kategorija sastojakas	<a href="#">+ Add</a> <a href="#">Change</a>
Korisniks	<a href="#">+ Add</a> <a href="#">Change</a>
Mjerne jedinices	<a href="#">+ Add</a> <a href="#">Change</a>
Recenzijas	<a href="#">+ Add</a> <a href="#">Change</a>
Recepts	<a href="#">+ Add</a> <a href="#">Change</a>
Sadrzis	<a href="#">+ Add</a> <a href="#">Change</a>
Sastojcis	<a href="#">+ Add</a> <a href="#">Change</a>

Slika 7. Prikaz Django baze podataka

## 5.5. Navigacija

Navigacija je definirana u svim html dokumentima unutar *headera*, te je definiran dizajn i njezina funkcija. Sastoji se od loga na lijevoj strani i od dva gumba na desnoj strani. Logo se nalazi u aplikaciji na lokaciji *main/static/logo/logo\_ema.png* koji je napravljen u *Inscap* programu te ujedno služi i kao link na početnu stranicu. Za logo je odabran naziv restorana *Home Chef*. Za prijavljene korisnike s desne strane prvi gumb je *UNESITE SVOJ RECEPT* koji klikom na njega otvara se obrazac koji korisnik može ispuniti za unos novog recepta. Pored njega nalazi se gumb koji sadržava korisničko ime te klikom na njega otvara se padajući izbornik s dvije opcije *Moji profil* i *Odjava*. Za neprijavljene korisnike nalazi se gumb *UNESITE SVOJ RECEPT* i *Prijavi se* koji će klikom na njih odvesti korisnika na isti skočni prozor kojim korisnik može birati da se prijavi ili registrira u aplikaciju. Kako bi se odredilo što se kojem korisniku prikazuje postavljaju se *if* i *else* uvjeti. Time se uvjetom *if user.is\_authenticated* definira način za prijavljene korisnike dok se pod uvjetom *else* definira načina za neprijavljene korisnike. Sljedeći kod prikazuje izgled i funkcije navigacije:

```

<header class="header-area">
  <div class="delicious-main-menu" >
    <div class="classy-nav-container breakpoint-off" >
      <nav class="classy-navbar justify-content-between shadow" style="height:
100px;" id="deliciousNav">
        <a class="nav-brand" style="margin-left: 80px;" href=" ">
          
        </a>
        <div class="classy-menu">
          <div class="classynav">
            <ul>
              {% if user.is_authenticated %}
                <li>
                  <button class="btn custom-btn btn-round" type="button"
onclick="location.href='/add_receipt'">
                    UNESITE SVOJ RECEPT
                  </button>
                </li>
                <li>
                  <div class="btn-group">
                    <button class="btn custom-btn btn-round" style="margin-right:
80px; border-radius: 50px; margin-left: 20px" type="button" data-toggle="dropdown"
aria-haspopup="true" aria-expanded="false">
                      {{ user.username }}
                    </button>
                    <div class="dropdown-menu dropdown-menu-right">
                      <a href="{% url 'main:profile' %}">Moji profil</a>
                      <a href="/logout">Odjava</a>
                    </div>
                  </div>
                </li>
              {% else %}
                <li>
                  <button class="btn custom-btn btn-round" type="button" data-
toggle="modal" data-target="#exampleModalCenter">
                    UNESITE SVOJ RECEPT
                  </button>
                </li>
                <li>
                  <button class="btn custom-btn btn-round" style="margin-right:
80px; margin-left: 20px" type="button" data-toggle="modal" data-
target="#exampleModalCenter">
                    Prijavi se
                  </button>
                </li>
            </ul>
          </div>
        </div>
      </nav>
    </div>
  </div>

```

```

        {% endif %}
    </ul>
</div>
</div>
</nav>
</div>
</div>

```

Slika 8. i Slika 9. prikazuju izgled navigacije prijavljenog korisnika.



Slika 8. Prikaz osnovne navigacije za prijavljenog korisnika



Slika 9. Prikaz osnovne navigacije i padajućeg izbornika za prijavljenog korisnika

Slika 10. prikazuje izgled navigacije za neprijavljene korisnike.



Slika 10. Prikaz osnovne navigacije za neprijavljene korisnika

Ako korisnik nije prijavljen u aplikaciju on nema pravo na unos novog recepta te nije mu kreiran profil, klikom na jedna od dva gumba u navigaciji odvest će ga na skočni prozor. U skočnom prozoru pruža se mogućnost korisniku da se prijavi ako već ima račun ili da se registriira ako nema već kreirani račun. Sljedeći kod prikazuje definiranje izgleda skočnog prozora:

```

<div class="modal fade" id="exampleModalCenter" tabindex="-1" role="dialog" aria-
labelledby="exampleModalCenterTitle" aria-hidden="true">
  <div class="modal-dialog modal-dialog-centered modal-sm" role="document">
    <div class="modal-content">
      <div class="modal-body text-center" style="padding: 2rem;">
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
        <div class="row">
          <div class="col-md-12 mb-3">
            <h5 class="modal-title">Već imate račun?</h5>
            <p class="text-muted mb-3">Prijavite se za pristup svom računu</p>
            <a href="/login" class="btn btn-primary btn-round">Prijava</a>
          </div>
        </div>
        <div class="row">
          <div class="col-md-12">

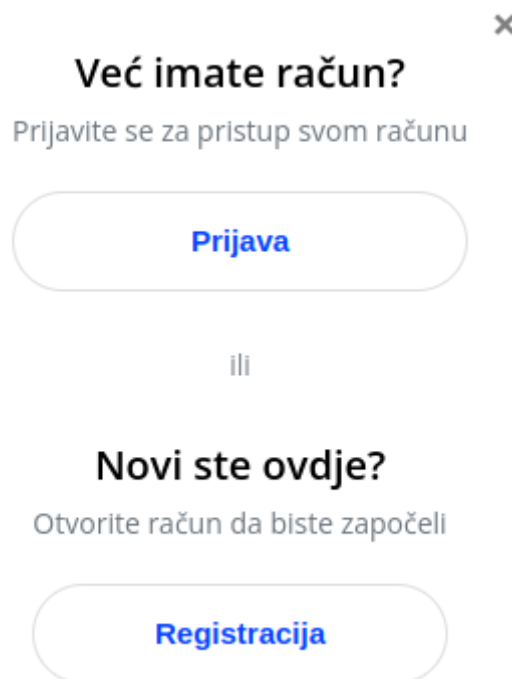
```

```

        <span style="display: flex; align-items: center;">
            <span style="padding: 10px 10px; font-size: 14px; color: #7b828a;
margin-bottom: 15px;">ili</span>
        </span>
    </div>
</div>
<div class="row">
    <div class="col-md-12">
        <h5 class="modal-title">Novi ste ovdje?</h5>
        <p class="text-muted mb-3">Otvorite račun da biste započeli</p>
        <a href="/register" class="btn btn-primary btn-round">Registracija</a>
    </div>
</div>
</div>
</div>
</div>
</div>
</div>

```

Slika 11. prikazuje izgled skočnog prozora za neprijavljene korisnike, ukoliko se žele prijaviti ili registrirati u aplikaciju.



Slika 11. Prikaz skočnog prozora za odabir prijave ili registracije

## 5.6. Prijava i registracija

Kako bi se korisnik ulogirao u aplikaciju ima dvije mogućnosti, prijavu ako već ima napravljeni račun ili registraciju ako je prvi put na stranici i želi napraviti račun. Ako korisnik odabere gumb *Prijava* odnijet će ga na stranicu na lokaciji '<http://127.0.0.1:8000/login/>' koja sadržava

obrazac za prijavu. Za prikaz ove stranice najprije je potrebno definirati funkciju koja se brine o prijavi unutra *main/views.py* datoteke. Definirana funkcija *login\_user* najprije provjerava je li metoda zahtjeva *POST*, što znači da je korisnik poslao obrazac prijave. Ako je metoda zahtjeva *POST* izvlači se korisničko ime *username* i lozinka *password* iz podataka zahtjeva *POST*. Potom se koristi funkcija *authenticate* za provjeru korisničkog imena i lozinke, ako su oni valjani funkcija vraća objekt *User*. Ako objekt *User* nije *None*, što znači da je provjera uspješna, korisnik se prijavljuje u aplikaciju s pomoću funkcije *login* te je nakon uspješne prijave preusmjeren na početnu stranicu. Ako provjera nije uspješna preglednik prikazuje poruku o uspjehu, koja je zapravo poruka o grešci, s pomoću okvira *messages* i preusmjerava korisnika natrag na stranicu prijave. Ako metoda zahtjeva nije *POST*, nego na primjer ako je *GET* zahtjev, onda se samo prikazuje obrazac prijave definiran u *login.html* datoteci. Sljedeći kod prikazuje definiranu funkciju *login\_user*.

```
def login_user (request):
    if request.method == "POST":
        username = request.POST['username']
        password = request.POST['password']
        user = authenticate(request, username=username, password=password)
        if user is not None:
            login(request, user)
            return redirect('main:pocetna')
        else:
            messages.success(request, ("Krivi unos! Pokušajte ponovo..."))
            return redirect('main:login')
    else:
        return render(request, 'registration/login.html', {})
```

Nakon definiranja funkcije u *views.py* potrebno je definirati i template pod nazivom *login.html*. U njemu je definiran izgled stranice te je pozvana forma za obrazac prijave i if-else uvjeti za prikaz poruke o grešci ako je potrebno. Sljedeći kod je iz template *login.html*:

```
<div class="shadow p-5 mb-6 bg-light rounded" style="margin: 40px auto; max-width: 500px;">
    <div style="text-align: center;">
        <h1>Prijava</h1>
    </div>
    <form method="POST" action="">
        {% csrf_token %}
        {% if messages %}
            <ul>
                {% for message in messages %}
                    <li>{{ message }}</li>
                {% endfor %}
            </ul>
        {% endif %}
    </div class="mb-3">
```

```

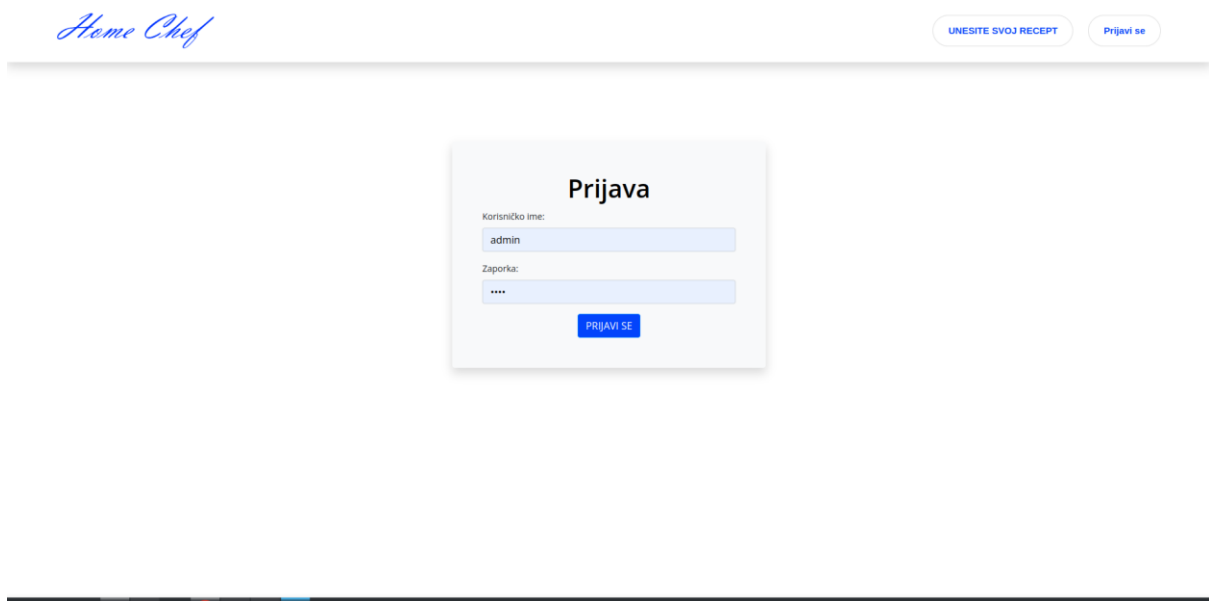
        <label for="exampleInputEmail1" class="form-label">Korisničko ime: </label>
        <input type="text" class="form-control" name="username" aria-
describedby="emailHelp">
    </div>
    <div class="mb-3">
        <label for="exampleInputPassword1" class="form-label">Zaporka: </label>
        <input type="password" class="form-control" name="password">
    </div>
    <div style="text-align: center;">
        <button type="submit" class="btn btn-primary" style="background-color:
#0244fb;" >PRIJAVI SE</button>
    </div>
</form>
</div>

```

Posljednje što se treba učiniti kako bi se stranica prikazivala je unutar *urls.py* definirati lokaciju na kojoj će se stranica nalaziti. U ovom slučaju stranica se nalazi na linku *login/*. Kod za definiranje linka je sljedeći

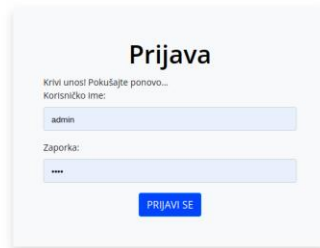
```
path('login/', views.login_user, name='login'),
```

Slika 12. prikazuje izgled forme za prijavu dok Slika 13. prikazuje poruku o pogrešci ukoliko su uneseni krivi podaci.



