# Web aplikacija za prodaju odjeće

**Danković, Marko**



**SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU**

**FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA**

**Stručni studij računarstva**

# WEB APLIKACIJA ZA PRODAJU ODJEĆE

**Završni rad**

**Marko Danković**

**Osijek, 2022.**

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

**Obrazac Z1S: Obrazac za imenovanje Povjerenstva za završni ispit na preddiplomskom stručnom studiju**

**Osijek, 17.09.2022.**

**Odboru za završne i diplomske ispite**

# Imenovanje Povjerenstva za završni ispit na preddiplomskom stručnom studiju

| | |
|---|---|
| **Ime i prezime Pristupnika:** | Marko Danković |
| **Studij, smjer:** | Preddiplomski stručni studij Računarstvo |
| **Mat. br. Pristupnika, godina upisa:** | AR 4717, 19.07.2019. |
| **OIB Pristupnika:** | 17166375780 |
| **Mentor:** | Robert Šojo, mag. ing. comp. |
| **Sumentor:** | , |
| **Sumentor iz tvrtke:** | |
| **Predsjednik Povjerenstva:** | mr.sc. Željko Štanfel |
| **Član Povjerenstva 1:** | Robert Šojo, mag. ing. comp. |
| **Član Povjerenstva 2:** | Marina Peko, dipl. ing. |
| **Naslov završnog rada:** | Web aplikacija za prodaju odjeće |
| **Znanstvena grana završnog rada:** | **Programsko inženjerstvo (zn. polje računarstvo)** |
| **Zadatak završnog rada** | Web aplikacija koja omogućava prodaju odjevnih predmeta. Postoje tri uloge korisnika koji koriste web aplikaciju. Prva uloga je administrator koji ima mogućnost kreiranje i brisanje artikala, uvid u komentare, analitiku korisnika. Druga uloga je registrirani korisnik kojemu se registracijom omogućavaju različite pogodnosti prilikom kupnje artikala. Treća uloga je gost korisnik koji može samo gledati proizvod i ne može komentirati. Student treba razviti funkcionalnu aplikaciju koristeći različite web tehnologije. Rezervirano za studenta: Marko Danković |
| **Prijedlog ocjene pismenog dijela ispita (završnog rada):** | Izvrstan (5) |
| **Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:** | Primjena znanja stečenih na fakultetu: 3 bod/boda<br>Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda<br>Jasnoća pismenog izražavanja: 3 bod/boda<br>Razina samostalnosti: 3 razina |
| **Datum prijedloga ocjene od strane mentora:** | 17.09.2022. |

| | |
|---|---|
| Potvrda mentora o predaji konačne verzije rada: | Mentor elektronički potpisao predaju konačne verzije. |
| | Datum: |

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

# IZJAVA O ORIGINALNOSTI RADA

**Osijek, 22.09.2022.**

| | |
|---|---|
| **Ime i prezime studenta:** | Marko Danković |
| **Studij**: | Preddiplomski stručni studij Računarstvo |
| **Mat. br. studenta, godina upisa:** | AR 4717, 19.07.2019. |
| **Turnitin podudaranje [%]:** | 14 |

Ovom izjavom izjavljujem da je rad pod nazivom**: Web aplikacija za prodaju odjeće**

izrađen pod vodstvom mentora Robert Šojo, mag. ing. comp.

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 studenta:

# SADRŽAJ

# 1. UVOD

Internet je u zadnjem desetljeću postao jako raširen i popularan. Razvojem tehnologije cijeli svijet je povezan putem Interneta koji omogućava međusobni kontakt među osobama u jako kratkom roku. Razvojem Interneta i privikavanjem ljudi na korištenje istog, gotovo sve obveze i zabavne aktivnosti ljudi se mogu obaviti na Internetu. Online kupnja ili kupovina preko web stranica je postala neizmjerno popularna u zadnje vrijeme, posebice za vrijeme pandemije Korona virusa. Web stranice za prodaju odjeće bilježe velik porast u prodaji putem Interneta u zadnjih par godina. Neke od pozitivnih stvari prelaska na online kupovinu odjeće su mogućnost kupovine odjeće iz udobnosti vlastitog doma, bez izlaska iz kuće, čime štedimo na vremenu, veći izbor artikala jer su prodavaonice ograničene veličinom i ne mogu skladištiti sve moguće artikle koje korisnici mogu naći na web stranicama. Međutim, najveća mana ovakvog načina kupovine je nemogućnost probavanja artikala prije same kupovine zbog provjere veličine ali i nepodudarnost artikala na slici i uživo.

## 1.1. Cilj završnog rada

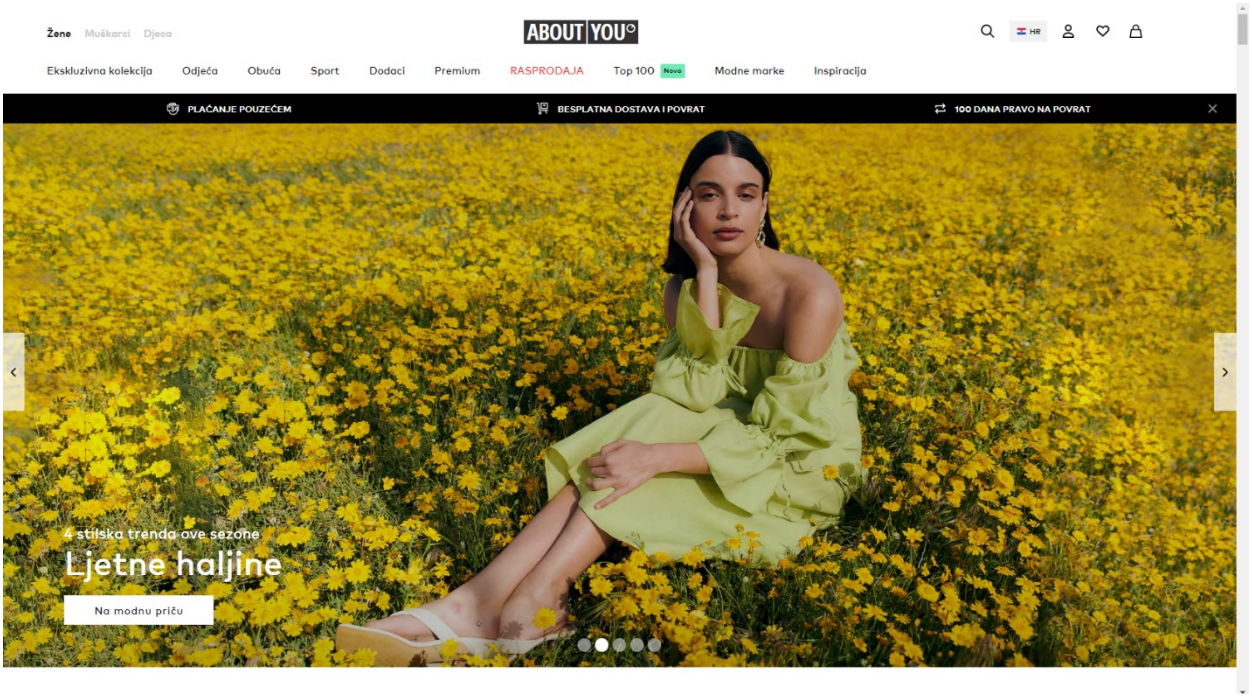
Cilj završnog rada je razviti web aplikaciju koja omogućava prodaju odjevnih predmeta. Postoje tri uloge korisnika koji koriste web aplikaciju. Prva uloga je administrator koji ima mogućnost kreiranja i brisanja artikala, uvid u komentare, analitiku korisnika i artikala. Druga uloga je registrirani korisnik kojemu se samom registracijom omogućavaju različite pogodnosti prilikom kupnje artikala. Treća uloga je gost korisnik koji može vidjeti artikle ali ih ne može kupovati i ne može vidjeti komentare.

# 2. POSTOJEĆE WEB STRANICE

U ovom poglavlju se nalaze slične već postojeće web aplikacije koje se koriste za online prodaju odjeće.

## 2.1. ABOUT YOU

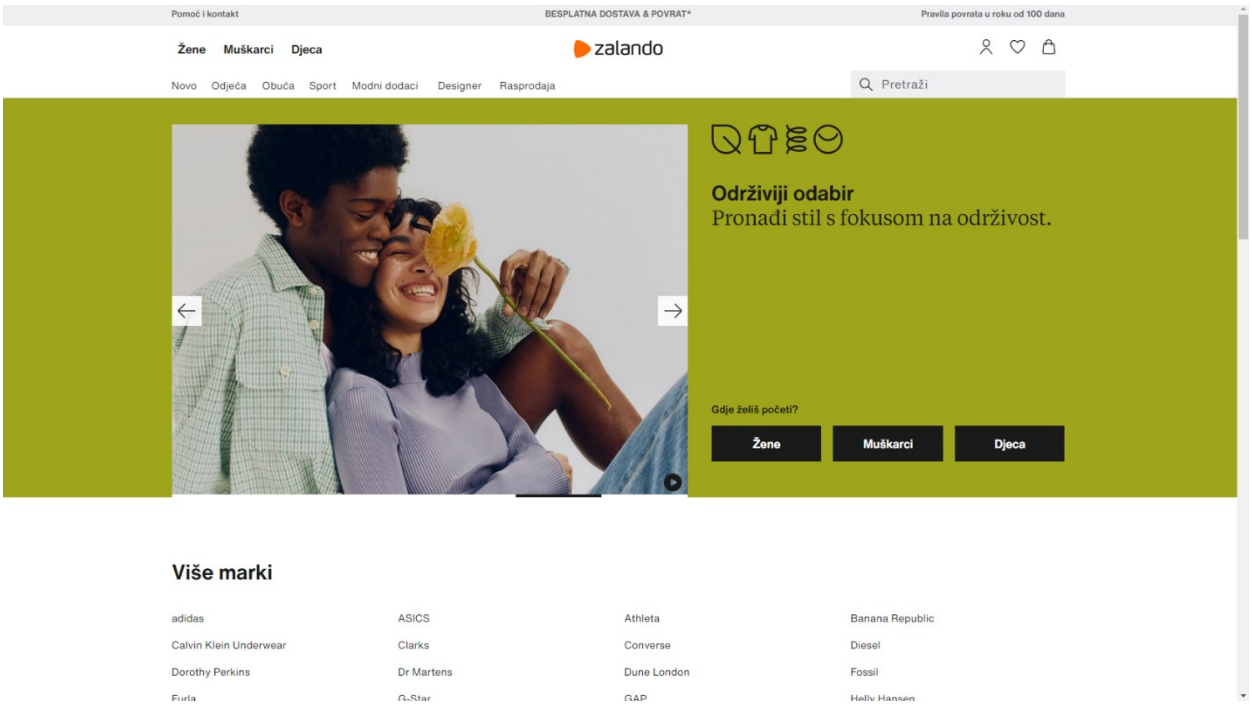
ABOUT YOU je jedna od najbrže rastućih tvrtki za e-trgovinu u Europi i postala je prva Hamburška tvrtka jednorog u 2018. godini. Kao modna i tehnološka korporacija, cilj im je digitalizirati klasičnu kupovinu šetnjom kroz trgovinu stvaranjem inspirativne i personalizirane kupovine na pametnom telefonu. Sadrži više od 2,000 različitih marki te više od 400,000 različitih artikala. Omogućuju kupovinu različitih artikala za svakoga, postoji mogućnost registracije nakon koje korisnik dobije određene benefite. Česti su kodovi za popuste, a dostava je besplatna. Slika 2.1. prikazuje izgled ABOUT YOU stranice [1].




**Slika 2.1.** *Izgled ABOUT YOU stranice [1].*

## 2.2. Zalando

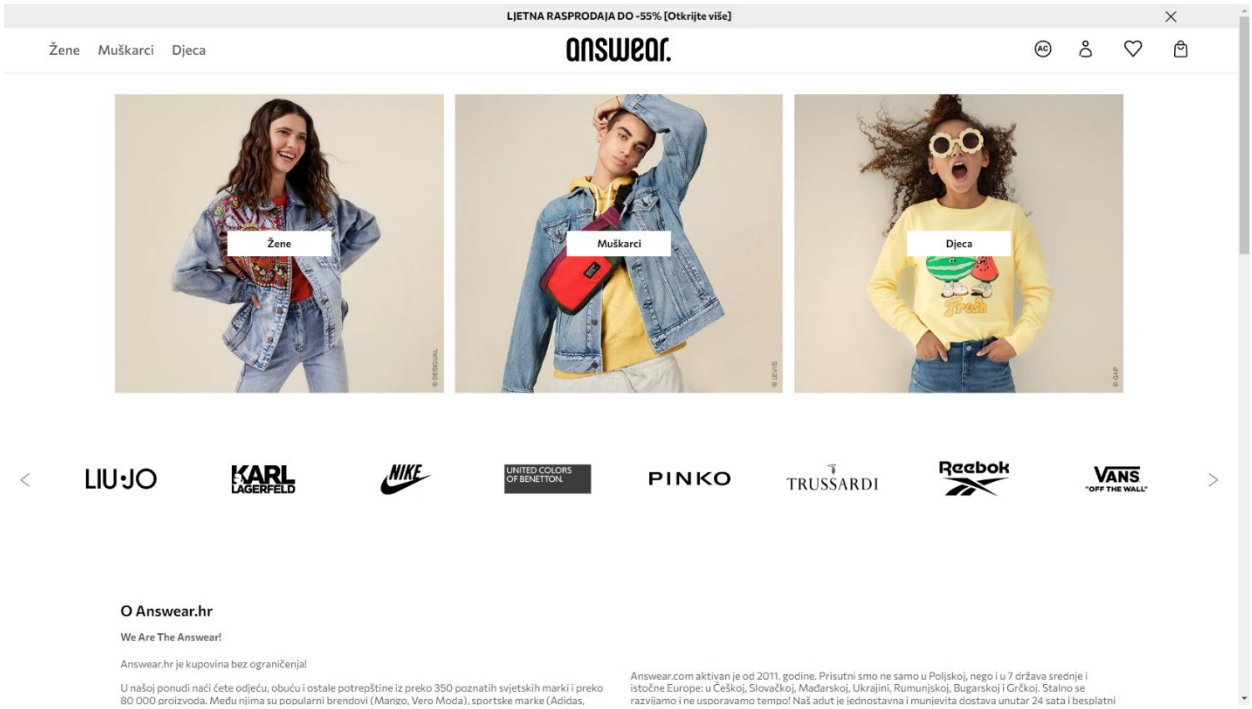
Zalando povezuje kupce, marke i partnere iz 23 zemlje. Ono što je 2008. godine započelo kao internetska trgovina cipela sa sjedištem u Berlinu pretvorilo se u vodeću europsku internetsku platformu za modu i stil života u samo nekoliko godina. Zalando broji više od 5,800 različitih marki te oko 1,4 milijuna artikala, oko 7 milijardi posjeta godišnje te dobit u 2021. od čak 10,4 milijarde eura. Postoji mogućnost registracije nakon koje se ostvaruju određene prednosti nad drugim neregistriranim korisnicima. Artikli se dostave na kuću adresu, a kodovi za popust nisu rijetkost. Slika 2.2. prikazuje Zalando web stranicu [2].




