IZRADA WEB TRGOVINE SPECIJALIZIRANE ZA PRODAJU KOŠARKAŠKE OPREME

Begović, Lovre

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijske tehnologije

LOVRE BEGOVIĆ

ZAVRŠNI RAD

IZRADA WEB TRGOVINE SPECIJALIZIRANE ZA PRODAJU KOŠARKAŠKE OPREME

Split, rujan 2022.

SVEUČILIŠTE U SPLITU
SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Informacijske tehnologije

Predmet: Baze podataka

Z A V R Š N I R A D

Kandidat: Lovre Begović

Naslov rada: Izrada web trgovine specijalizirane za prodaju košarkaške opreme

Mentor: mr. sc. Ivica Ružić, viši predavač

Split, rujan 2022.

SADRŽAJ

Sažetak

U ovom završnom radu je napravljena web aplikacija specijalizirana za prodaju košarkaške opreme. Aplikacija je napravljena tako da omogućuje korisniku registriranje, prijavljivanje i uređivanje korisničkog računa, te omogućuje pregledavanje svih dostupnih artikala i naručivanje istih. Također svi korisnici imaju mogućnost i pregledavati povijest svih svojih narudžbi kao i pregled najprodavanijih artikala. Poslužiteljska strana (engl. *backend*) napravljena je u okviru (engl. *framework*) Django u kojem se koristi programski jezik Python, dok je korisničko sučelje (engl. *frontend*) napravljen pomoću HTML-a (HyperText Markup Language), osnovnog CSS-a (Cascading Style Sheets), JavaScripta te pomoću CSS okvira Bootstrap koji omogućava jednostavno implementiranje responzivnog dizajna kao i uređivanja mrežne (engl. *web*) stranice. Za pohranu podataka korištena je zadana baza podataka od strane Djanga-a koja se zove SQLite.

Ključne riječi: web aplikacija, Django, Python, Bootrstap, SQLite

Summary

Development of a web shop specialized for the sale of basketball equipment

In this final work, a web application specialized for the sale of basketball equipment was created. The application is made in such a way that it allows the user to register, log in and edit the user account. It allows viewing all available items and ordering them. All users have the possibility to view the history of all their orders as well as an overview of the best-selling items. The server side is made in the framework called Django in which the programming language Python is used, while the frontend is made using HTML (HyperText Markup Language), basic CSS (Cascading Style Sheets), JavaScript and using the Bootstrap CSS framework that enables simple implementation of responsive design as well as editing of a web page. Django's default database called SQLite is used for data storage.

Keywords: web application, Django, Python, Bootrstap, SQLite

1. Uvod

Ovaj završni rad je napravljen zbog velikog porasta i potražnje za internetskim (engl. *online*) trgovinama. Internetske trgovine su u sve većem rastu i puno je lakše i isplativije otvoriti internetsku trgovinu od samog otvaranja dućana u nekom od tržišnih centara. Cilj ovakvih web trgovina je brzo i jednostavno pretraživanje artikala, filtriranje istih, brzo plaćanje, te sigurnost korisnika i posjetitelja trgovine. Aplikacija je podijeljena u dvije vrste prikaza. Jedna je ona koju vidi administrator, dok je druga je namijenjena za samog korisnika. Administrator ima više ovlasti od samog korisnika, neke od njih su mogućnost pregleda najboljih kupaca, dodavanje artikala, brisanje istih, te pregled i izmjena svih dosadašnjih narudžbi. Završni rad je podijeljen u četiri poglavlja.

U prvom poglavlju opisuje se razlog pisanja završnog rada na temu internetske trgovine. U poglavlju „Korištene tehnologije“ su opisane sve tehnologije koje su se koristile za vrijeme izrade samog rada. Unutar trećeg poglavlja „Funkcionalnost aplikacije“ je objašnjena detaljna izrada baze podataka. Prikazan je način pisanja koda i izgled same web aplikacije, te na kraju i sami zaključak unutar kojeg se argumentira i izvještava sveukupni završni rad.

2. Korištene tehnologije

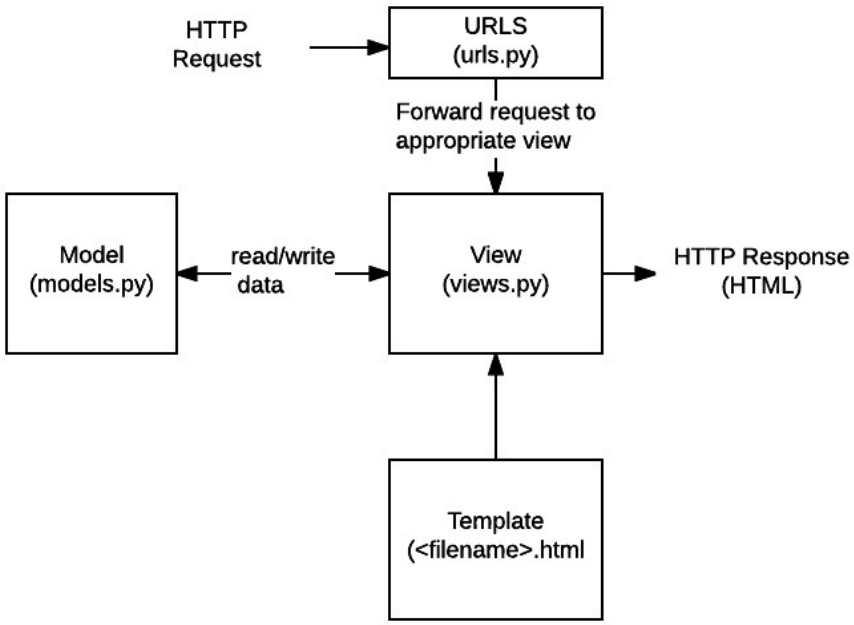
Kao što je spomenuto u sažetku, glavna tehnologija s kojom se napisao ovaj završni rad je Django. Django je Python web okvir visoke razine koji omogućuje brz razvoj sigurnih web stranica koje se mogu jednostavno održavati. Napravljen je 2003. godine od strane Adriana Holovatya i Simona Willisona, vrijeme kada se Python programski jezik počeo koristiti za izradu aplikacija. Trenutna standardna baza podataka od Djanga je SQLite.

Uz Django i Python korištene su tehnologije kao što su HTML, CSS, Bootstrap, te JavaScript. Ove tehnologije su se koristile isključivo za korisničko sučelje. HTML tehnologija se koristi kao standardni način za pisanje teksta koji će se prikazivati na web stranici. Tehnologije kao što su CSS i njegov okvir (engl. *framework*) Bootstrap su korištene isključivo za dizajn stranice. JavaScript je korišten da se prikažu animacije, skočni prozori te za prikaz botuna koji se koristio kako bi korisnika vratio na vrh stranice.

2.1. Django

Django je okruženje koje se bazira na potpunosti, svestranosti, sigurnosti, skalabilnosti, te mogućnosti jednostavnog održavanja i prijenosa.

Django web aplikacije se najčešće pišu tako da su najveći dijelovi koda odvojeni u datotekama: models.py, urls.py, views.py, te svi predlošci (engl. *template*) u datotekama s nastavkom .html u kojima se nalazi HTML kod koji će se prikazivati na web stranici. Način rada Django aplikacije prikazan je na slici 1.



Slika 1: *Način rada Django aplikacije [2]*

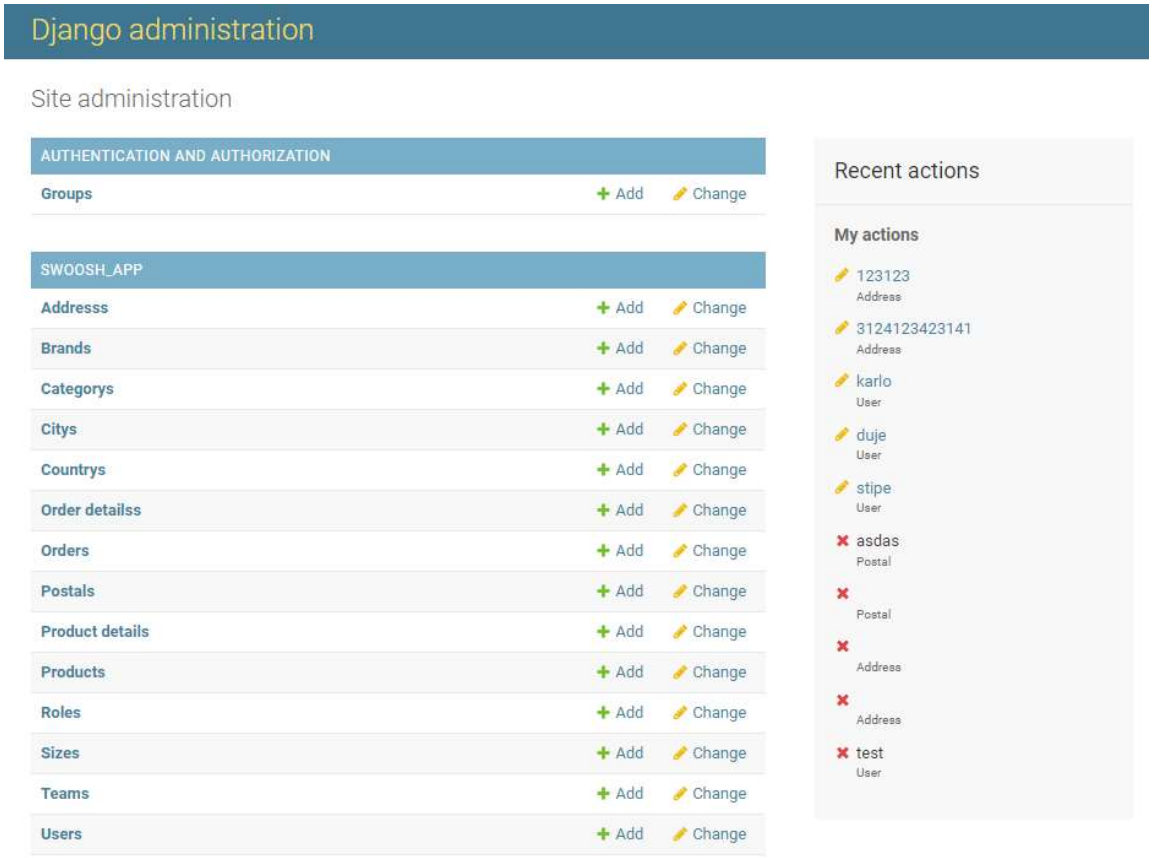
Kada se primi zahtjev, na temelju URL-a, ovisno o tome je li POST ili GET zahtjev, aplikacija radi ono što je potrebno ovisno o informacija dobivenim unutar tih zahtjeva. Kada se zahtjev primi može čitati ili pisati informacije iz baze podataka ili obavljati druge zadatke potrebne za udovoljavanje istoga. Aplikacija će zatim vratiti odgovor web-pregledniku, često dinamički stvarajući HTML stranicu koju preglednik prikazuje. Kada se dohvati URL zahtjev on se najčešće obrađuje u views.py datoteci unutar jedne ili više funkcije ili klase, te ta funkcija vrati odgovor u obliku HTTP (Hypertext Transfer Protocol) formata. Unutar models.py datoteke se nalaze objekti koji definiraju strukturu podataka aplikacije i daju mehanizme za upravljanje (dodavanje, mijenjanje, brisanje) i upite zapisa u bazi podataka, dok se u .html datotekama koje se najčešće nalaze u „templates“ direktoriju nalaze tekstualni sadržaji koje definiraju strukturu ili izgled datoteke (kao što je HTML stranica).