Slika 12. Prikaz obrasca za prijavu korisnika





Slika 13. Prikaz poruke o grešci kod obrasca za prijavu

Druga opcija za prijave u sustav je registracija. Za nju je ujedno potrebno definirati funkciju za registraciju korisnika te je definirana funkcija *register*. Prvo se provjerava koja je metoda zahtjeva stigla, ako je *POST* metoda to znači da je korisnik poslao podatke za prijavu. Ako je metoda *POST*, stvara se forma za registraciju korisnika s podacima koju su stigli iz zahtjeva, te se ona provjerava. Ako nije valjana, ispisuju se poruke o greškama. Ako je forma valjana, stvara se novi korisnik s podacima iz forme te se podaci spremaju u bazu podataka. Nakon što se korisnik sprema u bazu, autentificira se korisnik s korisničkim imenom i lozinkom te ako je ona uspješna korisnik se prijavljuje u sustav. Nakon što se korisnik prijavi, preusmjerava se na početnu stranicu web stranice. Ako metoda nije *POST* ili ako forma nije valjana prikazuje se forma za registraciju korisnika. Na kraju stvara se kontekst s formom za registraciju i renderira se stranica za registraciju. Sljedeći kod prikazuje funkciju *register* u *views.py*:

```
def register(request):
    if request.method == 'POST':
        form = RegisterUserForm(request.POST)
        print(form.error_messages)
        if form.is_valid():
            user = form.save(commit=False)
            user.first_name = form.cleaned_data['first_name']
            user.last_name = form.cleaned_data['last_name']
            user.email = form.cleaned_data['email']
            user.save()
            username = form.cleaned_data['username']
            password = form.cleaned_data['password1']
            user = authenticate(username=username, password=password)
            login(request, user)
            return redirect('main:pocetna')
    else:
```

```

    form = RegisterUserForm()
    context = {'form': form}
    return render(request, 'registration/registration.html', context)

```

U `views.py` funkciji `register` pozivali smo `RegisterUserForm` koji je potrebno definirati unutar `forms.py` datoteke. Ova datoteka ne postoji prilikom izrade projekta i aplikacije te ju je potrebno kreirati unutra `main` aplikacije. `Meta` klasa definira modele koji se koriste za kreiranje forme u ovom slučaju `User` model i polja koja se prikazuju u formu. Polja su `username`, `first_name`, `last_name`, `email`, `password1` i `password2`. S pomoću metode `__init__` inicijalizira se forma, najprije se poziva roditeljska metoda `__init__` s pomoću `super()` funkcije, a nakon toga se dodaju atributi klasama za svako polje forme. Atributi `class` se dodaju svakom polju forme kako bi se mogli stilizirati s pomoću CSS-a. Također dodjeljuje se svakom polju forme klasa `form-control`. Sljedeći kod prikazuje definiranje `RegisterUserForm` forme:

```

class RegisterUserForm(UserCreationForm):
    class Meta:
        model = User
        fields = ('username', 'first_name', 'last_name', 'email', 'password1',
                  'password2')
    def __init__(self, *args, **kwargs):
        super(RegisterUserForm, self).__init__(*args, **kwargs)
        self.fields['first_name'].widget.attrs['class'] = 'form-control'
        self.fields['last_name'].widget.attrs['class'] = 'form-control'
        self.fields['email'].widget.attrs['class'] = 'form-control'
        self.fields['username'].widget.attrs['class'] = 'form-control'
        self.fields['password1'].widget.attrs['class'] = 'form-control'
        self.fields['password2'].widget.attrs['class'] = 'form-control'

```

Sljedeće je potrebno napraviti HTML datoteku za prikaz registracijskog obrasca, u ovom slučaju naziva `register.html`. U ovoj datoteci definira se dizajn stranice, te svako polje u obrascu ima pripadajuću oznaku (`label`) koja opisuje što korisnik treba unijeti. Ako korisnik unese ne važeće podatke, prikazat će mu se poruka o grešci ispod odgovarajućeg polja. Na samom dnu nalaz se gumb `REGISTRIRAJ SE` kako bi se korisnik mogao registrirati. Sljedeći kod nalazi se u datoteci `register.html` i prikazuje definiciju obrasca na stranici:

```

<div class="container">
  <div class="row justify-content-center">
    <div class="col-md-6">
      <div class="card shadow-lg p-3 mb-5 bg-white rounded">
        <div class="card-body">
          <h2 class="text-center">Registracija</h2>
          <form method="POST">
            {% csrf_token %}
            <div class="form-group mb-3">
              <label for="first_name" class="form-label">Ime:</label>
              {{ form.first_name }}
              {% if form.first_name.errors %}

```

```

        <div class="text-danger">
            {{ form.first_name.errors|join:", " }}
        </div>
    {% endif %}
</div>
<div class="form-group mb-3">
    <label for="last_name" class="form-label">Prezime:</label>
    {{ form.last_name }}
    {% if form.last_name.errors %}
        <div class="text-danger">
            {{ form.last_name.errors|join:", " }}
        </div>
    {% endif %}
</div>
<div class="form-group mb-3">
    <label for="email" class="form-label">Email:</label>
    {{ form.email }}
    {% if form.email.errors %}
        <div class="text-danger">
            {{ form.email.errors|join:", " }}
        </div>
    {% endif %}
</div>
<div class="form-group mb-3">
    <label for="username" class="form-label">Korisničko ime:</label>
    {{ form.username }}
    {% if form.username.errors %}
        <div class="text-danger">
            {{ form.username.errors|join:", " }}
        </div>
    {% endif %}
</div>
<div class="form-group mb-3">
    <label for="password1" class="form-label">Zaporka:</label>
    {{ form.password1 }}
    {% if form.password1.errors %}
        <div class="text-danger">
            {{ form.password1.errors|join:", " }}
        </div>
    {% endif %}
</div>
<div class="form-group mb-3">
    <label for="password2" class="form-label">Potvrdite zaporku:</label>

```

```

    {{ form.password2 }}
    {% if form.password2.errors %}
        <div class="text-danger">
            {{ form.password2.errors|join:", " }}
        </div>
    {% endif %}
</div>
<div style="text-align: center;">
    <button type="submit" class="btn btn-primary" style="background-
color: #0244fb;" >REGISTRIRAJ SE</button>
</div>
</form>
</div>
</div>
</div>
</div>
</div>

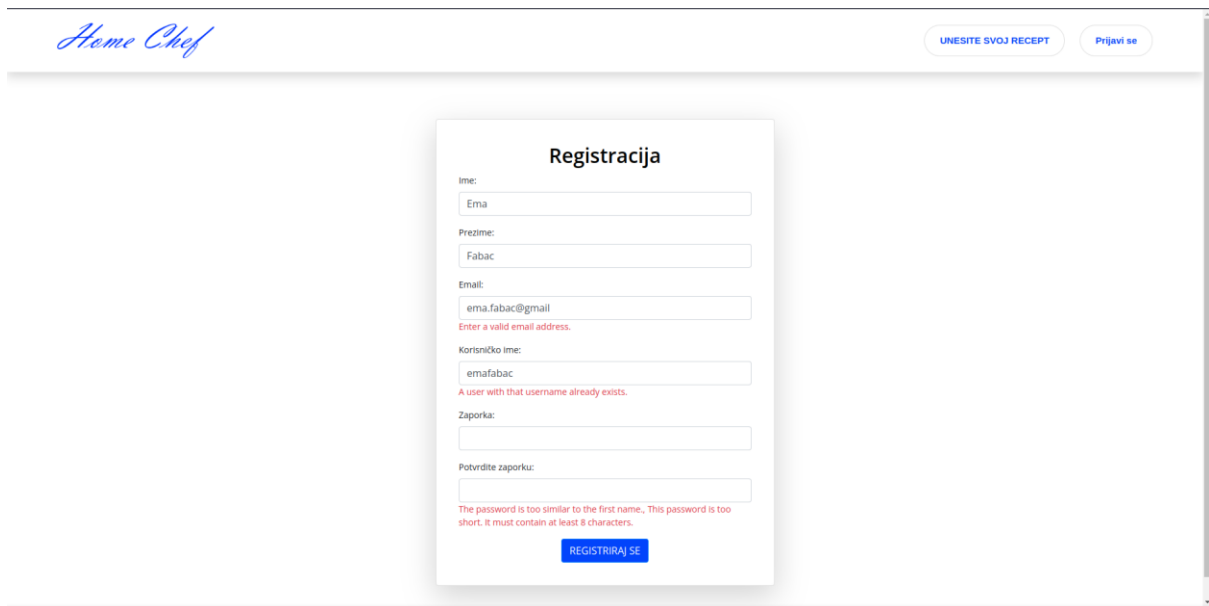
```

Posljednje što se treba učiniti kako bi se stranica prikazivala je unutar *urls.py* definirati lokaciju na kojoj će se stranca nalaziti. U ovom slučaju stranica se nalazi na linku *register/*. Kod za definiranje linka je sljedeći:

```
path('register/', views.register, name='register'),
```

Slika 14. prikazuje izgled obrasca za prijavu, dok Slika 15. prikazuje neke od mogućih poruka o grešci koje se ispisuju, kao što su da je mail adresa nevažeća, da korisnik s istim korisničkim imenom već postoji, da je lozinka prekratka te da treba imati najmanje osam znakova.

Slika 14. Prikaz obrasca za registraciju korisnika



Slika 15. Prikaz poruka o grešci kod registracije korisnika

## 5.7. Početna

Početna stranica je prva stranica s kojom korisnik ima kontakt i od nje sve kreće. Na njoj nalazi se prostor za pretraživanje recepta po nazivu, filtriranje recepta po kategorijama, te odabir sastojaka koji se nalaze u receptu i popis recepata. Kako bi se stranica prikazivala najprije je potrebno definirati *views.py*. Prvo je definirana klasa *Pocetna*, gdje se određuje da će se prikazivati objekti modela *Recept* i predložak *pocetna.html*. Nakon toga se metoda *get\_context\_data* poziva kako bi se spremili podaci za kontekst predloška. Ovdje dohvaćamo sve objekte iz modela *KategorijaRecepta*, *KategorijaSastojka*, *Recept* i *Recenzija*. Osim ove klase u *views.py* datoteci definirana je i funkcija *Filter* kako bi se mogli filtrirati recepti. Najprije se dohvaća listu odabranih kategorija koja se spremila u *kate[]* polje i odabranih sastojka koji su spremljeni u *sast[]* polje. Potom se filtriraju recepti da se dobiju samo oni koji se nalaze u kategoriji koja je označena i koji sadržavaju barem jedan sastojak koji je odabran. Nakon filtriranja vraća se odgovor s predloškom *product-list.html* s filtriranim receptima. Sljedeći kod prikazuje *views.py*:

```
class Pocetna(ListView):
    model = Recept
    template_name = 'pocetna.html'
    context_object_name = 'recept'
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        context['kategorije'] = KategorijaRecepta.objects.all()
        context['kategorije_sastojaka'] = KategorijaSastojaka.objects.all()
        context['recept'] = Recept.objects.all()
        context['komentari'] = Recenzija.objects.all()
        return context
```

```

def Filter(request):
    kategorije= request.GET.getlist("kate[]")
    sastojci = request.GET.getlist("sast[]")
    recept = Recept.objects.distinct()
    if len(kategorije) > 0:
        recept =
recept.filter(kategorija_recepta__id_kategorije_recepta__in=kategorije).distinct()
    if len(sastojci) > 0:
        sastojak_ids = [int(sastojak) for sastojak in sastojci]
        sastojak_query = Q()
        for sastojak in sastojak_ids:
            sastojak_query |= Q(sadrzi__id_sastojka=sastojak)
        recept = recept.filter(sastojak_query).distinct()
    data= render_to_string('product-list.html', {"recept": recept})
    return JsonResponse({"data": data})

```

Nakon definiranog *views.py* datoteke potrebno je kreirati *pocetna.html* i *product-list.html* datoteku. U datoteci *pocetna.html* najprije je definirano pretraživanje koje se nalazi na sredini stranice. Kako bi pretraživanje funkcionirao potrebno je definirati JavaScript kod. Definirano je da kada korisnik pritisne Enter u polje za pretraživanje, nakon unosa pojma za pretraživanje prikazuje se odgovarajući rezultat. Provjerava se naziv recepta te ako recept sadrži pojam onda se on prikazuje dok ako ne sadrži pojam recept se skriva. Ako nema recepta za upisani pojam ispisuje se poruka *Nema ponuđenih recepta!* te ako su pronađeni recepti ta se poruka skriva. Sljedeći kod prikazuje html za pretraživanje:

```

<div class="row">
  <div class="col-md-3" > </div>
  <div class="col-md-6" >
    <div class="row justify-content-center">
      <div class="col-12 col-md-10 col-lg-8">
        <form class="form-inline my-10 my-lg-0">
          <input id="search-input" class="form-control" style="width:
100%; height: 70px; border-radius: 50px; padding-left: 50px;" type="search"
placeholder="Unesite pojam koji želite pretražiti" aria-label="Search">
        </form>
      </div>
    </div>
  </div>
</div>
<div class="col-md-3" ></div>
</div>

```

Sljedeći kod prikazuje JavaScript za pretraživanje:

```

<script>
  const searchInput = document.getElementById('search-input');
  const recipesContainer = document.getElementById('filtered-recipes');
  const noResultsMessage = document.createElement('p');

```

```

noResultsMessage.textContent = 'Nema ponuđenih rezultata';
noResultsMessage.style.color = 'gray';
noResultsMessage.style.fontSize = '1.2em';
noResultsMessage.style.textAlign = 'center';
noResultsMessage.style.display = 'none';
recipesContainer.appendChild(noResultsMessage);
searchInput.addEventListener('keypress', (e) => {
  if (e.key === 'Enter') {
    e.preventDefault();
    const searchTerm = searchInput.value.trim().toLowerCase();
    const recipes = [...recipesContainer.children];
    let found = false;
    recipes.forEach((recipe) => {
      if (recipe.tagName === 'P') return;
      const recipeTitle =
recipe.querySelector('h5').textContent.toLowerCase();
      if (recipeTitle.includes(searchTerm)) {
        recipe.style.display = 'block';
        found = true;
      } else {
        recipe.style.display = 'none';
      }
    });
    if (!found) {
      noResultsMessage.style.display = 'block';
    } else {
      noResultsMessage.style.display = 'none';
    }
  }
});
</script>

```

Potom su u kodu definirani filteri sastojaka i kategorije. S lijeve strane najprije se nalazi padajući izbornik pod nazivom *KATEGORIJE* gdje se ispisuju nazivi svih kategorija. Korisnik može odabrati jednu ili više kategorija i prikazat će se oni recepti koji se nalaze u kategoriji. Kako su sastojci podijeljeni u kategorije ispisuju se nazivi svih kategorija sastojaka te odabirom jedne kategorije otvara se izbornik iz kojeg korisnik može izabrati jedan ili više sastojaka te će se prikazati svi oni sastojci koji sadržavaju barem jedan sastojak. Također vrijedi da se recepti filtriraju po sastojcima no ako se izabere kategorija, prikazat će se svi oni recepti koji se nalaze u odabranoj kategoriji i sadrže barem jedan od odabranih recepta. Sljedeći kod prikazuje HTML predložak za ispis filtera na lijevoj strani web aplikacije:

```

<section id="sidebar">
  <div class="box border-bottom">
    <div class="box-label text-uppercase d-flex align-items-
center">Kategorije

```

```

        <button class="btn ml-auto" style="border: black;" type="button"
data-toggle="collapse" data-target="#kategorije-inner-box" aria-expanded="false"
aria-controls="kategorije-inner-box">
            <span class="fa fa-angle-down"></span>
        </button>
    </div>
    <div id="kategorije-inner-box" class="collapse mt-2 mr-1">
        {% for k in kategorije %}
        <div class="my-1">
            <div class="checkbox-container">
                <input class="filter-checkbox" type="checkbox"
name="checkbox" data-filter="kate" id="example2" value="{{k.id_kategorije_recepta
}}">
                <label class="checkbox-label" style="margin-left:
30px; margin-top: 10px;">{{ k.naziv_kategorije_recepta }}</label>
            </div>
        </div>
        {% endfor %}
    </div>
    </div>
    {% for kategorija in kategorije_sastojaka %}
    <div class="box border-bottom">
        <div class="box-label text-uppercase d-flex align-items-center">{{
kategorija.naziv_kategorije_sastojaka }}
            <button class="btn ml-auto" style="border: black;"
type="button" data-toggle="collapse" data-target="#sastojci-inner-box-{{
forloop.counter }}" aria-expanded="false" aria-controls="sastojci-inner-box-{{
forloop.counter }}">
                <span class="fa fa-angle-down"></span>
            </button>
        </div>
    </div>
    <div id="sastojci-inner-box-{{ forloop.counter }}" class="collapse mt-2 mr-
1">
        <div class="row">
            {% for sastojci in kategorija.sastojci_set.all %}
            {% if sastojci %}
            <div class="col-auto mb-1">
                <label class="btn btn-outline-secondary btn-sm">
                    <input type="checkbox" name="checkbox" data-
filter="sast" value="{{ sastojci.id_sastojka }}" class="hidden form-check-input
filter-checkbox">
                    <span class="btn">{{ sastojci.naziv_sastojka
}}</span>
                </label>
            </div>
            {% endif %}
        </div>
    </div>

```



```

        {% empty %}
            <p class="col-12">Nema sastojka u ovoj kategoriji</p>
        {% endfor %}
    </div>
</div>
</div>
{% empty %}
    <p>Nije pronađena kategorija sastojka.</p>
{% endfor %}
</section>