**Slika 2.2.** *Izgled Zalando web stranice [2].*

## 2.3. Answear

Answear web stranica u svojoj ponudi nudi odjeću, obuću i ostale potrepštine od više od 400 poznatih svjetskih modnih marki i više od 94,000 proizvoda. Nude široku ponudu za muškarce, žene i djecu. Answear je aktivan od 2011. godine, a osim Poljske, prisutni su i u 10 država srednje i istočne Europe. Glavni adut im je dostava unutar 24 sata i besplatni povrat proizvoda. Postoje različiti načini plaćanja i oblici dostave, a registracijom na stranici korisnici ostvaruju dodatne pogodnosti. Slika 2.3. prikazuje izgled Answear web stranice [3].




**Slika 2.3.** *Izgled Answear web stranice [3].*

# 3. KORIŠTENE TEHNOLOGIJE KOD IZRADE WEB APLIKACIJE

Postoji puno različitih programskih jezika i tehnologija za razvoj web aplikacija. U ovom poglavlju su objašnjene one tehnologije koje su korištene za izradu ove web aplikacije. Podijeljene su u dvije grupe, *frontend* i *backend*. Prikaz različitih tehnologija za izradu web stranice se nalazi na slici 3.1 [4].


**Slika. 3.1.** *Tehnologije za izradu web stranice [4].*

*Frontend* tehnologije se odnose na prikaz izgleda aplikacije kao što su tekst, boja, slike, raspodjela elemenata na stranici te gumbi i ostali elementi. *Frontend* programeri programiraju web stranice koristeći jezike kao što su HTML, CSS i JavaScript na temelju dizajna koji dobiju od web dizajnera koji su zaduženi za osmišljavanje izgleda stranice. *Frontend* bez *backenda* je primjer statičke web stranice, njezin sadržaj se ne mijenja puno, nema baze u koju bi se spremali podaci. Statičke stranice su dobre za prikazivanje stvari koje su općenite i kod kojih nema puno izmjena, npr. prikaz osobnog profila.

*Backend* tehnologije se koriste za sve ono što korisnik ne može vidjeti u pregledniku, poput baze podataka i servera. Omogućuje komuniciranje s *frontendom*, a time i omogućava korisniku slanje

određenih zahtjeva *frontenda* koji se onda obrađuju na *backendu*. Nakon obrade, ovisno o zahtjevu, na *frontend* se šalje rezultat obrade kako bi se ispunio korisnički zahtjev.

## 3.1. HTML

HTML (eng. *HyperText Markup Language*) je jezik koji se koristi za izradu web stranica ali nije programski jezik jer ne može stvoriti dinamičku funkcionalnost. U HTML se pomoću oznaka i atributa govori pregledniku kako treba rasporediti sadržaj na web stranici i time se definira njezin izgled.

HTML je početkom 90-ih godina prošlog stoljeća razvio Tim Berners-Lee, fizičar zaposlen u CERN-u. HTML je vrlo brzo postao popularan pa je izašlo i nekoliko standarda koji nisu bili usklađeni [5].

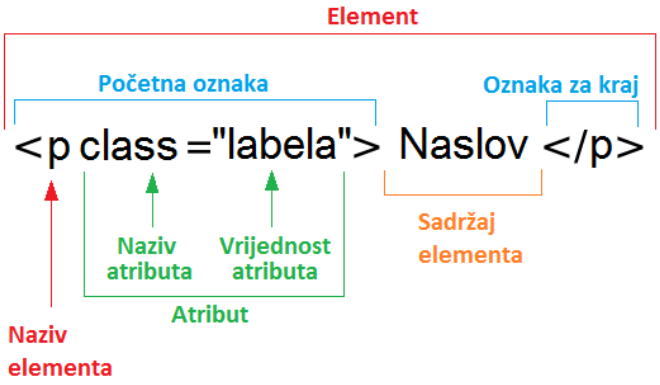
Prva verzija HTML-a je objavljena 1993. godine. U to je vrijeme bio još poprilično ograničen, pa nije bilo moguće dodati čak ni slike u HTML dokumente. Prva verzije HTML-a se sastojala od 18 oznaka dok najnovija verzija ima čak 140 HTML oznaka. Druga verzija HTML-a je objavljenja 1995. godine no ni ona nije postala standard. Godine 1997. HTML 3.2 je objavljen kao W3C (World Wide Web konzorcij) preporuka. Bila je to prva verzija koju je razvio i standardizirao isključivo W3C. Krajem 1999. godine W3C je objavio novu verziju HTML 4.2. Postojale su tri varijacije:

- Strict, u kojoj su zastarjeli elementi bili zabranjeni
- Transitional, u kojoj su zastarjeli elementi bili dopušteni
- Frameset, u kojoj su uglavnom dopušteni samo elementi povezani s okvirom

HTML 4.2 je usvojio mnoge tipove i atribute specifične za preglednike ali je u isto vrijeme nastojao ukinuti Netscapeove značajke vizualnog sustava u korist stilskih tablica.

Najnovija verzija HTML-a nazvana HTML5 je izašla 2014. godine dok je HTML 5.2 objavljen 2017. godine. Nova verzija HTML-a je uvela puno novih oznaka i značajki u jezik. Najbitnija je podrška za audio i video zapise.

HTML oznake i atributi se pišu kako je prikazano na slici 3.2 [5].




**Slika 3.2.** *Primjer pisanja oznake i atributa u HTML-u [5].*

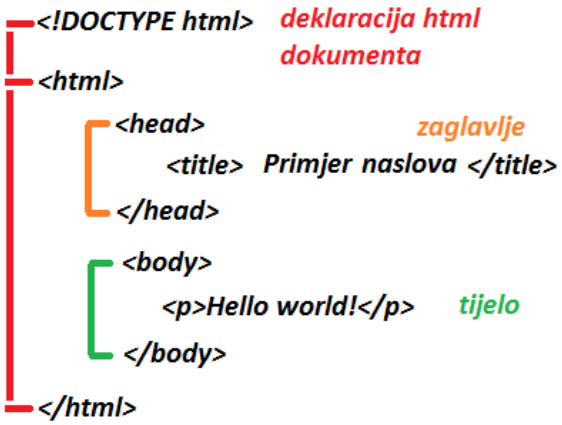
Svaki HTML dokument mora imati tri glavne oznake, a to su <*html*>, <*head*> i <*body*>.

<*html*> označava početak HTML dokumenta.

<*head*> označava zaglavlje HTML dokumenta u kojem se nalazi naslov stranice, stilska obilježja stranice te se definiraju JavaScript skripte.

<*body*> označava sadržaj koji se prikazuje na stranici.

Prikaz 3 glavne oznake je vidljiv na slici 3.3 [6].




**Slika 3.3.** *Prikaz glavnih oznaka u HTML-u [6].*

## 3.2. CSS

CSS (eng. *Cascading Style Sheets*), stilski je jezik za uređivanje i oblikovanje web stranica. Koristi se za uređivanje boja, slika, dodavanje margina, linkova, fontova i slično. Prva verzija CSS je definirana 1996. godine no do stvarnog usvajanja CSS-a je prošlo još dosta vremena. Glavna ideja CSS-a je odvajanje prezentacijskog koda u zasebne datoteke i njegovo definiranje pomoću pravila koja se mogu odnositi na više elemenata odjednom. Trenutačna verzija CSS-a je CSS 3, još uvijek se razvija i nadograđuje. Najbitnija stavka CSS 3 je da podržava responzivni dizajn [7].

CSS kôd se sastoji od tri glavna elementa:

1. selektori – određuje na koji se element utječe.
2. svojstva –opisuju pojedina svojstva koja se primjenjuju na elementu.
3. vrijednosti – predstavljaju vrijednost koju prima pojedino svojstvo.

Osnovna sintaksa CSS-a je prikazana na slici 3.4 [8].


**Slika 3.4.** *Osnovna sintaksa CSS-a [8].*

Postoje tri načina implementacije CSS-a s HTML-om [9]:

1. Linijski način, unutar elementa, slika 3.5.
2. Unutarnji način, skripta se nalazi u istoj datoteci, slika 3.6.
3. Vanjski način, skripta se nalazi u drugoj datoteci koja se uključuje, slika 3.7.

```
<p style="color:green;font-size:20px;">Ovo je paragraf</p>
```

**Slika 3.5.** *Linijski način implementacije CSS-a [9].*

```
<head>
<style type="text/css">
body {background-color:red;}
p {color:blue;}
</style>
</head>
```

**Slika 3.6.** *Unutarnji način implementacije CSS-a [9].*

```
<head>
<link rel="stylesheet" type="text/css" href="stil.css">
</head>
```

**Slika 3.7.** *Vanjski način implementacije CSS-a [9].*

U CSS-u postoji nekoliko osnovnih tipova selektora [10]:

- jednostavni selektori (eng. *type selectors*), najjednostavniji su od svih. Odgovaraju imenu HTML oznake i primjenjuju se na svaki istovrsni element u dokumentu. Kôd prikazan na slici 3.8.

```
p {
    font-family: Verdana, Helvetica, sans-serif;
}

h1 {
    color: #9b5c98;
    font-size: 24px;
    font-weight: bold;
}
```

**Slika 3.8.** *Prikaz jednostavnih selektora u CSS-u.*

- klasni selektori, dodavanje klase nekom HTML elementu i utjecanje na svojstva elemenata s tom klasom. Kôd prikazan na slici 3.9. uz dodavanje klasa HTML elementima na slici 3.10.

```css
.izreka {
    font-weight: bold;
    font-size: 11px;
}

.prijevod {
    font-style: italic;
    font-size: 12px;
}
```

**Slika 3.9.** *Prikaz klasnih selektora u CSS-u.*

```html
<p class="izreka">Navigare necesse est,
    vivere non est necesse.</p>
<p class="prijevod">Ploviti je nužno,
    živjeti nije nužno.</p>
```

**Slika 3.10.** *Prikaz dodavanja klasa HTML elementima.*

- ID selektori, dodavanje ID atributa nekom HTML elementu i utjecanje na svojstva samo elementa s tim ID-om. Prikaz CSS kôda na slici 3.11. te dodavanje ID atributa u HTML-u na slici 3.12.

```css
#izreka {
    font-weight: bold;
    font-size: 11px;
}

#prijevod {
    font-style: italic;
    font-size: 12px;
}
```

**Slika 3.11.** *Prikaz ID selektora u CSS-u.*

```html
<p id="izreka">Navigare necesse est,
    vivere non est necesse.</p>
<p id="prijevod">Ploviti je nužno,
    živjeti nije nužno.</p>
```

**Slika 3.12.** *Prikaz dodavanja ID atributa HTML elementima.*

- kontekstni selektori, gledaju na hijerarhiju elemenata u HTML-u te na temelju toga mijenjaju svojstva. Prikaz CSS kôda na slici 3.13. te HTML kôda na slici 3.14.

```css
h6 {
    font-weight: normal;
    font-size: 11px;
}

p {
    font-weight: bold;
    font-size: 11px;
}

p b {
    font-size: 12px;
    color: purple;
}
```

**Slika 3.13.** *Prikaz kontekstnih selektora u CSS-u.*

```html
<h6><b>Navigare</b> necesse est, ...</h6>
<p><b>Navigare</b> necesse est,
<b>vivere</b> non est necesse.</p>
```

**Slika 3.14.** *Prikaz HTML* kôda *za kontekstne selektore u CSS-u.*

- pseudoklase, prikaz CSS kôda na 3.15.

```css
a:hover {
    text-decoration: underline;
}

a:active {
    background-color: #ab1919;
    color: #ffffff;
    text-decoration: dashed;
}
```

**Slika 3.15.** *Prikaz pseudoklasnih selektora u CSS-u.*

## 3.3. Bootstrap

*Bootstrap* je najpopularniji besplatni *frontend* okvir za brži i lakši razvoj responzivnih web aplikacija. Sastoji se od HTML, CSS i JavaScript predložaka dizajna za različite komponente na stranici kao što su slike, gumbi, forme i druge komponente. Trenutna verzija Bootstrapa je Bootstrap 5. Uključivanje Bootstrapa u HTML je prikazano na slici 3.16 [11].

```
<!-- Latest compiled and minified CSS -->
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">

<!-- jQuery library -->
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js"></script>

<!-- Latest compiled JavaScript -->
<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
```

**Slika 3.16.** *Prikaz uključivanja Bootstrapa u HTML [11].*

## 3.4. JavaScript

JavaScript (skraćeno JS) je objektno orijentirani jezik, poznatiji kao skriptni jezik za web stranice ali se koristi i u mnogim okruženjima koja nisu preglednici. Izvodi se u klijentskoj strani u pregledniku što se može koristiti za programiranje ponašanja web stranica u slučaju nekih događaja [12]. Podržavaju ga svi noviji web preglednici, moćan je i naširoko se koristi za kontrolu ponašanja web stranica. Stvoren je 1996. godine za Netscape Navigator 2.0. od Brendana Eicha.

