Web aplikacija za naručivanje hrane

Širić, Matej

Undergraduate thesis / Završni rad

2024

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

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:770666

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2025-03-23**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

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

Sveučilišni prijediplomski studij Računarstvo

WEB APLIKACIJA ZA NARUČIVANJE HRANE

Završni rad

Matej Širić

Osijek, 2024.

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

Obrazac Z1P: Obrazac za ocjenu završnog rada na sveučilišnom prijediplomskom studiju

Ocjena završnog rada na sveučilišnom prijediplomskom studiju

Ime i prezime pristupnika:	Matej Širić
Studij, smjer:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. pristupnika, god.	R 4597, 24.11.2020.
JMBAG:	0036532605
Mentor:	izv. prof. dr. sc. Ivica Lukić
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Web aplikacija za naručivanje hrane
Znanstvena grana završnog rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak završnog rada:	Web aplikacija za naručivanje hrane u kojoj će biti riješena problematika narudžbe hrane kod krajnjeg korisnika i pregled istih narudžbi kod pružatelja usluge. Rješenje problematike nudit će registraciju korisnika prilikom koje će korisnik aplikacije moći napraviti račun kao korisnik usluge i pružatelj usluge. Korisnik usluge imat će funkcionalnosti poput pregleda proizvoda, dodavanje proizvoda u košaricu, odabir adrese dostave i odabir načina plaćanja. Pružatelj usluge imat će funkcionalnosti poput pregleda narudžbi, dodavanja i uređivanja proizvoda. Tema rezervirana za: Matej Širić

Datum prijedloga ocjene završnog rada od strane mentora:	31.08.2024.
Prijedlog ocjene završnog rada od strane mentora:	Izvrstan (5)
Datum potvrde ocjene završnog rada od strane Odbora:	11.09.2024.
Ocjena završnog rada nakon obrane:	Izvrstan (5)
Datum potvrde mentora o predaji konačne verzije završnog rada čime je pristupnik završio sveučilišni prijediplomski studij:	16.09.2024.

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

IZJAVA O IZVORNOSTI RADA

Osijek, 16.09.2024.

Ime i prezime Pristupnika:	Matej Širić
Studij:	Sveučilišni prijediplomski studij Računarstvo
Mat. br. Pristupnika, godina upisa:	R 4597, 24.11.2020.
Turnitin podudaranje [%]:	8

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za naručivanje hrane**

izrađen pod vodstvom mentora izv. prof. dr. sc. Ivica Lukić

i sumentora

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

Potpis pristupnika:

SADRŽAJ

1. UVOD

Među mnogim nedavnim promjenama u ponašanju potrošača promijenio se i način kupovine hrane. Ranije su dominirale narudžbe putem telefona i fizički posjeti restoranima. U posljednje vrijeme, zbog napretka tehnologije, digitalne platforme koje nude učinkovitije i praktičnije mehanizme naručivanja hrane zamijenile su ove načine. Iz tog razloga, restorani i poslovi povezani s ugostiteljstvom morali su transformirati svoje aktivnosti kako bi uhvatili korak s promjenama u ponašanju potrošača.

U ovom radu napravljena je web aplikacija namijenjena za naručivanje hrane iz restorana. Aplikacija uključuje sustav u kojem korisnici mogu pregledavati opcije obroka putem interneta te izvršiti narudžbu. Pružatelji usluge, odnosno zaposlenici restorana, mogu koristiti aplikaciju za upravljanje narudžbama, kontrolirati njihov status i ažurirati jelovnike.

Znanja stečena na predmetima kao što su „Objektno orjentirano programiranje“, „Programsko inženjerstvo“ i „Baze podataka“ bila su ključna za uspješnu izradu aplikacije. Naglasak se stavlja na “Programsko inženjerstvo“ jer je taj predmet pružio znanje potrebno za planiranje projekta, analizu zahtjeva i dizajn sustava.

Rad počinje opisom evolucije načina na koji se hrana naručuje. Nakon uvoda u povijest naručivanja hrane, rad se osvrće na slična rješenja koja već postoje na tržištu. Potom se fokusira na sam proces izrade aplikacije, a u završnom dijelu, fokus se stavlja na prikaz izgleda aplikacije i objašnjenje načina korištenja.

1.1. Zadatak završnog rada

Zadatak završnog rada je izraditi web aplikaciju za naručivanje hrane koja omogućava korisnicima registraciju kao kupci ili djelatnici restorana. Kupci mogu pregledavati proizvode, dodavati ih u košaricu, birati adresu dostave i način plaćanja, dok djelatnici imaju mogućnost upravljanja narudžbama, uređivanja jelovnika i pregledavanja povijesti narudžbi. Korisnik s administratorskim ovlastima ima potpuni uvid u sve korisnike i narudžbe te ovlasti za uklanjanje djelatnika, kupaca i narudžbi.

2. PREGLED PODRUČJA TEME

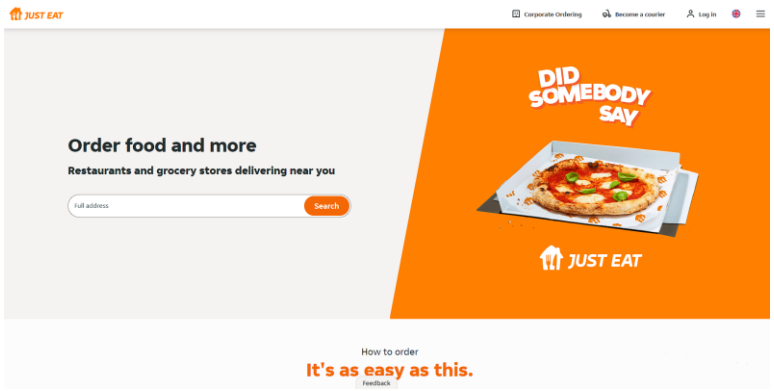
U ovom poglavlju opisano je trenutno stanje u sustavima za naručivanje hrane usporedbom značajki, korisničkog iskustva i tehničkih pristupa postojećih aplikacija.

2.1 Evolucija sustava za internetsko naručivanje hrane

Internetsko naručivanje hrane značajno se promijenilo tijekom posljednjih deset godina zahvaljujući napretku u tehnologiji, promjenama u preferencijama potrošača i učestalosti korištenja mobilnih uređaja. U početku su takvi sustavi koristili jednostavne internetske stranice kod kojih su korisnici birali artikle sa statičkog jelovnika i na taj način vršili narudžbe. Najveći je problem kod takvog pristupa nedostatak interakcije i ažuriranja u stvarnom vremenu, što je ograničilo njihovu funkcionalnost i samim time su bili manje privlačni korisnicima. Popularne platforme za naručivanje hrane u današnje vrijeme poput Domino's Pizza [1], UberEats [2] i Papa John's [3] preuzele su ovaj tržišni segment uvođenjem dinamičnih sučelja koja se ažuriraju u stvarnom vremenu. To znači da se integriraju sa sustavima za upravljanje restoranima, uključujući GPS (engl. *Global Positioning System*) praćenje za potrebe dostave te sigurnosne sustave za plaćanje. Ove platforme mogu obraditi velik broj istovremenih transakcija dok osiguravaju neometano funkcioniranje na više uređaja.

2.2 Usporedba postojećih sustava

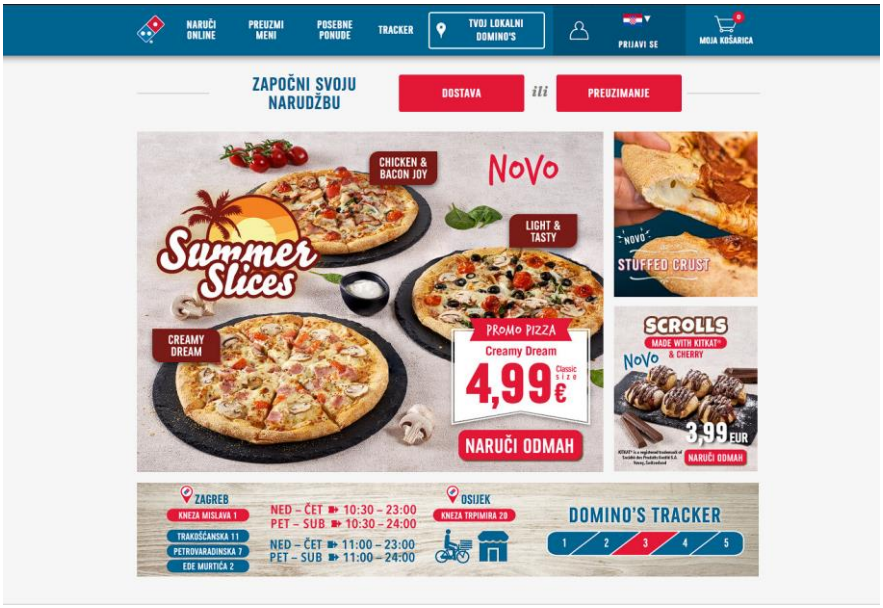
Postojeći sustavi za naručivanje hrane mogu se podijeliti u dvije kategorije: platforme posrednika koje okupljaju ponude većeg broja restorana na jednome mjestu i sustavi specifični za restorane. Platforme posrednika, poput JustEat [4] objedinuju nekoliko restorana pod jednom platformom i time omogućuju korisnicima odabir iz širokog raspona restorana i jednostavno pregledavanje različitih restorana na jednome mjestu. Naslovna stranica JustEat vidljiva je na slici 2.1