```

Osim HTML-a za filtriranje repata koristi se i JavaScript koji se poziva kada je neki od sastojak ili kategorija odabrana. Tada kod dohvaća odabrane vrijednosti filtera i šalje AJAX zahtjev na server s filterima i ažurira prikaz recepata u aplikaciji. Ako nema ponuđenih recepata na stranci ispisuje se poruka *Nema ponuđenih recepata*. Sljedeći kod prikazuje JavaScript kod za filtriranje recepata po sastojcima i kategorijama:

```

<script>
    $(document).ready(function() {
        $(".filter-checkbox").on("click", function() {
            let filter_object={
                $(".filter-checkbox").each(function(index) {
                    let filter_value = $(this).val()
                    let filter_key = $(this).data("filter")
                    filter_object[filter_key] =
Array.from(document.querySelectorAll('input[data-
filter='+filter_key+']:checked')).map(function(element) {
                        return element.value
                    })
                })
            console.log(filter_object)
            $.ajax({
                url: '/filter',
                data: filter_object,
                dataType: 'json',
                beforeSend: function(){
                },
                success: function(response) {
                    console.log(response);
                    console.log('Data filter successfully...')
                    if (response.data.trim() === '') {
                        $("#filtered-recipes").html("<p>Nema pronađenih recepata</p>");
                    } else {
                        $("#filtered-recipes").html(response.data)
                    }
                }
            });
        });
    });

```

```

    }
  })
  })))
</script>

```

U HTML dokumentu je definiran prikaz popisa recepta u aplikaciji. Za svaki recept prikazuje se njegova slika, naziv i broj zvjezdica, što prikazuje sljedeći kod:

```

<div class="row" id="filtered-recipes">
  {% for recept in recept %}
  <div class="col-12 col-sm-6 col-lg-4">
    <div class="single-best-receipe-area mb-30">
      
      <div class="receipe-content" >
        <a href="{% url 'main:recept_details' recept.id_recepta
%}" >
          <h5>{{recept.naziv_recepta}}</h5>
          </a>
          {% with recept.recenzija_set.all as komentari %}
          {% with komentari|average_rating_for_recept:recept
as avg_rating %}
              {% for i in
avg_rating|round_half_up|rating_range %}
                  <span class="fa fa-star star-active mx-
1"></span>
              {% endfor %}
              {% for i in
avg_rating|round_half_up|inactive_range %}
                  <span class="fa fa-star star-inactive mx-
1"></span>
              {% endfor %}
          {% endwith %}
          {% endwith %}
        </div>
      </div>
    </div>
  {% endfor %}
</div>

```

Kod unutar *product-list.html* predložka se poziva kada se filtriraju recepti prema pretraživanju ili odabranim filterima. Tada se poziva ova datoteka za ispis recepata koji zadovoljavaju filtere. Sljedeći kod nalazi se u navedenoj datoteci:

```

{% for recept in recept %}
  <div class="col-12 col-sm-6 col-lg-4">
    <div class="single-best-receipe-area mb-30">
      

```

```

<div class="receipe-content">
    <a href="{% url 'main:recept_details' recept.id_recepta %}" >
        <h5>{{recept.naziv_recepta}}</h5>
    </a>
    {% with recept.recenzija_set.all as komentari %}
        {% with komentari|average_rating_for_recept:recept
as avg_rating %}
            {% for i in
avg_rating|round_half_up|rating_range %}
                <span class="fa fa-star star-active mx-
1"></span>
            {% endfor %}
            {% for i in
avg_rating|round_half_up|inactive_range %}
                <span class="fa fa-star star-inactive mx-
1"></span>
            {% endfor %}
        {% endwith %}
    {% endwith %}
</div>
</div>
</div>
{% endfor %}

```

Dodatno za funkcioniranje ove aplikacije bilo je potrebno definirati posebni filter za izračunavanje broja zvjezdica za svaki recept. Ovaj filter sprema se u datoteci *templetags/rating\_filter.py* koju je potrebno dodati pod instalirane aplikacije u *setting.py* datoteci. Također potrebno je i učitati datoteku na početku predloška. Time definira se funkcija *average\_ratings\_for\_recepies* koja prima dva argumenta, listu svih recenzija i pojedinačni recept. Najprije filtrira recenzije za dani recept, te provjerava da li postoje recenzije za recept te ako ih nema vraća 0 kao prosječnu ocjenu. Ako postoje recenzije za pojedini recept, stvara listu ocjena iz filtriranih recenzija te izračunava prosječnu ocjenu kao sumu svih recepata podijeljenu brojem ocjena. Na kraju ocjena se zaokružuje na jednu decimalu. Sljedeći kod prikazuje *average\_ratings\_for\_recepies* filter:

```

@register.filter
def average_rating_for_recept(komentari, recept):
    komentari_for_recept = komentari.filter(recept=recept)
    if not komentari_for_recept:
        return 0
    ratings = [komentar.ocjena for komentar in komentari_for_recept]
    avg_rating = sum(ratings) / len(ratings)
    return round(avg_rating, 1)

```

Posljednje što je potrebno napraviti kao bi sve uredno radilo je definirati lokacije stranica u *urls.py* datoteci. Definiraju se dva urla za početnu stranicu te za filtere. Kod definiranih urla je:

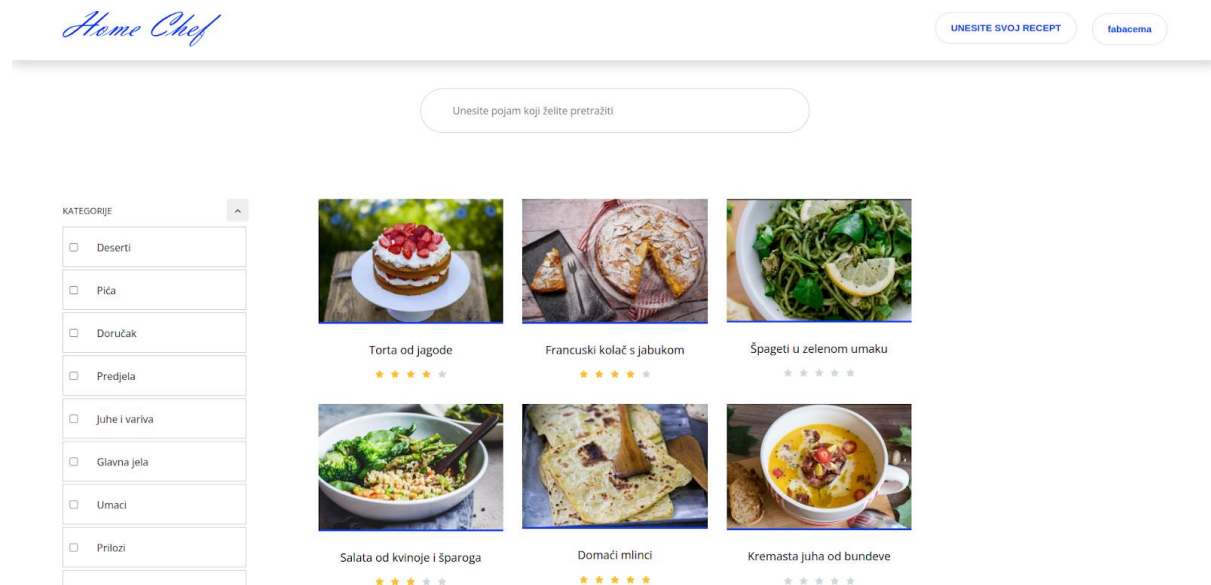
```

path('filter/', views.Filter, name='filter'),
path('', views.Pocetna.as_view(), name='pocetna'),

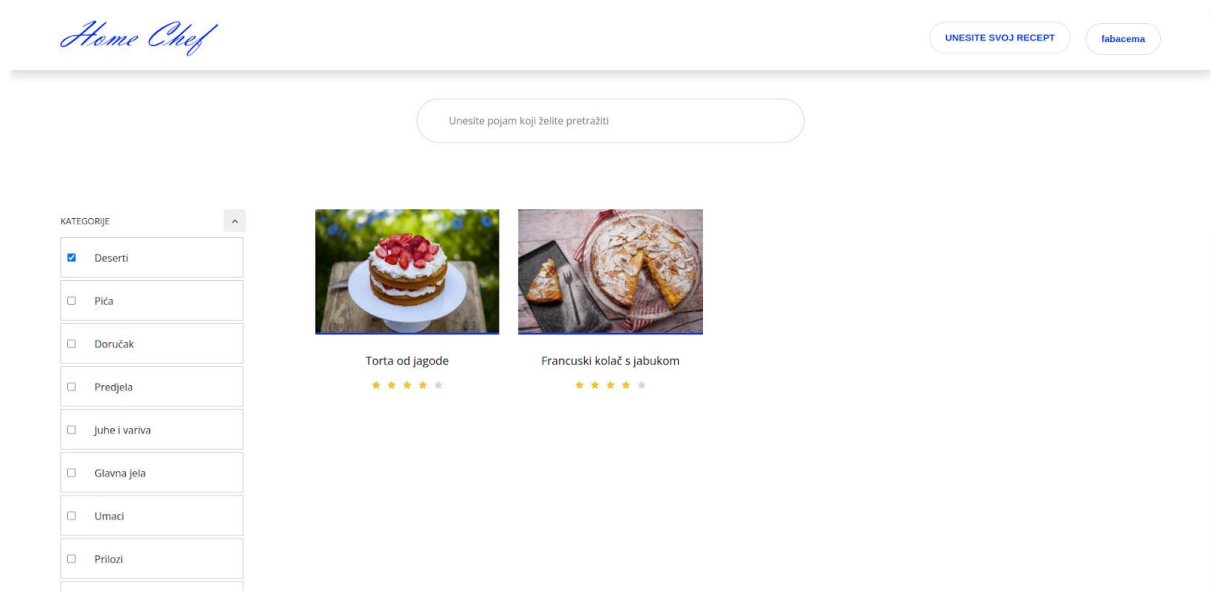
```

Za primjere slika i postupka pripreme recepta koriste se podaci sa *Index Recepti* stranice (<https://recepti.index.hr/>).

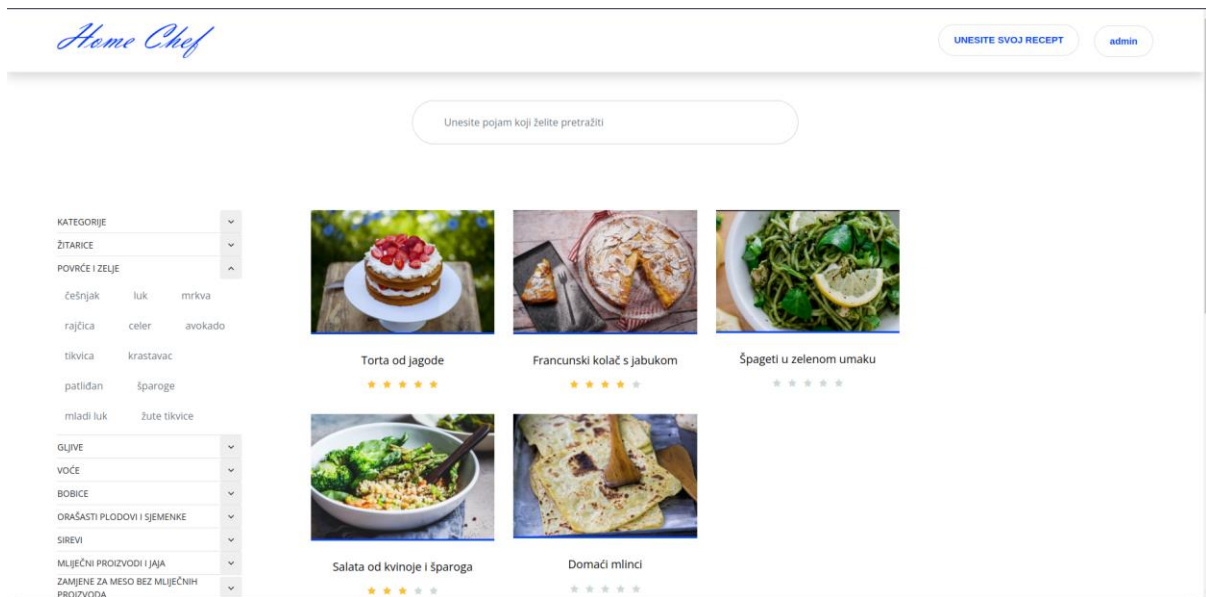
Slika 16. prikazuje početnu stranicu, dok Slika 17., Slika 18., Slika 19. i Slika 20. prikazuju kategorije i sastojke, te filtriranje recepata po određenim kategorijama i sastojcima. Slika 21. i Slika 22. prikazuju pretraživanje recepata.



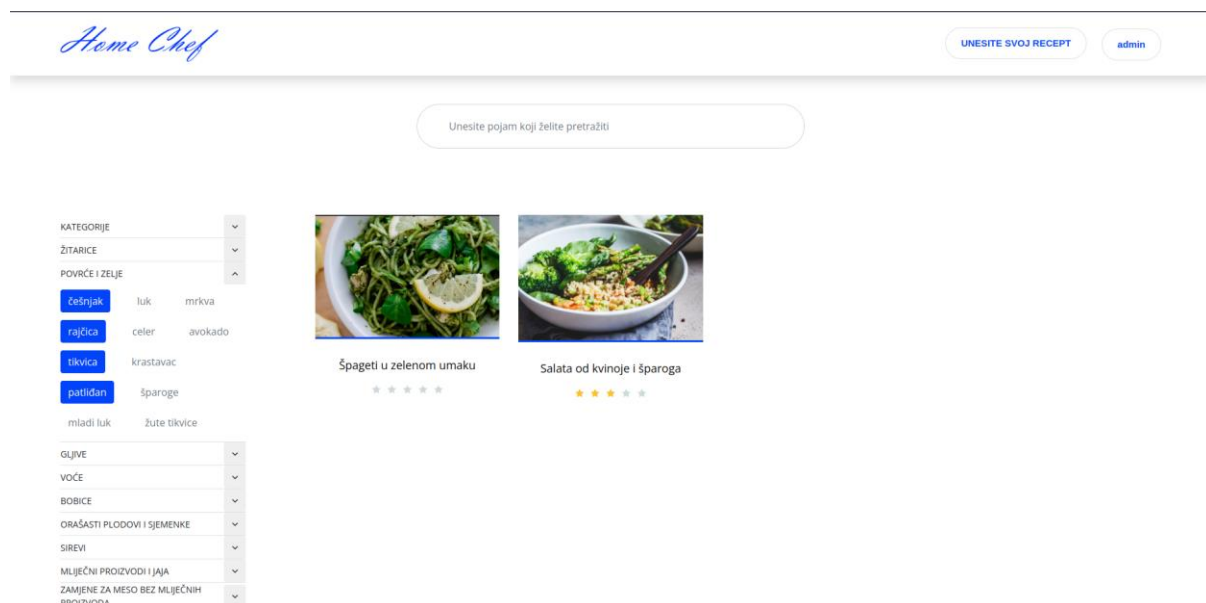
Slika 16. Prikaz početne stranice



Slika 17. Prikaz filtriranja recepata po kategoriji na početnoj stranici



Slika 18. Prikaz kategorija sastojaka na početnoj stranici



Slika 19. Prikaz filtriranja recepta po odabranim sastojcima

Unesite pojam koji želite pretražiti

- KATEGORIJE
- ŽITARICE
- POVRĆE I ZELJE
- češnjak luk mrkva
  - rajčica celer avokado
  - tikvica krastavac
  - patliđan **šparoge**
  - mliadi luk žute tikvice
- GLJIVE
- VOĆE
- BOBICE
- ORAŠASTI PLODOVI I SJEMENKE
- SIREVI
- MUJEČNI PROIZVODI I JAJA
- ZAMJENE ZA MESO BEZ MUJEČNIH PROIZVODA