JavaScript se najčešće koristi za [13]:

- Dodavanje interaktivnosti web mjestima – za stvaranje dinamičke stranice.
- Razvoj mobilnih aplikacija – koristi se i za izradu mobilnih aplikacija.
- Izradu igara – koristi se za programiranje igara koje se pokreću na web pregledniku.
- Pozadinski razvoj weba – većinom se koristi na *frontendu* ali je dovoljno svestran da se koristi i na *backendu*.

JavaScript se može pisati prema standardu Unicode ali se preporuča pisanje prema ASCII standardu. JS razlikuje velika i mala slova, pa je bitna dosljednost prilikom pisanja koda.

Uključivanje JS kôda u HTML kôd se može izvršiti na 4 načina [14]:

- Pisanjem kôda između oznaka <*script*> i </*script*> unutar same datoteke kao što je prikazano na slici 3.17 [15].
- Iz vanjske datoteke uporabom atributa *src* u oznaci <*script*> kao što je prikazano na slici 3.18 [15].
- Uključivanjem u URL-u koji se koristi posebnim protokolom *javascript:*

```
<script type="text/javascript">
<!--
    // JavaScript program
// -->
</script>
```

**Slika 3.17.** *Primjer pisanja JS kôda u HTML datoteci [15].*

```
<script type="text/javascript" src="datoteka.js"></script>
```

**Slika 3.18.** *Primjer uključivanja iz vanjske datoteke [15].*

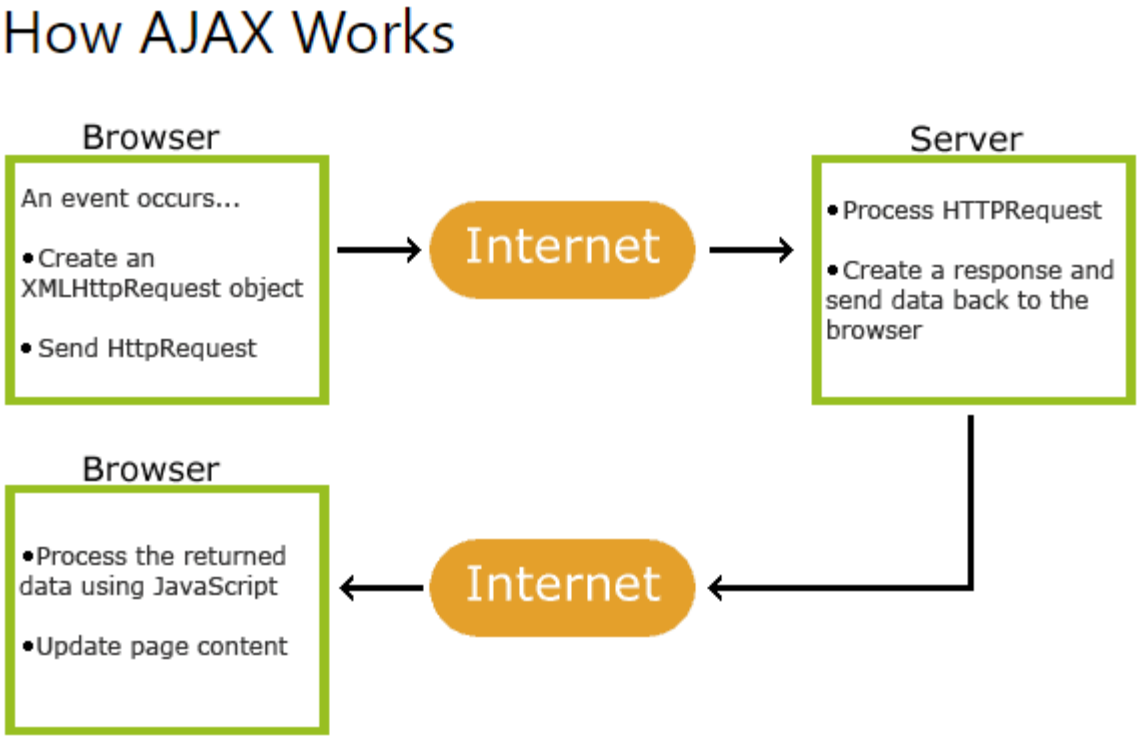
Deklaracija varijabli se obavlja pomoću ključne riječi *var*, a same varijable se dijele na cjelobrojne, decimalne, logičke i znakovne.

JavaScript omogućuje korisnikovu interakciju sa dijelovima web stranice kao što su gumbi koji reagiraju na klik ili prijelaz miša i ostalo.

## 3.5. AJAX

Ajax (*eng. Asynchronous JavaScript and XML*) je skaćenica za asinkroni JavaScript i XML je skup tehnika razvoja weba koji koristi različite web tehnologije na klijentskoj strani za stvaranje asinkronih web aplikacija. Uz Ajax, web aplikacije mogu slati i dohvaćati podatke s poslužitelja asinkrono bez ometanja prikaza i ponašanja stranice. Ajax omogućuje web aplikacijama da dinamički mijenjaju sadržaj bez potrebe za ponovnim učitavanjem cijele stranice. Podaci se mogu tražiti Ajax-a ili standardno gdje se cijela stranica mora osvježiti kako bi se podaci prikazali [16].

Rad Ajax-a je prikazan na slici 3.19 [17].




**Slika 3.19.** Rad Ajax-a [17].

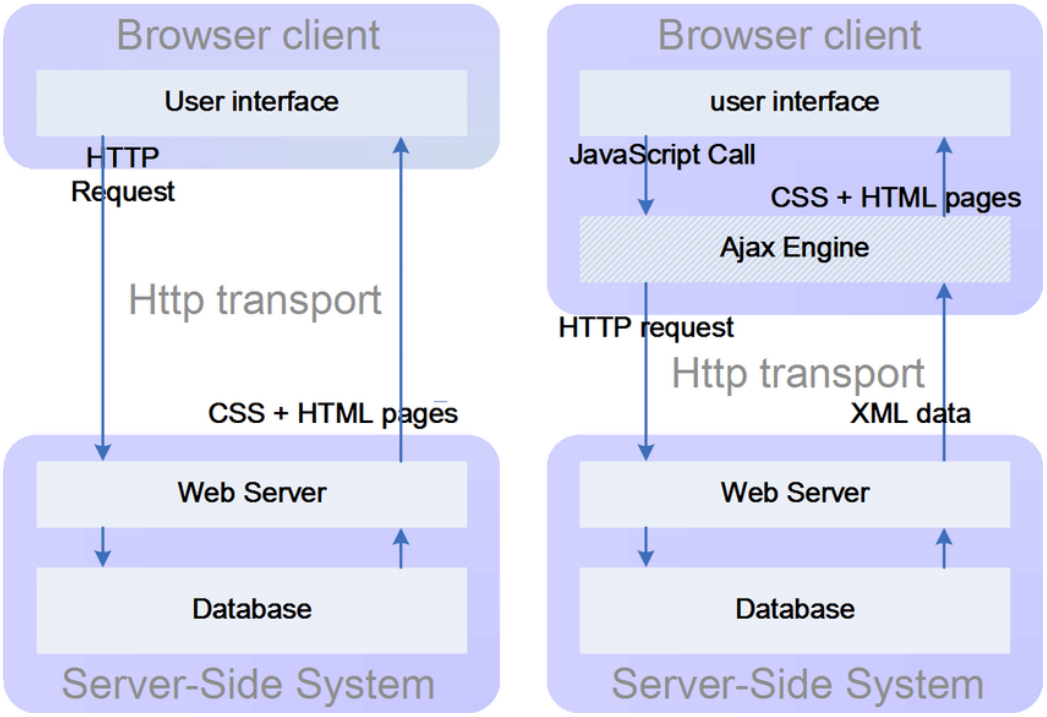
Prema standardnoj metodi zahtjeva podataka:

1. HTTP zahtjev se šalje iz preglednika na poslužitelj. Korisnik ne vidi promjene dok se zahtjev ne obradi.
2. Zahtjev dolazi do poslužitelja koji dohvaća tražene podatke.
3. Traženi podaci se prikazuju korisniku.

Prema Ajax metodi zahtjeva podataka:

1. Preglednik izvodi JavaScript te se stvara *XMLHttpRequest* objekt koji se koristi za komunikaciju sa serverom.
2. Zahtjev dolazi do poslužitelja koji dohvaća tražene podatke.
3. Traženi podaci se vraćaju u obliku HTML-a, XML-a i JavaScript-a Ajaxovom mehanizmu tražene podatke šalje nazad na preglednik.

Usporedba standardne metode i Ajax metode je prikazana na slici 3.20 [18].




**Slika 3.20.** *Usporedba standardne metode i Ajax metode traženja podataka [18].*

## 3.6. Python

Python je programski jezik opće namjene, interpretiran i visoke razine. Stvorio ga je Guido van Rossum 1990. godine. Python programerima omogućuje nekoliko stilova programiranja, objektno orijentirano, strukturno i aspektno orijentirano programiranje te je zbog te fleksibilnosti Python postaje sve popularniji [19].

Jedna velika razlika između Python i drugih programskih jezika je ta što nakon naredbe nema potrebe stavljati „;“ , a kao metodu razlikovanja programskih blokova koristi uvlačenje, a ne vitičaste zagrade ili ključne riječi kao većina programskih jezika.

Python kôd se piše u skripti s proširenjem *.py* .Prikaz Python skripte koja ispisuje nekakav tekst je vidljiv na slici 3.21 [19].

```
print ("Hello World")
```

**Slika 3.21.** *Prikaz Python skripte koja ispisuje tekst [19].*

## 3.7. PostgreSQL

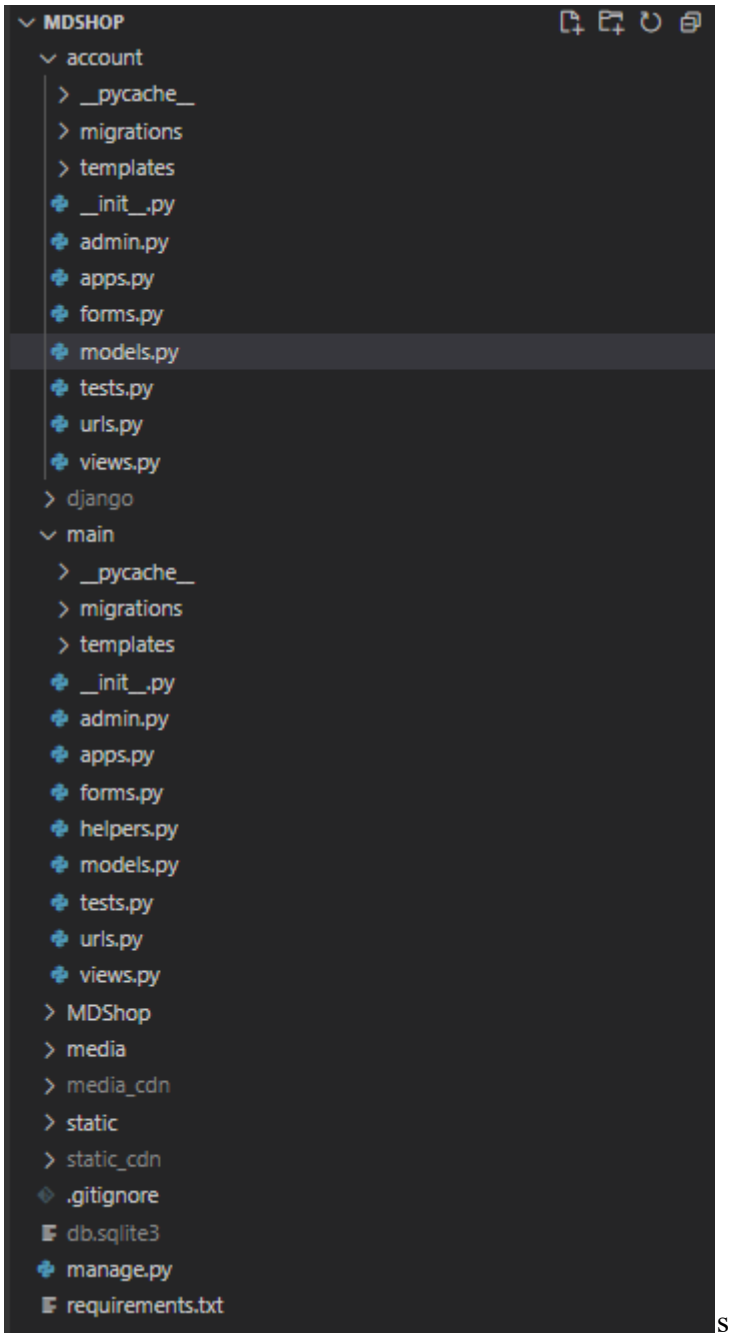
PostgreSQL je objektno-relacijski sustav baze podataka otvorenog kôda koji koristi i proširuje SQL jezik. PostgreSQL je započeo 1986. godine, kao dio projekta POSTGRES na Kalifornijskom sveučilištu u Berkeleyu i ima više od 30 godina aktivnog razvoja na temeljnoj platformi. PostgreSQL je stekao snažnu reputaciju svojom dokazanom arhitekturom, pouzdanošću, integritetom podataka, robusnim skupom značajki, proširivošću i predanošću zajednice otvorenog kôda koja stoji iza softvera kako bi se dosljedno isporučila učinkovita i inovativna rješenja. PostgreSQL radi na svim glavnim operativnim sustavima, kompatibilan je s ACID-om (eng. a*tomicity, consistency, isolation, durability*) od 2001. i ima moćne dodatke kao što je popularni postGIS geoprostorni ekstender baze podataka [20].

## 3.8. Visual Studio Code

Visual Studio Code (poznat kao VS Code) je Microsoftov besplatni uređivač teksta otvorenog kôda. VS Code je dostupan za Windows, Linux, i macOS. Uključuje neke moćne značajke koje su VS Code učinile jednim od najpopularnijih razvojnih okruženja u posljednje vrijeme. VS Code podržava velik broj programskih jezika kao što su Java, C++, Python, CSS i drugi. Omogućuje dodavanje pa čak i stvaranje novih proširenja, uključujući alate za ispravljanje pogrešaka i podršku za razvoj u oblaku i webu [21].

# 4. IZRADA APLIKACIJE

U ovom poglavlju opisan je kôd koji je napisan tokom izrade web stranica, korištenjem različitih tehnologija. Na početku stvaranja web stranice napravljen je projekt MDShop u django framework-u kao što je prikazano na slici 4.1.