Django nam također daje i mogućnost instalacije velikog broja manjih aplikacija koje se mogu instalirati u našoj glavnoj aplikaciji. Neke od tih aplikacija su „Stripe“, tehnologija pomoću koje se izvršava plaćanje pomoću kartice unutar aplikacije, zatim „django_filters“, tehnologija koja se koristila da omogući jednostavno filtriranje podataka po tipu proizvoda, marki, nazivu itd. Uz navedene aplikacije može se izdvojiti i aplikacija „pillow“ bez koje ne bi mogli spremati slike u bazu podataka, te „django-phonenumber-field“ - tehnologija pomoću koje se provjerava točnost unesenog telefonskog broja za određeno područje koje je navedemo unutar settings.py datoteke. Sve ove aplikacije koje se dodatno instaliraju

potrebno je spremiti u datoteku requirements.txt da bi aplikacija u potpunosti funkcionirala i na drugim računalima na kojima ju testiramo. U datoteci requirements.txt se nalaze sve dodatne aplikacije koje su instalirane unutar naše Django aplikaciji te se spremaju pomoću naredbe „pip freeze > requirements.txt“.

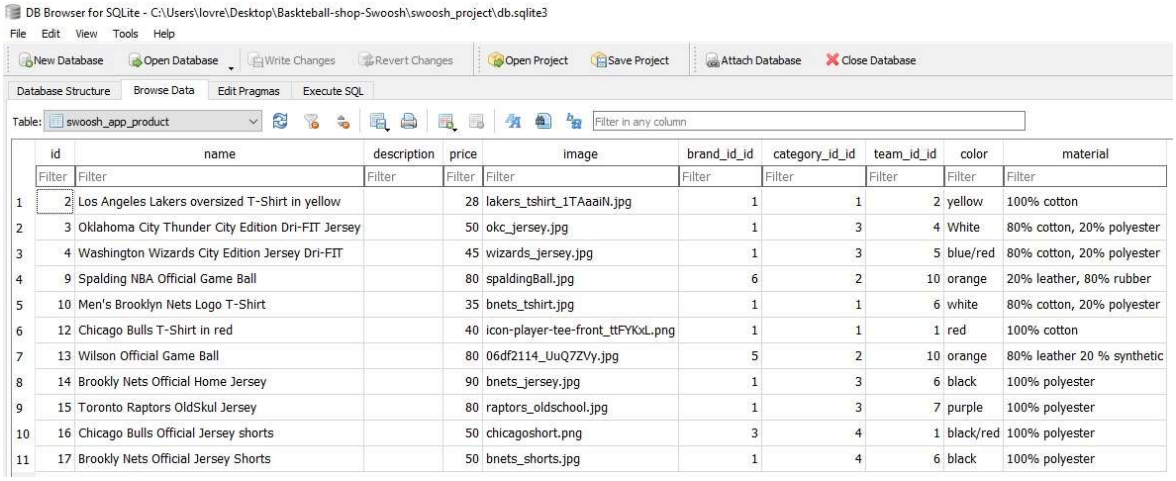
2.2. SQLite

SQLite je baza podataka koja je korištena od strane Djanga. Bazi podataka možemo pristupiti otvaranjem admin sučelja na web lokaciji „localhost/admin“ te prijavljivanjem sa super korisnikom kojeg smo morali prethodno napraviti unutar naše aplikacije. Na admin sučelju naše aplikacije super korisnik ima mogućnost čitanja, pisanja ažuriranja i brisanja svih elemenata koji se nalaze unutar baze. Na slici 2 je prikazan izgled baze podataka unutar admin sučelja.



Slika 2: *Primjer baze podataka unutar admin sučelja*

Modeli koji su se kreirali u aplikaciji se spremaju u zadanu bazu podataka SQLite koja omogućava brz i jednostavan pristup podacima te pregled istih. Također bazu podataka možemo otvoriti i pomoću SQLite aplikacije otvaranjem „db.sqlite3“ datoteke koja se automatski kreira kada napravimo sam projekt i aplikaciju. Na slici 3 se nalazi primjer modela „Product“ unutar aplikacije SQLite. Ova aplikacija nam ne omogućuje izmjenu podataka, ali nam može pomoći za sam pregled iste.


DB Browser for SQLite - C:\Users\lovre\Desktop\Baskteball-shop-Swoosh\swoosh_project\db.sqlite3

File Edit View Tools Help

New Database | Open Database | Write Changes | Revert Changes | Open Project | Save Project | Attach Database | Close Database

Database Structure | Browse Data | Edit Pragmas | Execute SQL

Table: swoosh_app_product | Filter in any column

	id	name	description	price	image	brand_id_id	category_id_id	team_id_id	color	material
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	2	Los Angeles Lakers oversized T-Shirt in yellow		28	lakers_tshirt_1TAaaiN.jpg	1	1	2	yellow	100% cotton
2	3	Oklahoma City Thunder City Edition Dri-FIT Jersey		50	okc_jersey.jpg	1	3	4	White	80% cotton, 20% polyester
3	4	Washington Wizards City Edition Jersey Dri-FIT		45	wizards_jersey.jpg	1	3	5	blue/red	80% cotton, 20% polyester
4	9	Spalding NBA Official Game Ball		80	spaldingBall.jpg	6	2	10	orange	20% leather, 80% rubber
5	10	Men's Brooklyn Nets Logo T-Shirt		35	bnets_tshirt.jpg	1	1	6	white	80% cotton, 20% polyester
6	12	Chicago Bulls T-Shirt in red		40	icon-player-tee-front_ttFYKxL.png	1	1	1	red	100% cotton
7	13	Wilson Official Game Ball		80	06df2114_UuQ7ZVy.jpg	5	2	10	orange	80% leather 20 % synthetic
8	14	Brookly Nets Official Home Jersey		90	bnets_jersey.jpg	1	3	6	black	100% polyester
9	15	Toronto Raptors OldSkul Jersey		80	raptors_oldschool.jpg	1	3	7	purple	100% polyester
10	16	Chicago Bulls Official Jersey shorts		50	chicagoshort.png	3	4	1	black/red	100% polyester
11	17	Brookly Nets Official Jersey Shorts		50	bnets_shorts.jpg	1	4	6	black	100% polyester

Slika 3: *Model “Product“ unutar aplikacije SQLite*

2.3. Bootstrap

Bootstrap je moćan set alata namijenjen za korisničko sučelje prepun značajki. Omogućuje nam da napravimo bilo što, od prototipa do proizvodnje u malo vremena. Jako je jednostavan za instalaciju. Možemo ga preuzeti sa službene Bootstrap stranice i dodati instaliranu datoteku u direktorij ili jednostavnu ubaciti link unutar našeg HTML-a u <head> zaglavlje, ili na dnu <body> zaglavlja ako se želi dodati JavaScript datoteka. Bootstrap nam omogućuje jednostavno, efikasno i brzo dizajniranje responzivne web stranice s korištenjem mnoštva animacija proizvoljnog izgleda. Cilj responzivnog dizajna je izrada web stranica koje detektiraju veličinu i orijentaciju zaslona posjetitelja te u skladu s tim mijenjaju izgled. Većina sadašnjih web stranica koristi Bootstrap okvir kako bi dizajnirali izgled svoje aplikacije te je to jedan od razloga zašto puno stranica izgleda na isti princip. Naravno da se istraže sve mogućnosti koje nam Bootstrap nudi, u njemu sigurno ima materijala za napraviti dizajn kakav drugi nemaju. Zadnja službena verzija Bootstrapa je Bootstrap 5.2.

3. Funkcionalnost aplikacije

Cilj ove web aplikacije je omogućiti sigurnu i jednostavnu mrežnu kupovinu košarkaških proizvoda putem interneta. U ovom poglavlju bit će opisane sve funkcionalnosti aplikacije, od registracije i prijave do samog izvršavanja narudžbe. Prikazat će se ovlasti administratora i samog korisnika. Isto tako bit će objašnjeni dijelovi poslužiteljske i klijentske strane aplikacije te njihova uloga. Uz samo aplikaciju bit će detaljno opisana baza podataka, pojedini entiteti i kardinalnost.

3.1. Izrada baze podataka

Baza podataka u ovom radu je napravljena od četrnaest tablica, od kojih svaka ima određen broj atributa koji je opisuju. Po pravilu bi tablica koja međusobno povezuje dvije druge tablice trebala imati dva strana ključa spojena kao primarni ključ, no pošto Django automatski napravi primarni ključ za određeni model, nije bilo potrebe za tim.

U poglavlju „Entiteti i kardinalnost“ su prikazane sve tablice s njihovim atributima, ključevima, te njihova kardinalnost. Polje s kardinalnošću prima dva broja. Prvi broj određuje minimalni broj atributa (engl. min card), odnosno je li atribut obavezan ili ne (0 označava da nije, 1 da je), dok drugi broj označava maksimalni broj atributa (engl. *max card*), broj koji označava koliko ih može biti u tom polju.

U poglavlju „Dijagram E-R“ je prikazana slika potpune baze podataka sa svim relacijama između entiteta.

3.1.1. Entiteti i kardinalnost

Tablica 1: *Entitet "User"*

User				
Naziv atributa	Tip	Veličina	Primarni ključ	Kardinalnost
ID	Broj	30	DA	(1,1)
Username	Char	150		(1,1)
First_name	Char	30		(1,1)
Last_name	Char	30		(1,1)
Email	Email	320		(1,1)
Phone	PhoneNumber	12		(1,1)
Role_id	Broj	30		(1,1)
Address	Char	30		(1,1)
Location_id	Broj	30		(1,1)

Entitet „User“ predstavlja sve standardne atribute koji mogu opisivati jednog korisnika uz njegov ID. Sve ove informacije korisnik će morati unijeti tijekom registracije i imat će mogućnost izmjene istih jednom kada napravi račun. U tablici imamo i dva strana ključa, a to su ključ „role_id“, ključ koji će opisivati korisnikovu ulogu (administrator ili obični korisnik) i strani ključ „location_id“ koji povezuje korisnika s gradom i državom. Sva polja u tablici su obavezna jer će se koristiti u slučaju da korisnik odluči naručiti neki od proizvoda.

Tablica 2: *Entitet "Role"*

Role				
Naziv atributa	Tip	Veličina	Primarni ključ	Kardinalnost
ID	Broj	30	DA	(1,1)
Role	Char	50		(1,1)

Entitet „Role“ predstavlja tablicu kojoj će biti opisane sve moguće uloge korisnika. Po zadanome svaki novi korisnik će dobiti ulogu „user“. To je standardna uloga pomoću koje korisnik može raditi izmjene samo na svome račune. Uz ulogu „user“ postojat će i uloga

„admin“, ona će davati administratoru veće uloge od standardnog korisnika. Administrator će imati priliku raditi izmjene puno većeg broja komponenata, te uvid u svakog korisnika.

Tablica 3: *Entitet "City"*

City				
Naziv atributa	Tip	Veličina	Primarni ključ	Kardinalnost
ID	Broj	30	DA	(1,1)
Name	Char	100		(1,1)
Postal_code	Char	30		(1,1)
Country_id	Broj	30		(1,1)

U entitetu „City“ je detaljno opisan grad iz kojeg korisnik dolazi. Entitet „City“ u sebi ima strani ključi naziva „Country_id“ koji je povezan s relacijom jedan na prema više. Odnosno jedna država može imati više gradova, dok taj jedan grad može pripadati samo jednoj državi.

Tablica 4: *Entitet " Country "*

Country				
Naziv atributa	Tip	Veličina	Primarni ključ	Kardinalnost
ID	Broj	30	DA	(1,1)
Name	Char	100		(1,1)

U ovom entitetu se nalazi ime države i njen ID. Entitet „Country“ će biti međusobno povezan s entitetom „City“ te će opisivati državu u kojoj se taj grad nalazi.