Salata od kvinoje i šparoga



torta

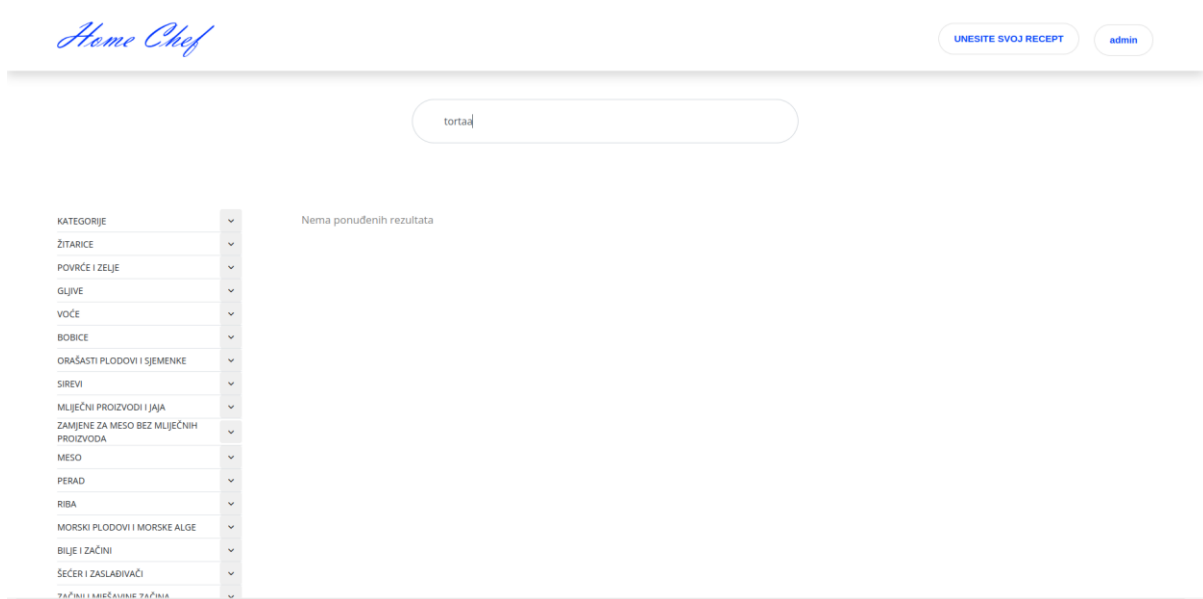
- KATEGORIJE
- ŽITARICE
- POVRĆE I ZELJE
- češnjak luk mrkva
  - rajčica celer avokado
  - tikvica krastavac
  - patliđan šparoge
  - mliadi luk žute tikvice
- GLJIVE
- VOĆE
- BOBICE
- ORAŠASTI PLODOVI I SJEMENKE
- SIREVI
- MUJEČNI PROIZVODI I JAJA
- ZAMJENE ZA MESO BEZ MUJEČNIH PROIZVODA



Torta od jagode



Slika 21. Prikaz rezultat pretraživanja recepta



Slika 22. Prikaz rezultata pretraživanja ukoliko nema niti jednog recepta

## 5.8. Detalji o receptu

Za svaki objavljeni recept postoji i dodatna stranica na kojoj se mogu vidjeti detaljnije informacije o receptu. Te informacije uključuju datum objave, korisnika koji je objavio recept, broj zvjezdica koje je recept dobio, naslov, postupak pripreme recepta, sve potrebne sastojke i recenzije koje su ostali korisnici ostavili. Za izradu ove stranice najprije je potrebno definirati funkciju u *views.py* datoteci, koja je u ovom slučaju nazvana *DetaljiRecepta*. u klasi najprije je definiran model *Recept* od kojega će se prikazivati detalji te definirana je i html datoteka *detalji\_recept.html* u kojoj će biti definiran izgled stranice i *context\_object\_name* je postavljen na *recept*, što znači da će objekt recepta biti dostupan u predlošku pod nazivom *recept*. Potom je definirana metoda *get\_context\_data* kako bi se moglo prikazati i dodatne podatke osim onih spremljenih u model *Recept*. U ovoj metodi dohvaća se model *Sadrzi* koji je povezan s receptom i stvara popis rječnika koji sadrže naziv sastojka, količinu i mjernu jedinicu. Također dohvaća se i objekt *Recenzija* koja je isto povezana s modelom *Recept* te u prikazu su poredane po vremenu objave. Za svaku instancu *Recepta* zbraja se i broj recenzija koje je ta instanca dobila te se zbraja i broj recenzija za svaku ocjenu od 1 do 5, što služi kako bi se u detaljima o recenzijama moglo vizualno prikazati koliko je ukupno recenzija te kako su one raspoređene po ocjenama. Korisnicima je omogućeno i dodavanje recenzija na svaki recept i time je sljedeći dio koda definiran za upravljanje dodavanja novih recenzija. Najprije se provjerava ako je metoda zahtjeva *POST*, onda se stvara instanca *ReviewForm* s poslanim podacima. Ako je obrazac valjan, poziva se metoda *form\_valid* u suprotnom dodaje se obrazac u kontekst te ako metoda nije *POST* stvara se prazna instanca *ReviewForm* i dodaje se u novi kontekst. Definirana je i metoda *post* kako bi rukovala slanjem popunjenog obrasca recenzija. Ako je obrazac valjan poziva se metoda *form\_valid* u suprotnom poziva se metoda *form\_invalid*. *form\_valid* metoda dohvaća objekt *Recept* koristeći *get\_object* te stvara novi objekt *recenzija* koristeći podatke iz obrasca, ali ga zasad ne sprema. Već dohvaća objekt *Korisnik* za trenutnog korisnika te dodjeljuje objekt *Korisnik* i *Recept* objektu *Recenzija* i sprema ga. Na kraju preusmjerava korisnika na stranicu o detaljima recepta, odnosno samo se ponovo učita stranica kako bi se

prikazao novo dodani recept. Sljedeći kod prikazuje klasu *DetaljiRecepta* definiranu u *views.py* datoteci:

```
class DetaljiRecepta(DetailView):
    model = Recept
    template_name = 'detalji_recepta.html'
    context_object_name = 'recept'
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        recept = self.get_object()
        sadrzi = Sadrzi.objects.filter(id_recepta=recept)
        context['sadrzi'] = []
        for s in sadrzi:
            mjerna_jedinica = s.id_mjerne_jedinice.naziv_mjerne_jedinice if
s.id_mjerne_jedinice else ''
            context['sadrzi'].append({
                'sastojak': s.id_sastojka.naziv_sastojka,
                'kolicina': s.kolicina,
                'mjerna_jedinica': mjerna_jedinica
            })
        context['komentar_i'] = Recenzija.objects.filter(recept=recept).order_by("-
vrijeme_recenzije")
        komentari = Recenzija.objects.filter(recept=recept)
        rating_counts =
komentari.values('ocjena').annotate(count=Count('ocjena')).order_by('ocjena')
        rating_dict = {}
        for rating in rating_counts:
            rating_dict[rating['ocjena']] = rating['count']
        context['excellent_count'] = rating_dict.get(5, 0)
        context['good_count'] = rating_dict.get(4, 0)
        context['average_count'] = rating_dict.get(3, 0)
        context['poor_count'] = rating_dict.get(2, 0)
        context['terrible_count'] = rating_dict.get(1, 0)
        if self.request.method == 'POST':
            form = ReviewForm(self.request.POST)
            if form.is_valid():
                return self.form_valid(form)
        else:
            form = ReviewForm()
            context['form'] = form
        return context
    def post(self, request, *args, **kwargs):
        form = ReviewForm(self.request.POST)
        if form.is_valid():
            return self.form_valid(form)
```



```

    else:
        return self.form_invalid(form)
def form_valid(self, form):
    receipt = self.get_object()
    komentar = form.save(commit=False)
    komentar.receipt = receipt
    racun = Korisnik.objects.get(user=self.request.user.id)
    komentar.korisnik = racun
    komentar.save()
    return redirect('main:receipt_details', pk=receipt.pk)
def form_invalid(self, form):
    return self.render_to_response(self.get_context_data(form=form))

```

Kako bi se u klasi *DetaljiRecenzije* mogla koristiti *ReviewForm* potrebno ju je prije definirati u *forms.py* datoteci. U formi najprije se povezuje s modelom *Recenzija*, te se pozivaju polja koja će se prikazivati u obrascu. Za polje *ocjena* definirano je *IntegerField*, te su postavljena ograničenja da je najveća vrijednost 5, a najmanje 1, što znači da se mogu unijeti samo ocjene od 1 do 5. Također dodan je placeholder s tekstom *Ocjena:1 do 5* kako bi korisnicima bilo jasnije što trebaju unijeti. Za polje *tekst\_ocjene* definiran je *CharField*, te koristi se *Textarea* za prikaz polja dimenzija 30 stupaca i 10 redaka. Također kao i za ocjenu postavljen je placeholder na tekst *Komentar*. Sljedeći kod prikazuje navedenu formu:

```

class ReviewForm(forms.ModelForm):
    class Meta:
        model = Recenzija
        fields = ('ocjena', 'tekst_recenzije')
    ocjena = forms.IntegerField(
        min_value=1,
        max_value=5,
        widget=NumberInput(attrs={
            'class': 'form-control',
            'placeholder': 'Ocjena: 1 do 5'
        })
    )
    tekst_recenzije = forms.CharField(
        widget=Textarea(attrs={
            'class': 'form-control',
            'id': 'message',
            'cols': 30,
            'rows': 10,
            'placeholder': 'Komentar'
        })
    )

```

Sljedeće što je potrebno napraviti je kreirati *detalji\_recepta.html* predložak za prikaz stranice o detaljima recepta na web aplikaciji. Na samom početku web aplikacije definirana je slika

recepta koja se dohvaća iz baze podataka s pomoću *recept.slika\_recepta.url*. Odmah ispod su prikazani datum objave recepta u formatu dd.MM.yyyy., gdje je za mjesec definiran hrvatski naziv i korisnik koji je objavio recept. Ispod toga nalazi se naziv recepta te informacije o broju osoba za koje je recept namijenjen i vrijeme potrebno za pripremu recepta u minutama. S desne strane prikazuju se zvjezdice za ocjenu recepta te se koristi for petlja koja prolazi kroz raspon ocjena i prikazuje aktivne zvjezdice i petlja koja prikazuje neaktivne zvjezdice. Sljedeći kod u HTML obrascu prikazuje definirani dio:

```
<div class="container" style="height: 50%;">
  <div class="row" style="height: 30%;">
    <div class="col-12" style="height: 100%;">
      
    </div>
  </div>
</div>
<div class="row">
  <div class="col-12 col-md-8">
    <div class="recept-headline my-5">
      <span>
        {{ recept.datum_objave|date:"d." }}
        {% with month=recept.datum_objave|date:"m" %}
          {% if month == "01" %}siječanj
          {% elif month == "02" %}veljača
          {% elif month == "03" %}ožujak
          {% elif month == "04" %}travanj
          {% elif month == "05" %}svibanj
          {% elif month == "06" %}lipanj
          {% elif month == "07" %}srpanj
          {% elif month == "08" %}kolovoz
          {% elif month == "09" %}rujan
          {% elif month == "10" %}listopad
          {% elif month == "11" %}studeni
          {% elif month == "12" %}prosinac
          {% endif %}
        {{ recept.datum_objave|date:"Y." }}
        {% endwith %}
        korisnik {{recept.id_korisnika.user}}
      </span>
      <h2>{{recept.naziv_recepta}}</h2>
      <div class="recept-duration">
        <h6>Osoba: {{recept.broj_osoba}}</h6>
        <h6>Vrijeme: {{recept.vrijeme_pripreme}} min</h6>
      </div>
    </div>
  </div>
</div>
```

```

        </div>
    </div>
    <div class="col-12 col-md-4">
        <div class="receipe-ratings text-right my-5">
            <div class="ratings">
                {% for i in
komentari|average_rating|round_half_up|rating_range %}
                    <span class="fa fa-star star-active mx-1"></span>
                {% endfor %}
                {% for i in
komentari|average_rating|round_half_up|inactive_range %}
                    <span class="fa fa-star star-inactive mx-1"></span>
                {% endfor %}
            </div>
        </div>
    </div>
</div>

```

Slika 23. prikazuje prikaz slike recepta te informacije o datumu i korisniku objave, naslovu recepta te broj osoba, vrijeme pripreme recepta i njegove zvjezdice.



03. srpanj 2024. korisnik emafabac

## Torta od jagode



Osoba: 8  
Vrijeme: 50 min

Slika 23. Prikaz osnovnih detalja o receptu

Nakon osnovnih informacija o receptu prikazuju se postupak pripreme i sastojci recepta. Kod unosa novog recepta od korisnika se tražilo da svaki novi postupak napiše u novom redu baš iz razloga kako bi se ovaj dio koda mogao izvršavati i sve točno prikazati. Za prikaz postupka koristi se petlja koja prolazi kroz korake pripreme recepta odvojene novim redom, te se pomoću *forloop.counter* za svaki novi red prikazuje redni broj koraka uz koji se prikazuje tekst koraka pripreme. S desne strane stupca za pripremu recepta nalazi se stupca za prikaz sastojka. Za njihov prikaz koristi se for petlja koja prolazi kroz listu sastojaka te ih ispisuje u formatu naziv sastojka: količina i mjerna jedinica. Dodatno kako je jedino naziv sastojka obavezno polje, odnosno ostala polja mogu i ostati prazna, koriste se if- else metoda za dobivanje odgovarajućeg prikaza za svaki od mogućih slučajeva. Sljedeći kod u HTML datoteci prikazuje definirani dio:

```
<div class="row">
    <div class="col-12 col-lg-8">
        <h4>Priprema</h4>
        <br>
        {% for step in recept.priprema_recepta|split_by_newline %}
        <div class="single-preparation-step d-flex">
            <h4>{{ forloop.counter }}.</h4>
            <p>{{ step }}</p>
        </div>
        {% endfor %}
    </div>
    <div class="col-12 col-lg-4">
        <div style="margin-left: 100px;">
            <h4>Sastojci</h4>
            <br>
            {% for s in sadrzi %}
            <div>
                <label style="font-weight:bold; padding-bottom: 5px;">
                    {{ s.sastojak }}
                    {% if s.kolicina and s.mjerna_jedinica %}
                        : {{ s.kolicina }} {{ s.mjerna_jedinica }}
                    {% elif s.kolicina %}
                        : {{ s.kolicina }}
                    {% elif s.mjerna_jedinica %}
                        : {{ s.mjerna_jedinica }}
                    {% endif %}
                </label>
            </div>
            {% endfor %}
        </div>
    </div>
</div>
```

Slika 24. prikazuje izgled pripreme i sastojaka na web stranici.