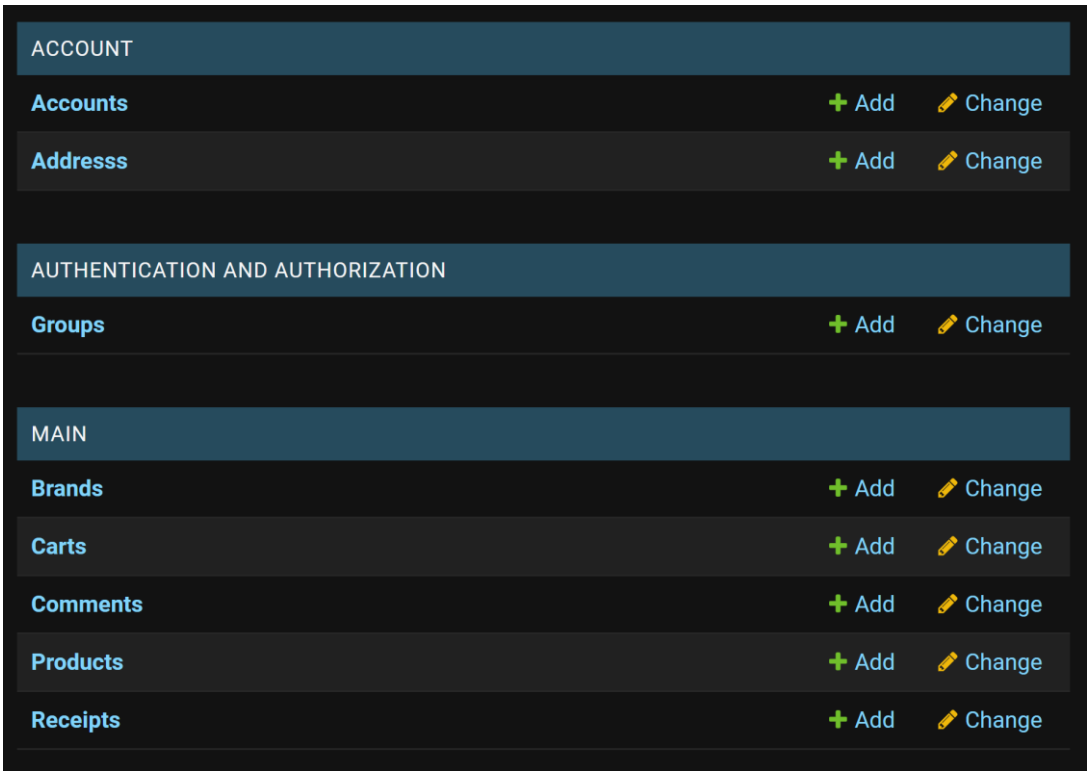


s

**Slika 4.1.** *Prikaz direktorija MDShop.*

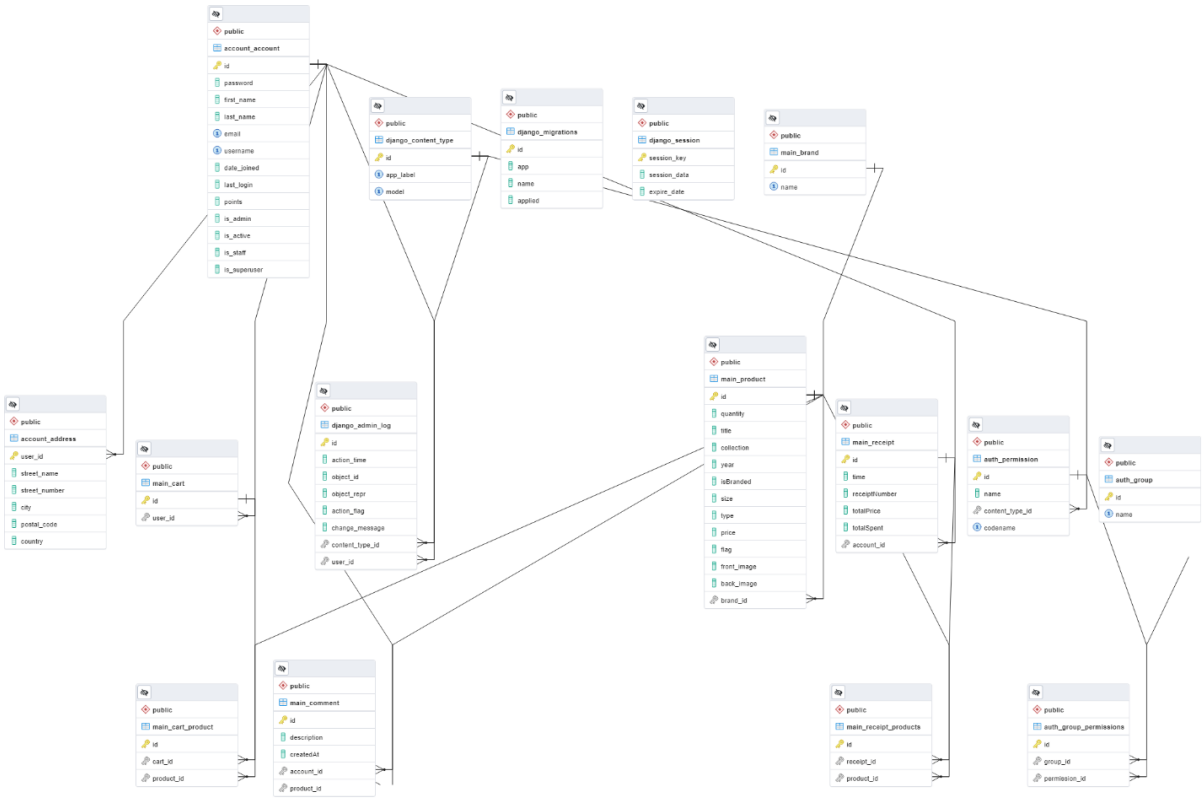
## 4.1. Baza podataka

U datoteci models.py koja je automatski stvorena, napravljeni su modeli objekata koji predstavljaju tablice u bazi podataka koji su vidljivi na slici 4.2.




**Slika 4.2.** *Prikaz svih tablica u bazi.*

Na slici 4.3. vidljiv je ER dijagram baze podataka.



**Slika 4.3.** *ER dijagram baze.*

Dio kôda pomoću kojeg je stvorena tablica za korisnika u bazi podataka se nalazi na slici 4.4. Naziv tablice u bazi je *Account,* a atribute koje sadrži tablica su *first_name, last_name* i *username* koji su tipa charfield odnosno tekstualnog tipa, *email* koji ima poseban tip specifičan za email, *date_joined* koji je tipa date te se u njega zapisuje datum, te još pomoćni atributi *last_login, is_admin, is_active, is_staff* i *is_superuser* koji su tipa boolean i imaju dvije moguće vrijednosti, true ili false.

```
class Account(AbstractBaseUser):
    first_name = models.CharField(verbose_name='First name', max_length=60)
    last_name = models.CharField(verbose_name='Last name', max_length=60)
    email = models.EmailField(verbose_name='E-mail', max_length=60, unique=True)
    username = models.CharField(max_length=16, unique=True)
    date_joined = models.DateField(verbose_name='Date joined', auto_now_add=True)
    last_login = models.DateField(verbose_name='Last login', auto_now_add=True)
    points = models.IntegerField(default=0)
    is_admin = models.BooleanField(default=False)
    is_active = models.BooleanField(default=True)
    is_staff = models.BooleanField(default=False)
    is_superuser = models.BooleanField(default=False)
```

**Slika 4.4.** *Prikaz tablice za Account u bazi stvorene u models.py.*

## 4.2. Prijava i registracija korisnika

Registracija i prijava su potrebni na web stranici kako bi korisnici mogli kupovati artikle i vidjeti sve dostupne podatke na web stranici. Neregistrirani korisnici ne mogu kupovati artikle niti vidjeti komentare na određene artikle. Za registraciju korisnika se u *models.py* skripti koristi kôd prikazan na slici 4.5. koji provjerava je li korisnik unio *email* i *username*. U skripti *forms.py* nalazi se kôd koji predstavlja formu za registraciju, a vidljiv je na slici 4.6. Također, prilikom registracije, korisnik unosi i svoju adresu koja će kasnije biti korištena za dostavu artikala ako se korisnik odluči na kupnju i dostavu na adresu. Forma, koju korisnik popunjava prilikom registracije za unos adrese, se nalazi na slici 4.7.

```
class AwesomeAccountManager(BaseUserManager):
    def create_user(self, email, username, password):
        if not email:
            raise ValueError("Users must have an email address.")
        if not username:
            raise ValueError("Users must have a username.")
        user = self.model(
            email=self.normalize_email(email),
            username=username,
        )
        user.set_password(password)
        user.save(using=self._db)
        return user
```

**Slika 4.5.** *Prikaz klase za registraciju korisnika.*

```
class Meta:
    model = Account
    fields = ['first_name', 'last_name', 'email', 'username', 'password1', 'password2']
    widgets = {
        'first_name': forms.TextInput(attrs={
            'name': 'first_name',
            'class': 'form-control form-control-lg',
            'placeholder': 'First Name',
            'autofocus': 'true'
        }),
        'last_name': forms.TextInput(attrs={
            'name': 'last_name',
            'class': 'form-control form-control-lg',
            'placeholder': 'Last Name'
        }),
        'email': forms.EmailInput(attrs={
            'name': 'email',
            'class': 'form-control form-control-lg',
            'placeholder': 'Electronic mail'
        }),
        'username': forms.TextInput(attrs={
            'name': 'username',
            'class': 'form-control form-control-lg',
            'placeholder': 'Username'
        }),
    }
```

**Slika 4.6.** *Prikaz forme za registraciju korisnika i unos podataka o korisniku.*

```
class AddressForm(forms.ModelForm):
    class Meta:
        model = Address
        fields = ['street_name', 'street_number', 'city', 'postal_code', 'country']
        widgets = {
            'street_name': forms.TextInput(attrs={
                'name':'Street Name',
                'class': 'form-control form-control-lg',
                'placeholder': 'Street Name'
            }),
            'street_number': forms.TextInput(attrs={
                'name': 'Street Number',
                'class': 'form-control form-control-lg',
                'placeholder': 'Street Number'
            }),
            'city': forms.TextInput(attrs={
                'name': 'City',
                'class': 'form-control form-control-lg',
                'placeholder': 'City Name'
            }),
            'postal_code': forms.NumberInput(attrs={
                'name': 'Postal Code',
                'class': 'form-control form-control-lg',
            }),
            'country': forms.TextInput(attrs={
                'name': 'Country',
                'class': 'form-control form-control-lg',
                'placeholder': 'Country Name'
            }),
        }
```

**Slika 4.7.** *Prikaz forme za registraciju korisnika i unos podataka o adresi.*

Prilikom prijave na stranicu korisnik unosi email i lozinku s kojima se registrirao. Forma za prijavu je prikazana na slici 4.8. Ukoliko uneseni podaci nisu točni, korisnik će dobiti obavijest da prijava nije bila uspješna, taj dio kôda se također nalazi u istoj formi, a prikazan je na slici 4.9.

```
class LoginForm(forms.ModelForm):
    class Meta:
        model = Account
        fields = ['email', 'password']
        widgets = {
            'email': forms.EmailInput(attrs={
                'name': 'email',
                'class': 'form-control form-control-lg',
                'placeholder': 'Electronic mail',
                'autofocus': 'true'
            }),
            'password': forms.PasswordInput(attrs={
                'name': 'password1',
                'class': 'form-control form-control-lg',
                'placeholder':'Password'
            }),
        }
```

**Slika 4.8.** *Prikaz forme za prijavu korisnika.*

```
if not authenticate(email=email, password=password):
    raise forms.ValidationError('Invalid login.')
```

**Slika 4.9.** *Prikaz forme za prijavu korisnika.*

## 4.3. Početna stranica

Ukoliko korisnik nije prijavljen na početnoj ili drugim stranicama web aplikacije, dobit će poruku kako samo registrirani korisnici mogu kupovati proizvode i vidjeti komentare na proizvode. Neregistrirani korisnici mogu samo vidjeti proizvode na glavnim stranicama ali i detalje o pojedinom proizvodu klikom na gumb *info.* Slika 4.10. prikazuje kôd za ispis poruke neregistriranim korisnicima, a nalazi se u *navbar.html.*

```
{% if not request.user.is_authenticated %}
<div class="sticky-top bg-warning text-center p-2">
    <h5 class="text-center">You have to login in order to buy items and view comments!</h5>
</div>
{% endif %}
```

**Slika 4.10.** *Prikaz klase za registraciju korisnika.*

Nakon uspješne prijave korisnik je odveden na početnu i glavnu stranicu koja se zove *index* na kojoj se nalaze izdvojeni proizvodi, poput onih koji su novi ili trenutno popularni. Svi ostali proizvodi imaju vrijednost *flag* atributa *NONE* pa se na jednostavan način mogu prikazati samo oni koji imaju neku konkretnu vrijednost. Kôd koji to prikazuje se nalazi na slici 4.11.

```
{% if product.getFlag != 'NONE' and product.quantity > 0 %}
```

**Slika 4.11.** *Prikaz klase za registraciju korisnika.*

Na svakoj stranici se nalaze *navbar* i *footer* pa se oni nalaze u posebnoj *html* skripti koja se zove *base.html*. Uz njih, u skripti se također nalaze i linkovi i skripte potrebne za bootstrap, javascript i fontawesome. Skripta *base.html* je uključena u druge skripte kako se kôd ne bi morao ponavljati. Kôd za uključivanje skripte se nalazi na slici 4.12.

```
{% extends 'main/base.html' %}
```

**Slika 4.12.** *Prikaz kôda za uključivanje skripte base.html.*

Za prikaz pojedinog artikla koristila se bootstrapova klasa *card*, koja svaki proizvod stavlja u karticu na kojoj se nalazi slika proizvoda ali i bitni podaci i pojedinom proizvodu poput naziva, cijene, marke, veličine i spola za koji je artikl namijenjen. Kôd za karticu proizvoda se nalazi na slici 4.13.

```
<div class="card text-center">
    <p class="flag">{{ product.getFlag }}</p>
    <img src="{{ product.front_image.url }}" alt="{{ product.title }}" id="image-{{ product.id }}" class="img-fluid border-bottom">
    <script>
        $("#image-{{ product.id }}").hover(function () {
            $(this).attr("src", "{{ product.back_image.url }}");
        }, function () {
            $(this).attr("src", "{{ product.front_image.url }}");
        });
    </script>
    <div>
        <h2 class="p-2 text-uppercase">{{ product.title }}</h3>
            <h4 class=""> {{ product.brand }}</h5>
    </div>
    <div class="card-body mx-auto">
        <p><b>{{ product.price }}kn</b></p>
        <p>{{ product.getSize }} - {{ product.size }}</p>
        <p> {{ product.getType }}</p>

        <div class="row">
            {% if request.user.is_authenticated %}
            {% include './includes/cartManagment.html' %}
            {% endif %}
            <a class="btnDetails mt-2" href="{% url 'main:productDetail' product.id %}">
                <i class="fa-solid fa-circle-info"></i> Info</a>
        </div>
    </div>
</div>
```

**Slika 4.13.** *Prikaz kartice za prikaz artikla.*

Ostale stranice na koje korisnik može otići klikom na navbaru su *MEN, WOMEN,* i *KIDS*. Ovisno o odabiru, na pojedinoj stranici će se prikazivati proizvodi za muškarce, žene ili djecu.