Tablica 5: *Entitet "OrderDetails"*

OrderDetails				
Naziv atributa	Tip	Veličina	Primarni ključ	Kardinalnost
Product_id	Broj	30	DA	(1,1)
Order_id	Broj	30	DA	(1,1)
Date	Datum			(1,1)
Status	Char	100		(1,1)
Quantity	Broj	30		(1,1)

U Tablici 5 u kojoj je prikazan entitet „OrderDetails“ je opisana količina jednog proizvoda koji se nalazi u košarici, kao datum i status tog proizvoda u narudžbi. U entitetu se nalaze dva strana ključa „Product_id“ i „Order_id“ koji zajedno stvaraju jedan primarni ključ. Ova tablica je napravljena da predstavi vezu između entiteta „Order“ i entiteta „ProductDetail“. Entitet je nastao jer se na jednoj narudžbi može nalaziti više proizvoda, a taj isti proizvod se može nalaziti na više narudžbi.

Tablica 6: *Entitet "Order"*

Order				
Naziv atributa	Tip	Veličina	Primarni ključ	Kardinalnost
ID	Broj	30	DA	(1,1)
Customer_id	Broj	30		(1,1)
Number	Char	100		(1,1)
Date	Datum			(1,1)
Status	Char	100		(1,1)

Entitet „Order“ predstavlja tablicu u kojoj su naveden sve potrebne informacije za pojedinu narudžbu. Ovaj entitet je povezan s entitetom „ProductDetail“ pomoću dodatno nastale tablice „OrderDetails“. Tablica također u sebi ima i strani ključ „Customer_id“ . Pošto jedan korisnik može imati više narudžbi, a jedna narudžba može pripadati samo jednom korisniku, ova tablica je povezana s tablicom „User“ relacijom jedan na prema više pomoću atributa „Customer_id“. Uz naveden atribute svaka narudžba ima i svoj „status“ koji prikazuje status narudžbe (košarica ili plaćeno), broj narudžbe „number“, te atribut „Date“ koji određuje kada je narudžba zaprimljena.

***Tablica 7**: Entitet "ProductDetail"*

ProductDetail				
Naziv atributa	Tip	Veličina	Primarni ključ	Kardinalnost
ID	Broj	30	DA	(1,1)
Product_id	Broj	30		(1,1)
Size_id	Broj	30		(1,1)
Quantity	Broj	30		(1,1)

Entitet „ProductDetail“ predstavlja tablicu u kojoj je detaljno opisan određeni proizvod. U ovom entitetu atribut „Quantity“ predstavlja stvarnu količinu preostalih proizvoda određene veličine dok u entitetu „OrderDetail“ atribut „Quantity“ predstavlja količinu ovog proizvoda unutar košarice. Ovaj entitet ima dva strana ključa, to su „Size_id“ i „Product_id“. „Product_id“ predstavlja proizvod, dok „Size_id“ predstavlja veličinu. Ova tablica je potrebna jer proizvod u jednoj veličini može biti dostupan za prodaju, dok taj isti proizvod u drugoj veličini možda više nije dostupan.

***Tablica 8:** Entitet "Size"*

Size				
Naziv atributa	Tip	Veličina	Primarni ključ	Kardinalnost
ID	Broj	30	DA	(1,1)
Size	Char	20		(1,1)

Entitet Size predstavlja vrlo jednostavan entitet sa samo dva atributa. Atribut „ID“ i atribut „Size“. Ovaj entitet služi da bi se opisale sve moguće veličine proizvoda. Povezan je s entitetom „ProductDetail“ u kojem opisuje njegovu veličinu. Povezan je relacijom jedan na prema više zato što jedna veličina može pripadati više proizvoda, a jedan proizvod može imati samo jednu veličinu.

Tablica 9: *Entitet "Product"*

Product				
Naziv atributa	Tip	Veličina	Primarni ključ	Kardinalnost
ID	Broj	30	DA	(1,1)
Team_id	Broj	30		(1,1)
Category_id	Broj	30		(1,1)
Brand_id	Broj	30		(1,1)
Name	Char	50		(1,1)
Material	Char	50		(1,1)
Description	Char	400		(0,1)
Color	Char	12		(1,1)
Price	Decimal	10		(1,1)
Image	Slika			

U entitetu „Product“ su opisani svi detalji proizvoda koji se kupuje i njegove karakteristike. Tablica ima tri strana ključa koji daju informaciju o timu kojem pripada, kategoriji i marki. Uz to može se spomenuti i atribut „Image“ koji predstavlja sliku proizvoda i zbog kojeg se trebao instalirati dodatak „pillow“ unutar Django aplikacije kako bi se ta slika mogla pohraniti.

Sljedeće tri tablice prikazuju entitete koji su povezani na entitet „Product“. To su tri jednostavne tablice s dva atributa u svakoj. To su atributi „ID“ i „Name“. Sva tri entiteta su povezana s entitetom „Product“ u relaciji jedan na prema više jer jedna marka, jedna kategorija i jedan tim može pripadati više proizvoda dok taj jedan proizvode može pripadati samo jednoj marki, kategoriji i timu.

Tablica 10: *Entitet "Team"*

Team				
Naziv atributa	Tip	Veličina	Primarni ključ	Kardinalnost
ID	Broj	30	DA	(1,1)
Name	Char	50		(1,1)

Tablica 11: *Entitet "Brand"*

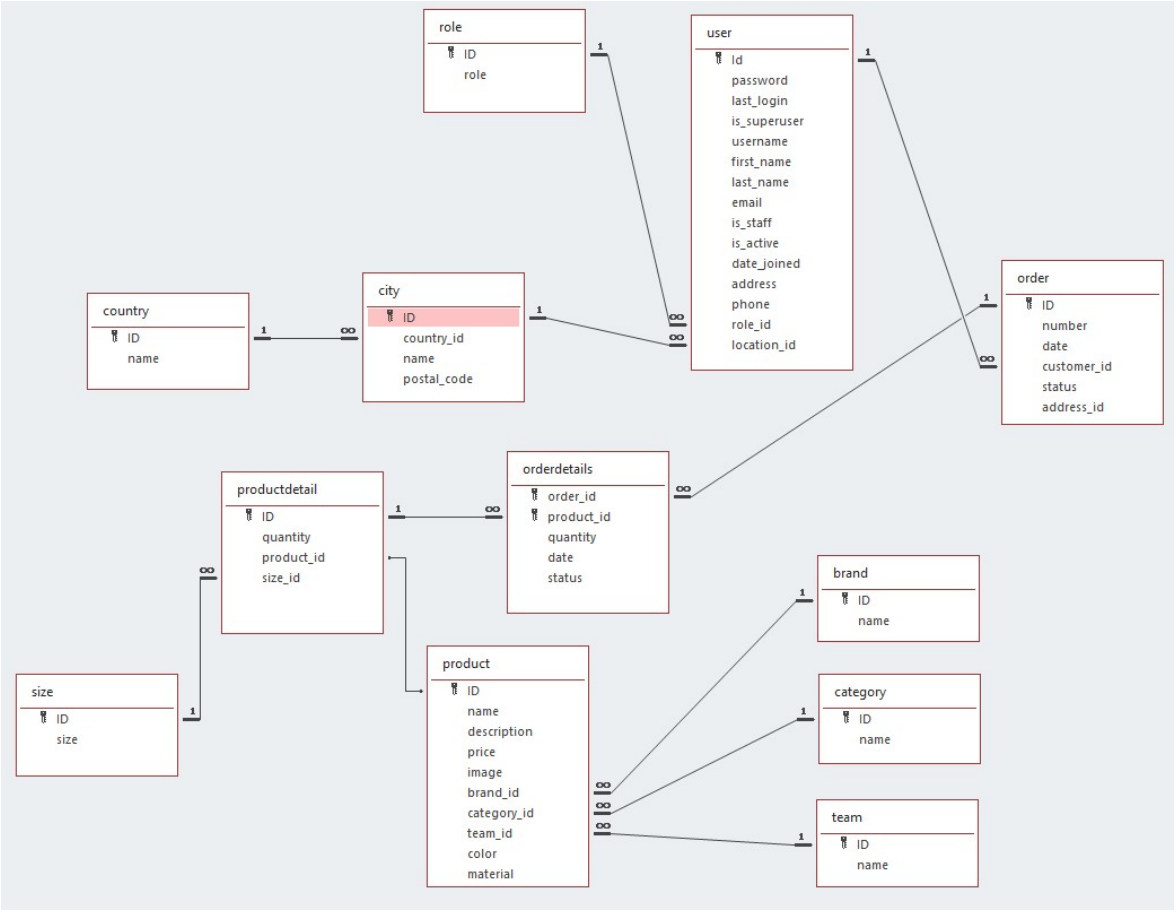
Brand				
Naziv atributa	Tip	Veličina	Primarni ključ	Kardinalnost
ID	Broj	30	DA	(1,1)
Name	Char	50		(1,1)

Tablica 12: *Entitet "Category"*

Category				
Naziv atributa	Tip	Veličina	Primarni ključ	Kardinalnost
ID	Broj	30	DA	(1,1)
Name	Char	50		(1,1)

3.1.2. Dijagram E-R

Na slici 4 se nalazi dijagram E-R koji predstavlja međusobne odnose između svih entiteta koji postoje unutar baze podataka.

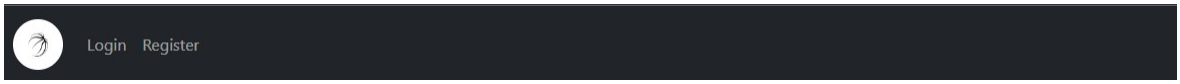


Slika 4: *Dijagram E-R*

3.2. Izrada aplikacije

3.2.1. Izrada korisničkog računa i prijava

Kod prvog pristupa stranici ako korisnik nije prijavljen u navigacijskoj traci ima tri poveznice: Logo, Login i Register. Klikom na Logo korisniku se otvara početna stranica web trgovine s prikazom svih proizvoda, dok se pomoću preostale dvije poveznice može prijaviti u svoj postojeći račun ili izraditi novi. Izgled navigacije se nalazi na slici 5.



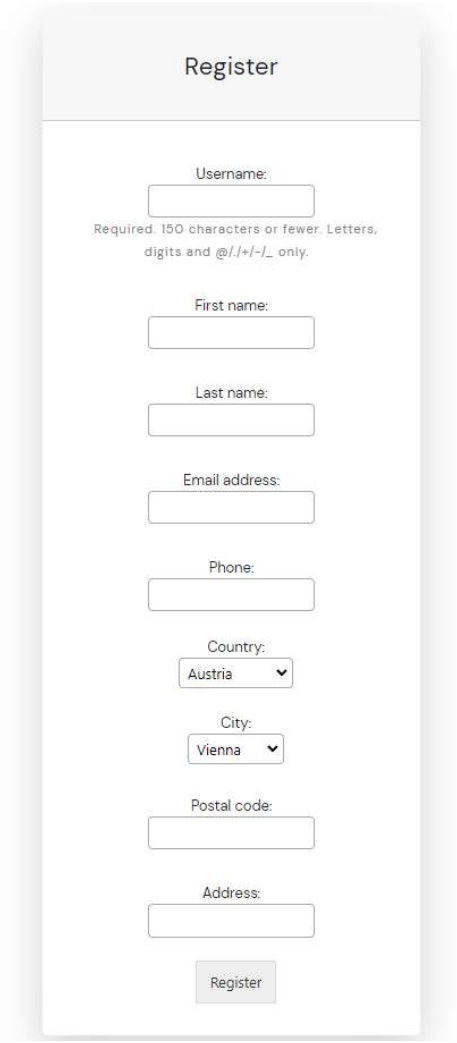
Slika 5: *Navigacijska traka kada korisnik nije prijavljen*

U poveznici „Register“ korisnik mora ispuniti sve podatke unutar forme kako bi napravio svoj korisnički račun. Za model „User“ se koristio standardni oblik Django korisnika tako što je dodana apstraktna klasa „Django Abstract User“ koja nasljeđuje sva polja od Django korisnika i daje nam mogućnost dodavanja vlastitih polja i metoda profila. Na ovoj aplikaciji je korištena Djangova „UserCreationForm“ forma pomoću koje se kreira novi korisnik. Registracija je prikazna na slici 6.