Sl. 2.1 *Naslovnica JustEat web aplikacije*

S druge strane, sustavi specifični za restorane dizajnirani su kako bi zadovoljili potrebe pojedinih restorana, pružajući prilagođenije iskustvo usmjereno na njihove vlastite brendove. Jedan od najboljih predstavnika ove skupine je ranije spomenuti Domino's. Na stranici je naglašena jednostavnost naručivanja i privlačna prezentacija proizvoda. Na središnjem dijelu stranice nalazi se promotivna sekcija gdje korisnici mogu vidjeti trenutno dostupne akcijske ponude. Cijena je jasno istaknuta uz svaku od ponuda, a gumb *Naruči odmah* jasno je vidljiv za brzu reakciju korisnika. Domino's web aplikacija strukturirana je tako da korisnicima omogućuje odabir jela s jelovnika te obavljanje narudžbi i plaćanja. Korisnicima je također omogućeno praćenje napretka njihove narudžbe zahvaljujući usluzi Domino's Tracker [1], koja omogućuje praćenje procesa narudžbe od trenutka njezina postavljanja do dostave. Domino's pruža različite personalizirane preporuke prema povijesti korisnika te sigurno plaćanje. Na dnu stranice pružene su informacije o dvama gradovima gdje je Domino's dostupan u Hrvatskoj. Sučelje Domino's web aplikacije snažno je orijentirano na promocije. Žarke boje i mnoštvo promotivnih elemenata na Domino's stranici, prikazanoj slikom 2.2 privlačni su korisnicima. Aplikacija FoodOS nudi funkcionalnosti koje su usporedive s onima koje nudi Domino's, omogućujući korisnicima registraciju, prijavu, upravljanje i praćenje narudžbi, s tim da je veći naglasak stavljen na specifične ponude restorana.

Neke od tehnologija na koje se Domino's fokusira uključuju strojno učenje koje omogućuje prikaz relevantnih ponuda korisniku, kao i dizajn koji omogućuje korisnicima korištenje bilo kojeg uređaja za pristup. Na slici 2.2 vidljiva je Domino's naslovnica.



Sl. 2.2 *Naslovnica Domino's web aplikacije*

2.3 Tehnološki pristupi

Moderne platforme za naručivanje hrane oslanjaju se na sofisticirane tehnološke pristupe kako bi pružile besprijekorno korisničko iskustvo, osigurale učinkovitost poslovanja i održale konkurentsku prednost na tržištu. Tri su ključna tehnološka pristupa koji su od posebnog značaja.

- **Računarstvo u oblaku** omogućuje visoku skalabilnost i pouzdanost aplikacije za obradu velikog broja korisničkih zahtjeva i transakcija istovremeno. Kao rezultat, aplikacije se mogu distribuirati po raznim regijama bez utjecaja na performanse.
- **Integracija platnih pristupnika** omogućuje sigurnu obradu plaćanja putem elektroničkih transakcijskih sustava. Platni pristupnik djeluje kao posrednik između trgovca i banke, provjeravajući i autorizirajući transakcije kako bi osigurao dostupnost sredstva i sigurno plaćanje. Sustavi poput Stripea [5] najčešće se koriste zbog svoje jednostavnosti u korištenju, kako za korisnike tako i za pružatelje usluga, a cijela transakcija je potpuno sigurna.
- **Sinkronizacija podataka u stvarnom vremenu**: AJAX i WebSockets su neke od tehnologija koje omogućuju ažuriranje podataka na stranici u stvarnom vremenu, što znači da korisnik ne mora osvježavati ili ponovno učitavati stranicu kako bi dobio važne informacije.

3. KORIŠTENE TEHNOLOGIJE

U ovom poglavlju opisane su glavne tehnologije koje su korištene prilikom razvoja i implementacije sustava. Naglasak je stavljen na popularne alate i okvire koji omogućuju brži razvoj, veću sigurnost i jednostavno održavanje web aplikacija.

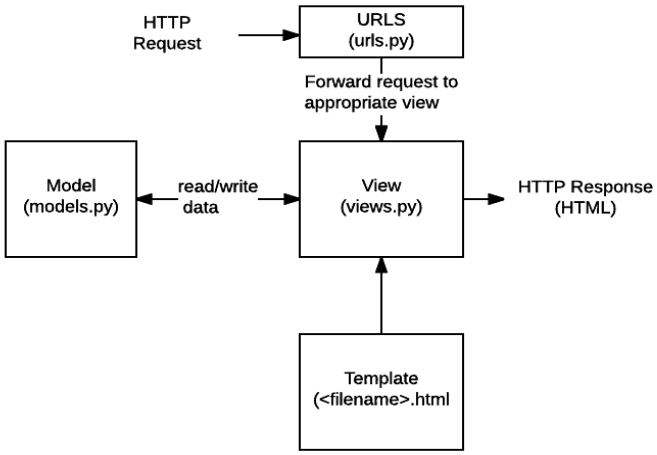
3.1. Django

Django je robustan web okvir izgrađen na Pythonu koji olakšava brz razvoj sigurnih i održivih web stranica. Ublažavanjem izazova povezanih s web razvojem, Django omogućuje razvojnim programerima koncentriranje na izradu vlastitih aplikacija bez potrebe da ponovno izmišljaju postojeća rješenja. Njegova je struktura dizajnirana da omogućuje brzi napredak od koncepta do izvedbe. Usvajajući filozofiju *baterije uključene*, Django nudi široku lepezu ugrađenih značajki. S jakim fokusom na ponovnu upotrebu, modularnost i ubrzani razvoj, Django predstavlja privlačan izbor i za male tvrtke u nastajanju i za etablirana poduzeća koja žele stvoriti svoju mrežnu prisutnost [6].

Prvobitno ga je razvio web tim odgovoran za izradu i upravljanje novinskim web stranicama, a projekt je dobio oblik između 2003. i 2005. godine. Nakon izrade nekoliko stranica, tim je počeo izdvajati i ponovno koristiti brojne uobičajene obrasce dizajna (engl. *design pattern*) i kod. Ovaj dijeljeni kod na kraju se transformirao u generički okvir za razvoj weba, koji je pokrenut kao *open-source Django* projekt u srpnju 2005. godine [7].

Na tradicionalnoj web stranici koja se temelji na podacima, web aplikacija čeka HTTP (engl. *Hypertext Transfer Protocol*) zahtjeve web preglednika ili drugog klijenta. Kada se primi HTTP zahtjev, aplikacija radi ono što je potrebno na temelju URL-a (engl. *Uniform Resource Locator*) i mogućih informacija u *post* podacima ili *get* podacima. Ovisno o tome što je potrebno, tada može čitati ili pisati informacije iz baze podataka ili obavljati druge zadatke potrebne za udovoljavanje HTTP zahtjevu. Aplikacija će zatim vratiti HTTP odgovor web pregledniku, često dinamički stvarajući HTML (engl. *Hypertext Markup Language*) stranicu koju preglednik prikazuje umetanjem dohvaćenih podataka na rezervirana mjesta u HTML predlošku.

Obično Django web aplikacije organiziraju kod odgovoran za svaki od ovih procesa u različite datoteke [7]. Izgled Django strukture prikazan je slikom 3.1 .



Sl. 3.1 *Django struktura*

Django slijedi MVT (engl. *Model View Template*) obrazac dizajna. Ovaj obrazac predstavlja Djangovu interpretaciju široko poznatog MVC (engl. *Model-View-Controller*) obrasca, prilagođenog specifičnostima web razvoja i Python ekosustava. Razdvajanjem različitih aspekata aplikacije, upravljanja podacima, poslovne logike i prezentacije, MVT olakšava razvoj i održavanje kompleksnih web sustava.