## 4.4. Detaljan prikaz artikla

Kod prikaza svakog proizvoda pojavljuju se dva gumba, a jedan od njih je gumb *info* koji nakon klika, korisnika vodi na novu stranicu koja detaljno prikazuje sami proizvod. To je postignuto tako da se prilikom klika u uzima *id* proizvoda te se otvara nova stranica koja prikazuje taj pojedini proizvod ali s većim slikama i više podataka o artiklu. URL kreiran za to nalazi se u skripti *urls.py* za to je prikazan na slici 4.14.

```
path('product_detail/<int:productId>',productDetailView, name='productDetail'),
```

**Slika 4.14.** *Prikaz URL-a za detaljan prikaz artikla.*

S desne strane se nalaze komentari koje su pojedini korisnici napisali o tome proizvodu. Svaki registrirani korisnik može napisati komentar, vidjeti druge komentare i obrisati komentare koje je

on sam napisao, dok *admin* može obrisati sve komentare. Metoda pomoću koje su dohvaćeni podaci o artiklu i komentarima se nalazi na slici 4.15. U dolje vidljivoj metodi predaje se *id* artikla, te se onda pokušava dohvatiti proizvod koji ima taj *id*, ukoliko proizvod ne postoji, dolazi do greške i korisniku će se prikazati *Http 404 error*. Međutim, nakon uspješnog dohvaćanja proizvoda, u detaljan prikaz se šalje proizvod i komentari koji su napisani za njega.

```
def productDetailView(request,productId):
    context = {}
    try:
        product = get_object_or_404(Product, id=productId)
        comments = product.comments.all()
    except Http404:
        return redirect('main:index')

    context = {
        'product' : product,
        'comments' : comments

    }

    return render(request, 'main/productDetail.html', context)
```

**Slika 4.15.** *Prikaz metode za detaljan prikaz artikla.*

## 4.5. Proces kupnje

Ukoliko se korisnik odluči na kupnju, klikom na gumb *Buy now* korisnik će dodati proizvod u košaricu. Kôd kojim je proces napravljen nalazi se na slici 4.16. To je postignuto tako da prilikom klika na taj gumb, korisnik šalje *id* proizvoda koji hoće kupiti u košaricu, a u košarici se onda prikazuje proizvod s tim *id-em*. Također, ukoliko je proizvod dodan u košaricu, mijenja se klasa i tekst gumba pa gumb *Buy now* sada prikazuje text *Remove* i ima bootstrapovu klasu koja ga stilizira u crvenu boju. Osim teksta mijenjaju se i ikone koje se nalaze pokraj.

```
$("#btnCartManage-{{ product.id }}").click(function () {
    $.ajax({
        url: "{% url 'main:manageCart' %}",
        type: "POST",
        data: {
            "id": "{{ product.id }}",
            csrfmiddlewaretoken: '{{ csrf_token }}',
        },
        cache: false,
        dataType: "json",
        success: function () {
            isInCart = !isInCart;
            if (isInCart) {
                $("#btnCartManage-{{ product.id }}").removeClass("btnBuy").addClass("btn-outline-danger");
                $("#btnCartManage-{{ product.id }}").html('<i class="fa-solid fa-square-minus"></i> Remove');
            }
            else {
                $("#btnCartManage-{{ product.id }}").addClass("btnBuy").removeClass("btn-outline-danger");
                $("#btnCartManage-{{ product.id }}").html('<i class="fa-solid fa-circle-plus"></i>  Buy now');
            }
        }
    });
});
```

**Slika 4.16.** *Prikaz funkcije za kupnju artikla.*

## 4.6. Košarica

Košarica je tablica u bazi koja u sebi sadrži artikle koje je korisnik dodao te samog korisnika kako bi se znalo koji je korisnik dodao proizvode u košaricu. Model košarice se nalazi na slici 4.17.

```
class Cart(models.Model):
    user                = models.OneToOneField(Account, on_delete=models.CASCADE, verbose_name='cart')
    product             = models.ManyToManyField(Product, related_name='products')
```

**Slika 4.17.** *Prikaz modela košarice.*

Za prikaz košarice se koristi metoda *cartView* u kojoj se dohvaća košarica te se onda predaje u kontekstu koji se šalje na stranicu *cart.html* na kojoj se prikazuju proizvodi koje je korisnik odabrao. Slika 4.18. prikazuje metodu za prikaz košarice.

```
def cartView(request):
    cart = get_object_or_404(Cart, user=request.user)
    return render(request, 'main/cart.html', {'cart': cart, })
```

**Slika 4.18.** *Prikaz metode za dohvaćanje i prikaz košarice.*

Metoda *cartMenagmentView* je metoda pomoću koje se proizvod dodaje u košaricu pomoću funkcije *cart.product.add()* ili uklanja iz košarice pomoću funkcije *cart.product.remove()*. Nakon dodavanja ili uklanjanja artikla iz košarice, objekt košarice se sprema. Kôd koji obavlja te radnje se nalazi na slici 4.19.

```
def cartManagmentView(request):
    if request.method == "POST" and isAjax(request):
        productId = request.POST.get('id', None)
        user = Account.objects.get(id=request.user.id)
        product = Product.objects.get(id=productId)
        cart = get_object_or_404(Cart, user=user)
        if product in cart.product.all():
            cart.product.remove(product)
        else:
            cart.product.add(product)
        cart.save()
        return HttpResponse(json.dumps({ "good": True }), content_type="application/json")
    else:
        return redirect('main:index')
```

**Slika 4.19.** *Prikaz metode za spremanje i uklanjanje artikala iz košarice.*

Nakon što je korisnik dodao sve artikle u košaricu, može potvrditi kupnju klikom na gumb *Go to checkout.* Klikom na taj gumb se otvara modal u kojem se nalaze podaci o korisniku koji obavlja kupnju, ukupnoj cijeni svih proizvoda te o načinu preuzimanja proizvoda. Korisnik ima opciju odabrati preuzimanje u trgovini, ili dostavu na adresu koju je korisnik unio prilikom registracije. U slučaju dostave, dostupna je opcija plaćanja pouzećem. Model računa, potvrda kupnje, vidljiva je na slici 4.20.

```
class Receipt(models.Model):
    products = models.ManyToManyField(Product, related_name='receipt_products')
    time = models.DateTimeField(auto_now_add=True)
    account = models.ForeignKey(Account, on_delete=models.CASCADE, verbose_name='receipt')
    receiptNumber = models.CharField(max_length=255, null=False, blank=False)
    totalPrice = models.FloatField()
    totalSpent = models.FloatField()
```

**Slika 4.20.** *Prikaz modela računa.*

Prilikom potvrde kupnje korisnikov račun se sprema u bazu podataka. Potvrda kupnje se obavlja klikom na gumb *Confim purchase* koji je dostupan u modalu. Klikom na njega, poziva se metoda

*checkoutFinalView* koja provjerava radi li se o *POST* metodi te dohvaća podatke koji su potrebi poput *id*-a košarice i broja računa. Dalje se dohvaća košarica sa svim proizvodima, te se u *for* petlji prolazi kroz sve proizvode koji se uklanjaju iz košarice, te im se umanjuje ukupna količina. Račun se kreira i sprema u bazu podataka kao potvrda kupnje. Kôd koji prikazuje funkcioniranje gore objašnjenog se nalazi na slici 4.21.

```
def checkoutFinalView(request):
    user = request.user
    if request.method == "POST":
        cartId = request.POST.get('id', None)
        receiptNumber = request.POST.get('receiptNumber', None)
        cart = get_object_or_404(Cart, id=cartId)
        receipt = Receipt.objects.create(
            account=request.user, receiptNumber=receiptNumber, totalPrice=0.0, totalSpent=0.0)
        receipt.save()
        for product in cart.product.all():
            receipt.products.add(product)
            product.quantity -= 1
            product.save()
            cart.product.remove(product)
            receipt.totalPrice += float(product.price)
        cart.save()
        receipt.save()
        if receipt.totalPrice >= 100:
            if user.points < 10:
                user.points += 1
            elif user.points == 10:
                user.points = 1
        user.save()
        return JsonResponse({"good": "True"})
```

**Slika 4.21.** *Prikaz potvrde kupnje i stvaranja računa.*

## 4.7. Mogućnosti admina

Admin ili administrator je osoba koja je registrirana na stranici ali ima veća prava i obveze od običnih korisnika. Admin može obavljati sve isto kao i običan registrirani korisnik ali ima mogućnosti kao što su: dodavanje, uređivanje i brisanje artikala, dodavanje i brisanje marki tj. brendova, brisanje svih komentara te uređivanje i brisanje registriranih korisnika. Za tu svaku, dodatnu mogućnost mora postojati provjera je li korisnik admin, ukoliko je, ima pravo pristupiti i obavljati određene promjene, ukoliko nije, korisniku se ne smije dopustiti obavljanje promjena te

se vraća na početnu stranicu. Kôd koji prikazuje način provjere u metodi nalazi se na slici 4.22., a način provjere u *HTML*-u na slici 4.23.

```
if not request.user.is_superuser:
    return redirect('main:index')
```

**Slika 4.22.** *Prikaz provjere je li korisnik admin u metodi.*

```
{% if request.user.is_superuser %}
```

**Slika 4.23.** *Prikaz provjere je li korisnik admin u HTML-u.*

Admini imaju pristup listi svih proizvoda. Imaju pravo uređivanja postojećih ali i dodavanja novih proizvoda. Također, smiju i brisati određene proizvode ukoliko ih više nema ili se takvi proizvodi više ne proizvode. Na slikama 4.24. i 4.25. su prikazane metode za prikaz proizvoda u tablici.

```
def productsView(request):
    if not request.user.is_superuser:
        return redirect('main:index')

    return render(request,"main/products.html",{})
```

**Slika 4.24.** *Metoda koja vodi admina na html dokument za prikaz proizvoda.*

Metoda *productsView* provjerava je li korisnik koji joj pristupa admin, ukoliko nije, vraća ga se na početnu stranicu, ukoliko je, šalje ga se na *product.html* gdje se pomoću metode *productsJsonView* prikazaju svi proizvodi koji se nalaze u bazi podataka.

```python
def productsJsonView(request):
    products = Product.objects.all()
    total = products.count()
    _start = request.GET.get('start')
    _length = request.GET.get('length')
    if _start and _length:
        start = int(_start)
        length = int(_length)
        page = math.ceil(start / length) + 1
        per_page = length
        products = products[start:start + length]
    data = [product.to_dict_json() for product in products]
    response = {
        'data': data,
        'page': page,
        'per_page': per_page,
        'recordsTotal': total,
        'recordsFiltered': total,
    }
    return JsonResponse(response)
```

**Slika 4.25.** *Metoda koja dohvaća proizvode i vraća ih na stranicu u json obliku*

Na stranici se dohvaćeni *json* objekti prikazuju u tablici. Kôd koji prikazuje funkciju za dohvaćanje proizvoda i njihov stilizirani prikaz nalazi se na slici 4.26.

```javascript
$(document).ready(function () {
    table = $('#productsTable').DataTable({
        processing: true,
        serverSide: true,
        ajax: {
            type: "GET",
            datatype: 'json',
            url: 'products/json',
        },
        success: function (data) {
            console.log(data);
        },
        columns: [
            {
                data: "image", render: function (data, type, row, meta) {
                    return '<script>' +
                        '$("#imageProducts-' + row["pk"] + '").hover(function () {' +
                        '$(this).attr("src", "' + row["back_image"] + '");' +
                        '}, function () {' +
                        '$(this).attr("src", "' + row["front_image"] + '");' +
                        '});' +
                        '<\/script>' +
                        '<a><img src="' + row["front_image"] + '" id="imageProducts-' + row["pk"] + '" alt="' + row["title"] + '" class="cartImage"></a>'
                }
            },
            { "data": "title" },
            { "data": "brand" },
            {
                data: "price", render: function (data, type, row, meta) {
                    var eur = 7.5345;
                    return row['price'] + ' HRK <br> ' + (row['price']/eur).toFixed(2) + ' EUR' ;
                } },
            { "data": "type" },
            { "data": "quantity" },
            { "data": "size" },
            { "data": "collection" },
            { "data": "flag" },
            {
                data: "id", render: function (data, type, row, meta) {
                    return '<button onclick=visitEditProduct(' + row["pk"] + ') class="btn btn-outline-info"><i class="fa-solid fa-pen-to-square text-dark"></i></button>' + " " +
                        '<button onclick=deleteProduct(' + row["pk"] + ') class="btn btn-outline-danger" id="removeProduct-' + row["pk"] + '"><i class="fa-solid fa-ban text-dark"></i></button>';
                }
            },

        ],
    });
});
```