Kako bi dodali „Django Abstract User“ korisnika moramo ga implementirati na početku naše models.py datoteke. Isto tako u našu klasu User moramo u zagrade dodati ime implementacije. Na ispisu 1 se nalazi primjer modela „User“.

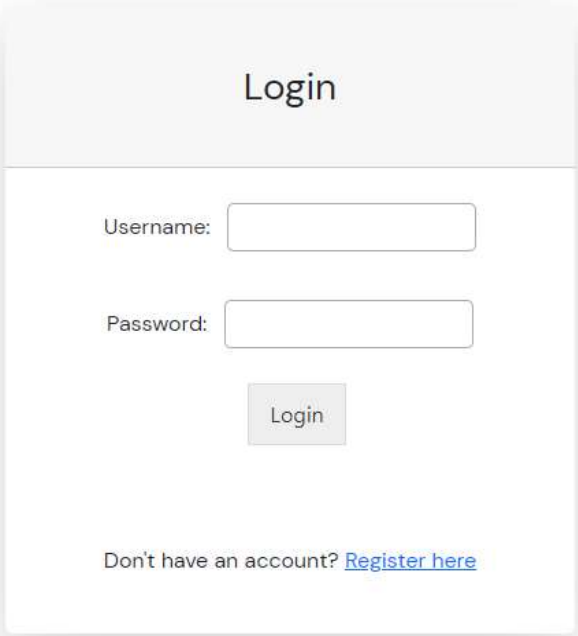
```
from django.contrib.auth.models import AbstractUser

class User(AbstractUser):
      role_id = models.ForeignKey(Role, on_delete=models.CASCADE,
      null=True, default=1)
      phone = PhoneNumberField(null=True, blank=False, unique=False)
      location = models.ForeignKey(Address, on_delete=models.SET_NULL,
      null=True, blank=True)
```

***Ispis 1**: Model User*

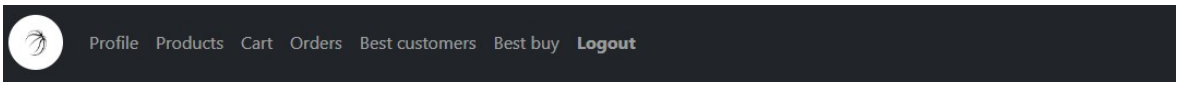


***Slika 6:** Izgled registracijske forme*

Forma za prijavljivanje (engl. *login*) funkcionira tako da na nakon što unesemo korisničko ime i lozinku prvo pretražuje u bazi podataka postoji li korisnik s tim korisničkim imenom. Kada se taj korisnik pronađe uspoređuje se hashirana lozinka poslana preko forme s hashiranom lozinkom koja je spremljena za tog korisnika unutar baze podataka. Ako se hash podudara korisnik je uspješno prijavljen i otvaraju mu se i ostala polja unutar navigacijske trake, kao i mogućnost same narudžbe proizvoda i dodavanja istih u košaricu. Forma za prijavu je napravljena pomoću „LoginView-a“ koji omogućuje prijavu korisnika. Na nama je da samo napravimo .html predložak u kojem će biti forma za prijavu i njen dizajn, te u postavka aplikacije moramo odrediti gdje će nas odvesti ako se uspješno prijavimo. LOGIN_REDIRECT_URL = 'naziv našeg url-a'. Dizajn od stranice za prijavu se može vidjeti na slici 7, dok je izgled navigacije nakon uspješne prijave prikazan na slici 8.



***Slika** 7: Izgled forme za prijavljivanje*



Slika 8: *Navigacijska traka nakon uspješne prijave*

3.2.2. Početna stranica i pregled proizvoda

Početna stranica je dizajnira tako da ispiše listu svih proizvoda s mogućnošću otvaranja svakog proizvoda pojedinačno. Svaki proizvod ima svoju sliku, ime, opis i cijenu. Korisnik može otvoriti detalje o proizvodu klikom na sliku ili klikom vezu s desne strane „Buy here“ koju vide obični korisnici, dok admin u tom području vidi opciju za uređivanje i brisanje tog proizvoda. Neprijavljeni korisnici također imaju mogućnost vidjeti sve proizvode, no ako se odluče dodati proizvod u košaricu ili ga kupiti moraju se prije toga prijaviti ili napraviti novi račun.

```
def productlist(request):
    info = ''
    if request.method == 'POST':
        product = request.POST['product']
        action = request.POST['action']
        if action == 'delete':
            info = deleteproduct(product)

    products = Product.objects.all().order_by('id')
    myFilter = ProductFilter(request.GET, queryset=products)
    products = myFilter.qs
    paginator = Paginator(products, 5)
    page = request.GET.get('page')
    products_pages = paginator.get_page(page)
    data = {
        'info':info,
        'products': products,
        'myFilter': myFilter,
        'products_pages': products_pages
    }
    return render(request, 'products/productlist.html', data)
```

Ispis 2: *Funkcija za dohvacanje svih proizvoda*

U ispisu 2 prikazana je funkcija pomoću koje se mogu dohvatiti svi proizvodi. Oni se dohvaćaju tako da se dohvate svi objekti tipa „Product“ te se filtriraju po njihovom idu-u. Također ova funkcija poziva i funkciju ProductFilter, funkciju koja je kreirana u drugoj datoteci. „ProductFilter“ je kreiran pomoću „django-filters“ instalacije i daje nam mogućnost filtriranja proizvoda na način koji želimo. To može biti padajuća lista (engl. *drop down list*), dohvaćanje elemenata koji se zovu točno onako kako smo upisali u tražilici ili dohvaćanje elemenata koji počinju s onim što je korisnik počeo unositi. U ovom slučaju se koristila padajuća lista koja je omogućavala filtriranje po kategoriji, timu marki, te mogućnost unosa imena proizvoda. Također se poziva i Django dodatak „Paginator“ koji nam omogućuje da razdvojimo naše proizvode u više stranica da se ne prikazuje svi na jednoj, te nam omogućuje da dohvatimo informaciju na kojoj se stranici trenutno nalazimo. U funkciji se isto tako

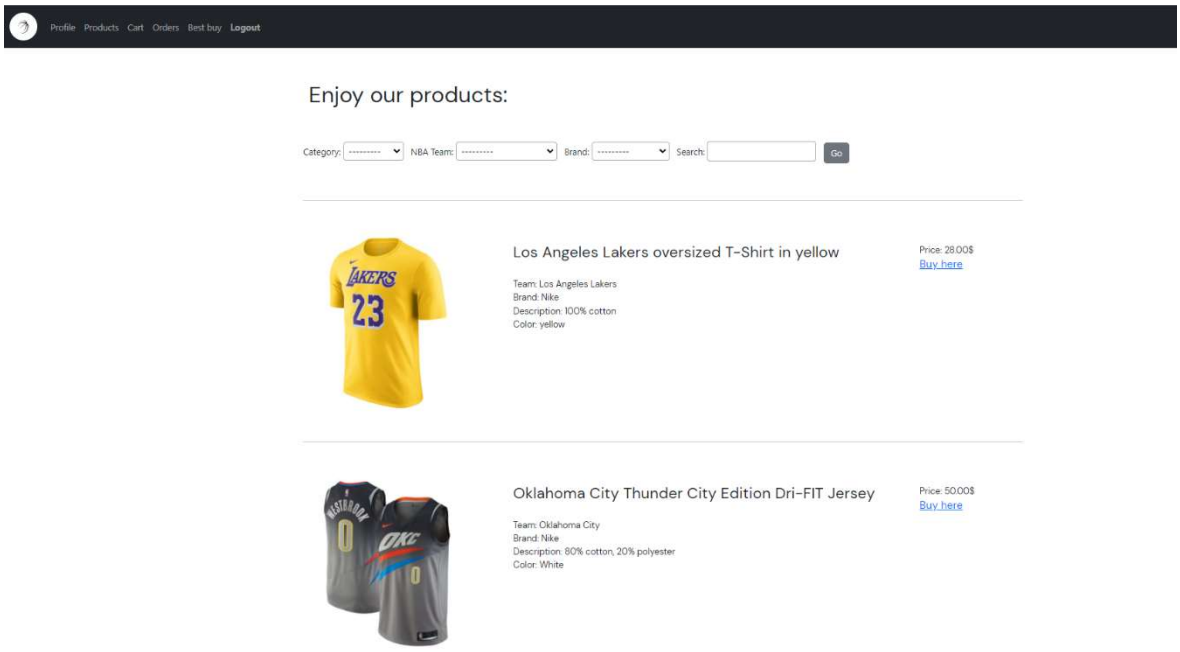
poziva i dodatna funkcija „deleteproduct“ koja prima proizvod koji se želi izbrisati. Jedini tko ima pravo koristiti ovu funkciju je administrator i ona se može napraviti pomoću „POST“ metode nakon što administrator odredi proizvod koji želi izbrisati.

Kada se piše python kod unutar .html datoteke moramo koristiti Djangov format. Primjer tog formata je „{% for product in products_pages %}“. Odnosno prije svakog pisanja logičkog koda mora se dodati uglata zagrada i znak postotka, te se zatvoriti po završetku. Kako bi ova for petlja bila gotova moramo na kraju dodati još „{% endfor %}“. Unutar ove for petlje možemo jednostavno dodati proizvode koje smo poslali iz funkcije „productlist“ koja je prikazana u ispisu 3.

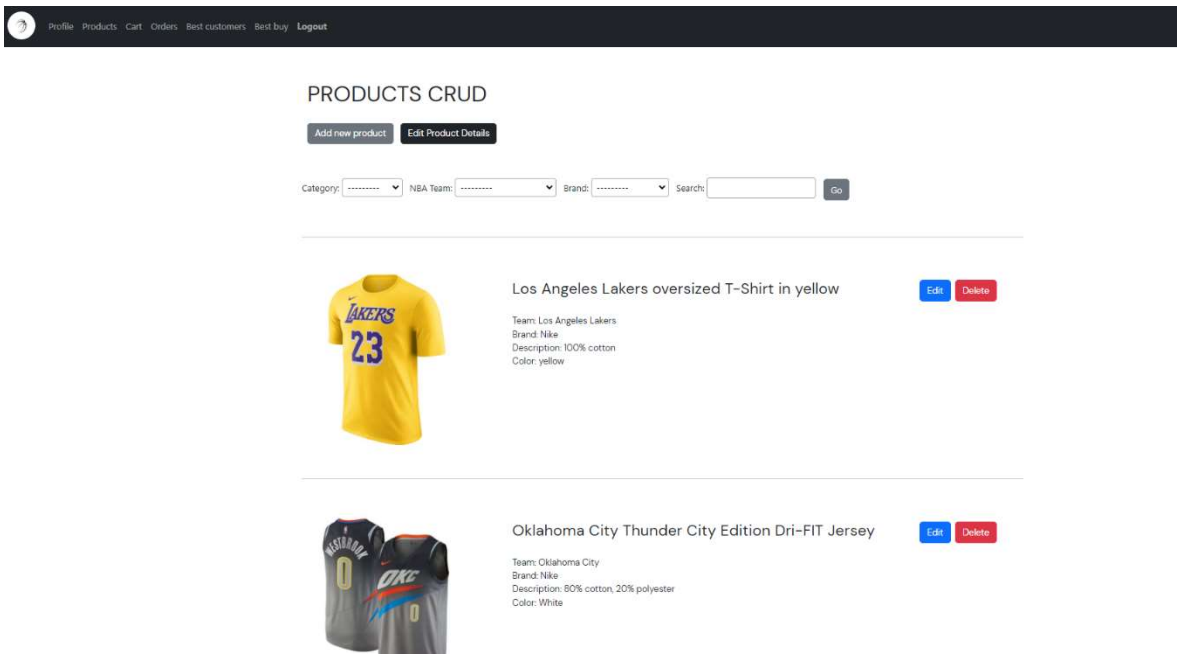
```
{% for product in products_pages %}
      <h3>{{product.name}}</h3>
      <p>
            Team: {{product.team_id}}<br>
            Brand: {{product.brand_id}}<br>
            Description: {{product.material}}<br>
            Color: {{product.color}}<br>
      </p>
{% endfor %}
```

Ispis 3: *Primjer for petlje unutar Djanga*

Prikaz svih proizvoda za običnog korisnika je prikazan na slici 9, dok je na slici 10 prikaz svih proizvoda od strane administratora.