3.1.1. Model

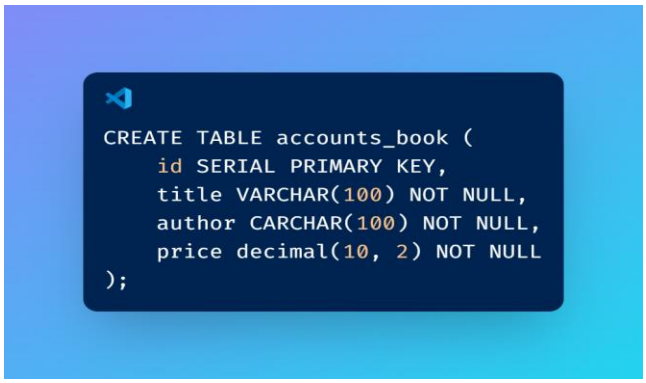
Model predstavlja izvor podataka iz baze podataka. U Djangu se podaci predstavljaju u obliku ORM (engl. *Object Relational Mapping*). ORM je moćan alat koji omogućava programerima rad s bazama podataka i korištenje Python objekata umjesto pisanja SQL (engl. *Structured query language*) upita. ORM povezuje modele s tablicama u bazi podataka, čime omogućuje interakciju s bazom podataka na apstraktnom nivou. Obično se modeli nalaze u datoteci pod nazivom *models.py*, a jednostavan model prikazan je na slici 3.2

```python
from django.db import models

class Book(models.Model):
    title = models.CharField(max_length=100)
    author = models.CharField(max_length=100)
    price = models.DecimalField(max_digits=10, decimal_places=2)
```

Sl. 3.2 *Django model*

Na slici 3.3 vidljiv je odgovarajući SQL upit koji kreira tablicu pod nazivom *accounts_book* s četiri atributa koji predstavljaju stupce u bazi podataka.



Sl. 3.3 *Kreiranje tablice*

Nakon što je model definiran ili promijenjen, potrebno je kreirati i primijeniti migraciju kako bi se te promjene odrazile na stvarnu bazu podataka.

Kreiranje migracija:

Linija ***Kod***

```
1:  python manage.py makemigrations
```

Programski kod 3.1 *Kreiranje migracija*

Primjena migracija:

Linija ***Kod***

```
1:  python manage.py migrate
```

Programski kod 3.2. *Primjena migracija*

3.1.2. Pogled

Pogled je sloj korisničkog sučelja. On kontrolira što korisnik vidi i s čim komunicira u pregledniku. U Djangu, pogledi su Python funkcije koje primaju HTTP zahtjeve i vraćaju HTTP odgovore. U pogledima se implementira logika aplikacije, koja određuje koje će se informacije prikazati korisniku. Na slici 3.4 vidljiv je primjer jednostavnog pogleda koji se nalazi u *views.py* datoteci i prikazuje poruku dobrodošlice.

Sl. 3.4 *Pogled*

3.1.3. Predložak

Predložak je sloj prezentacije koji obavlja prikazivanje HTML/CSS (engl. *Cascading Style Sheets*) datoteka. Predlošci su HTML datoteke koje se kombiniraju s podacima iz modela kako bi se stvorile dinamičke web stranice. Djangov mehanizam za predloške omogućuje uključivanje dinamičkog sadržaja u web stranice i podržava nasljeđivanje predložaka, što olakšava upravljanje promjenama u dizajnu cijele stranice.

3.1.4. Html obrasci

Django koristi HTML obrasce za prikupljanje korisničkih podataka. Obrasci omogućuju korisnicima unos podataka koji se, nakon potvrde, šalju web poslužitelju. Na poslužitelju se ti podaci obrađuju, a zatim se korisniku šalju odgovarajuće povratne informacije. Rad s obrascima može biti složen, posebno kada se obrađuju različiti tipovi podataka. Primjer složenog rada s obrascima može se vidjeti u Djangovom administracijskom sučelju, gdje se prikupljeni podaci prikazuju u HTML formatu, omogućuju uređivanje putem sučelja, a nakon toga se ponovno šalju poslužitelju na daljnju obradu, validaciju i konačno spremanje [7].

Djangova funkcionalnost za rad s obrascima značajno olakšava proces kreiranja i obrade podataka unesenih u obrasce. Django automatski generira HTML kod za obrasce na temelju definiranih modela ili prilagođenih obrazaca, što smanjuje količinu ručnog pisanja koda. Također, Django automatski upravlja validacijom unesenih podataka, osiguravajući da su podaci ispravni prije nego što ih aplikacija obradi ili pohrani u bazu podataka. Django rukuje s tri različita dijela posla povezanih s obrascima:

- pripremom i restrukturiranjem podataka
- stvaranjem HTML obrazaca za te podatke
- primanjem i obradom predanih obrazaca i podataka od klijenta

3.1.5. Sigurnosni aspekti

Zaštita od XSS napada (engl. *cross-site scripting*) važan je aspekt sigurnosti. Ovi napadi omogućuju napadaču ubacivanje zlonamjernog koda na strani klijenta, koji se izvršava u preglednicima drugih korisnika. Najčešće se to postiže tako da napadač unese skripte u bazu podataka. Kasnije, kada drugi korisnici pregledavaju stranicu koja prikazuje te podatke, skripta se automatski izvršava u njihovim preglednicima. Drugi način je da napadač prevari korisnika navodeći ga na klikanje na poveznicu koja izvršava napadačev JavaScript kod. XSS napadi mogu dolaziti iz bilo kojeg nepouzdanog izvora podataka, uključujući kolačiće (engl. *cookies*) ili web servise, osobito kada podaci nisu pravilno *očišćeni* prije prikazivanja na stranici. Srećom, Django predlošci pružaju zaštitu od većine XSS napada automatskim *čišćenjem* podataka prije prikazivanja, čime se sprječava izvršenje takvih skripti [7].

Drugi primjer je zaštita od CSRF (engl. *cross-site request forgery*) napada. CSRF napadi omogućuju zlonamjernom korisniku izvršavanje radnje na web stranici koristeći vjerodajnice drugog korisnika, bez njegovog znanja ili pristanka. Ovaj tip napada može prisiliti korisnika na nenamjerno izvršavanje neke radnje, poput promjene postavki ili slanja novca. Django ima ugrađenu zaštitu protiv većine CSRF napada, pod uvjetom da je ta zaštita aktivirana i pravilno korištena. Ipak, postoje ograničenja – na primjer, moguće je onemogućiti CSRF zaštitu globalno ili za pojedine poglede, što može otvoriti potencijalne ranjivosti. Dodatno, problemi mogu nastati ako stranica koristi poddomene koje nisu pod kontrolom korisnika. CSRF zaštita funkcionira tako što svaka *post* metoda zahtijeva tajni token specifičan za korisnika. Ovaj token sprječava ponovno slanje obrasca u ime drugog korisnika, što onemogućuje neovlašteno izvršenje radnje. Kada korisnik ispunjava obrazac i klikne na gumb za slanje, uz ostale podatke, također se šalje ovaj CSRF token. Poslužitelj tada provjerava je li taj token valjan i pripada li trenutnom korisniku [7].

Napad SQL injekcijom događa se kada zlonamjerna osoba uspije izvršiti proizvoljni SQL kod na bazi podataka, što potencijalno dovodi do curenja podataka ili brisanja zapisa. Curenje podataka odnosi se na situaciju kada se povjerljivi ili osjetljivi podaci nenamjerno otkriju, pristupe ili prenesu neovlaštenim osobama. Djangovi skupovi upita nude zaštitu od takvih SQL injekcija korištenjem parametariziranih upita, što osigurava da se SQL kod jasno odvaja od korisničkih unosa. Budući da su korisnički unosi nepredvidljivi i potencijalno opasni, parametre obrađuje temeljni upravljački program baze podataka, čime se sprječava direktna manipulacija SQL kodom te se značajno smanjuje rizik od SQL injekcija [7].

3.1.6. Django administracija

Jedan od najmoćnijih dijelova Djanga je automatizirano administracijsko sučelje. Čita metapodatke iz modela kako bi pružio sučelje orijentirano na model gdje korisnici mogu upravljati sadržajem na stranici. Preporučena upotreba sučelja za upravljanje ograničena je na interne alate za upravljanje unutar organizacije. Nije namijenjeno izradi cijelog sučelja stranice [7].

Ako se želi kreirati *admin* korisnik za prijavu, koristi se naredba *createsuperuser*. Prema zadanim postavkama, prijava u administracijsko sučelje zahtijeva da atribut *is_staff* bude postavljen na *True*. Nakon toga odlučuje se koje modele u aplikaciji treba moći uređivati u administracijskom sučelju. Svaki model treba biti registriran u administracijskom sučelju kao što je vidljivo u datoteci *admin.py* [7], prikazanoj slikom 3.5.

```
from django.contrib import admin
from myapp.models import Author


class AuthorAdmin(admin.ModelAdmin):
    pass


admin.site.register(Author, AuthorAdmin)
```

Sl. 3.5 *Registracija modela u admin*

3.2. HTML

HTML je jezik koji koristi oznake za strukturiranje i objavljivanje hipertekstualnih dokumenata. On je osnova za izradu web sadržaja, omogućujući programerima oblikovanje rasporeda i formatiranje teksta, slika, poveznica i multimedijskih komponenti unutar web stranica. Ovi hipertekstualni dokumenti pregledavaju se putem WWW-a (engl. *World Wide Web*), u daljnjem tekstu koristit će se izraz web, koji je globalna, distribuirana platforma povezanih izvora informacija.