**Slika 4.26.** *Funkcija za dohvaćanje i prikaz proizvoda u tablici*

Ukoliko admin želi dodati novi proizvod, to može napraviti klikom na gumb *Add product* koji se nalazi iznad tablice izlistanih proizvoda. Klikom na gumb za dodavanje proizvoda, admina se preusmjerava na novu stranicu gdje se nalazi forma za unos novog proizvoda. Dio forme za dodavanje novog proizvoda iz datoteke *forms.py* nalazi se na slici 4.27. U formi se nalaze svi atributi koje je potrebno unijeti u bazu podataka. Na slici 4.27. je vidljivo da je *title* tipa *textinput* te su mu dodijeljene klasa, ime, id, i ostali atributi.

```
class Meta:
    model = Product
    fields = ['title', 'collection', 'year', 'quantity', 'type', 'flag', 'brand', 'size', 'price', 'front_image', 'back_image']
    widgets = {
        'title': forms.TextInput(attrs={
            'class': 'form-control',
            'type': 'text',
            'name': 'Title',
            'id': 'id_title',
            'placeholder': 'Title',
            'maxlength': '80',
            'required': True,
            'autofocus': True,
        }),
        'collection': forms.Select(choices=Product.ClothesCollection, attrs={
            'class': 'form-control',
            'name': 'Collection',
            'id': 'id_collection',
            'required': True
        }),
```

**Slika 4.27.** *Forma za dodavanje novog proizvoda*

Novi proizvod se dodaje pomoću metode koja se nalazi na slici 4.28. Prvo se radi provjera je li korisnik admin jer obični korisnici ne smiju moći dodavati proizvode. Nakon uspješne potvrde da se radi o adminu, provjerava se je li metodi predan *id* marke tj. brenda kako bi novi proizvod bio te marke. Ukoliko je, u formi se u polju *brand* dodjeljuje ta marka. Ukoliko nije predan nikakav *id*, kreira se prazna forma te admin unosi podatke u polja koja su mu pojavila. Nakon unosa, potvrđuje dodavanje novog proizvoda te se novi proizvod dodaje u bazu podataka.

```python
def addProductView(request, brandId=None):
    context = {}
    isBranded = False
    if not request.user.is_superuser:
        return redirect('main:index')
    if brandId:
        brand = get_object_or_404(Brand, id=brandId)
        isBranded = True
        form = CreateProductForm(request.POST or None, request.FILES or None, initial = { 'brand': brand, 'isBranded': True })
    else:
        form = CreateProductForm(request.POST or None, request.FILES or None, initial = { 'isBranded': False })
    if form.is_valid():
        obj = form.save(commit=False)
        obj.isBranded = True if brandId else False
        obj.save()
        productId = obj.id
        form = CreateProductForm()
        return redirect('main:productDetail', productId=productId)
    context = {
        'form': form,
        'isBranded': isBranded,
    }
    return render(request, 'main/addProduct.html', context)
```

**Slika 4.28.** *Metoda za dodavanje novog proizvoda*

Admin postojeće proizvode može i uređivati. Za to također postoji forma koja se nalazi u datoteci *forms.py* i jako je slična formi za stvaranje novog proizvoda. Metoda za uređivanje postojećeg proizvoda poziva formu za uređivanje te joj predaje podatke koje proizvod koji želimo urediti već ima pa admin može izmijeniti samo neke podatke, a ostali mogu ostati isti. Prikaz metode za uređivanje proizvoda je na slici 4.29.

```python
def editProductView(request,productId):
    context = {}
    product = get_object_or_404(Product, id=productId)
    if not request.user.is_superuser:
        return redirect('main:index')
    if request.POST:
        form = EditProductForm(request.POST or None, request.FILES or None, instance=product)
        if form.is_valid():
            obj = form.save()
            context['success_message'] = 'Product updated!'
            product = obj
    form = EditProductForm(
        initial = {
            'title': product.title,
            'quantity':product.quantity,
            'collection':product.collection,
            'brand': product.brand,
            'type': product.type,
            'size': product.size,
            'price' : product.price,
            'flag': product.flag,
            'front_image': product.front_image,
            'back_image': product.back_image,
        }
    )
    context = {
        'form': form,
        'product': product,
    }
    return render(request, 'main/editProduct.html', context)
```

**Slika 4.29.** *Metoda za uređivanje proizvoda*

Admin također može i brisati proizvode ako tako želi. Metoda za brisanje proizvoda provjerava radi li se o *POST* metodi te o *AJAX* zahtjevu, ako da, onda dohvaća *id* proizvoda kojeg treba obrisati i jednostavnom narednom *product.delete()* proizvod se briše. Metoda je vidljiva na slici 4.30. dok je na slici 4.31. prikazana funkcija za brisanje proizvoda, gdje *ajax* poziva *deleteProductView* metodu te predaje *id* proizvoda kao podatak koji metoda koristi kako bi obrisao traženi proizvod.

```python
def deleteProductView (request):
    if request.method == "POST" and isAjax(request):
        productId = request.POST.get('id', None)
        try:
            product = get_object_or_404(Product, id=productId)
            product.delete()
            return HttpResponse(json.dumps({ "good": True }), content_type="application/json")
        except Account.DoesNotExist:
            return HttpResponse(json.dumps({ "good": False }), content_type="application/json")
```

**Slika 4.30.** *Metoda za brisanje proizvoda*

```javascript
function deleteProduct(data) {
    $.ajax({
        url: "{% url 'main:deleteProduct' %}",
        type: "POST",
        data: {
            "id": data,
            csrfmiddlewaretoken: '{{ csrf_token }}',
        },
        cache: false,
        dataType: "json",
        success: function () {
            table.row("#productRow-{{ product.id }}").remove().draw();
            table.draw();
        }
    });
}
```

**Slika 4.31.** *Funkcija za brisanje proizvoda*

Na isti način se obavlja i dohvaćanje, uređivanje i brisanje drugih podataka koji su omogućeni adminu.

Admin također ima pristup analitici korisnika, gdje se adminu u tablici prikazuje ukupna potrošnja svakog korisnika. Kôd koji prikazuje metodu koja dohvaća račun za analitiku se nalazi na slici 4.32. a metoda koja dohvaća ukupan iznos koji je korisnik potrošio se nalazi na slici 4.33.

```python
def analyticsView(request):
    context = {}
    if not request.user.is_superuser:
        return redirect('main:index')
    user = get_object_or_404(Account, id=request.user.id)

    if not user.is_authenticated:
        return redirect('main:index')
    receipts = Receipt.objects.distinct('account').all()
    context['receipts'] = receipts
    context['user'] = user
    return render(request,"account/analytics.html",context)
```

**Slika 4.32.** *Prikaz metode koja dohvaća račun za analitiku*

```python
def getTotalSpent(self):
    receipts = Receipt.objects.filter(account=self.account).all()
    price = 0
    for receipt in receipts:
        price += receipt.totalPrice
    return price

def getTotalSpentEUR(self):
    return round((self.getTotalSpent() / float(TO_EUR)), 2)
```

**Slika 4.33.** *Prikaz kôda koji dohvaća ukupnu potrošnju svakog korisnika*

# 5. KORIŠTENJE APLIKACIJE

Za uspješno pokretanje aplikacije potrebno je preuzeti kôd, instalirati potrebne ekstenzije koje se nalaze u datoteci requirements.txt i pokrenuti server. Naredba pomoću koje se instaliraju sve potrebne ekstenzije se nalazi na slici 5.1.

```
$ pip install -r requirements.txt
```

**Slika 5.1.** *Naredba za instaliranje potrebnih ekstenzija*

Nakon uspješne instalacije, potrebno je aktivirati virtualno okruženje pomoću naredbe koja je vidljiva na slici 5.2.

```
$ source django/Scripts/activate
```

**Slika 5.2.** *Naredba za aktiviranje virtualnog okruženja*

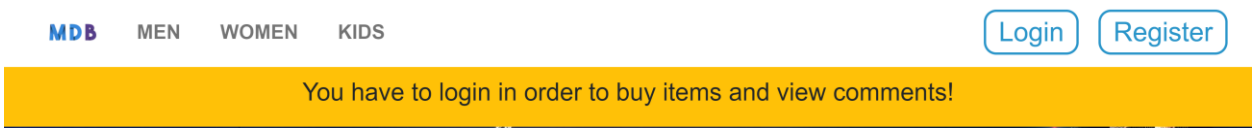
Nakon toga, potrebno je pokrenuti server naredbom koja se nalazi na slici 5.3. Server će se pokrenuti te će stranica biti dostupna na *http://localhost:8000.*

```
$ python manage.py runserver
```

**Slika 5.3.** *Naredba za pokretanje servera*

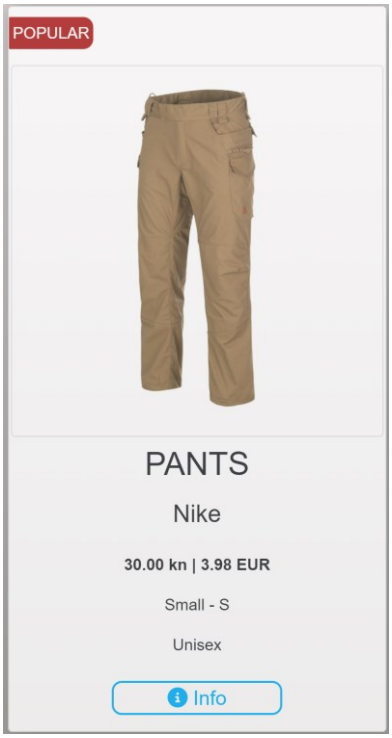
## 5.1. Neregistrirani korisnik

Neregistrirani korisnik nema velika prava u usporedbi sa registriranim korisnicima ili administratorima. Ulaskom na stranicu neregistrirani korisnici vide poruku koja im govori da ne mogu kupovati artikle i vidjeti komentare. Izgled navbara i poruke se nalazi na slici 5.4.



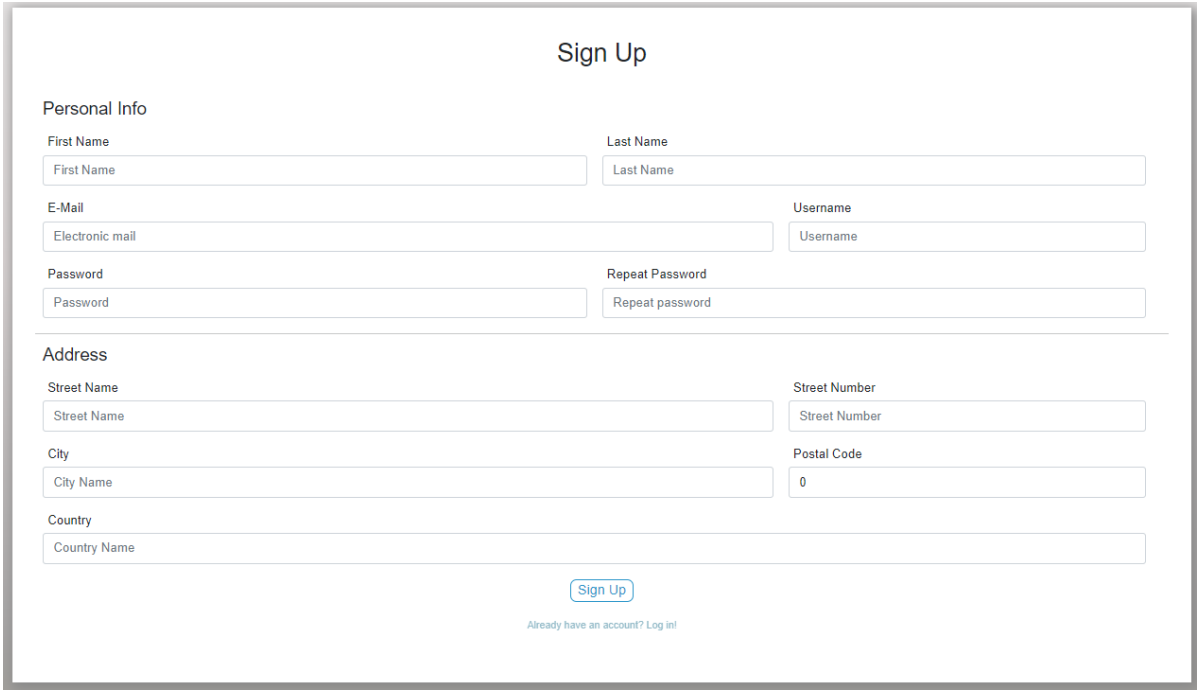

**Slika 5.4.** *Navbar i poruka za neregistrirane korisnike*

Osim toga, na početnoj stranici na kojoj su izlistani proizvodi, neregistrirani korisnici neće imati gumb za dodavanje artikla u košaricu, nego samo gumb za detaljan prikaz artikla. To je vidljivo na slici 5.5.



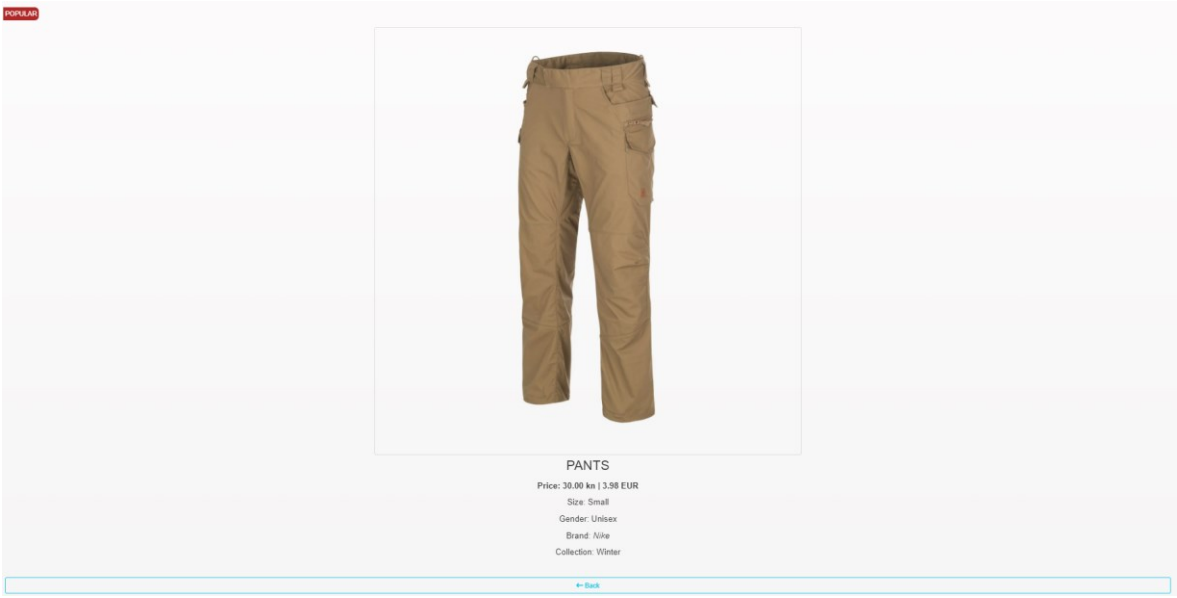

**Slika 5.5.** *Prikaz kartice artikla za neregistrirane korisnike*

U navbaru se nalazi gumb za registraciju koji neregistrirani korisnici mogu kliknuti kako bi se registrirali. Klikom na njega, otvara se nova stranica koja predstavlja formu za registraciju. Izgled stranice za registraciju je na slici 5.6. Nakon unošenja svih podataka, klikom na gumb *Sign Up* korisnik se registrira i preusmjerava se na početnu stranicu.