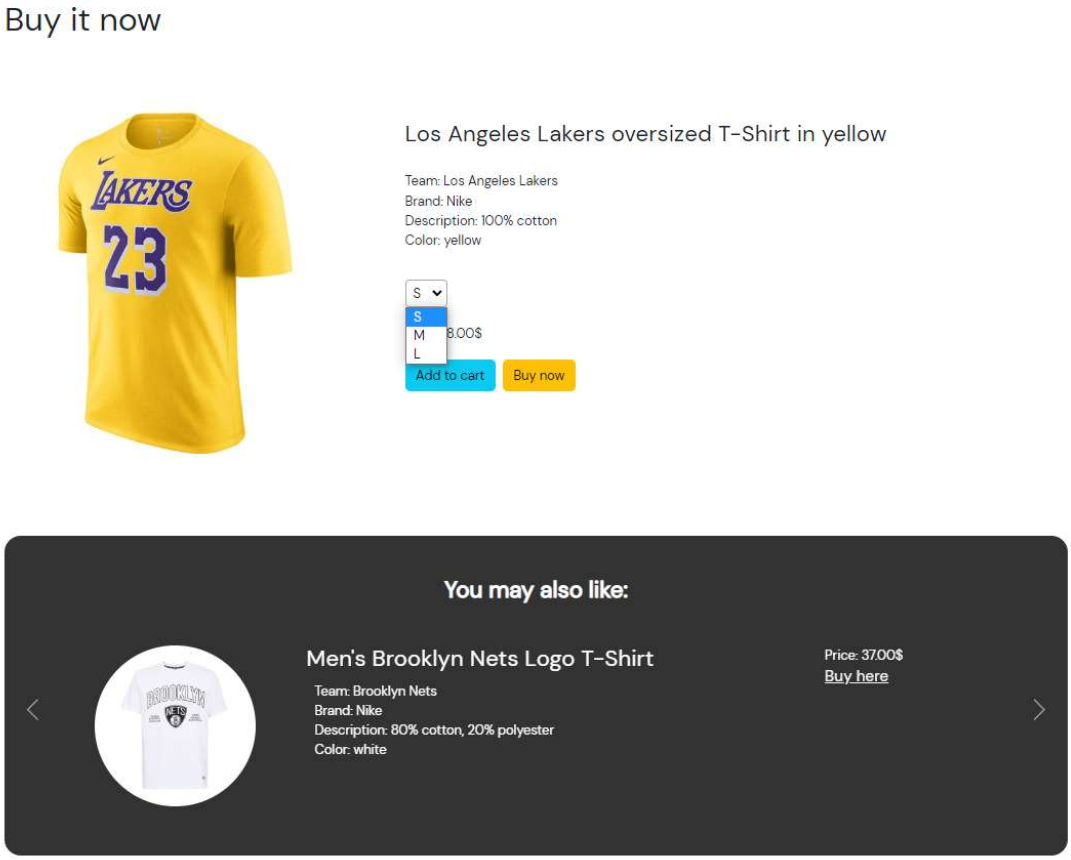


Slika 9: *Pregled proizvoda za običnog korisnika*



Slika 10: *Pregled proizvoda za admina*

Nakon što korisnik klikne na sliku nekog od proizvoda ili na „Buy here“ vezu s desne strane preusmjeri ga se na stranicu u kojoj može izabrati veličinu proizvoda i dodati ga u košaricu. Klikom na vezu „Add to cart“ korisnik može dodati proizvod u košaricu, pritom ostajući i dalje na istoj stranici, dok klikom na vezu „Buy now“ korisnik dodaje proizvod u košaricu i automatski ga se preusmjerava na stranicu „Cart“ koja predstavlja njegovu trenutnu košaricu. Ispod proizvoda se nalazi popis slika koje pripadaju istoj kategoriji koje se naizmjenično pojavljuju jedna za drugom. Detalji proizvoda se nalaze na slici 11.



Slika 11: *Detalji proizvoda*

Ako proizvod više nije u dostupan ne možemo ga dodati u košaricu i umjesto botuna „Add to cart“ i „Buy now“ vidimo upozorenje koje nam govori da ga nema više u ponudi.

3.2.3. Dodavanje proizvoda u košaricu

Kada je proizvod dostupan i kliknemo na jednu od veza pomoću kojih se proizvod dodaje u košaricu poziva se funkcija koja kreira narudžbu za prijavljenog korisnika, ako ona već ne postoji. Ta narudžba bude postavljena u status „cart“. Ukupan broj tog proizvoda se smanji za jedan, dok se u košarici prijavljenog korisnika količina dodanog proizvoda poveća za jedan.

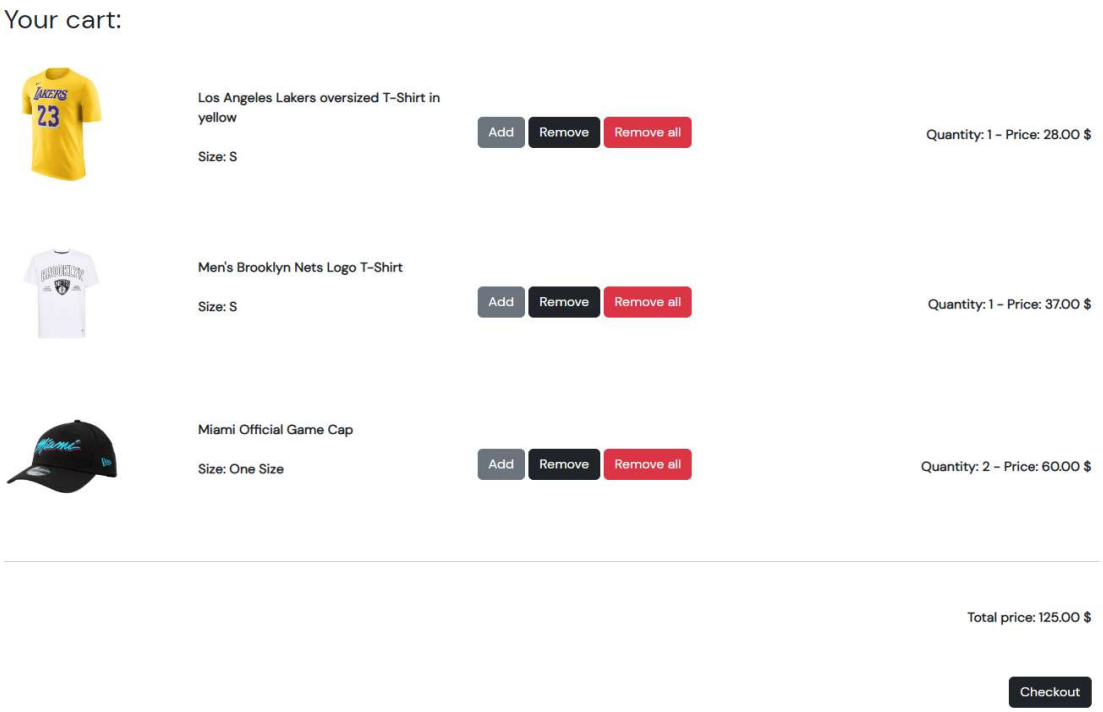
```
@login_required(login_url='login')
def addToCart(request, myProduct):
    customer = request.user
    order, created = Order.objects.get_or_create(customer_id =
customer, status='cart')
    itemDetail, created =
OrderDetails.objects.get_or_create(order_id=order, product=myProduct)
    if myProduct.quantity > 0:
        itemDetail.quantity += 1
        itemDetail.save()
        myProduct.quantity -= 1
        myProduct.save()
        return 'Product addedd successfully!'
    else:
        return 'No more products left in stock!'
```

***Ispis 4**: Funkcija "addToCart"*

U funkciji postoji i dodatna provjera koja pregledava je li proizvod i dalje dostupan. Ona nam u ovom slučaju nije potrebna, jer ako proizvod nije dostupan nećemo ga ni moći dodati u košaricu, ali će nam biti potrebna kada budemo htjeli povećati količinu proizvoda unutar košarice. Nakon svake promjene koja radi promjene na našem objektu i utječe na našu bazu podataka potrebno je nove izmijenjene podatke spremiti, a to se može učiniti pozivom na funkciju „.save()“ kao što je prikazano u ispisu 4. „@login_required(login_url='login')“ je linija koja se nalazi ispred same funkcije. Ona nagovještava da je za ovu radnju potrebno biti prijavljen, te ako korisnik koji nije prijavljen pokuša dodati neki od proizvoda u košaricu preusmjerit će ga na stranicu za prijavu.

3.2.4. Košarica

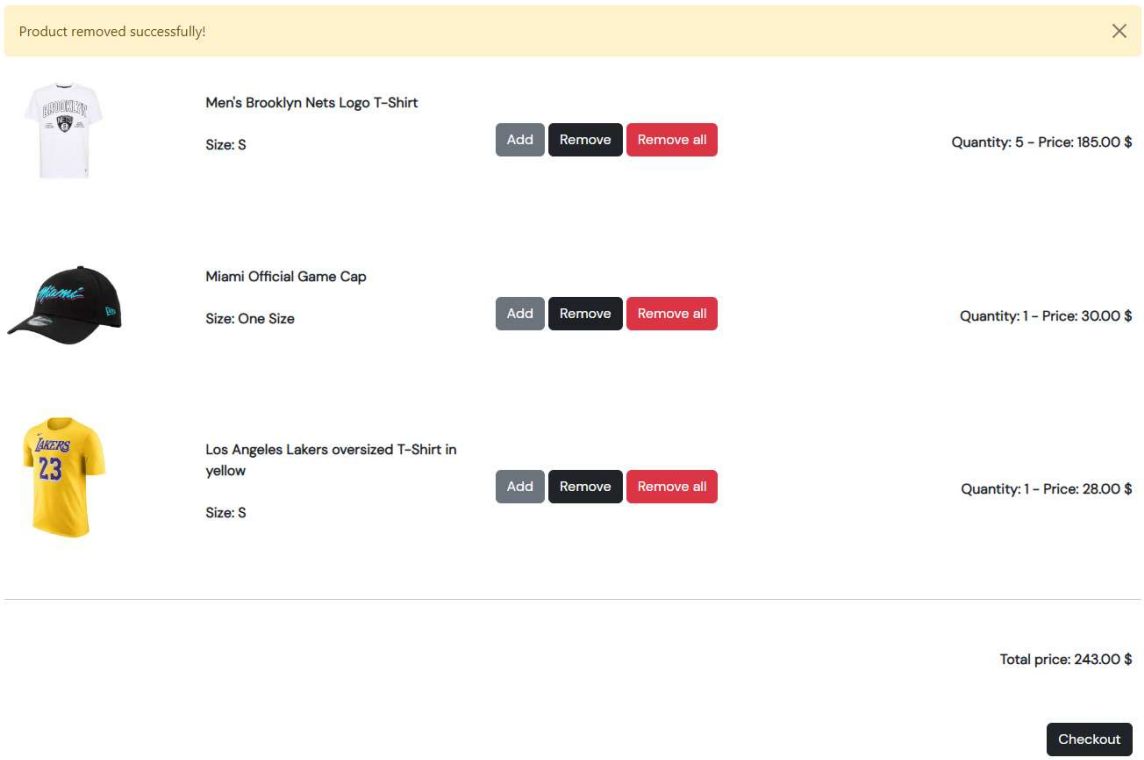
Prije samog plaćanja narudžbe svi proizvodi koje korisnik želi kupiti se prvo dodaju u košaricu. Unutar košarice korisnik ima mogućnost povećavati i smanjivati količinu svih proizvoda te kompletno maknuti proizvod kojeg nije odlučio kupiti. U košarici je prikazana cijena svakog proizvoda na osnovu njegove količine te sveukupna cijena. Unutar košarice su prikazani svi proizvodi iz narudžbe koja se odnosni na prijavljenog korisnika i koja ima statsu „cart“. Košarica je prikana na slici 12.