Web je ogroman sustav međusobno povezanih dokumenata i resursa, koji omogućuje pristup informacijama s bilo kojeg mjesta u bilo koje vrijeme. HTML igra važnu ulogu u ovom sustavu jer omogućuje strukturiranje web stranica s elementima poput poveznica, što korisnicima olakšava navigaciju između različitih web stranica. Osim toga, HTML omogućuje integraciju drugih alata i jezika, poput CSS-a za stiliziranje i JavaScripta za dodavanje interaktivnosti, što ga čini ključnim elementom u izradi modernih i učinkovitih web stranica.

3.2.1. Konstrukcije u HTML-u

HTML element osnovni je dio HTML koda. Elementi su dijelovi koji se koriste za strukturiranje sadržaja na webu. Oni su identificirani početnom oznakom, tekstom i završnom oznakom. Sadržaj može biti tekst, slike, poveznice ili drugi elementi.

Linija ***Kod***

```
1:      <ime_elementa>Sadržaj</ime_elementa>
```

Programski kod 3.3 *Primjer HTML elementa*

Početna oznaka: < *ime_elementa* > - Ovo je oznaka koja označava početak elementa. Sadržaj je informacija koju element sadrži ili obuhvaća. To može biti tekst, slike ili drugi elementi. Završna oznaka: </ *ime_elementa* > - Ovo je oznaka koja označava kraj elementa.

Atributi pružaju dodatne informacije o HTML elementu. Uvijek su uključeni unutar početne oznake elementa i sastoje se od imena i vrijednosti.

Linija ***Kod***

```
1:      <ime_elementa atribut="vrijednost">Sadržaj</ime_elementa>
```

Programski kod 3.4 *Primjer HTML atributa*

4. IZRADA APLIKACIJE

U ovom poglavlju opisan je proces izrade aplikacije, od postavljanja okruženja i pokretanja projekta do razvoja funkcionalnosti specifičnih za web aplikaciju. Kroz modularni pristup, Django omogućuje jednostavnu organizaciju projekta i fleksibilnost u razvoju aplikacije.

4.1. Postavljanje okruženja

Nakon instalacije Pythona potrebno je stvoriti virtualno okruženje za projekt kako bi izolirali ovisnosti projekta od drugih projekata i globalne Python instalacije. Stvaranje virtualnog okruženja vidljivo je u programskom kodu 4.1.

Linija	***Kod***
1:	`python -m venv env`

Programski kod 4.1 *Stvaranje virtualnog okruženja pod nazivom env*

Potom je potrebno aktivirati stvoreno virtualno okruženje što je vidljivo u kodu 4.2 :

Linija	***Kod***
1:	`source env/Scripts/activate`

Programski kod 4.2 *Aktivacija virtualnog okruženja pod nazivom env*

4.2. Pokretanje projekta

Naredbom *pip install django* instalira se Django, a Django projekt u kojem se nalazi skup svih postavki, uključujući konfiguraciju baze podataka, stvara se naredbom *django-admin startproject mysite*. Projekt koji se kreira ima stukturu prikazanu na slici 4.1

```
mysite/
    manage.py
    mysite/
        __init__.py
        settings.py
        urls.py
        asgi.py
        wsgi.py
```


Sl. 4.1 *Struktura Django projekta*

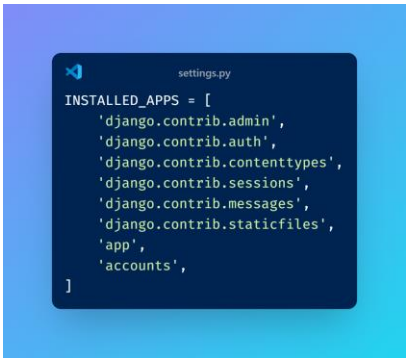
Django automatski generira set direktorija i datoteka koji čine osnovu projekta. Ova struktura nije samo organizacijski okvir, već odražava Djangovu filozofiju *baterije uključene,* što znači da pruža većinu potrebnih funkcionalnosti unutar samog okvira, bez potrebe za instalacijom dodatnih alata ili biblioteka

- Spremnik za projekt je vanjski *mysite* direktorij. Može biti preimenovan u bilo koji naziv jer ta informacija Djangu nije bitna.
- Interakciju s Django projektom olakšava *manage.py*, koji nudi niz naredbi. Na primjer, može se pokrenuti razvojni poslužitelj pomoću *python manage.py runserver*, migrirati baze podataka s *python manage.py migrate* ili stvoriti nove aplikacije naredbom *python manage.py startapp*, između ostalih funkcija. Ovaj alat pojednostavljuje izvršenje ovih zadataka, eliminirajući potrebu za izravnim radom s Python skriptama ili internim okvirima Djanga.
- Unutarnji *mysite* direktorij stvarni je Python paket za projekt. Njegov naziv je naziv Python paketa koji će se koristiti za uvoz bilo čega unutar njega (npr. *mysite.urls*.).
- Prazna datoteka koja govori Pythonu da se ovaj direktorij treba smatrati Python paketom je *mysite/__init__.py*.
- U *mysite/settings.py* mogu se pronaći sve bitne konfiguracije projekta, kao što su postavke baze podataka, parametri aplikacije, pojedinosti o autentikaciji, lokacije statičnih datoteka i razne druge postavke specifične za projekt.
- U *mysite/urls.py* datoteci postavljene su putanje koje upućuju zahtjeve prema odgovarajućim *view* funkcijama ili klasama. Ovo je ključna datoteka za mapiranje URL-ova na odgovarajuće prikaze koji će obraditi zahtjeve korisnika.
- *mysite/asgi.py* je ulazna točka za ASGI-kompatibilne (engl. *Asynchronous Server Gateway Interface*) web poslužitelje koji služe projektu.
- *mysite/wsgi.py* je ulazna točka za web poslužitelje kompatibilne s WSGI (engl. *Web Server Gateway Interface*) koji služe projektu [8].

4.3. Web aplikacija FoodOS

Nakon što je osnovni projekt postavljen, sljedeći korak je dodavanje aplikacije. Aplikacija u Django projektu predstavlja modularni dio funkcionalnosti i može se uspješno razvijati i održavati kao zaseban entitet, što znači da može biti korištena i u drugim projektima. Aplikacija je kreirana

naredbom *python manage.py startapp app* te su automatski stvorene pripadajuće datoteke. Kako bi Django prepoznao aplikaciju i mogao je koristiti, potrebno ju je registrirati unutar glavnih postavki projekta u datoteci *settings.py*. U listi *INSTALLED_APPS*, vidljivoj na slici 4.2, registriran je *app*, zajedno s ostalima koji su automatski registrirani pri izradi projekta.



Sl. 4.2 *Lista registriranih aplikacija*

4.3.1. Definiranje modela

Novi modeli definirani su u datoteci *models.py* unutar aplikacije *app*. Kako bi isti bili prepoznati, iz datoteke *django.db* uvezeni su modeli naredbom *from django.db import models.* Model se definira na način da klasa naslijedi *models.Model*, a svako svojstvo klase predstavlja jedno polje u tablici baze podataka. Osnovni model u ovoj web aplikaciji je *Account*. *Account* može imati dvije vrste korisnika, što je vidljivo iz *USER_TYPE_CHOICES.* Korisnik može biti kupac (*customer*) ili djelatnik (*employee*). Time su ograničene vrijednosti koje korisnik može odabrati za atribut *user_type.* Cijeli model *Account* prikazan je na slici 4.3.

Sl. 4.3 *Account model*

Drugi model je *FoodItem* koji predstavlja stavke hrane koje se mogu ponuditi u aplikaciji, kao što su različite vrste pizza, salata, jela s roštilja i kuhana jela. Ovaj model omogućuje pohranu i upravljanje informacijama o svakom jelu, uključujući naziv, opis, cijenu i sliku. Na slici 4.4 vidljiv je izgled cijelog modela *FoodItem.*

Sl. 4.4 *FoodItem model*

Model *Order* predstavlja narudžbu koju korisnik kreira u aplikaciji. Ovaj model pohranjuje sve informacije povezane s narudžbom, uključujući korisnika koji je naručio, status narudžbe, ukupnu cijenu, metode dostave i plaćanja te adresu za dostavu. Pomaže u praćenju narudžbi kroz različite faze, od kreiranja do završetka. Model *Order* prikazan je slikom 4.5.

Sl. 4.5 *Order model*

Model *Basket* označava košaricu koja privremeno pohranjuje proizvode hrane prije nego što korisnik napravi narudžbu. U modelu se nalaze podaci o tome tko posjeduje ovu košaricu. Model također automatski izračunava ukupnu cijenu svih stavki u košarici na temelju količine i cijene svakog proizvoda. Košarica služi kao privremeni spremnik proizvoda dok korisnik ne završi proces narudžbe, a model je prikazan slikom 4.6.