**Slika 5.6.** *Prikaz kartice artikla za neregistrirane korisnike*

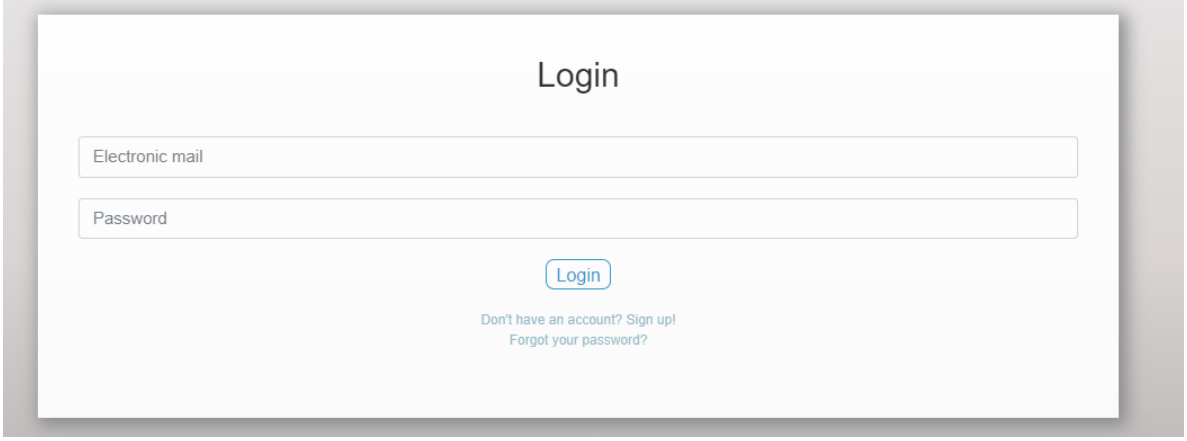
Klikom na gumb *Info* koji se nalazi na svakom proizvodu, korisnika gosta će se preusmjeriti na detaljan prikaz odabranog proizvoda ali korisnik i dalje neće moći dodati proizvod u košaricu, niti vidjeti komentare drugih za taj proizvod. Slika 5.7. prikazuje detaljan prikaz jednog proizvoda.




**Slika 5.7.** *Detaljan prikaz artikla za neregistrirane korisnike*

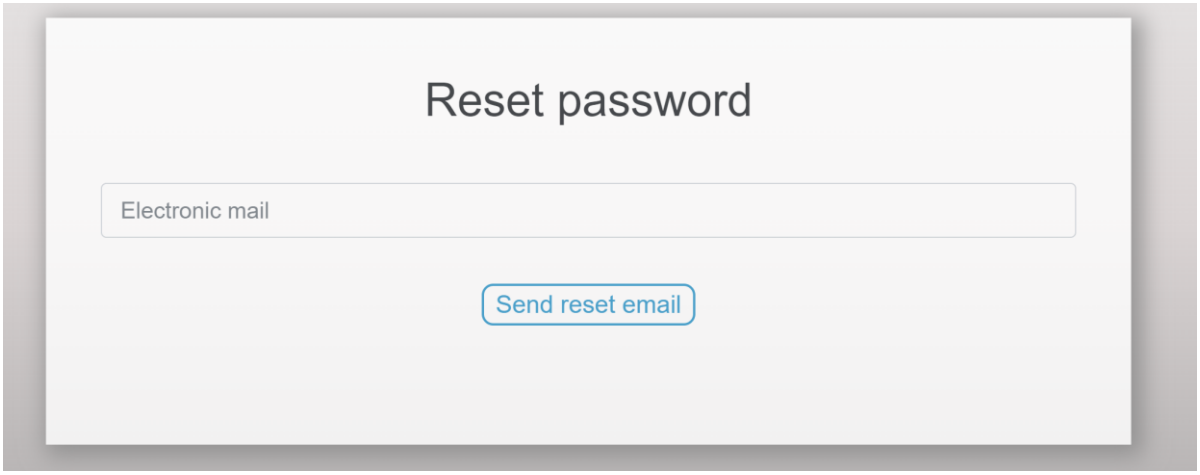
## 5.2. Registrirani korisnik

Ukoliko je korisnik registriran i želi se prijaviti, u navbaru se nalazi gumb *Login.* Klikom na taj gumb korisnik se vodi na formu za prijavu. Izgled stranice za prijavu je vidljiv na slici 5.8.

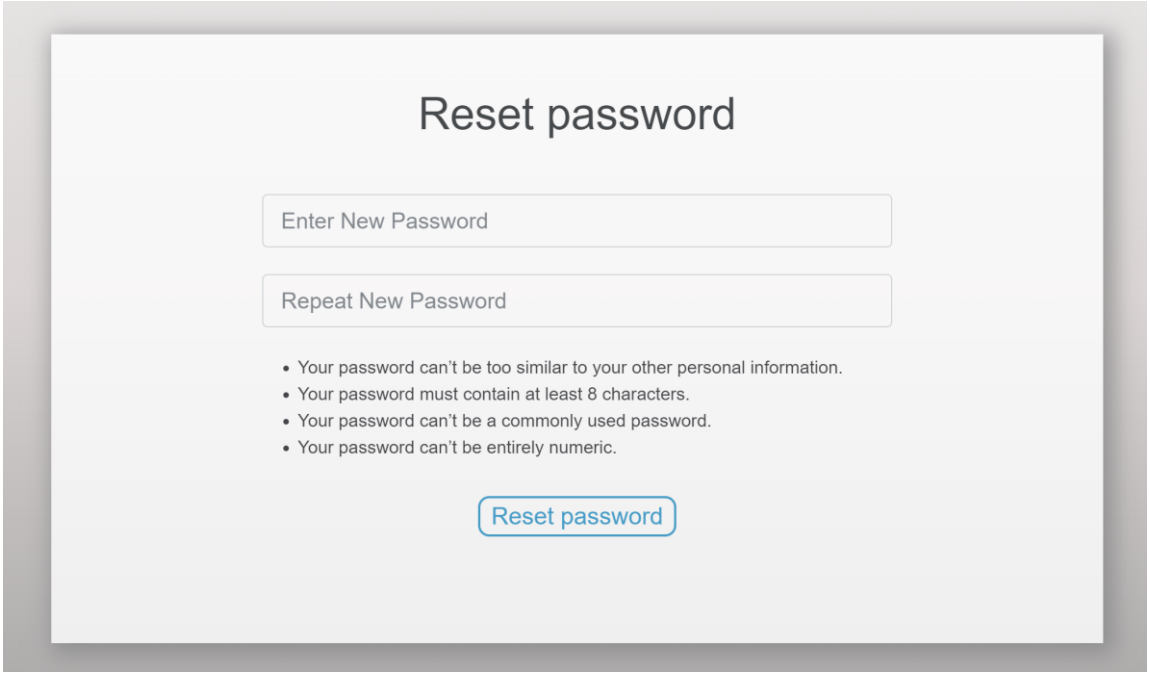



**Slika 5.8.** *Prikaz kartice za prijavu korisnika*

Ukoliko je korisnik zaboravio lozinku, klikom na *Forgot your password?* Vodi ga se na formu za vraćanje lozinke. U formi na slici 5.9. korisnik unosi svoj email te mu se na taj email šalje link pomoću kojeg unosi novu lozinku. Slika 5.10. prikazuje formu za unos nove lozinke.

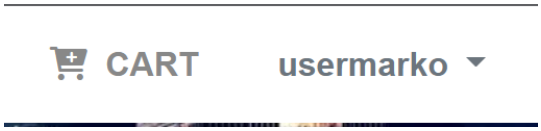



**Slika 5.9.** *Prikaz forme za vraćanje lozinke*




**Slika 5.10.** *Prikaz forme za unos nove lozinke*

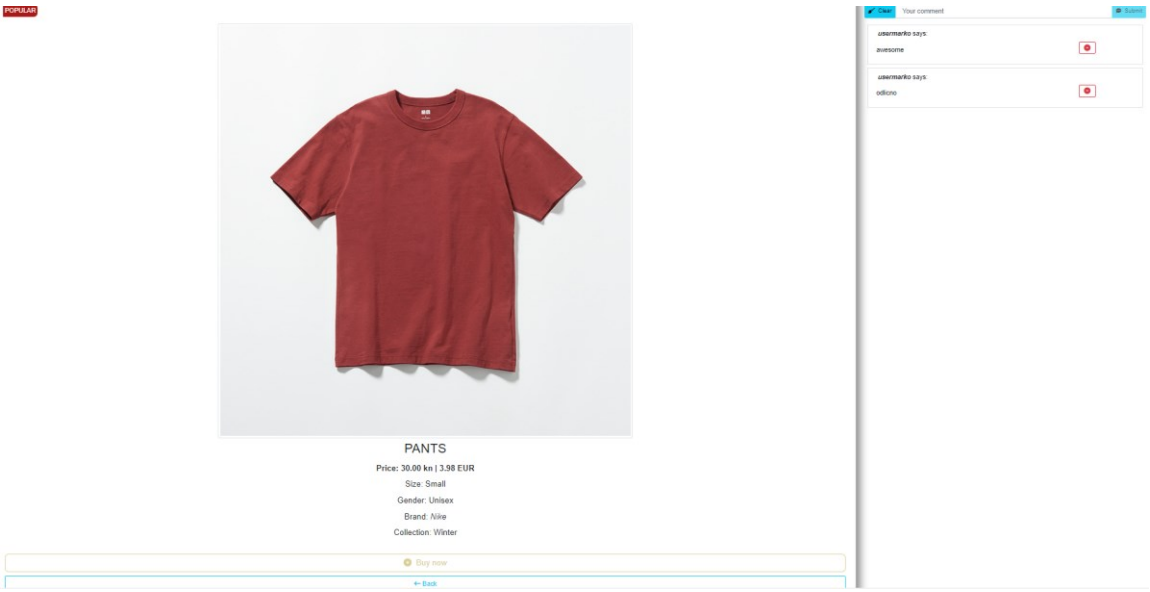
Nakon uspješne prijave, korisnik je preusmjeren na početnu stranicu gdje mu je, za razliku od neregistriranog korisnika, dostupan gumb za dodavanje artikla u košaricu. Izgled kartice proizvoda za prijavljenog korisnika je vidljiva na slici 5.12. Također, nadimak registriranog korisnika će se pojaviti u navbaru kao i košarica. Navbar se nalazi na slici 5.11.




**Slika 5.11.** *Prikaz dijela navbara za prijavljenog korisnika*

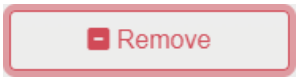

**Slika 5.12.** *Prikaz kartice artikla za prijavljenog korisnika*

Prijavljeni korisnik također ima opciju vidjeti detaljan opis artikla klikom na gumb *Info*. Izgled detaljne stranice odabranog artikla se nalazi na slici 5.13. Na toj stranici će prijavljenom korisniku biti omogućene dodatne funkcionalnosti poput čitanja postojećih komentara za taj proizvod, dodavanja ili brisanja vlastitog komentara.

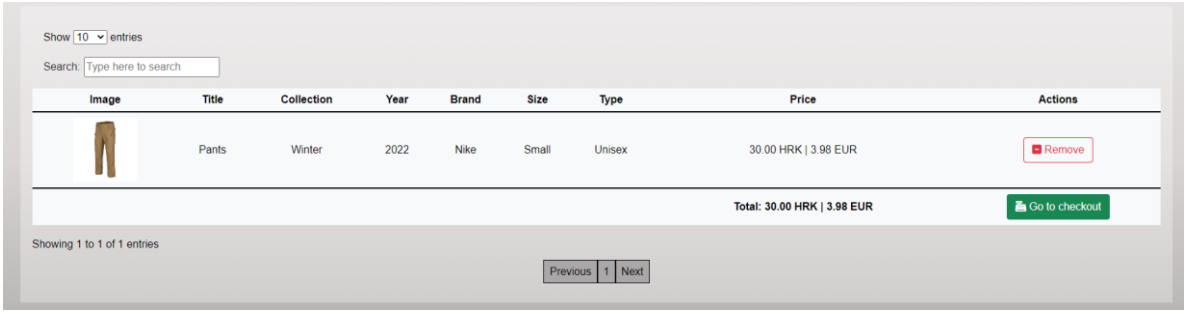



**Slika 5.13.** *Detaljan prikaz za prijavljenog korisnika*

Klikom na gumb za kupnju, taj isti gumb mijenja boju i tekst, a artikl je dodan u košaricu. Izgled gumba nakon dodavanja artikla se nalazi na slici 5.14. a košarice na slici 5.15.