### Priprema

1. Jaja miksajte sa šećerom da nastane pjenasta i svijetla smjesa. Dodajte vodu, glatko brašno i prašak za pecivo. Izmišljajte da nestanu eventualne grudice.
2. Okrugli kalup za tortu premažite s malo ulja ili maslaca (to radimo da se papir za pečenje zaljeplj) i obložite papirom za pečenje. Izlijte smjesu, poravnajte i stavite peći u zagrijanu pećnicu na 180 stupnjeva 20 minuta. Pečeni biskvit ohladite i prerežite na pola da dobijete dva tanja biskvita.
3. Polovicu jagoda narežite na četvrtine i pomiješajte sa šećerom. Drugu polovicu narežite na pola, one će ići na vrh torte. Slatko vrhnje izmiksajte u šlag.
4. Prvi dio biskvita prelijte s malo soka koje su pustile jagode. Premažite s polovicom tučenog slatkog vrhnja i stavite jagode koje ste pomiješali sa šećerom. Poklopite drugom polovicom biskvita pa premažite ostatkom tučenog slatkog vrhnja. Ukrasite s jagodama koje ste prerezali na pola. Ohladite prije rezanja, bit će dovoljno i sat vremena, čisto da se šlag malo stisne i biskvit natopi sokom od jagoda.

### Sastojci

jaja : 5  
šećer : 180 g  
paprika : 5 žlica  
prašak za pecivo : 1 žličica  
jagoda : 500 g  
slatko vrhnje : 500 ml

Slika 24. Prikaz pripreme i sastojka na stranici detalja o receptu

Nakon prikaza pripreme i sastojaka nalazi se prikaz recenzija koji se sastoji od forme za dodavanje nove recenzije, lista već objavljenih recenzija i kratkog prikaza detalja o recenzijama. Prvo je definirana forma za dodavanje recenzija i prikaz postojećih recenzija na stranici. Za dodavanje recenzija koristi se obrazac gdje korisnik može unijeti ocjenu od 1 do 5 i komentar. Dodavanje recenzija je omogućeno samo za korisnike koji su prijavljeni u sustav time se prije prikaza forme provjerava je li korisnik prijavljen, ako nije obrazac se ne prikazuje. Dok za prikaz postojećih recenzija prikazuje se ime i prezime korisnika koji je objavio recenziju, vrijeme objave, prikaz ocjene u obliku zvjezdica i tekst recenzije te ako nema niti jedne recenzije prikazuje se tekst *Trenutno nema recenzija!* i u tom slučaju prikazuje se 0 aktivnih zvjezdica. Za prikaz postojećih recenzija poziva se template *recenzije.html* u kojem je definiran dizajn prikaza recenzija. S desne strane nalazi se kartica s prosječnom ocjenom i prikaz raspodjele ocjena. Kartica s prosječnom ocjenom prikazuje prosječnu ocjenu svih dobivenih ocjena za određeni recept, računa se kao zbroj svih ocjena podijeljen s ukupnim brojem recenzija te se prikazuje kao broj od 1 do 5. Dobivena ocjena prikazuje se s pomoću zvjezdica, tako se broj zvjezdica određuje da se zaokružuje na najbliži cijeli broj, npr. ako je ocjena od 1 do 1.49 prikazat će se samo jedna zvjezdica dok ako je od 1.5 do 2 prikazat će se dvije zvjezdice. U donjem dijelu kartice nalazi se prikaz raspodjele ocjena. Prikazuju se različite kategorije ocjena a one su: odličan, dobar prosječan, loš i užasan. Za svaku kategoriju prikazan je broj recenzija i grafički prikaz udjela te kategorije u ukupnom broju recenzija. Grafički prikaz se ostvaruje s pomoću *bar-container* i *bar-5* elementa čija se širina temelji na izračunatim varijablama *excellent\_count*, *good\_count*, *average\_count*, *poor\_count* i *terrible\_count*. Svaka od ovih varijabli prikazuje broj recenzija za svaku ocjenu od 1 do 5. Sljedeći kod iz HTML datoteke prikazuje definirano:

```

<div class="row">
    <div class="col-md-8">
        {% if user.is_authenticated %}
        <div class="row">
            <div class="col-12">
                <div class="contact-form-area">
                    <form method="POST" action="{% url
'main:recept_details' pk=recept.id_recepta %}">
                        {% csrf_token %}
                        {{ form.as_p }}
                        <input type="hidden" name="rec_id"
value="{{ recept.id_recepta }}">
                        <div class="row">
                            <div class="col-12">
                                <button class="btn
delicious-btn mt-30" type="submit">Dodaj recenziju</button>
                                </div>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>
        {% endif %}
        <div class="review-list">
            {% include 'recenzije.html' with
komentarikomentari %}
        </div>
        <div class="col-md-4">
            <div class="card">
                <div class="row justify-content-left d-flex">
                    <div class="col-md-12 d-flex flex-column
align-items-center">
                        <div class="rating-box text-
center">
                            <h1 class="pt-4">{{
komentarikomentari|average_rating }}
                            </h1>
                            <p class="">od 5</p>
                        </div>
                        <div class="text-center">
                            {% for i in
komentarikomentari|average_rating|round_half_up|rating_range %}
                                <span class="fa fa-star
star-active mx-1"></span>
                            {% endfor %}
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>

```

```

                                {% for i in
komentari|average_rating|round_half_up|inactive_range %}
                                <span class="fa fa-star
star-inactive mx-1"></span>
                                {% endfor %}
                                </div>
                                </div>

                                {% with
total_count=excellent_count|add:good_count|add:average_count|add:poor_count|add:ter
rible_count %}

                                <div class="rating-bar0 justify-
content-center">
                                <table class="text-left mx-
auto">
                                <tr>
                                <td class="rating-
label">Odličan</td>
                                <td class="rating-bar">
                                <div class="bar-
container">
                                <div class="bar-5"
style="width: {% if total_count > 0 %}{{
excellent_count|width_percentage:total_count }}{% else %}0{% endif %}"></div>
                                </div>
                                </td>
                                <td class="text-right">{{
excellent_count }}</td>
                                </tr>
                                <tr>
                                <td class="rating-
label">Dobar</td>
                                <td class="rating-bar">
                                <div class="bar-
container">
                                <div class="bar-5"
style="width: {% if total_count > 0 %}{{ good_count|width_percentage:total_count
}}{% else %}0{% endif %}"></div>
                                </div>
                                </td>
                                <td class="text-right">{{
good_count }}</td>
                                </tr>
                                <tr>
                                <td class="rating-
label">Prosječan</td>
                                <td class="rating-bar">
                                <div class="bar-
container">

```

```

                                <div class="bar-5"
style="width: {% if total_count > 0 %}{{ average_count|width_percentage:total_count
}}{% else %}0{% endif %}"></div>
                                </div>
                                </td>
                                <td class="text-right">{{
average_count }}</td>
                                </tr>
                                <tr>
                                <td class="rating-
label">Loš</td>
                                <td class="rating-bar">
                                <div class="bar-
container">
                                <div class="bar-5"
style="width: {% if total_count > 0 %}{{ poor_count|width_percentage:total_count
}}{% else %}0{% endif %}"></div>
                                </div>
                                </td>
                                <td class="text-right">{{
poor_count }}</td>
                                </tr>
                                <tr>
                                <td class="rating-
label">Užasan</td>
                                <td class="rating-bar">
                                <div class="bar-
container">
                                <div class="bar-5"
style="width: {% if total_count > 0 %}{{
terrible_count|width_percentage:total_count }}{% else %}0{% endif %}"></div>
                                </div>
                                </td>
                                <td class="text-right">{{
terrible_count }}</td>
                                </tr>
                                </table>
                                </div>
                                {% endwith %}
                                </div>
                                </div>
                                </div>

```

Sljedeći kod se nalazi u *recenzije.html* datoteci za prikaz popisa postojećih recenzija:

```

{% for komentar in komentari %}
                                <div class="card">
                                <div class="row d-flex">

```



```

                                <div class="d-flex flex-row">
                                    <h3 class="mt-2 mb-
0">{{komentar.korisnik.user.first_name}} {{komentar.korisnik.user.last_name}}</h3>
                                    <div>
                                        <p style="margin-top: 10px; margin-
left: 10px;">

                                                {% for i in
komentar.ocjena|rating_range %}
                                                    <span class="fa fa-star
star-active"></span>
                                                        {% endfor %}
                                                {% for i in
komentar.ocjena|inactive_range %}
                                                    <span class="fa fa-star
star-inactive"></span>
                                                        {% endfor %}
                                                <span class="text-muted">({{
komentar.ocjena }})</span>
                                                    </p>
                                        </div>
                                    </div>
                                <div class="ml-auto">
                                    <p class="text-muted pt-5
pt-sm-3">
                                                {{
komentar.vrijeme_recenzije|date:"d." }}
                                                    {% with
month=komentar.vrijeme_recenzije|date:"m" %}
                                                        {% if month == "01"
%}siječanj
                                                        {% elif month ==
"02" %}veljača
                                                        {% elif month ==
"03" %}ožujak
                                                        {% elif month ==
"04" %}travanj
                                                        {% elif month ==
"05" %}svibanj
                                                        {% elif month ==
"06" %}lipanj
                                                        {% elif month ==
"07" %}srpanj
                                                        {% elif month ==
"08" %}kolovoz
                                                        {% elif month ==
"09" %}rujan
                                                        {% elif month ==
"10" %}listopad

```

```

                                {% elif month ==
"11" %}studeni
                                {% elif month ==
"12" %}prosinac
                                {% endif %}
                                {{
komentar.vrijeme_recenzije|date:"Y." }} {{ komentar.vrijeme_recenzije|date:"H:i" }}
                                {% endwith %}
                                </p>
</div>
                                </div>
                                <div class="row text-left">
                                <p class="content">{{
komentar.tekst_recenzije }} </p>
                                </div>
</div>
<br>
{% empty %}
<br>
<p>Trenutno nema recenzija!</p>
{% endfor %}

```

Slika 25. prikazuje obrasca za dodavanje recenzije, listu već objavljenih recenzija te karticu o detaljima recenzija za prijavljenog korisnika

## Recenzije

Ocjena:

Ocjena: 1 do 5

Tekst recenzije:

Komentar

Dodaj Recenziju

Ana Anić ★★★★★ (4)

03. srpanj 2024. 14:18

Ukusna torta, prikladna za ovo vruće razdoblje

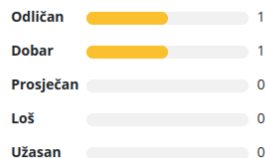
Luka Lukić ★★★★★ (5)

03. srpanj 2024. 14:16

Veoma ukusna torta!

4.5

od 5



Slika 25. Prikaz bloka za recenzije kod prijavljenog korisnika

Slika 26. prikazuje listu već objavljenih recenzija te karticu o detaljima recenzija za neprijavljenog korisnika

## Recenzije

Ana Anić ★★★★★ (4)

03. srpanj 2024. 14:18

Ukusna torta, prikladna za ovo vruće razdoblje

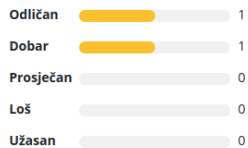
Luka Lukić ★★★★★ (5)

03. srpanj 2024. 14:16

Veoma ukusna torta!

4.5

od 5

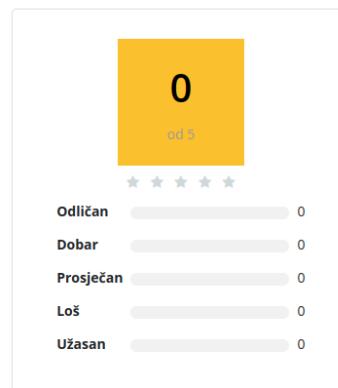


Slika 26. Prikaz bloka za recenzije kod neprijavljenog korisnika

Slika 27. prikazuje stranicu ukoliko nema niti jedne recenzije

## Recenzije

Trenutno nema recenzija!



Slika 27. Prikaz bloka za recenzije ukoliko nema niti jedne recenzije

Za prikaz i izračunavanje mnogih elemenata s ove web stranice definirani su vlastiti filteri koji se nalaze u `main/templatetags/rating_filters.py`. Kako bi se ove filtere moglo koristiti potrebno ih je učitati na samom početku stranice `{% load rating_filters %}`, te je potrebno i navesti direktorij `templatetags` kao instaliranu aplikaciju u `settings.py` datoteci unutra projekt `'main.templatetags'`. Filteri koji se koriste za prikaz podataka na ovoj stranici su: `rating_range`, `inactive_range`, `average_rating`, `round_half_up`, `split_by_newline` i `width_percentage`.

Filter `rating_range` koristi se za prikaz aktivnih zvjezdica i radi tako da za vrijednost `value` vraća listu brojeva od 1 do `value` koji predstavlja broj zvjezdica koje će biti aktivne. Ako je `value None` onda vraća praznu listu odnosno niti jedna zvjezdica neće biti aktivna odnosno ocjena recepta je 0. Sljedeći kod prikazuje `rating_range` filter:

```
@register.filter
def rating_range(value):
    if value is None:
        return []
    return range(1, int(value) + 1)
```

`inactive_range` radi suprotno od prethodnog filtera, odnosno on određuje koliko će se neaktivnih zvjezdica prikazati. Ovaj filter prima vrijednost `value` i vraća listu brojeva od 0 do 4 isključujući `value`, dok ako je `value None` vraća listu brojeva od 0 do 4. Sljedeći kod prikazuje `inactive_range` filter:

```
@register.filter
def inactive_range(value):
```

```

if value is None:
    return range(5)
return range(int(5 - value))

```

*average\_rating* filter prima listu *komentar* i izračunava prosječnu ocjenu. Ako je lista *komentar* prazna vraća 0. Izračun prosječne ocjene vrši se tako da se zbroje sve ocijene podijeljene s brojem komentara, te se rezultat zaokružuje na jednu decimalu. Sljedeći kod prikazuje *average\_rating* filter:

```

@register.filter
def average_rating(komentari):
    if not komentari:
        return 0
    ratings = [komentar.ocjena for komentar in komentari]
    return round(sum(ratings) / len(ratings), 1)

```

*round\_half\_up* filter prima vrijednost *value* i zaokružuje je na najbliži cijeli broj. Ako je decimalni dio veći ili jednak 0.5 zaokružuje se na veći cijeli broj, inače na manji. Ovaj filter se koristi kako bi dobili broj zvjezdica za prosječnu ocjenu recepta. Sljedeći kod prikazuje *round\_half\_up* filter:

```

@register.filter
def round_half_up(value):
    return math.ceil(value) if value % 1 >= 0.5 else math.floor(value)

```

*split\_by\_newline* filter prima vrijednost *value* i razdvaja je na listu podstringova koristeći znak reda '\n'. Ovaj filter koristi se za razdvajanje postupaka pripreme, tako da se svaki postupak nalazi u novom redu. Sljedeći kod prikazuje *split\_by\_newline* filter:

```

@register.filter
def split_by_newline(value):
    return value.split("\n")

```

*with\_percentage* filter prima dva argumenta *count* i *total\_count* i vraća postotak *count* u odnosu na *total\_count*, izražen kao cijeli broj. Ovaj se filter koristi kako bi se za kategorije recenzija od 1 do 5 moglo vizualno prikazati postotak pojedinih ocjena. Sljedeći kod prikazuje *width\_percentage*:

```

@register.filter
def width_percentage(count, total_count):
    return (count / total_count) * 100

```

Posljednje što se treba učiniti kako bi se stranica prikazivala je unutar *urls.py* definirati lokaciju na kojoj će se stranica nalaziti. U ovom slučaju stranica se nalazi na linku `<int:pk>/` gdje *pk* predstavlja primarni ključ recepta. Kod za definiranje linka je sljedeći:

```

path('<int:pk>/', views.DetaljiRecepta.as_view(), name='recept_details'),