Sl. 4.6 *Basket model*

4.3.2. Definiranje pogleda

Pogled *add_to_basket* omogućuje korisniku dodavanje jela u košaricu. Kada je određeno jelo odabrano, zadatak tog pogleda je pronalazak istog. Pogled detektira košaricu za prijavljenog korisnika ili je stvara ukoliko već ne postoji. Ukoliko stavka u košarici već postoji, klikom na gumb *Dodaj u košaricu* povećava se količina, a u suprotnom se stvara nova stavka u košarici. Korisnik ima mogućnost odabira kategorije hrane koju želi naručiti te se na temelju kategorije (*pizza_gallery, salad_gallery, cooked_meals_gallery, grill_gallery*) preusmjerava na odgovarajuću galeriju hrane, a ako kategorija nije prepoznata, korisnik se preusmjerava na početnu stranicu. Iznad same funkcije nalazi se dekorator *@login_required*. Dekorator osigurava da tom pogledu mogu pristupiti samo prijavljeni korisnici. Ukoliko neprijavljeni korisnik pokuša pristupiti ovom pogledu, biva preusmjeren na stranicu za prijavu. Na slici 4.7 prikazan je pogled *add_to_basket.*

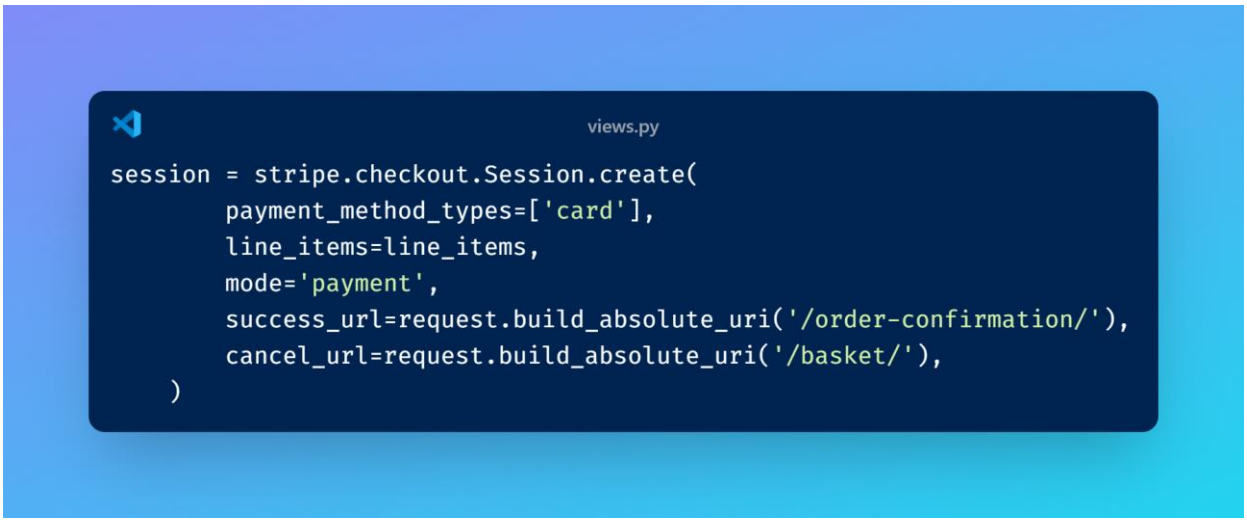
Sl. 4.7 *Pogled add_to_basket*

Pogled *view_basket* prikazuje sadržaj korisnikove košarice i računa ukupnu cijenu po proizvodu te zajednički iznos svih proizvoda u košarici, a to se onda šalje kroz kontekst za prikaz stranice. Logika računanja ukupnog iznosa u košarici može se vidjeti na slici 4.8.

Sl. 4.8 *Logika računanja ukupnog iznosa stavki u košarici*

Pogled *remove_from_basket* omogućava korisniku uklanjanje stavke iz košarice, dok pogled *create_checkout_session* kreira sesiju za plaćanje. Ovaj pogled prikuplja informacije o dostavi te stvara narudžbu i dodaje sve stavke iz košarice u tu narudžbu. Ovdje je dodana Stripe integracija jer omogućuje korisnicima sigurno i jednostavno online plaćanje. Nakon što korisnik završi s dodavanjem stavki u košaricu, podaci o dostavi i sadržaju košarice prikupljaju se iz *post* zahtjeva linijom koda *request.POST.get*. Ove informacije koriste se za kreiranje narudžbe i za

pripremu podataka koji se šalju Stripeu te se kreira sjednica koristeći Stripe API (engl. *Application Programming Interface*). Kreiranje Stripe sjednice za plaćanje prikazano je slikom 4.9.



Sl. 4.9 *Kreiranje Stripe sjednice*

Integracijom Stripea u web aplikaciju, korisnici mogu sigurno izvršiti plaćanja bez potrebe za kompleksnim implementacijama sigurnosnih mehanizama, jer Stripe pruža alate za zaštitu osjetljivih podataka i sprječavanje prijevara.

Korisnicima se nudi i mogućnost plaćanja pouzećem. Ta funkcionalnost opisana je u pogledu *cod_checkout*. Ukoliko je korisnik odabrao dostavu i plaćanje gotovinom, nudi mu se unos adrese za dostavu koja je integrirana pomoću Google Maps API [9] zajedno s Places bibliotekom. Ova integracija koristi se za poboljšanje korisničkog iskustva prilikom unosa adrese. Na slici 4.10 prikazan je pogled *cod_checkout.*

views.py

```
@login_required
def cod_checkout(request):
    if request.method == 'POST':
        delivery_method = request.POST.get('delivery_method')
        address = request.POST.get('address') if delivery_method == 'delivery' else "nedostupno(osobno preuzimanje)"
        basket = Basket.objects.get(user=request.user)

        order = Order.objects.create(
            user=request.user,
            status='pending',
            total_price=basket.total_price(),
            delivery_method=delivery_method,
            address=address,
            payment_method='cod'
        )
        for item in basket.items.all():
            OrderItem.objects.create(
                order=order,
                food_item=item.food_item,
                quantity=item.quantity
            )
        basket.items.all().delete()

        return redirect('order_confirmation')
    return redirect('view_basket')
```

Sl. 4.10 *cod_checkout pogled*

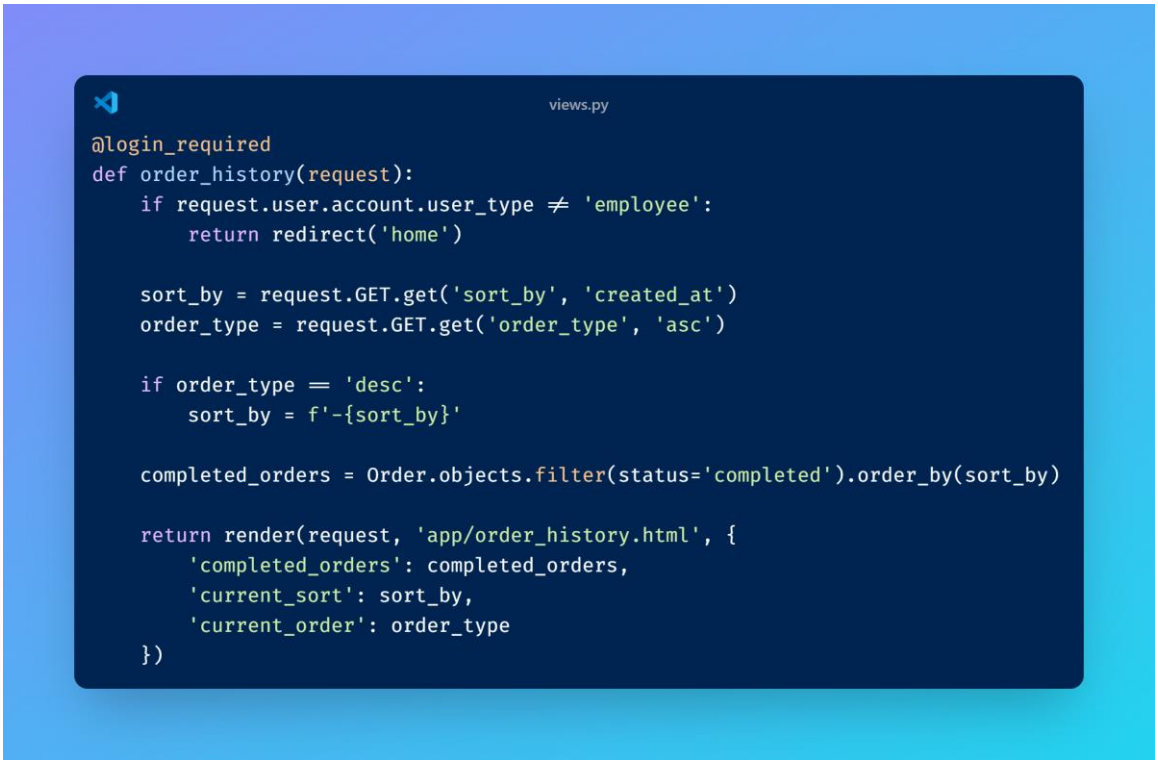
Korisnici prijavljeni kao djelatnici mogu uređivati informacije o dostupnim proizvodima. Ta funkcionalnost dodana je u pogledu *edit_food_item.* Naredba *if request.user.account.user_type != 'employee'* provjerava je li korisnik djelatnik te na temelju toga određuje može li pristupiti toj funkcionalnosti. Ukoliko je metoda *post*, kreira se instanca *FoodItemForm* s podacima iz *post* zahtjeva te se, ukoliko je forma valjana, podaci spremaju u bazu podataka*: form.save().* Sama provjera radi li se o *post* metodi obavlja se linijom koda *if request.method == 'POST'***.** Valjanost forme provjerava se linijom koda *if form.is_valid()***.** Ova naredba provjerava ispunjavaju li podaci u formi potrebne uvjete kao što je provjera je li cijena broj te provjera jesu li polja prazna. Forma i stavka hrane prosljeđuju se predlošku *edit_food_item.html* kako bi se prikazali korisniku. Na slici 4.11 vidljiv je pogled *edit_food_item.*