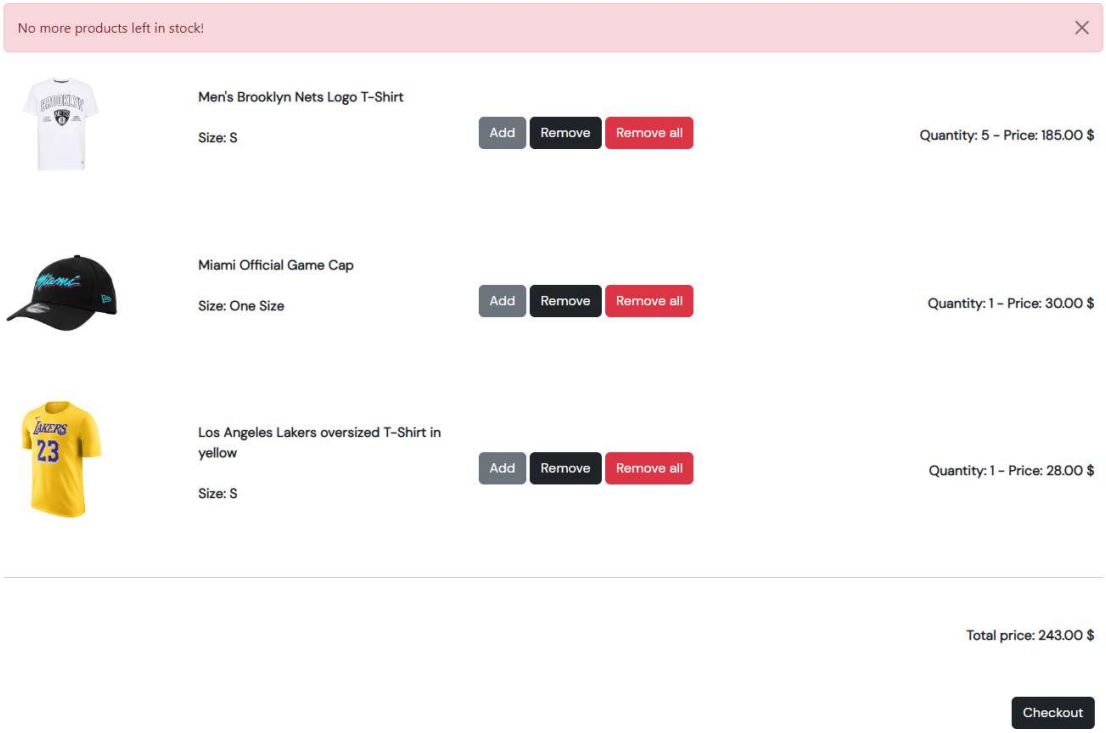
Slika 12: *Košarica*

Kada unutar košarice pokušamo dodati još jedan primjerak već dodanog proizvoda klikom na vezu „Add“ na vrhu ekrana će nam se pojaviti žutom bojom obavijest da smo uspješno dodali taj proizvod u košaricu. Na isti način funkcionira i kada se proizvod uklanja iz košarice klikom na vezu „Remove“ ili „Remove all“. Međutim ako pokušamo dodati još jedan primjerak, a nestalo je svih zaliha na vrhu ekrana će nam se u crvenom okviru pojaviti obavijest koja nam govori da je nemoguće dodati još jedan takav isti proizvod. Zbog ovog razloga je korištena dodatna provjera u ispisu 4 koja provjerava je li moguće dodati još jedan takav proizvod u košaricu ili ne. Funkcija vraća jednostavan string koji će se ispisati na stranici. Okvir u kojem se taj string prikazuje će sam nestati nakon jedne i pol sekunde ili

klikom na „x“ s desne strane. To je napravljeno pomoću jednostavne JavaScript funkcije koja je prikazna u ispisu 5. Uklanjanje proizvoda iz košarice je prikazano na slici 13, a na slici 14 se vidi poruka koja iskoči ako se pokuša dodati proizvod za kojeg više nema preostalih zaliha.



Slika 13: *Uklanjanje proizvoda iz košarice*



Slika 14: *Pokušaj dodavanja proizvoda koji više nije dostupan*

```
$("#success-alert").fadeTo(1500, 250).slideUp(250, function(){
      $("#success-alert").slideUp(250);
  });
```

Ispis 5: *Funkcija koja briše element s id-om "success-alert" nakon 1,5 sekunde*

Kada želimo izbrisati sve primjerke određenog proizvoda moramo kliknuti na vezu „Remove all“. Kada se klikne ova veza poziva se funkcija „removeall“ koja prvo provjeri koliko je tih proizvoda u košarici te na osnovu toga smanjuje količinu u samoj košarici dok se s druge strane količina tog proizvoda u zalihama povećava. Funkcija također vraća jednostavni string koji se ispisuje na samoj stranici kao i funkcije za dodavanje i brisanje jednog proizvoda. Način na koji je napisana možemo vidjeti na ispisu 6.

```
def removeAll(request, myProduct):
    customer = request.user
    order = Order.objects.get(customer_id = customer, status='cart')
    itemDetail = OrderDetails.objects.get(order_id=order,
product=myProduct)
    while itemDetail.quantity > 0:
        itemDetail.quantity -= 1
        myProduct.quantity += 1
    itemDetail.delete()
    myProduct.save()
    return 'Products removed successfully!'
```

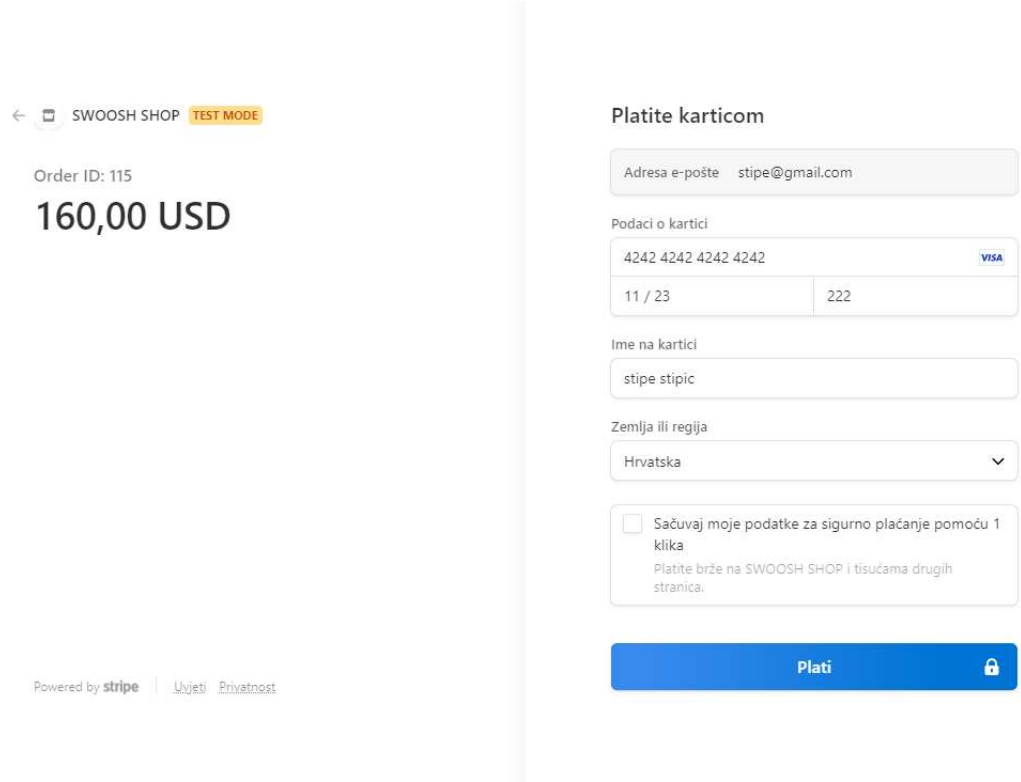
Ispis 6: *Funkcija "removeAll"*

3.2.5. Plaćanje narudžbe

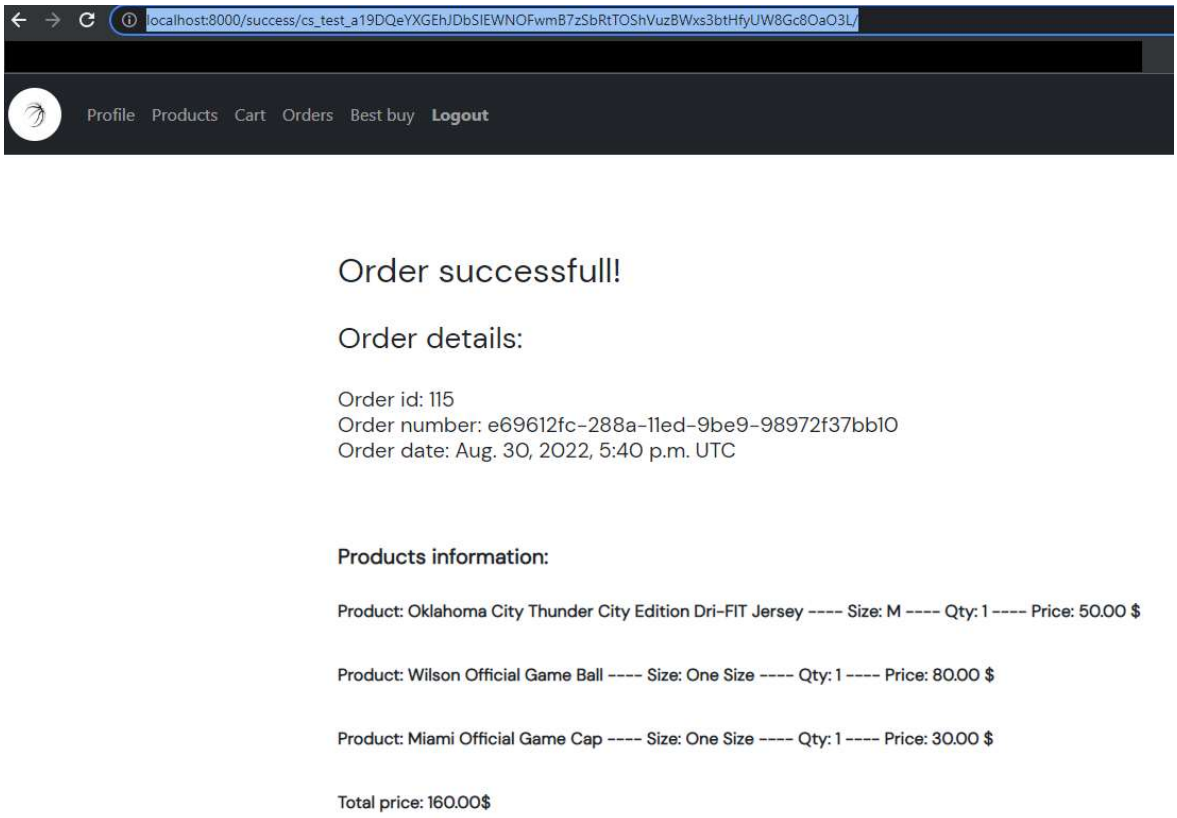
Samo plaćanje je napravljeno pomoću dodatka „Stripe“ koji je instaliran u ovu Django aplikaciju. Stripe omogućuje brzo i jednostavno dodavanje kartice te izvršavanje same narudžbe. On vodi brigu o tome da je korisnik platio iznos koji je dodao u košaricu i ako je plaćanje uspješno izvršeno preusmjerava ga na stranicu koja prikazuje potvrdu narudžbe i sve proizvode koji su se naručili. Kada se narudžba uspješno izvrši kreira se njen id unutar Stripe-a.