```

## 5.9. Profil

Prilikom registracije u sustav korisniku se kreira profilna stranica. Na njoj se nalaze informacije o korisniku, slika te popis recepata koje je korisnik objavio. Također korisniku je ovdje i omogućeno brisanje i ažuriranje vlastitih recepata. Za izradu profile stranice najprije je definirana klasa u *views.py* datoteci naziva *ProfileView*. Ova klasa nasljeđuje *ListView* i predstavlja pogled za prikaz profila korisnika. Za dohvaćanje podataka definira se model *Korisnik* i HTML datoteka *profile.html*. U *get\_context\_data()* dohvaća se korisnik na temelju trenutno prijavljenog korisnika i dodaje ga se u kontekst. Također dohvaćaju se i svi recepti tog korisnika i dodaje ih se u kontekst pod ključem *recepts*. Sljedeći kod prikazuje *ProfileView* klasu:

```
class ProfileView(ListView):
    model = Korisnik
    template_name = 'profile.html'
    context_object_name = 'korisnik'
    def get_context_data(self, **kwargs):
        context = super().get_context_data(**kwargs)
        korisnik = Korisnik.objects.get(user=self.request.user)
        context['korisnik'] = Korisnik.objects.get(user=self.request.user)
        context['recepts'] = Recept.objects.filter(id_korisnika=korisnik)
        return context
```

Nadalje potrebno je kreirati *profile.html* datoteku za prikaz podataka o korisniku. Prvi dio koda prikazuje podatke o korisniku kao što su korisničko ime, ime, prezime, email i slika korisnika. Dodatno na kraju nalazi se gumb *Ažuriraj profil* koji omogućuje korisnicima da ažuriraju svoje osobne podatke. Također kako pri registraciji nije potrebno unositi profilnu fotografiju, modelu se automatski dodjeljuje osnovni prikaz avatara. Ako korisnik odluči postaviti vlastitu fotografiju to može napraviti prilikom ažuriranja recepta te će mu se prikazati nova profilna fotografija. Sljedeći kod prikazuje prvi dio koda u datoteci *profile.html* za prikaz podataka o korisniku:

```
<section class="w-100 px-4 py-3">
    <div class="row d-flex justify-content-center">
        <div class="col col-md-9 col-lg-7 col-xl-6">
            <div class="card" style="border-radius: 15px; box-shadow: 0 0 10px rgba(0,
0, 255, 0.2);">
                <div class="card-body p-3">
                    <div class="d-flex justify-content-center mb-3">
                        {% if korisnik.slika_korisnika %}
                            
                        {% else %}
                            
                    {% endif %}
                </div>
            </div>
        </div>
    </div>
```

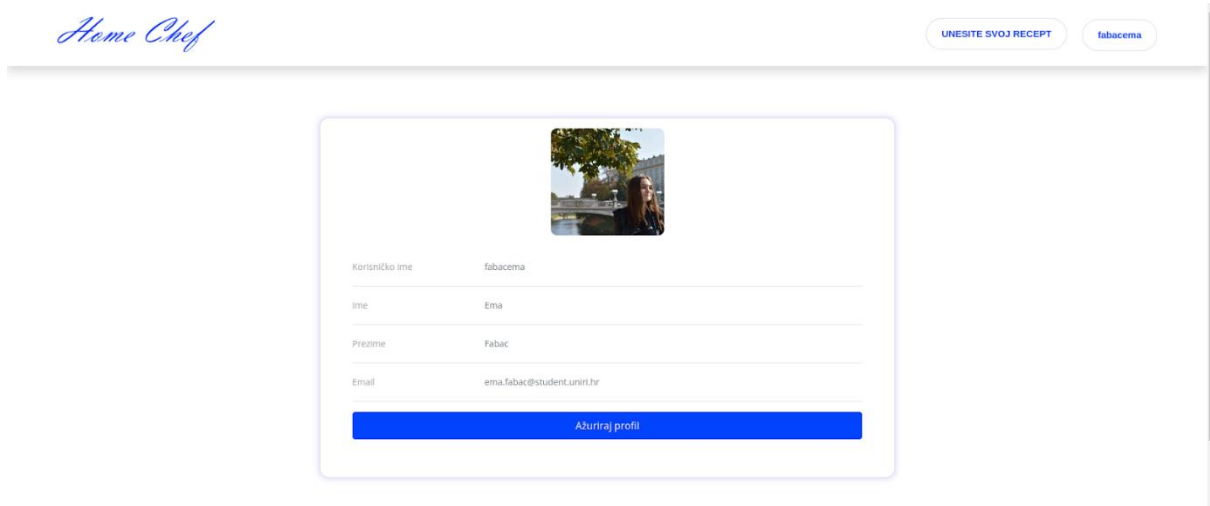
```

</div>
<div class="col-lg-12">
  <div class="card mb-4" style="border: none; box-shadow: none;">
    <div class="card-body">
      <div class="row">
        <div class="col-sm-3">
          <p class="mb-0">Korisničko ime</p>
        </div>
        <div class="col-sm-9">
          <p class="text-muted mb-0">{{ user.username }}</p>
        </div>
      </div>
      <hr>
      <div class="row">
        <div class="col-sm-3">
          <p class="mb-0">Ime</p>
        </div>
        <div class="col-sm-9">
          <p class="text-muted mb-0">{{ user.first_name }}</p>
        </div>
      </div>
      <hr>
      <div class="row">
        <div class="col-sm-3">
          <p class="mb-0">Prezime</p>
        </div>
        <div class="col-sm-9">
          <p class="text-muted mb-0">{{ user.last_name }}</p>
        </div>
      </div>
      <hr>
      <div class="row">
        <div class="col-sm-3">
          <p class="mb-0">Email</p>
        </div>
        <div class="col-sm-9">
          <p class="text-muted mb-0">{{ user.email }}</p>
        </div>
      </div>
      <hr><div >
        <a href="{% url 'main:update_profile' %}" class="btn btn-primary
btn-block">Ažuriraj profil</a>
      </div>

```

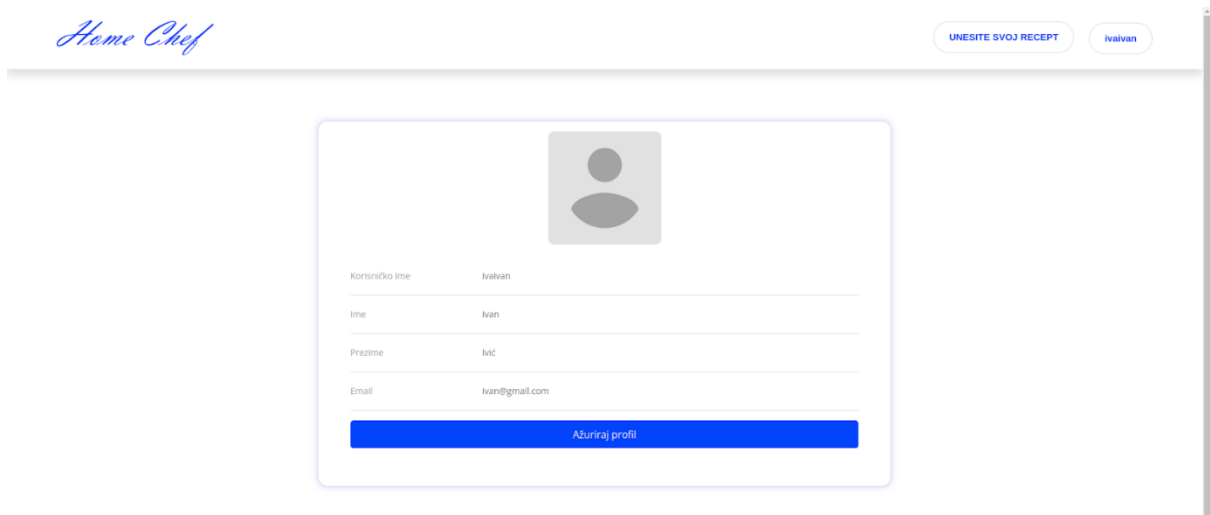
```
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</div>
</section>
```

Slika 28. prikazuje profil i detalje o korisniku koje je moguće mijenjati.



Slika 28. Prikaz podataka o korisniku kada je odabrana profilna fotografija

Slika 29. prikazuje korisnički profil ukoliko još nije odabran profilna fotografija.



Slika 29. Prikaz podataka o korisniku kada nije odabrana profilna fotografija

Kako bi se prilikom klika na gumb *Ažuriraj recept* otvorila nova stranica za ažuriranje recepata najprije je potrebno definirati funkciju *views.py* datoteci. U funkciji se najprije provjerava je li zahtjev *POST*, onda se ažuriraju podaci o korisniku. Također, ako je na zahtjev poslana nova



profilna slika, ona se sprema u model *Korisnik* i na kraju se spremaju promjene u bazu i preusmjeruje se korisnika na stranicu profila.

```
@login_required
def update_profile(request):
    if request.method == 'POST':
        user = request.user
        user.username = request.POST['username']
        user.first_name = request.POST['first_name']
        user.last_name = request.POST['last_name']
        user.email = request.POST['email']
        if 'profile_picture' in request.FILES:
            profile_picture = request.FILES['profile_picture']
            korisnik = Korisnik.objects.get(user=user)
            korisnik.slika_korisnika = profile_picture
            korisnik.save()
        user.save()
        return redirect('main:profile')
    return render(request, 'update_profile.html', {'korisnik':
Korisnik.objects.get(user=request.user)})
```

Za prikaz obrasca za ažuriranje osobnih podataka kreira se datoteka *update\_profile.html*. U njoj je definiran dizajn obrasca i pozvano je svako input polje s već gotovim obrascima te omogućuje korisniku njegovo ažuriranje. Sljedeći kod prikazuje obrazac za ažuriranje koda:

```
<form method="post" action="{% url 'main:update_profile' %}"
enctype="multipart/form-data">
    {% csrf_token %}
    <div class="container">
        <div class="row justify-content-center">
            <div class="col-lg-8">
                <div class="card mb-4">
                    <div class="card-body">
                        <h4 class="card-title">Ažuriraj profil</h4>
                        <hr>
                        <div class="form-group row">
                            <label for="profile_picture" class="col-sm-3 col-form-
label">Profilna fotografija</label>
                            <div class="col-sm-9">
                                <div class="custom-file">
                                    <input type="file" class="custom-file-input"
id="profile_picture" name="profile_picture">
                                    <label class="custom-file-label" for="profile_picture">Izaberi
sliku</label>
                                </div>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </div>
    </div>
```

```

        <div class="form-group row">
            <label for="username" class="col-sm-3 col-form-label">Korisničko
ime</label>
            <div class="col-sm-9">
                <input type="text" class="form-control" id="username"
name="username" value="{{ user.username }}">
            </div>
        </div>
        <div class="form-group row">
            <label for="first_name" class="col-sm-3 col-form-label">Ime</label>
            <div class="col-sm-9">
                <input type="text" class="form-control" id="first_name"
name="first_name" value="{{ user.first_name }}">
            </div>
        </div>
        <div class="form-group row">
            <label for="last_name" class="col-sm-3 col-form-
label">Prezime</label>
            <div class="col-sm-9">
                <input type="text" class="form-control" id="last_name"
name="last_name" value="{{ user.last_name }}">
            </div>
        </div>
        <div class="form-group row">
            <label for="email" class="col-sm-3 col-form-label">Email</label>
            <div class="col-sm-9">
                <input type="email" class="form-control" id="email" name="email"
value="{{ user.email }}">
            </div>
        </div>
        <div class="form-group row">
            <div class="col-sm-3"></div>
            <div class="col-sm-9">
                <button type="submit" class="btn btn-primary btn-block">Ažuriraj
profil</button>
            </div>
        </div>
    </div>
</div>
</div>
</div>
</div>
</div>
</form>

```

Slika 30. prikazuje obrazac za ažuriranje profila.

## Ažuriraj profil

Profilna fotografija	<input type="text" value="Izaberi sliku"/>	<input type="button" value="Browse"/>
Korisničko ime	<input type="text" value="fabacema"/>	
Ime	<input type="text" value="Ema"/>	
Prezime	<input type="text" value="Fabac"/>	
Email	<input type="text" value="ema.fabac@student.uniri.hr"/>	
<input type="button" value="Ažuriraj profil"/>		

Slika 30. Prikaz obrasca za ažuriranje podataka o korisniku

Ispod osnovnih informacija o korisniku nalazi se popis recepata koje je objavio. Za svaki recept prikazuje se slika, naziv i broj zvjezdica. Na slici svakog od recepta nalaze se i dva gumba *Izbriši* i *Ažuriraj* koji omogućuju korisniku da izbriše ili ažurira pojedini recept. Ako korisnik još nije objavio niti jedan recept prikazuje mu se poruka *Trenutno nemate niti jedan recept!*. Sljedeći kod prikazuje definiranu listu recepta na profilnoj stranici:

```

div class="section-header text-center mb-4">
    <h1>Objavljeni recepti</h1>
</div>

<div class="row" id="filtered-recipes">
    {% for recept in receipts %}
    <div class="col-12 col-sm-6 col-lg-4">
        <div class="single-best-recipe-area mb-30">
            <div class="image-container">
                
            <div class="icon-container">
                <a href="{% url 'main:update_recept2'
recept.id_recepta %}" class="btn btn-primary">
                    Ažuriraj
                </a>
                <a href="{% url 'main:delete_recept'
recept.id_recepta %}" class="btn btn-danger">
                    Izbriši
                </a>
            </div>
        </div>
    </div>
    </div>

```

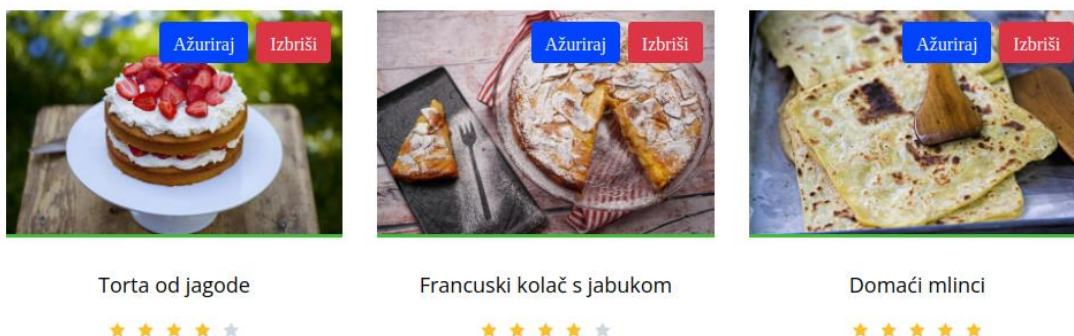
```

recept.id_recepta %}">
    <div class="receipe-content">
        <a href="{% url 'main:recept_details'
            <h5>{{ recept.naziv_recepta }}</h5>
        </a>
        {% with recept.recenzija_set.all as komentari %}
            {% with
komentari|average_rating_for_recept:recept as avg_rating %}
                {% for i in
avg_rating|round_half_up|rating_range %}
                    <span class="fa fa-star star-active mx-
1"></span>
                {% endfor %}
                {% for i in
avg_rating|round_half_up|inactive_range %}
                    <span class="fa fa-star star-inactive mx-
1"></span>
                {% endfor %}
            {% endwith %}
        {% endwith %}
        </div>
    </div>
</div>
{% empty %}
<br>
<p>Trenutno nemate niti jedan recept!</p>
{% endfor %}
</div>

```

Slika 31. prikazuje popis objavljenih recepata korisnika.