Sl. 4.11 *Edit_food_item pogled*

Jedna od najbitnijih funkcionalnosti za djelatnike omogućena je pogledom *manage_orders* jer omogućuje djelatnicima pregledavanje i upravljanje narudžbama u sustavu. Prije svega pogled provjerava vrstu prijavljenog korisnika *user_type* te se, ukoliko nije djelatnik, preusmjerava na početnu stranicu. Korisniku je omogućeno sortiranje narudžbi prema različitim kriterijima koji su preuzeti iz *get* zahtjeva, a ako takav ne postoji koriste se zadane vrijednosti. Kriterij za sortiranje može biti *created_at* kako bi mogli sortirati po vremenu kada je narudžba kreirana, a sortiranje može biti uzlazno *asc* ili silazno *desc*. Pogled preuzima narudžbe koje su na čekanju *pending* i odobrene narudžbe *approved*. U skladu s tim kreirane su dvije tablice za prikaz narudžbi s odgovarajućim statusom. Djelatnik tada može kliknuti gumb za odobravanje ili odbijanje narudžbe te se time šalje *post* zahtjev. Pogled tada uzima *id* narudžbe i akciju koja može biti odobri (engl. *approve)* ili odbij (engl. *decline),* a na temelju odabrane akcije status narudžbe se postavlja na *approved* ili *declined*. Ukoliko je narudžba odobrena, prebacuje se u tablicu odobrenih narudžbi te se djelatniku prikaže gumb gotovo (engl. *done)* kojeg treba kliknuti nakon što je narudžba preuzeta i plaćena. Na slici 4.12 je pogled *manage_orders.*

Sl. 4.12 *Manage_orders pogled*

Isključivo za djelatnike implementiran je pogled *order_history* koji omogućuje pregled cijele povijesti završenih narudžbi tako što preuzima sve narudžbe koje su u statusu *completed* iz baze podataka i sortira ih prema odabranom kriteriju. Ovaj pogled igra ključnu ulogu u praćenju poslovanja i analizi prodaje. Na slici 4.13 prikazana je stuktura pogleda *order_history*.



Sl. 4.13 *Order_history pogled*

4.3.3. Definiranje predložaka

Svaka HTML stranica u projektu temelji se na *base.html* predlošku koji predstavlja osnovni dizajn i izgled. Na taj se način održava red i dosljednost kroz cijelu stranicu, što zauzvrat olakšava upravljanje. Linija koda *{% extends 'app/base.html' %}* dio je Django sustava predložaka i koristi se za implementaciju nasljeđivanja predložaka.

Ključna komponenta *base.html* predloška je navigacijska traka koja omogućuje brz i učinkovit pristup svim aspektima web stranice. Kombinacijom Bootstrap klasa i JavaScript koda postignuta je responzivnost navigacijske trake. Element *button* koristi Bootstrapovu klasu *navbar-toggler* koja stilizira gumb za preklapanje navigacijske trake na manjim ekranima. Ova je klasa dio Bootstrapovih komponenti za responzivnu navigaciju. Atribut *aria-expanded="false"* dinamički se ažurira kako bi označio je li navigacijski izbornik trenutno proširen ili skriven, a vrijednost mu se mijenja u *True* ukoliko je izbornik proširen. Bootstrapova JavaScript biblioteka upravlja funkcionalnošću gumba za preklapanje navigacijske trake. Kada se gumb klikne, Bootstrapov JavaScript preklapa vidljivost elementa specificiranog atributom *data-bs-target*. Važno je primjetiti da se stavka *Kontakt* prikazuje samo korisnicima koji su prijavljeni. To je postignuto provjerom *if user.is_authenticated*. Ukoliko je korisnik prijavljen *user.is_authenticated*

je postavljen na *True*, a u suprotnom na *False*. Linija koda *if user.account.user_type != 'employee'* provjerava je li tip korisničkog računa različit od *employee*. To znači da korisnik može biti bilo koji tip osim *employee* kako bi mu taj gumb bio vidljiv. Slično kao za *Kontakt*, postoji provjera je li korisnik prijavljen za prikaz košarice. Uz to se provjerava i je li korisnik isključivo kupac na sljedeći način: *if user.account.user_type == 'customer'*. Prikaz ključnih dijelova u kodu za navigacijsku traku je na slici 4.14.

base.html

```
<li class="nav-item">
                    <a class="nav-link" href="{% url 'home' %}">Naslovnica</a>
                </li>
                {% if user.is_authenticated and user.account.user_type != 'employee' %}
                <li class="nav-item">
                  <a class="nav-link" href="{% url 'contact' %}">Kontakt</a>
                </li>
                {% endif %}
                {% if user.is_authenticated %}
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'logout' %}">Odjava</a>
                    </li>
                {% else %}
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'login' %}">Prijava</a>
                    </li>
                    <li class="nav-item">
                        <a class="nav-link" href="{% url 'register' %}">Registracija</a>
                    </li>
                {% endif %}
                {% if user.is_authenticated and user.account.user_type == 'customer' %}
                    <li class="nav-item position-relative">
                        <a class="nav-link" href="{% url 'view_basket' %}">
                            <i class="ri-shopping-cart-line"></i>
                            Basket
                            {% if basket_item_count > 0 %}
                                <span class="badge bg-danger position-absolute top-0 start-100 translate-middle">{{
basket_item_count }}</span>
                            {% endif %}
                        </a>
                    </li>
                {% endif %}
```

Sl. 4.14 *Navigacijska traka u base.html predlošku*

Ukoliko je korisnik prijavljen, na desnoj strani navigacijske trake prikazat će se korisničko ime. U slučaju da je prijavljeni korisnik djelatnik, prikazuje se odgovarajući tekst pokraj korisničkog imena: *Djelatnik*. Klasa *sticky-top* je Bootstrap klasa koja čini element *ljepljivim* na vrhu. Kada se korisnik na web stranici spušta prema dolje, navigacijska traka će se *zalijepiti* za vrh ekrana i ostati vidljiva, umjesto da nestane s vrha stranice.

U predlošku za registraciju korisnika također je naslijeđen osnovni predložak *base.html* kako bi se održala konzistentnost cijele web aplikacije. Ukoliko postoji greška u unesenim podacima, tada se prikazuje odgovarajući blok s greškama. Linija koda za provjeru grešaka je *{% if form.errors %}*. *UserCreationForm* je ugrađena forma u Djangu koja se koristi za registraciju

novih korisnika. Ona automatski kreira polja za korisničko ime, zaporku i potvrdu zaporke. Na dnu predloška za registraciju dodana je linija koda koja uključuje JavaScript kod za dodavanje funkcionalnosti kod registracije novih djelatnika: *<script src="{% static 'js/registration.js' %}"></script>*. Cjelokupan sadržaj *registration.js* datoteke je na slici 4.15.

Sl. 4.15 *Registration.js*

S ciljem sprječavanja lažnih registracija kao djelatnik, dodana je skrivena funkcionalnost za zaposlenike. Za pristup opciji za registraciju kao djelatnik, korisnici moraju kliknuti na određeni dio stranice 10 puta. Nakon toga, pojavljuje se skriveni gumb koji omogućuje registraciju kao djelatnik. Ovaj mehanizam služi kao zaštita, jer samo oni koji su upoznati s ovim postupkom, poput budućih djelatnika restorana, mogu aktivirati ovu opciju. Time je izbjegnuta potreba za ručnim pregledavanjem i provjeravanjem svih registracija.