Taj id se zove „CHECKOUT_SESSION_ID“ i on predstavlja narudžbu koju je korisnik upravo izvršio. Na ovaj način unutar Stripe-a možemo pronaći tu narudžbu. Provjeru radimo tako da se „CHECKOUT_SESSION_ID“ pošalje u URL unutar kojega se narudžba prihvati i prikaže uspješna kupnja. Ako se uspije pronaći narudžba s tim brojem koji je proslijeđen unutar URL-a narudžba je uspješno plaćena i smatra se izvršenom te se korisniku prikazuje stranica uspješne kupnje. Ovo je sigurnosni korak koji je morao biti dodan zato što bi korisnici mogli pristupiti ovoj stranici s bilo kojim slučajnim brojem i uspješno obaviti kupnju bez uplaćenih sredstava.

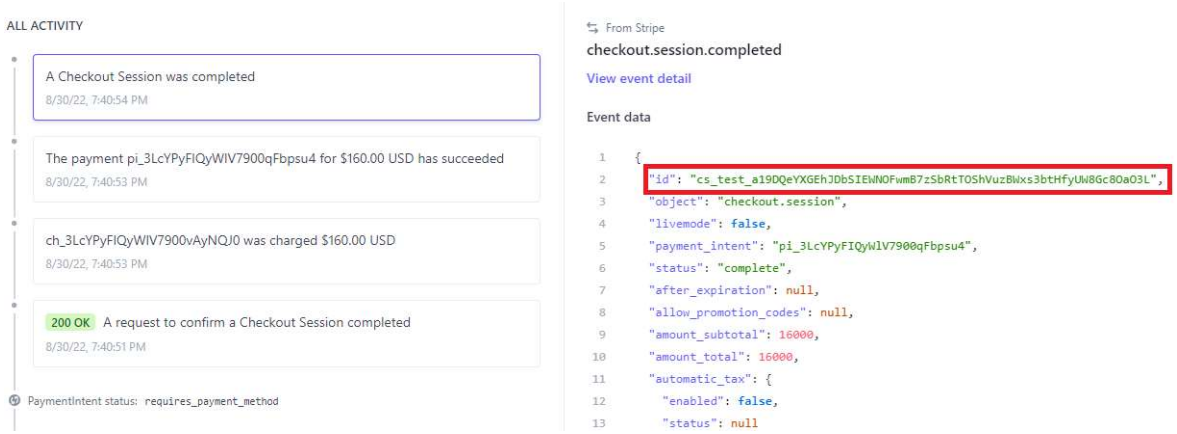
Stranica na kojoj se izvršava samo plaćanje je prikazana na slici 15, dok se stranica na koju korisnik bude preusmjeren nakon uspješne kupnje nalazi na slici 16. Na slici 17 možemo vidjeti kako se narudžba kreirala na Stripe sučelju nakon što se plaćanje izvrši.



Slika 15: *Stripe plaćanje*



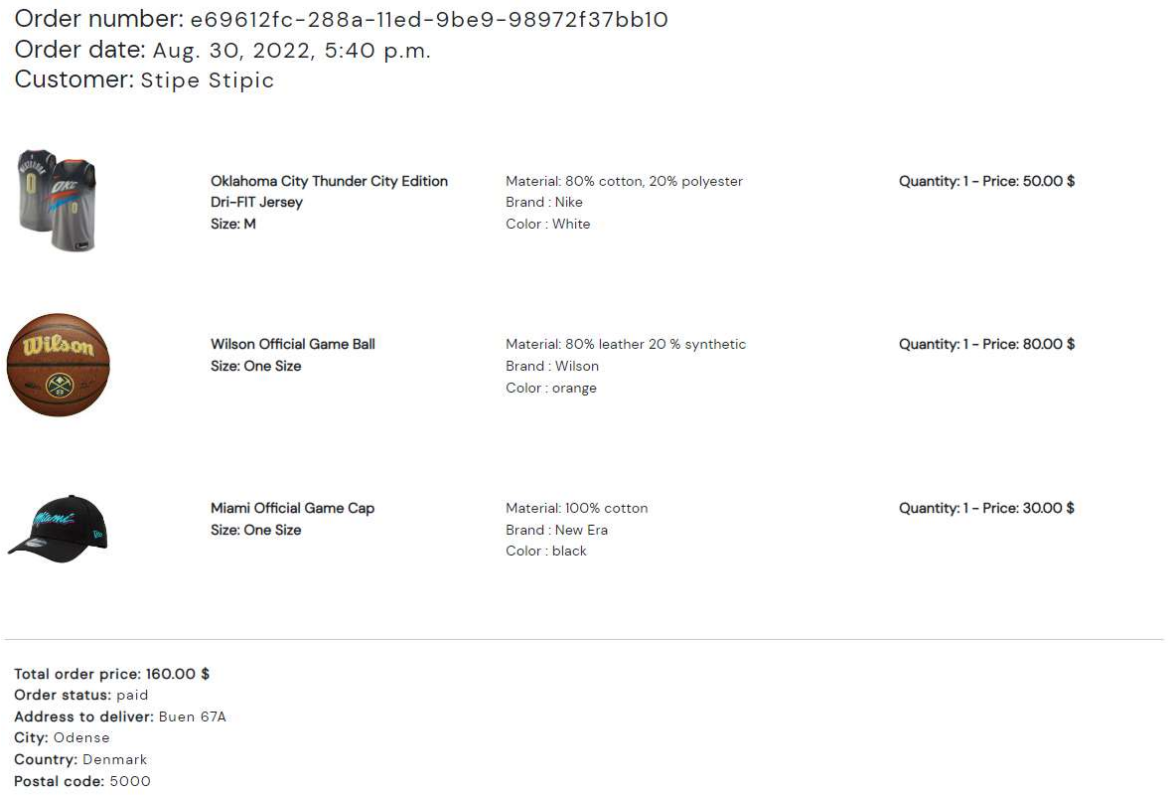
Slika 16: *Stranica uspješne kupnje i prikaz „CHECKOUT_SESSION_ID-a“ unutar URL-a*



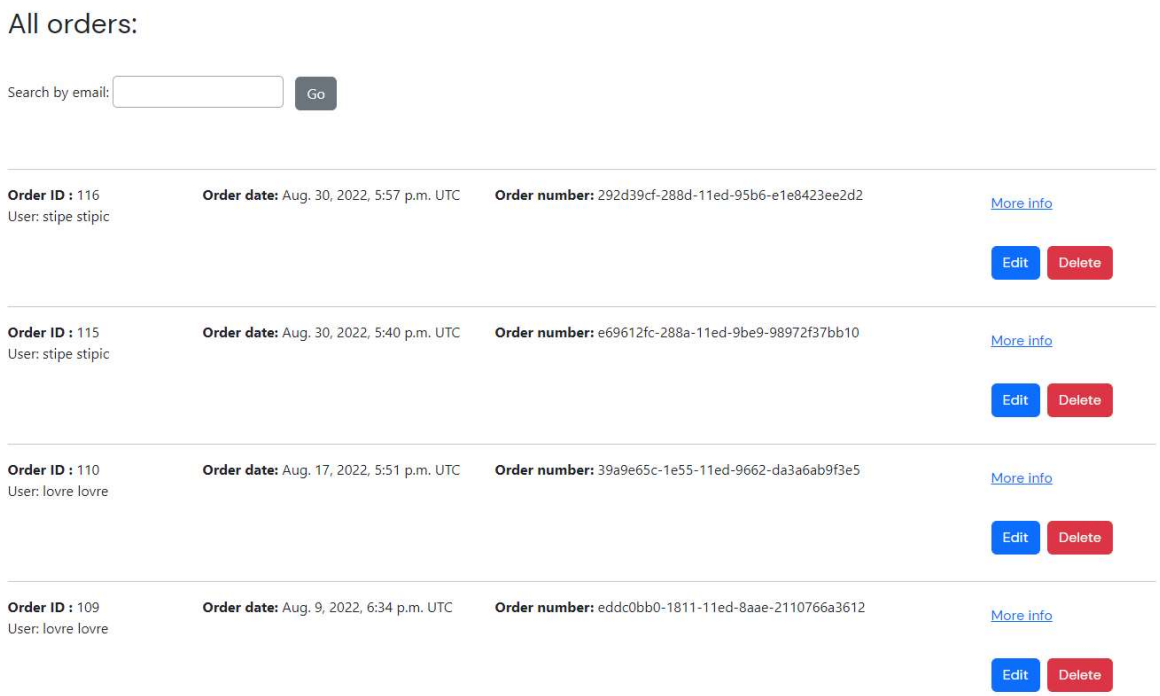
Slika 17: *Stripe objekt koji se stvorio nakon uspješnog plaćanja sa svojim „CHECKOUT_SESSION_ID-em“*

3.2.6. Narudžbe

Stranica svih narudžbi nalazi se na poveznici „Orders“ kojoj možemo pristupiti pomoću navigacijske trake. Na toj stranici se nalaze sve narudžbe prijavljenog korisnika koje su u statusu „paid“, odnosno plaćene. Također moguće je pristupiti detaljima pojedine narudžbe i vidjeti sve informacije vezanu za nju te sve proizvode koji su kupljeni u toj narudžbi. To je prikazano na slici 18. Administrator uz pregled svojih narudžbi ima pravo pregleda i narudžbi svih korisnika, te ima mogućnost uređivanja i brisanja pojedine narudžbe. Uz sve to ima i opciju pretraživanja narudžbi po korisnikovoj email adresi. Slika 18 prikazuje detalje narudžbe koja je izvršena, dok se na slici 19 nalazi pregled svih narudžbi od strane administratora.