## Objavljeni recepti



Slika 31. Prikaz objavljenih recepata korisnika na profilnoj stranici

Slika 32. prikazuje izgled stranice ako korisnik nije objavio niti jedan recept.

# Objavljeni recepti

Trenutno nemate niti jedan recept!

Slika 32. Prikaz poruke ukoliko korisnik nema niti jedan objavljen recept

Posljednje što se treba učiniti kako bi se stranica prikazivala je unutar *urls.py* definirati lokaciju na kojoj će se stranica nalaziti. U ovom slučaju stranica se nalazi na linku *profile/* dok se za ažuriranja podataka o korisniku koristi link *profile/update-profile*. Kod za definiranje linka je sljedeći:

```
path('profile/update-profile/', views.update_profile, name='update_profile'),
path('profile/', views.ProfileView.as_view(), name='profile'),
```

## 5.10. Unos recepta

U navigaciji se nalazi gumb *UNESITE SVOJ RECEPT* koji omogućuje prijavljenim korisnicima da objave svoj vlastiti recept. Nakon što se klikne na gumb otvara se nova stranica s obrascem koji mogu ispuniti i na kraju objaviti recept. Ispunjavanjem ovog obrasca istovremeno se stvaraju nove instance u modelima *Recept* i *MjereneJedinice*, time je potrebno definirati dvije forme. Definirane su forme *ReceptForm* i *SadrziForm* te formset *SadrziFormSet*. *ReceptForm* je forma za kreiranje i uređivanje recepta. Sadrži meta klasu koja definira model, polja i widgete za forum. Polja forme iz modela *Recept*, su *naziv\_recepta* kao tekstualno polje s placeholderom *'Torta od jagode'* kao primjer naslova nekog recepta, *slika\_recepta* kao polje u kojemu se može unositi slika, *priprema\_recepta* kao textarea s placeholder tekstem u kojem su objašnjeni dva koraka izrade recepta, *broj\_osoba* kao numeričko polje i placeholder primjerom *'2'*, *vrijeme\_pripreme* kao numeričko polje s placeholder primjerom *'30'* i *kategorija\_recepta* kao padajući izbornik s kategorijama, te ako je vrijednost prazna stoji *'Izaberite kategoriju'*. Druga forma je *SadrziForm* koja ujedno ima Meta klasu te sadrži polja *id\_sastojka* kao padajući izbornik s popisom sastojaka, prazna vrijednost je *'Izaberite sastojke'* i polje nije obavezno, *id\_mjerne\_jedinice* koje je padajući izbornik s popisom mjernih jedinica, prazna vrijednost je *'Izaberite mjernu jedinicu'* i polje nije obavezno i *kolicina* koja je tekstualno polje s placeholder primjerom *'1'*. Ova forma služi za kreiranje i uređivanje modela *Sadrzi*. Definiran je također, i *SadrziFormSet* koji je formset kreiran pomoću *'forms.inlineformset\_factory()'*. Formset omogućuje kreiranje, uređivanje i brisanje više *Sadrzi* instanca odjednom te je povezan s jednim *Recept* objektom. Parametri *forms.inlinedormset\_factory* su *Recept* kao model roditelja, *Sadrzi* kao model dijete, *form* kao forma za *Sadrzi* model i *extra* kao broj dodatnih praznih formi za unos sastojaka. Ovaj kod omogućuje kreiranje recepta s pripadajućim sastojcima i izgleda kao:

```
class ReceptForm(forms.ModelForm):
    class Meta:
        model = Recept
        fields = ('naziv_recepta', 'slika_recepta', 'priprema_recepta', 'broj_osoba',
                 'vrijeme_pripreme', 'kategorija_recepta')
        labels = {
```

```

        'kategorija_recepta' : 'Kategorija',
    }
    widgets = {
        'naziv_recepta': forms.TextInput(attrs={'class': 'form-control', 'placeholder': 'Torta od jagode'}),
        'priprema_recepta': forms.Textarea(attrs={'class': 'form-control', 'placeholder': 'Jaja miksajte sa šećerom da nastane pjenasta i svijetla smjesa. Dodajte vodu, glatko brašno i prašak za pecivo. Izmiksajte da nestanu eventualne grudice \nOkrugli kalup za tortu premažite s malo ulja ili maslaca (to radimo da se papir za pečenje zalijepi) i obložite papirom za pečenje.'}),
        'broj_osoba': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': '2'}),
        'vrijeme_pripreme': forms.NumberInput(attrs={'class': 'form-control', 'placeholder': '30'})
    }

    kategorija_recepta = forms.ModelChoiceField(queryset=KategorijaRecepta.objects.all(), empty_label="Izaberite kategoriju")

class SadrziForm(forms.ModelForm):
    id_sastojka = forms.ModelChoiceField(queryset=Sastojci.objects.all(), empty_label="Izaberite sastojak", required=False)
    id_mjerne_jedinice = forms.ModelChoiceField(queryset=MjerneJedinice.objects.all(), empty_label="Izaberite mjernu jedinicu", required=False)

    class Meta:
        model = Sadrzi
        fields = ('id', 'id_sastojka', 'id_mjerne_jedinice', 'kolicina')
        widgets = {
            'kolicina': forms.TextInput(attrs={'placeholder': '1'}),
        }

SadrziFormSet = forms.inlineformset_factory(Recept, Sadrzi, form=SadrziForm, extra=0)

```

Nadalje je potrebno definirati funkciju u *views.py* datoteci gdje će se pozvati napravljene forme. Napravljena je funkcija *add\_recept* koja omogućuje kreiranje novog recepta. Prvo je inicijaliziran *SadrziFormSet* koji koristi *inlineformset\_factory()* funkciju za kreiranje formset-a za model *Sadrzi*, povezan s modelom *Recept*. U formset-u se postavlja *extra=1*, što znači da će formset imati jednu dodatnu praznu formu za unos sastojka. Potom se provjerava HTTP metoda zahtjeva, te ukoliko je ona POST znači da je korisnik pokušao spremiti novi recept, time se kreira *ReceptForm* instanca s podacima iz *request.POST* i *request.FILES* za sliku. Ako je forma ispravna, sprema se novi *Recept* objekt, ali se još ne sprema u bazu podataka te se postavlja *commit=False*. Već se najprije za *id\_korisnika* polje u *Recept* modelu sprema podatak o korisniku koji je trenutno prijavljen te se onda novi *Recept* objekt sprema u bazu podataka. Nakon toga poziva se *SadrziFormSet* instanca s podacima iz *request.POST* i povezuje se s upravo spremljenim *Recept* objektom. Ako je *SadrziFormSet* ispravan sprema se u bazu podataka i korisnika se preusmjerava na početnu stranicu. Ako metoda nije POST znači da korisnik pristupa stranici za dodavanje novog recepta, i kreira praznu *ReceptForm* instancu i *SadrziFormSet* instance. Konačno, funkcija renderira html datoteku *add\_recept.html*. Sljedeći kod prikazuje funkciju *add\_recept*:

```

def add_recept(request):
    SadrziFormSet = inlineformset_factory(Recept, Sadrzi, form=SadrziForm, extra=1)
    if request.method == 'POST':
        recept_form = ReceptForm(request.POST, request.FILES)
        if recept_form.is_valid():
            recept = recept_form.save(commit=False)
            recept.id_korisnika = Korisnik.objects.get(user=request.user)
            recept.save()
            sadrzi_formset = SadrziFormSet(request.POST, instance=recept)
            if sadrzi_formset.is_valid():
                sadrzi_formset.save()
                return redirect('main:pocetna')
        else:
            recept_form = ReceptForm()
            sadrzi_formset = SadrziFormSet()
    return render(request, 'add_recept.html', {'recept_form': recept_form,
'sadrzi_formset': sadrzi_formset})

```

Nadalje je potrebno kreirati *add\_recept.html* datoteku kao bismo mogli prikazati formu u web aplikaciji. Forma za unos receptata koristi *POST* metodu i *multipart/form-data* enkodiranje za slanje podataka uključujući i slike. Uključen je i *sadrzi\_formset.management\_form* koji sadrži skrivena polja potrebna za rad formset-a. Za svako se polje pozivaju *.lable\_tag* za prikaz naslova, polje za unos te za pojedina polja dodan je i tekst s napomenom kako bi korisnici znali na koji način da unose podatke. Za popunjavanje sastojaka koristi se for petlja za svaki obrazac *sadrzi\_formset* koja se sastoji od polja za količinu, naziv sastojka i mjerne jedinice. Na kraju se prikazuje gumb *Dodaj Recept* za slanje forme. Sljedeći kod prikazuje formu u *add\_recept.html* datoteci:

```

<div class="row">
    <div class="col-md-3">
    </div>
    <div class="col-md-6">
        <form method="post" enctype="multipart/form-data">
            {% csrf_token %}
            <h2 class="mb-4">Objavite svoj recept</h2>
            {{ sadrzi_formset.management_form }}
            <div class="form-group">
                <div class="row">
                    <div class="col-md-6">
                        {{ recept_form.naziv_recepta.label_tag }}
                        {{ recept_form.naziv_recepta }}
                    </div>
                    <div class="col-md-6">
                        {{ recept_form.slika_recepta.label_tag }}
                        <br>

```

```

        {{ receipt_form.slika_recepta }}
    </div>
</div>
<div class="row">
    <div class="col-md-12">
        {{ receipt_form.priprema_recepta.label_tag }}
        <p class="text-muted">*Svaki postupak odvojite u novi
red kao što je prikazano u polju</p>
        {{ receipt_form.priprema_recepta }}
    </div>
</div>
<div class="row">
    <div class="col-md-6">
        {{ receipt_form.broj_osoba.label_tag }}
        <p class="text-muted">*Za koliko osoba je definirana
količina sastojka</p>
        {{ receipt_form.broj_osoba }}
    </div>
    <div class="col-md-6">
        {{ receipt_form.vrijeme_pripreme.label_tag }}
        <p class="text-muted">*Vrijeme unosite u minutama</p>
        {{ receipt_form.vrijeme_pripreme }}
    </div>
</div>
<div class="row">
    <div class="col-md-12">
        {{ receipt_form.kategorija_recepta.label_tag }}
        {{ receipt_form.kategorija_recepta }}
    </div>
</div>
</div>
<h4 class="mb-3">Sastojci:</h4>
<p class="text-muted">*za svaki sastojak odaberite njegov naziv,
mjernu jedinicu i količinu sastojka, no ukoliko se radi o sastojcima poput rum,
limunova korica, odaberite samo naziv sastojka, te količinu i mjernu jedinicu
pustite prazno, te unesite sastojke redom kojim želite da se prikazuju</p>
<div id="form_set">
    {% for form in sadrzi_formset %}
        <div class="form-row mb-3">
            <div class="col-md-4">
                {{ form.kolicina.label_tag }} <br>
                {{ form.kolicina }}
            </div>
            <div class="col-md-4">
                {{ form.id_mjerne_jedinice }}
            </div>
        </div>
    {% endfor %}
</div>

```



```

        </div>
        <div class="col-md-4">
            {{ form.id_sastojka }}
        </div>

    </div>
    {% endfor %}
</div>

        <button type="submit" class="btn btn-primary btn-
round" style="margin-top: 20px;">Dodaj Recept</button>
    </form>
</div>
<div class="col-md-3">
</div>
</div>

```

Dodatno je u *add\_recept.html* datoteci potrebno i JavaScript kod, kako bi se omogućilo korisnicima da dinamički dodaju i uklone obrasce sastojaka na stranici za dodavanje recepta. Koristi se kako bi korisnici mogli dodati onoliko sastojka koliko im je potrebno bez da trebaju unaprijed definirati broj sastojaka. Konkretno JavaScript prilikom učitavanja stranice pokreće *\$(document).ready()* funkciju koja inicijalizira *\$('#form\_set').formset()* s određenim opcijama koje su: *prefix* se postavlja na vrijednost *sadrzi\_fromset.prefix* što je Django varijabla koja sadrži prefiks *formset-a*, nakon toga *addText* postavlja tekst gumba za dodavanje novog obrasca sastojka na *Dodaj sastojak* i posljednji *deleteText* koji postavlja tekst gumba za brisanje obrasca sastojka na crveni tekst *Izbriši sastojak*. Sljedeći kod prikazuje JavaScript u datoteci *add\_recept.html*:

```

<script>
    $(document).ready(function() {
    $('#form_set').formset({
        prefix: '{{ sadrzi_formset.prefix }}',
        addText: 'Dodaj sastojak',
        deleteText: '<span style="color: red;">Izbriši sastojak</span>',
    });
    });
</script>

```

Posljednje što se treba učiniti kako bi se stranica prikazivala je unutar *urls.py* definirati lokaciju na kojoj će se stranica nalaziti. U ovom slučaju stranica se nalazi na linku *add\_recept/*. Kod za definiranje linka je sljedeći:

```

path('add_recept/', views.add_recept, name='add_recept'),

```

Slika 33. prikazuje obrazac za unos novog recepta.

*Home Chef* UNESITE SVOJ RECEPT admin

### Objavite svoj recept

Naziv recepta:  Slika recepta:  Nije odabrana niti jedna datoteka.

Priprema recepta:  
\*Svaki postupak odvojite u novi red kao što je prikazano u polju

Jaja miksajte sa šećerom da nastane pjenasta i svijetla smjesa. Dodajte vodu, glatko brašno i prašak za pecivo. Izmiksajte da nestanu eventualne grudice. Okrugli kalup za tortu premažite s malo ulja ili maslaca (to radimo da se papir za pečenje zalijepi) i obložite papirom za pečenje.

Broj osoba:  \*Za koliko osoba je definirana količina sastojka

Vrijeme pripreme:  \*Vrijeme unosite u minutama

Kategorija recepta:

#### Sastojci:

\*za svaki sastojak odaberite njegov naziv, mjernu jedinicu i količinu sastojka, no ukoliko se radi o sastojcima poput rum, limunova korica, odaberite samo naziv sastojka, te količinu i mjernu jedinicu pustite prazno, te unesite sastojke redom kojim želite da se prikazuju

Količina:

Izbriši sastojak  
Dodaj sastojak

Slika 33. Prikaz obrasca za dodavanje novog recepta

## 5.11. Brisanje i ažuriranje recepta

Svakom korisniku je omogućeno ažuriranje i brisanje vlastitih recepata koje može naći na profilnoj stranici. Nalazi se lista recepta, gdje svaki recept iznad slike dodatno ima *Ažuriraj* i *Izbriši* gumb za obavljanje te dvije funkcije. Ako korisnik odabere *Izbriši* gumb otvara mu se nova stranica s upitom je li siguran da želi obrisati odabrani recept, ako odgovoriti potvrdno recept se kompletno briše iz baze podataka. Za obavljanje ove funkcije najprije je definirana klasa *DeleteReceptView* koja nasljeđuje *DeleteView* iz Django web framework-a. Pomoću ove klase definiramo koji će se objekti modela *Recept* obrisati, te definira se HTML predložak za prikaz forme za potvrdu brisanja i nakon uspješnog brisanja korisnika se preusmjeri na profilnu stranicu. Sljedeći kod prikazuje *DeleteReceptView* klasu:

```
class DeleteReceptView(DeleteView):
    model = Recept
```

```

template_name = 'recept_confirm_delete.html'
success_url = reverse_lazy('main:profile')
def get_context_data(self, **kwargs):
    context = super().get_context_data(**kwargs)
    context['pocetna'] = self.get_object()
    return context

```

Nadalje kreiran je *recept\_confirm\_delete.html* datoteka u kojoj je definiran obrazac za potvrdu brisanja određenog recepta. Obrazac se sastoji od upita hoće li se određeni recept obrisati te ako se izabere gumb *Potvrdite brisanje* taj se recept briše. Dodatno korisniku je prikazan i gumb X, kojim korisnik može odustati od brisanja recepta i vratiti se na profilnu stranicu. Sljedeći kod nalazi se u *recept\_confirm\_delete.html* datoteci:

```

<body>
  <h1>Brisanje recepta</h1>
  <div class="delete-form">
    <a href="{% url 'main:profile' %}" class="close-button">x</a>
    <p>Jeste li sigurni da želite obrisati recept "{ { recept.naziv_recepta
}}"?</p>
    <form method="post">
      {% csrf_token %}
      <input type="submit" value="Potvrdite brisanje">
    </form>
  </div>
</body>

```

Slika 34. prikazuje obrazac za potvrdu brisanja primjera recepta *'Torta od jagode'*.