4.3.4. Django admin

Kako bi bio omogućen pristup *admin* sučelju bilo je potrebno u glavnom direktoriju unutar *urls.py* datoteke dodati *admin*. To se dodaje u listu *urlpatterns* tako što se doda *path('admin/', admin.site.urls)*. *Admin* korisnik kreiran je naredbom *python manage.py createsuperuser*, pri čemu

je bilo potrebno unijeti korisničko ime i zaporku. Tada je pristup *admin* sučelju dostupan na ruti */admin/*. U datoteci *admin.py* registrirani su svi modeli koje će korisnik s administratorskim ovlastima moći uređivati u svom sučelju. Korisnik s administratorskim ovlastima ima pregled svih korisnika uz informaciju o tome jesu li djelatnik ili kupac, kao i mogućnost uklanjanja i dodavanja istih. Korisnik s administratorskim ovlastima može korisniku promijeniti i vrstu računa te iz svog sučelja dodavati nove artikle hrane te uređivati i brisati postojeće. Može vidjeti sve narudžbe koje su korisnici napravili, uključujući korisnika koji je napravio narudžbu, ukupnu cijenu, način dostave, način plaćanja, adresu i status narudžbe te promijeniti status narudžbe. Korisnik s administratorskim ovlastima može vidjeti sve košarice koje su korisnici kreirali, uključujući korisnika kojem pripada košarica i datum kreiranja. Registracija modela u *admin.py* datoteci prikazana je slikom 4.16 .

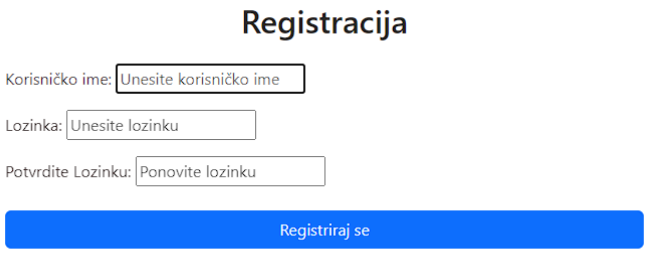
Sl. 4.16 *Registracija modela*

5. IZGLED APLIKACIJE

Izgled aplikacije mijenja se ovisno o tome je li korisnik prijavljen ili odjavljen, te razlikuje sadržaj za djelatnike i kupce. Ako korisnik nije prijavljen, navigacijska traka prikazuje opcije za prijavu i registraciju, dok se svima prikazuje prigodna poruka dobrodošlice. Na lijevoj strani navigacijske trake nalazi se logo restorana, koji je povezan s naslovnicom web stranice. Klikom na logo korisnik se preusmjerava na početnu stranicu, što je ostvareno dodavanjem hiperveze na *home* stranicu putem oznake: *<a class="navbar-brand" href="{% url 'home' %}">*. Na slici 5.1 prikazan je izgled naslovnice za neprijavljene korisnike.

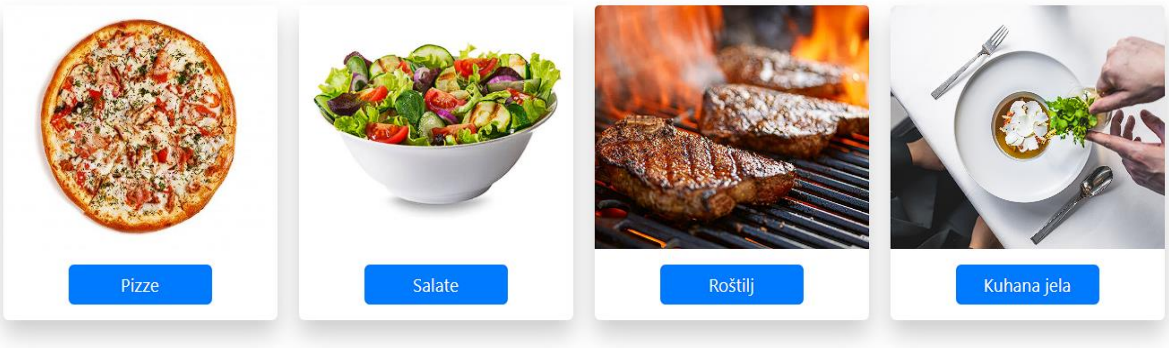
Sl. 5.1 *Naslovnica FoodOS*

Kada korisnik želi registrirati novi račun, može to učiniti klikom na gumb *Registracija* u navigacijskoj traci. Time se preusmjerava na stranicu s registracijskom formom, kao što je prikazano na slici 5.2. Registracijska forma sadrži polja za unos korisničkog imena, zaporke i potvrde zaporke. Nakon što korisnik ispuni sva polja i klikne na gumb *Registriraj se*, podaci se šalju poslužitelju, gdje se provjerava ispravnost unosa te kreira novi korisnički račun.



Sl. 5.2 *Forma za registraciju*

Ispod poruke dobrodošlice na početnoj stranici, neprijavljeni korisnici imaju mogućnost pregledavanja dostupnih kategorija hrane u restoranu. Kategorije su jasno predstavljene vizualno atraktivnim slikama koje odgovaraju svakoj vrsti hrane, što korisnicima omogućuje brzu i jednostavnu navigaciju. Klikom na neku od kategorija otvara se odgovarajuća galerija dostupnih jela. Na slici 5.3 je prikaz kategorija hrane.

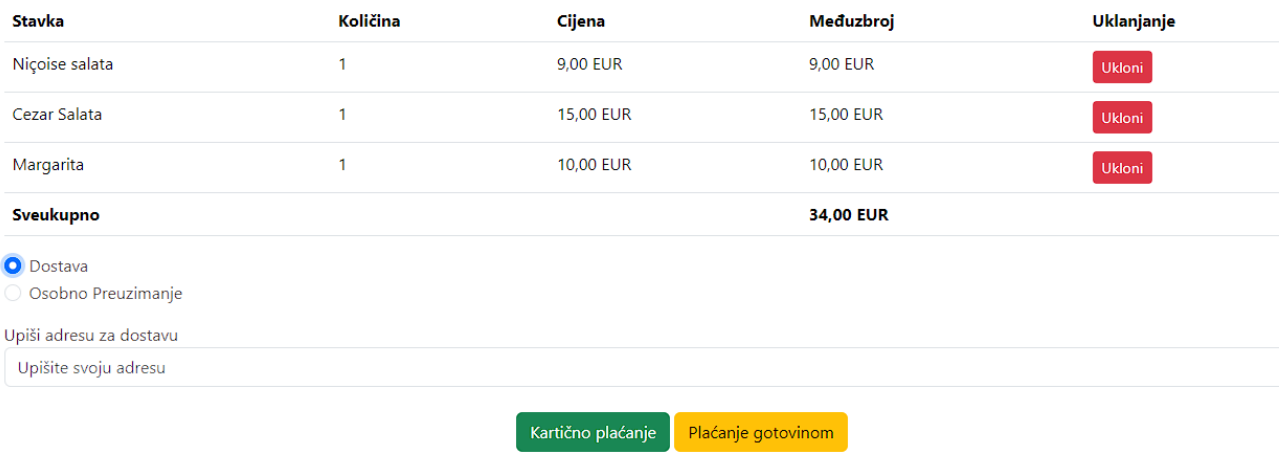


Sl. 5.3 *Kategorije hrane*

Ukoliko korisnik nije prijavljen, u galeriji hrane prelaskom pokazivača preko jela, prikazat će se samo sastojci. Ako se korisnik prijavi, na toj poziciji prikazan je gumb *Dodaj u košaricu*. Ta funkcionalnost vidljiva je na slici 5.4.

Sl. 5.4 *Galerija salata za prijavljenog korisnika*

Dodavanjem stavki u košaricu u navigacijskoj traci iznad košarice pojavljuje se broj koji predstavlja količinu stavki u košarici. Klikom na košaricu korisnika se preusmjerava na sve proizvode koje je u nju stavio, te ima mogućnost uklanjanja stavki. Automatski je izračunata ukupna cijena svih stavki te je ispod ponuđen odabir načina preuzimanja narudžbe. Nudi se osobno preuzimanje i dostava, za koju, ukoliko ju korisnik odabere, treba upisati adresu. Potom korisnik bira način plaćanja, a ukoliko odabere kartično plaćanje, preusmjerava se na Stripe sustav za plaćanje te nakon unosa svih podataka dobije prigodnu poruku zahvale. Primjer košarice sa stavkama prikazan je na slici 5.4.