Slika 18: *Pregled narudžbe*



***Slika 19**: Pregled svih narudžbi od strane administratora*

Administratora se klikom na poveznicu „Edit“ preusmjeri na novu stranicu u čijem URL-u se nalazi ID odabrane narudžbe. To preusmjeravanje je napravljeno pomoću HTML atributa „href“ koji u sebi sadrži ime URL-a i ID narudžbe koju želimo izmijeniti. Na isti ovaj način je implementirano i uređivanje pojedinog proizvoda. Kod se nalazi na ispisu 7.

```
<a href="{%url 'editorder' order.id %}" class="btn btn-primary"
role="button" style="color: #fff;">Edit</a> 
```

***Ispis 7:** Poveznica koja vodi na stranicu za izmjenjivanje odabrane narudžbe*

Kada se ta stranica dohvati na njoj bude prikazana odabrana narudžba sa svim informacijama koje su vezane za nju. Da bi se unaprijed ispisali podatci u formi od narudžbe koja se pokuša otvoriti moramo u formu dodati instancu (engl. *instance*) narudžbe koju šaljemo u URL. Funkcija koja ovo radi je prikazana u ispisu 8. Funkcija na web stranicu šalje formu, poruku koja će se prikazati nakon uspješnog uređivanja i samu narudžbu. Linija koda „form = OrderForm(instance=order)“ ispunjava formu s objektom koji smo poslali u tu instancu.

```
@login_required(login_url='login')
def editorder(request, id):
    info = ''
    order = Order.objects.get(id = id)
    form = OrderForm(instance=order)
    if request.method=='POST':
        form = OrderForm(request.POST, instance=order)
        if form.is_valid():
            form.save()
            info = 'Product successfully edited!'

    data = {
        'info': info,
        'form': form,
        'order': order
    }
    return render(request, "admin/editorder.html", data)
```

Ispis 9: *Funkcija za uređivanje narudžbe*

Brisanje narudžbe se radi pomoću „POST“ zahtjeva. Kada administrator klikne na poveznicu „Delete“ na stranici se prikaže skočni prozor s kojim se potvrđuje brisanje te se nakon toga poziva funkcija koja briše narudžbu poslanu pomoću „POST“ zahtjeva. Funkcija je vrlo jednostavna, prima narudžbu koju je potrebno izbrisati, izbriše je te vrati povratnu informaciju. Prikazana je u ispisu 9.

```
def deleteproduct(product):
product = Product.objects.get(id=product)
product.delete()
return 'Product successfully deleted!'
```

Ispis 8: *Funkcija za brisanje naružbe*

3.2.7. Statistički podaci

Od statističkih podataka evidentirani su najbolji kupci i najprodavaniji artikli na dnevnoj, tjednoj i mjesečnoj bazi. Administrator može pristupiti svim statističkim podacima, dok korisnik ima pristup samo statističkim podacima vezanim za najprodavanije artikle. Za prikaz ovih podataka bilo je potrebno saznati trenutni dan u kojem se nalazimo, dan s kojim počinje tjedan, zadnji dan u tjednu, te prvi i zadnji dan u mjesecu. Način na koji je to napravljeno se nalazi na ispisu 10.

```
current_day = datetime.date(timezone.now().year, timezone.now().month,
timezone.now().day)


_, lastday = calendar.monthrange(timezone.now().year,
timezone.now().month)


first_day_month = datetime.date(timezone.now().year,
timezone.now().month, 1)

last_day_month = datetime.date(timezone.now().year,
timezone.now().month, lastday)


date = datetime.date.today()

start_week = date - datetime.timedelta(date.weekday())

end_week = start_week + datetime.timedelta(6)
```

Ispis 10: *Dohvaćanje informacija o danu, tjednu i mjesecu*

Datum koji predstavlja dan u kojem se nalazimo je bilo jednostavno pronaći. On se dohvatio pomoću „timezone“ implementacije tako da se u „datettime.date“ funkciju unesu parametri koji definiraju današnju godinu, mjesec i dan.

Kod dohvaćanja prvog i zadnjeg dana u mjesecu koristila se funkcija „monthrange“ koja vraća zadnji dan u trenutnom mjesecu. Kada se dozna zadnji dan u mjesecu jednostavno je dohvatiti točan datum kada mjesec završava i datum kada mjesec počinje na isti način na koji se saznao datum trenutnog dana u kojem se nalazimo.

Za dohvaćanje prvog i zadnjeg dana u tjednu se koristila funkcija „datetime.date.today().weekday()“. Ova funkcija vraća broj trenutnog dana u tjednu (broj 0 označava ponedjeljak, broj 6 označava nedjelju). Da bi dobili datum prvog dana u tjednu

oduzimamo trenutni datum s tim brojem. Kada se dohvati prvi datum u tjednu lako je dohvatiti zadnji datum u tjednu tako što ćemo ovoga povećati za šest.

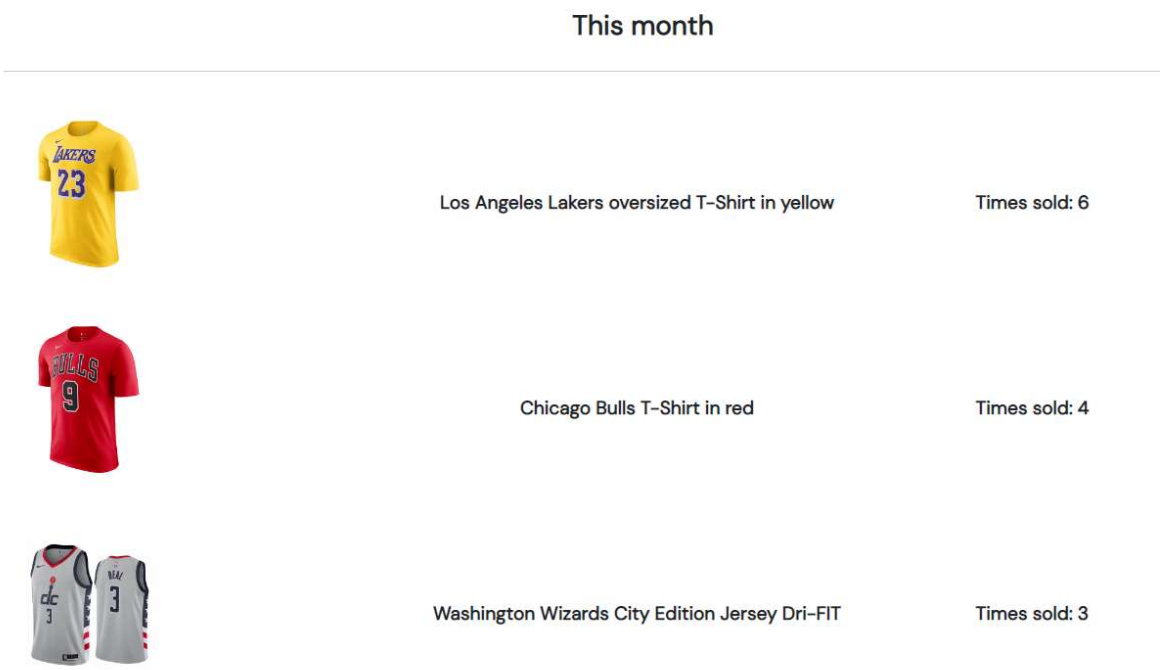
Kada se dobiju svi datumi pozivaju se funkcije za dohvaćanje podataka o najboljim kupcima i najprodavanijim artiklima.

```
def best_buy_range(start, end):


    products = Product.objects.all()
    bestBuyList = {}


    for product in products:
        num_of_prod = 0
        orderedItems = OrderDetails.objects.filter(status='paid',
product__product_id = product)
        for order in orderedItems:
            orderdatetime = order.date
            orderdate = orderdatetime.date()
            if orderdate >= start and orderdate <= end:
                num_of_prod += order.quantity


        if num_of_prod > 0:
            bestBuyList[product] = num_of_prod


    bestBuyList = dict(sorted(bestBuyList.items(), key=lambda item:
item[1], reverse = True)[:3])


    return bestBuyList
```

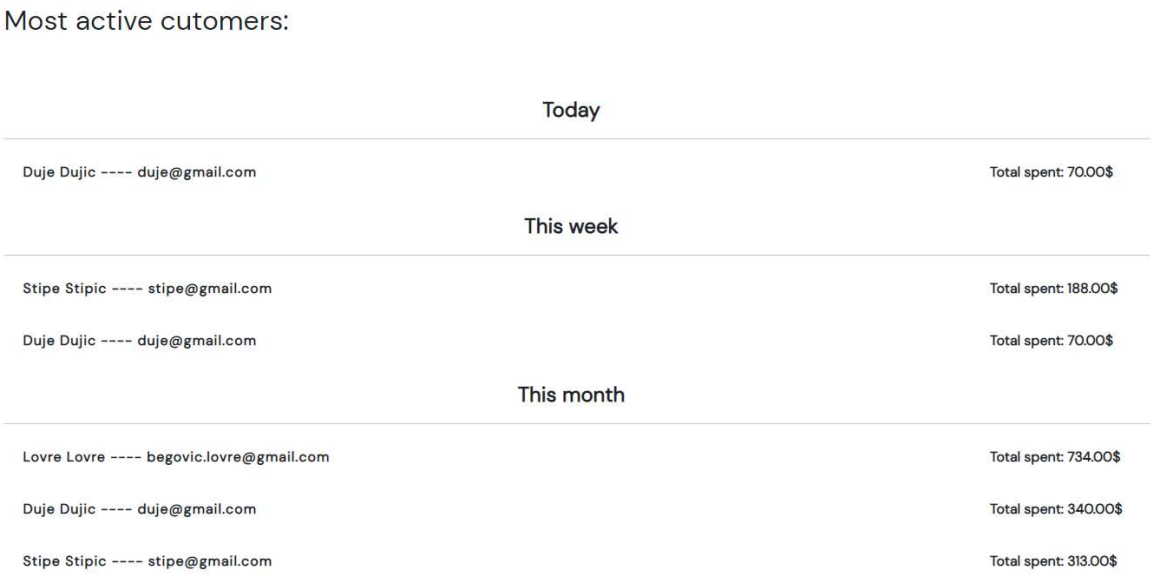
Ispis 11: *Funkcija za dohvaćanje najprodavanijih artikala*

U ispisu 11 se nalazi funkcija koja vraća tri najprodavanija artikla. Funkcija radi na principu da prolazi kroz sve plaćene artikle i za svaki artikal provjeri nalazi li se u intervalu koji trenutno testiramo (najbolje prodani artikli u trenutnom danu, tjednu, mjesecu). U slučaju da se artikal nalazi u tom intervalu dodaje ga se u rječnik s brojem prodanih primjeraka. Na kraju se rječnik sortira silaznim redoslijedom i iz njega se izvuku prva tri elementa.

Na isti način radi i funkcija za dohvaćanje najboljih kupaca. U toj funkciji umjesto artikala i njihovih broja prodanih primjeraka funkcija vraća kupce s iznosim koji su potrošili. Na slici 20 i slici 21 se nalazi izgled stranica koje predstavljaju najprodavanije artikle i najbolje kupce za trenutni dan, mjesec i godinu.



Slika 20: *Najprodavaniji artikli*



Slika 21*: Najbolji kupci*

4. Zaključak

U ovom završnom radu je napravljena aplikacija koja omogućuje brzo i jednostavno naručivanje košarkaške opreme. Odvojeno je sučelje administratora i običnog korisnika, omogućen je pregled vlastitog profila, pretraživanje artikala, filtriranje istih te samo naručivanje i pregled povijesti svih narudžbi. Uz to su dodani i statistički podaci koji mogu korisniku privući pažnju na najprodavanije proizvode ili administratoru na kupce koji su potrošili najviše sredstava. Napravljena je aplikacija klasičnog, jednostavnog i responzivnog dizajna pomoću Djanga. Django se pokazao kao jako dobra tehnologija za izradu web aplikacija koja ima mogućnost velikog broja instalacija i lako ga je naučiti. U ovoj aplikaciji je bilo jako bitno napraviti dobru bazu podataka. Ona je bila ključni dio izrade samog rada. Kada se napravi baza preostalo ju je samo implementirati unutar aplikacije.

Ova aplikacija bi se mogla i poboljšati s raznim funkcionalnostima. Neke od tih funkcionalnosti mogu biti mogućnost ostavljanja recenzije na proizvodu, slanje emaila korisniku nakon uspješne registracije i narudžbe, ili možda korištenje tokena za automatsko odjavljivanje s računa i brisanje trenutne košarice.

Naravno, uvijek postoji mjesta za napredak, no u konačnici mislim da je ova aplikacija zadovoljila svoju svrhu.

Literatura

[1] Django dokumentacija (posjećeno 10.8.2022.)

https://docs.djangoproject.com/en/4.0/

[2] Uvod u Django (posjećeno 10.8.2022.)

https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction

[3] Bootstrap dokumentacija (posjećeno 15.8.2022.)

https://getbootstrap.com/docs/5.2/getting-started/introduction/

[4] Stripe dokumentacija (posjećeno 20.8.2022)

https://stripe.com/docs/payments

[5] Stripe plaćanje (posjećeno 22.8.2022.)

https://www.youtube.com/watch?v=722A27IoQnk