Slika 34. Prikaz obrasca za potvrdu brisanja recepta

Ako korisnik odabere gumb *Ažuriraj* otvara se nova stranica koja mu omogućuje ažuriranje podataka o receptu. Najprije je potrebno definirati *views* funkciju *update\_recept* koja

omogućuje ažuriranje recepta i njihovih sastojaka. Funkcija prima *recept\_id* kao argument i koristi *get\_object\_or\_404* za dohvaćanje instance *Recept* modela s odgovarajućim *id\_recepta*. Koristi se *inlineformset\_factory* za kreiranje *SadrziFormSet* formset-a, koji je povezan s modelom *Sadrzi* i koristi *SadrziForm* obrazac. *extra=0* znači da će se prikazati samo postojeći sastojci bez dodatnih praznih formi, a *can\_delete=True* omogućava brisanje sastojaka, do svakog sastojka stoji *Delete* koji se može označiti te prilikom ažuriranja svi sastojci kojima je bio označen *Delete* bit će izbrisani iz baze. Sljedeće je definirana obrada *POST* zahtjeva, ako je zahtjev *POST*, stvaraju se instance *recept\_form* i *sadrzi\_formset* s podacima iz zahtjeva. Nakon toga se provjerava valjanost obrazaca s pomoću *is\_valid*, ako su obrasci valjani, poziva se *save()* metoda za spremanje promjena u bazu podataka, te nakon uspješnog spremanje korisnik se preusmjerava na stranicu profila. Ako zahtjev nije *POST* stvaraju se instance *recept\_form* i *sadrzi\_formset* s podacima iz postojećeg recepta te *instance=recept* povezuje obrazac i formset s postojećim receptom. Funkcija na kraju poziva *update\_recept2.html* predložak i prosljeđuje mu *recept\_form*, *sadrzi\_formset* i *sadrzi\_formset\_prefix*. Sljedeći kod prikazuje *update\_recept* funkciju:

```
def update_recept(request, recept_id):
    recept = get_object_or_404(Recept, id_recepta=recept_id)
    SadrziFormSet = inlineformset_factory(Recept, Sadrzi, form=SadrziForm, extra=0,
    can_delete=True)
    if request.method == 'POST':
        recept_form = ReceptForm(request.POST, request.FILES, instance=recept)
        sadrzi_formset = SadrziFormSet(request.POST, instance=recept)
        if recept_form.is_valid() and sadrzi_formset.is_valid():
            recept_form.save()
            sadrzi_formset.save()
            return redirect('main:profile')
    else:
        recept_form = ReceptForm(instance=recept)
        sadrzi_formset = SadrziFormSet(instance=recept)
    return render(request, 'update_recept2.html', {
        'recept_form': recept_form,
        'sadrzi_formset': sadrzi_formset,
        'sadrzi_formset_prefix': sadrzi_formset.prefix,
    })
```

Nakon definiranja *views* funkcije potrebno je kreirati *update\_recept2.html* u kojem se definira izgled web stranice za ažuriranje recepta. Forma za ažuriranje recepta koristi *POST* metodu i *multipart/form-data* enkodiranje za slanje podataka i datoteka. Polja recepta se prikazuju prva s pomoću *recept\_form.as\_p*, dok nakon njih svaka u novom redu prikazuju polja *Sadrzi* modela za koji se koristi *sadrzi\_formset.management\_form* za prikaz skrivenih polja potrebnih za upravljanje formset-om te se koristi for petlja za prikaz svih sastojaka uz pomoć *form.as\_p*. Također definiran je i prikaz praznog obrasca sastojka, koji se koristi za dodavanje novih

obrazaca sastojaka. Na kraju nalazi se gumb *Dodaj sastojak* kojim se dodaje novi obrazac sastojka. Dok se pored njega nalazi i gumb *Ažuriraj recept* koji služi za slanje forme i ažuriranje recepta. Sljedeći kod nalazi se u *update\_recept2.html* dokumentu:

```
<div class="row">
  <div class="col-md-3"></div>
  <div class="col-md-6">
    <h1>Ažuriraj recept</h1>
    <form method="post" enctype="multipart/form-data">
      {% csrf_token %}
      {{ receipt_form.as_p }}
      {{ sadrzi_formset.management_form }}
      <div id="form_set">
        {% for form in sadrzi_formset %}
          {{ form.as_p }}
        {% endfor %}
      </div>
      <script id="empty_form" type="form-template">
        {{ sadrzi_formset.empty_form.as_p }}
      </script>
      <div class="form-group">
        <button type="button" id="add_more" class="btn btn-primary">Dodaj
sastojak</button>
        <button type="submit" class="btn btn-success">Ažuriraj recept</button>
      </div>
    </form>
  </div>
  <div class="col-md-3"></div>
</div>
```

Osim HTML-a u datoteci je definiran i JavaScript funkcionalnost za dinamičko dodavanje obrazaca sastojka na stranici ažuriranja recepta. Kada se klikne na gumb *Dodaj sastojak* poziva se ova funkcija te se najprije dohvaća vrijednost skrivenog polja koje sadrži ukupan broj obrazaca sastojaka. Potom se dodaje novi prazan obrazac sastojaka na kraju forme te se ažurira vrijednost skrivenog polja za ukupan broj obrazaca uvećavajući je za 1. Sljedeći kod prikazuje pozivanje JavaScripta:

```
<script>
$(document).ready(function() {
  $('#add_more').click(function() {
    var form_idx = $('#id_{{ sadrzi_formset_prefix }}-TOTAL_FORMS').val();
    $('#form_set').append($('#empty_form').html().replace(/__prefix__/g,
form_idx));
    $('#id_{{ sadrzi_formset_prefix }}-TOTAL_FORMS').val(parseInt(form_idx) +
1);
  });
});
```

```
});  
</script>
```

Slika 35., Slika 36. i Slika 37. prikazuju obrasce za ažuriranje detalja o receptu.

The screenshot shows the 'Home Chef' website interface for updating a recipe. The page title is 'Ažuriraj recept'. The form contains the following fields and content:

- Naziv recepta:** Text input field containing 'Torta od jagode'.
- Slika recepta:** Text input field containing 'Currently: images/recepti/torta\_od\_jagode.jpg'.
- Change:** A button labeled 'Odaberi datoteku' and a message: 'Nije odabrana niti jedna datoteka.'
- Priprema recepta:** A large text area containing detailed preparation instructions in Croatian, such as 'Jaja miksajte sa šećerom da nastane pjenasta i svijetla smjesa...'.
- Broj osoba:** Text input field containing '8'.
- Vrijeme pripreme:** Text input field containing '50'.
- Kategorija recepta:** A dropdown menu with 'Deserti' selected.

Slika 35. Prikaz dijela obrasca za ažuriranje podataka modela 'Recept' za određeni recept

The screenshot shows the 'Sadržaj' (Ingredients) section of the recipe update form. It lists ingredients with dropdown menus for selection and input fields for quantity:

- Ingredient 1:** 'jaja' (eggs), quantity '5'.
- Ingredient 2:** 'šećer' (sugar), quantity '8'.
- Ingredient 3:** 'paprika' (pepper), quantity '5'.
- Ingredient 4:** 'žlica' (teaspoon), quantity '5'.

Each ingredient entry includes a 'Delete' checkbox and a 'Delete' button. A blue arrow button is visible at the bottom right of the form.

Slika 36. Prikaz dijela obrasca za ažuriranje podataka modela 'Sadržaj' za određeni recept

Slika 37. Prikaz obrasca dodavanja novog sastojka prilikom ažuriranja podataka o receptu

Još je potrebno povezati stranicu s projektom kako bi se ona mogla prikazati kao web stranica, odnosno je potrebno definirati poveznicu u *main/urls.py* datoteci. Za ažuriranje i brisanje željenog recepta definirane su sljedeće dvije poveznice u kojima su *pk* i *recept\_id* primarni ključevi recepta.

```

path('profile/delete/<int:pk>/', views.DeleteReceptView.as_view(),
name='delete_recept'),

path('profile/update_recept/<int:recept_id>/', views.update_recept,
name='update_recept2'),

```

## 6. Zaključak

U radu je opisan proces izrade web aplikacije za digitalnu kuharicu. Prije početka izrade aplikacije definirane su tri fiktivne persone koje predstavljaju korisnike aplikacije. Napisane su i sedam korisničkih priča koje predstavljaju što različiti korisnici očekuju da će u aplikaciji moći napraviti. Nakon toga, opisuje se proces izrade aplikacije od kreiranja projekta u kojem će se aplikacija nalaziti te definiranja tablica, od kojih treba paziti na tablice agregacije i tablice slabog entiteta kako bi se svi podaci dobro povezali i time kako bi im se uvijek moglo prikladno pristupiti. Nakon definiranja modela potrebno je definirati funkcije u views datoteci koje će pozivati kreirane modele. Zatim se funkcije povezuju sa odgovarajućim predloškom i url-om kako bi se stranice mogle prikazivati na webu. Django je veoma fleksibilan okvir te osim Pythona omogućuje i korištenje Bootstrapa, CSS, JavaScripta i sl. Kako Django ima već ugrađene dijelove koda, kao što su model User, funkcije i greške za odjavu, prijavu i registraciju olakšava izradu same aplikacije.

Web aplikacija za digitalnu kuharicu ima puno prostora za širenje. Kao na primjer, mogao bi se dodati odlomak za preporuke recepta te mogućnost planiranja obroka kako bi korisnici mogli kreirati tjedne ili mjesečne planove obroka. Također, mogli bi se proširiti i podaci o samom receptu kako bi se dodatno mogli filtrirati recepti po nutritivnoj vrijednosti, vremenu potrebno da se pripremi jelo, broju zvjezdica, težini pripreme jela i sl. Nadalje, može se dodati i mogućnost dodavanja recepta u omiljene te mogućnost praćenja profila i dobivanja obavijesti prilikom objave novog recepta. Osim navedenih proširenja u aplikaciju bi se mogle dodati još razne funkcionalnosti, no time je potrebno dodatno proširiti model podataka i definirati pojedine funkcionalnosti u kodu.



## 7. Literatura

- [1] "Django introduction - Learn web development | MDN," Mozilla Developer Network, 2024, <https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction>. (Preuzeto 13.06.2024.)
- [2] "What is Django?," Amazon Web Services, accessed July 4, 2024, <https://aws.amazon.com/what-is/django/>. (Preuzeto:13.06.2024.)
- [3] "Do you know who your target persona is and how to define it?," CX Croatia, accessed July 4, 2024, <https://www.cx.hr/korisnicko-iskustvo/zna-te-li-ko-je-vasa-ciljana-persona-te-kako-je-definirati/>. (Preuzeto 4.07.2024.)
- [4] "User Stories," Mountain Goat Software, accessed July 4, 2024, <https://www.mountaingoatsoftware.com/agile/user-stories>. (Preuzeto 4.7.224.)
- [5] "What is a User Story?," Project Management Serbia, accessed July 4, 2024, <https://project-management-srbija.com/project-management/sta-je-user-story>. (Preuzeto: 4.7.2024.)
- [6] Pavlić, Oblikovanje baze podataka, Odjel za informatiku Sveučilišta u Rijeci, Rijeka, 2011.

## 8. Popis slika

Slika 1. Prikaz prve persone .....	2
Slika 2. Prikaz druge persone .....	3
Slika 3. Prikaz treće persone .....	3
Slika 4. Prikaz modela podataka .....	6
Slika 5. Prikaz kreiranog projekta <i>mysite</i> i aplikacije <i>main</i> u Visual Studio Codeu .....	7
Slika 6. Prikaz Django welcome stranice .....	8
Slika 7. Prikaz Django baze podataka .....	14
Slika 8. Prikaz osnovne navigacije za prijavljenog korisnika .....	16
Slika 9. Prikaz osnovne navigacije i padajućeg izbornika za prijavljenog korisnika .....	16
Slika 10. Prikaz osnovne navigacije za neprijavljenog korisnika .....	16
Slika 11. Prikaz skočnog prozora za odabir prijave ili registracije .....	17
Slika 12. Prikaz obrasca za prijavu korisnika.....	19
Slika 13. Prikaz poruke o grešci kod obrasca za prijavu.....	20
Slika 14. Prikaz obrasca za registraciju korisnika .....	23
Slika 15. Prikaz poruka o grešci kod registracije korisnika .....	24
Slika 16. Prikaz početne stranice.....	31
Slika 17. Prikaz filtriranja recepata po kategoriji na početnoj stranici.....	31
Slika 18. Prikaz kategorija sastojaka na početnoj stranici.....	32
Slika 19. Prikaz filtriranja recepta po odabranim sastojcima .....	32
Slika 20. Prikaz filtriranja recepta po odabranoj kategoriji i sastojcima.....	33
Slika 21. Prikaz rezultat pretraživanja recepta .....	33
Slika 22. Prikaz rezultata pretraživanja ukoliko nema niti jednog recepta.....	34
Slika 23. Prikaz osnovnih detalja o receptu .....	38
Slika 24. Prikaz pripreme i sastojka na stranici detalja o receptu .....	40
Slika 25. Prikaz bloka za recenzije kod prijavljenog korisnika.....	46
Slika 26. Prikaz bloka za recenzije kod neprijavljenog korisnika .....	46
Slika 27. Prikaz bloka za recenzije ukoliko nema niti jedne recenzije.....	47
Slika 28. Prikaz podataka o korisniku kada je odabrana profilna fotografija.....	51
Slika 29. Prikaz podataka o korisniku kada nije odabrana profilna fotografija.....	51
Slika 30. Prikaz obrasca za ažuriranje podataka o korisniku .....	54
Slika 31. Prikaz objavljenih recepata korisnika na profilnoj stranici .....	55
Slika 32. Prikaz poruke ukoliko korisnik nema niti jedan objavljen recept .....	56
Slika 33. Prikaz obrasca za dodavanje novog recepta.....	61
Slika 34. Prikaz obrasca za potvrdu brisanja recepta .....	62
Slika 35. Prikaz dijela obrasca za ažuriranje podataka modela 'Recept' za određeni recept .....	65
Slika 36. Prikaz dijela obrasca za ažuriranje podataka modela 'Sadrzi' za određeni recept .....	65
Slika 37. Prikaz obrasca dodavanja novog sastojka prilikom ažuriranja podataka o receptu .....	66