Sl. 5.4 *Košarica sa stavkama*

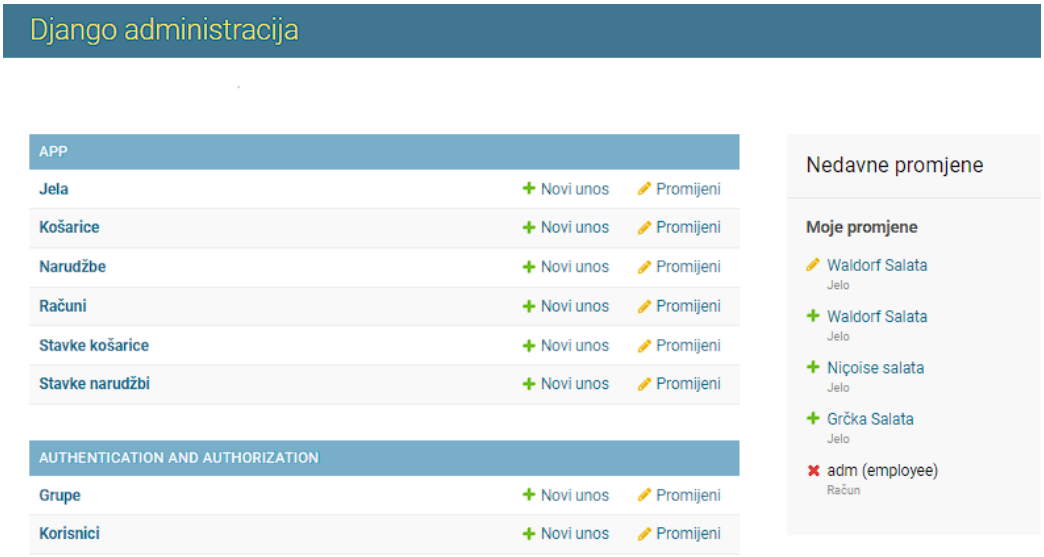
Nakon što korisnik postavi narudžbu, ona postaje vidljiva djelatniku kada pristupi sekciji za upravljanje narudžbama. Podijeljena je u dva dijela pri čemu je u prvom dijelu tablica s narudžbama koje još nisu prihvaćene. Djelatniku su tu vidljive razne korisne informacije o narudžbi te ima mogućnost prihvaćanja ili odbijanja te narudžbe i sortiranje po vremenu. Tablica s neriješenim narudžbama prikazana je slikom 5.5.

Sl. 5.5 *Tablica s neriješenim narudžbama*

Djelatnik na naslovnici ima gumb *Dodaj novo jelo* koje mu omogućava ispunjavanje svih informacija o novom jelu i dodavanje u odgovarajuću kategoriju. Dodavanje novog jela prikazano je slikom 5.6.

Sl. 5.6 *Dodavanje novog jela*

U okviru administracijskog sučelja, korisnik s administratorskim ovlastima može izvršavati ključne operacije koje su važne za upravljanje sadržajem aplikacije. To uključuje i funkcionalnosti dodavanja, brisanja i uređivanja objekata kao što su jela, korisnici i narudžbe. Na slici 5.7 je izgled *admin* sučelja.



Sl. 5.7 *Django admin sučelje*

6. ZAKLJUČAK

Ovaj završni rad postigao je ciljeve postavljene na početku projekta, a to su razvoj i implementacija web aplikacije za naručivanje hrane iz restorana koja odgovara potrebama krajnjih korisnika i pružatelja usluga. Kroz analizu postojećih sustava za naručivanje hrane te istraživanje najboljih praksi u industriji, identificirane su ključne značajke koje moderna aplikacija mora sadržavati kako bi bila konkurentna i relevantna na tržištu. Na temelju tih saznanja, razvijen je sustav koji omogućuje korisnicima pregledavanje dostupnih proizvoda, jednostavno dodavanje u košaricu, odabir opcija dostave i načina plaćanja, dok zaposlenicima nudi alate za upravljanje narudžbama i uređivanje jelovnika.

Izbor Django okvira pokazao se kao vrlo koristan jer je omogućio brzi razvoj aplikacije zahvaljujući svojoj strukturi koja podržava modularnost i olakšava održavanje. Implementirane su ključne sigurnosne mjere, poput zaštite od SQL injekcija, čime je osigurana sigurnost korištenja ove aplikacije te su, integracijom sa Stripe sustavom za plaćanje, osigurane pouzdane transakcije.

Postignuće koje se ističe jest proces registracije korisnika, pri čemu se korisnici mogu registrirati kao osoblje putem nevidljivog gumba, čime se minimiziraju šanse za lažne registracije. Administracijsko sučelje omogućuje potpun nadzor i upravljanje nad svim korisnicima i narudžbama, čime se olakšava poslovna administracija.

Unatoč postignućima u ovom projektu, uočena su i određena ograničenja. Na primjer, aplikacija trenutno ne uključuje napredne analitičke alate koji pomažu u praćenju ponašanja korisnika ili automatsko generiranje izvještaja, što bi moglo pomoći u optimizaciji poslovanja restorana unutar aplikacije. Osim toga, iako je razvijen responzivni dizajn, budući radovi mogli bi uključivati stvaranje nativne mobilne aplikacije kako bi se poboljšalo korisničko iskustvo na mobilnim uređajima.

Kako bi se dodatno poboljšalo zadovoljstvo korisnika, u budućem radu predlaže se proširenje funkcionalnosti aplikacije integracijom sustava personaliziranih preporuka temeljenih na prošlim narudžbama korisnika. Nadalje, mogli bi se uključiti vanjski sustavi za dostavu, poput GPS praćenja i optimizacije ruta, kako bi se povećala učinkovitost dostave i skratilo vrijeme isporuke.

Autor dokumenta vjeruje da postavlja čvrstu osnovu za bilo kakav budući rast koji se tiče web aplikacija povezanih s ugostiteljskom industrijom. Nadalje, dobiveni rezultati pokazali su da postoji mogućnost osmišljavanja učinkovite, ali sigurne platforme za naručivanje hrane koja bi odgovorila na želje i zahtjeve današnjeg društva.

LITERATURA

[1] „Domino’s Pizza / Online narudžbe za dostavu i preuzimanje“ [online]. Dostupno na: https://www.dominos.hr/hr/. [Pristupljeno: 26.8.2024.].

[2] „Uber Eats | Food & Grocery Delivery | Order Groceries and Food Online“ [online]. Dostupno na: https://www.ubereats.com/. [Pristupljeno: 10.9.2024.].

[3] „🏅 Pizzería Online » Pizza Delivery ▷ Papa John’s Costa Rica“ [online]. Dostupno na: https://www.papajohns.cr/. [Pristupljeno: 26.8.2024.].

[4] „Order takeaway online from 30,000+ food delivery restaurants | Just Eat“ [online]. Dostupno na: https://www.just-eat.co.uk/. [Pristupljeno: 26.8.2024.].

[5] „Stripe | Financial Infrastructure to Grow Your Revenue“ [online]. Dostupno na: https://stripe.com/en-hr. [Pristupljeno: 26.8.2024.].

[6] „What is Django? - Django Framework Explained - AWS“ [online]. Dostupno na: https://aws.amazon.com/what-is/django/. [Pristupljeno: 26.8.2024.].

[7] „Django introduction - Learn web development | MDN“ [online], 25-srp-2024. Dostupno na: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction. [Pristupljeno: 27.8.2024.].

[8] „Writing your first Django app, part 1 | Django documentation“ [online]. Dostupno na: https://docs.djangoproject.com/en/5.1/intro/tutorial01/. [Pristupljeno: 28.8.2024.].

SAŽETAK

Ovaj završni rad bavi se razvojem web-baziranog sustava za naručivanje hrane iz restorana s ciljem poboljšanja korisničkog iskustva i optimizacije poslovnih procesa restorana. Ključno je bilo omogućiti krajnjim korisnicima jednostavno naručivanje hrane, dok se pružatelju usluge osiguralo učinkovito upravljanje narudžbama. Korištenjem Django okvira postignut je brz razvoj sustava, implementirane su sigurnosne mjere poput zaštite od SQL injekcija, te je integriran Stripe sustav za sigurno online plaćanje. Krajnji rezultat je operacijski sustav koji korisnicima omogućuje pretraživanje proizvoda, dodavanje u košaricu, odabir načina plaćanja i dostave, dok zaposlenici mogu upravljati narudžbama i jelovnicima.

Ključne riječi: aplikacija, Django, naručivanje hrane, sigurnost, web razvoj

ABSTRACT

Title: Web application for food ordering

This final paper focuses on the development of a web-based system for restaurant food ordering, with the aim of improving the user experience and optimizing the restaurant's business processes. The key goal was to enable end users to order food easily, while also providing the service provider with efficient order management. By using the Django framework, rapid system development was achieved, security measures such as SQL injection protection were implemented, and the Stripe payment system was integrated for secure online payments. The end result is an operational system that allows users to browse products, add items to the cart, choose payment and delivery options, while employees can manage orders and menus.

Keywords: Application, Django, food ordering, security, web development.

PRILOZI

Na Gitlab-u se nalazi cijeli projekt: https://gitlab.com/ferit3622643/aplikacija-foodos-zavrsni-rad