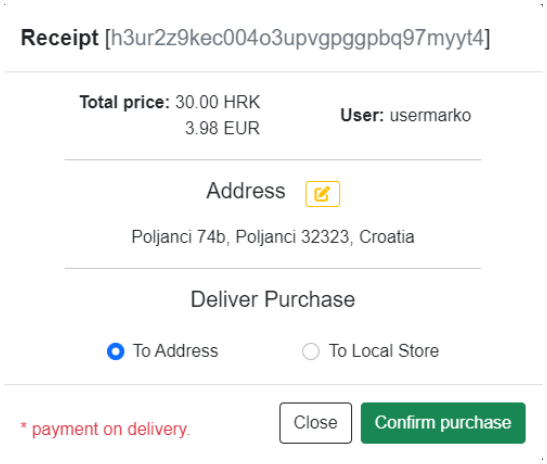



**Slika 5.14.** *Gumb za kupnju nakon dodavanja u košaricu*

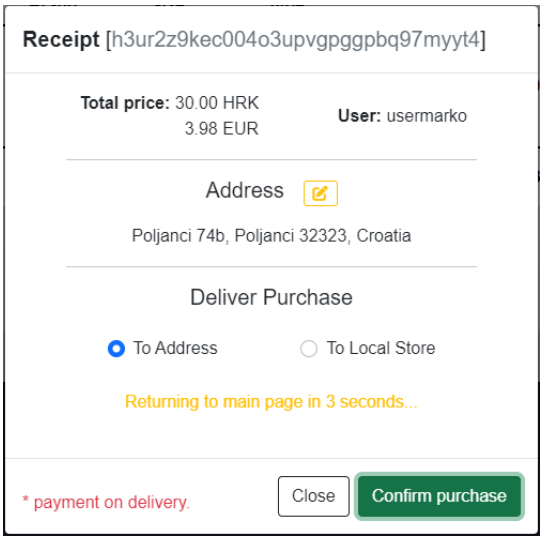



**Slika 5.15.** *Izgled košarice nakon dodavanja proizvoda*

Ukoliko korisnik želi potvrditi kupnju to može napraviti klikom na gumb *Go to checkout* nakon čega se otvara modal koji prikazuje najbitnije podatke o korisniku i košarici. Izgled modala je vidljiv na slici 5.16. Korisnik može birati želi ti dostavu kući ili će proizvode preuzeti u trgovini. Za potvrdu kupnje korisnik mora kliknuti na gumb *Confirm purchase* nakon čega se pojavljuje poruke o uspješnoj kupnji i preusmjeravanju na početnu stranicu. Prikaz toga je vidljiv na slici 5.17.

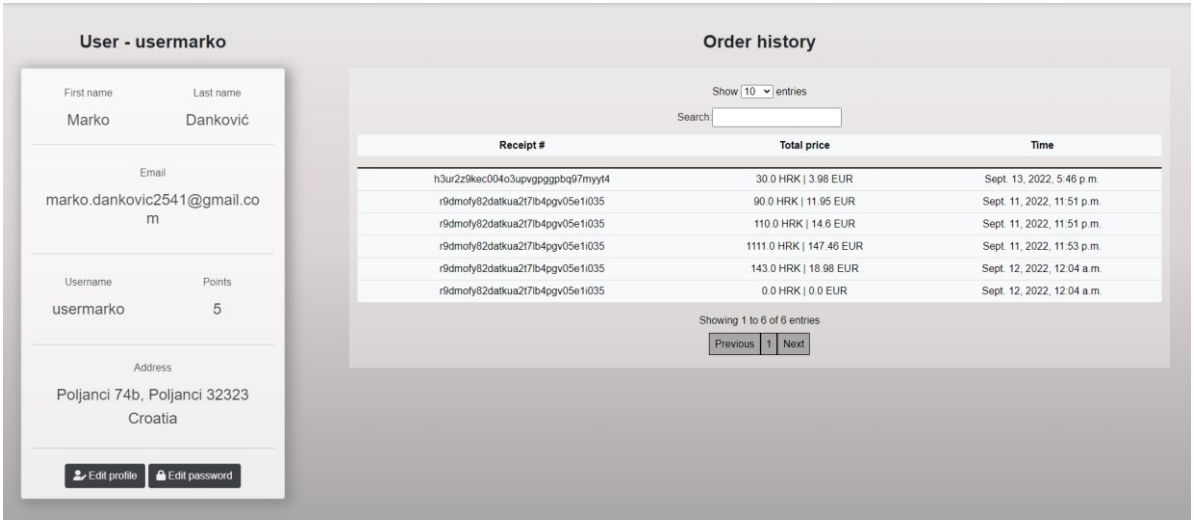



**Slika 5.16.** *Izgled košarice nakon dodavanja proizvoda*



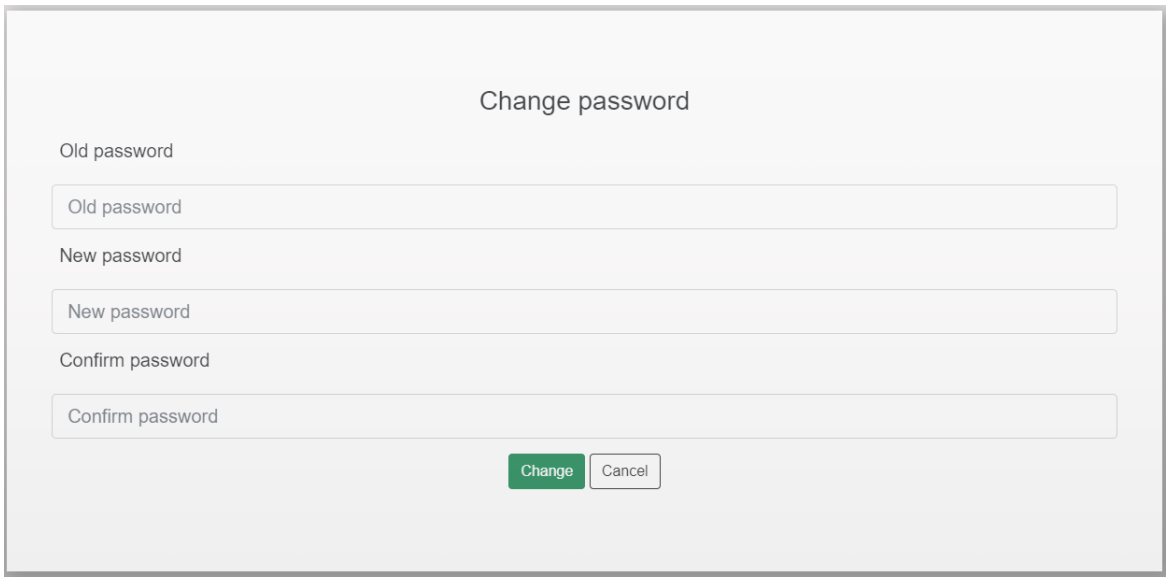

**Slika 5.17.** *Izgled košarice nakon potvrde kupnje*

Svaki registrirani korisnik ima mogućnost uređivanja vlastitog profila te uvida u povijest vlastitih narudžbi. Izgled stranice profila korisnika je vidljiva na slici 5.18.




**Slika 5.18.** *Izgled stranice profila korisnika*

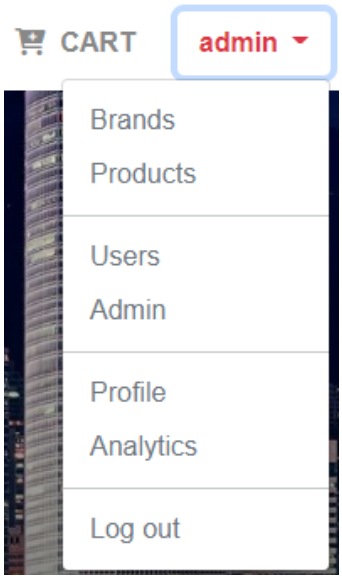
Također korisnik ima mogućnost promjene lozinke klikom na gumb *edit password* nakon čega se otvara forma za mijenjanje lozinke. Prikaz forme za promjenu lozinke se nalazi na slici 5.19.




**Slika 5.19.** *Prikaz forme za promjenu lozinke*

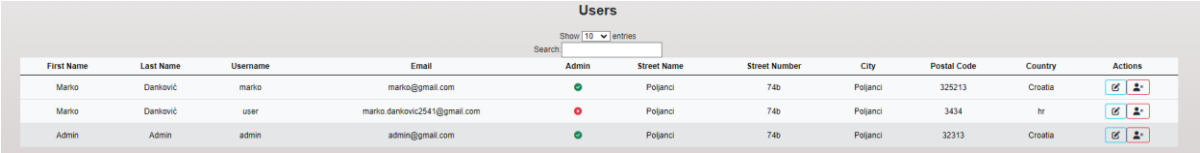
## 5.3. Korisnik administrator

Korisnik administrator ili kraće admin, ima najveća prava od svih korisnika. On upravlja proizvodima, korisnicima, komentarima i markama. Ukoliko se radi o adminu, nakon prijave će njegovo ime u navbaru biti crvene boje te će imati dodatne mogućnosti. Slika 5.20. to pokazuje.



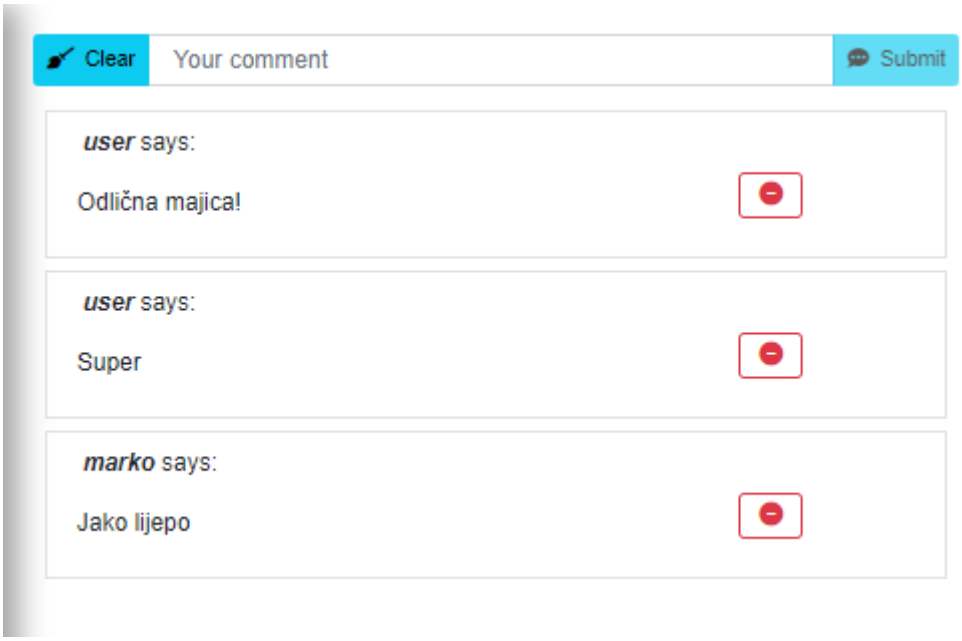

**Slika 5.20.** *Izgled navrbar-a nakon prijave administratora*

Admin ima pristup svim podacima, pa ih može mijenjati, dodavati i brisati. Prikaz tablice korisnika koju admin vidi nalazi se na slici 5.21. S desne strane se nalaze gumbi za uređivanje ili brisanje korisnika.




**Slika 5.21.** *Prikaz svih registriranih korisnika koje admin vidi*

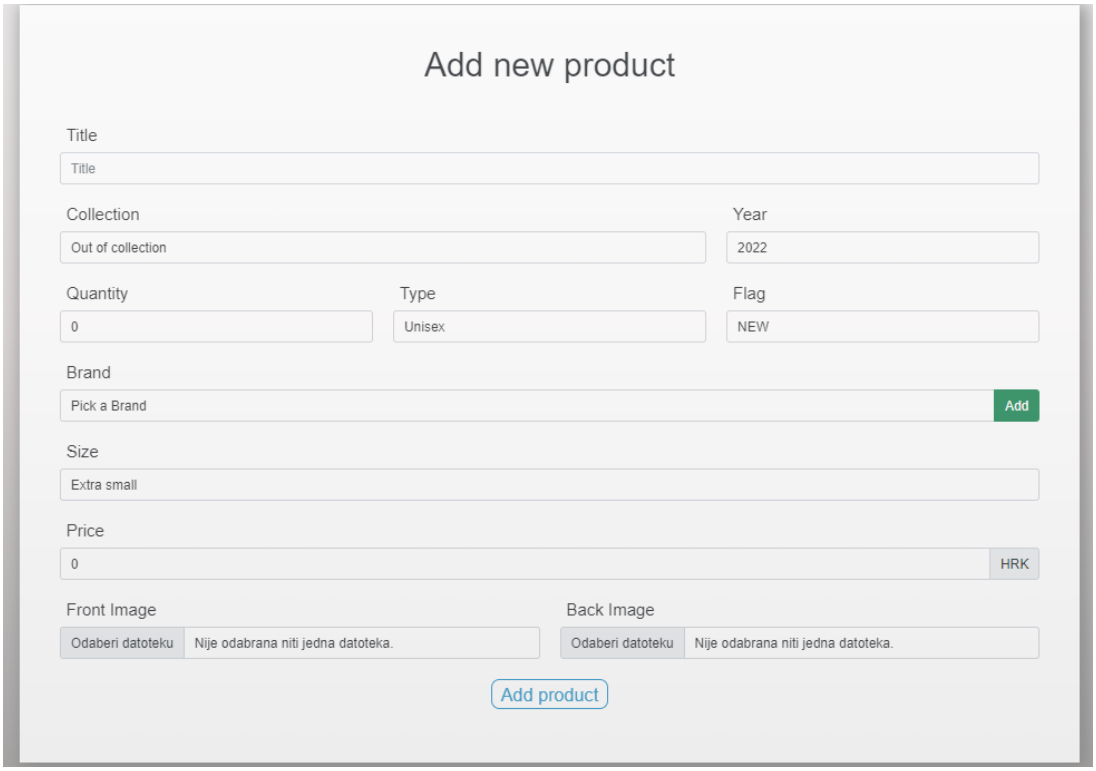
Isto vrijedi i za ostale podatke. Još jedan dodatna mogućnost koju admini imaju je brisanje svih komentara o nekom proizvodu. Prikaz izgleda komentara za admina je na slici 5.22.



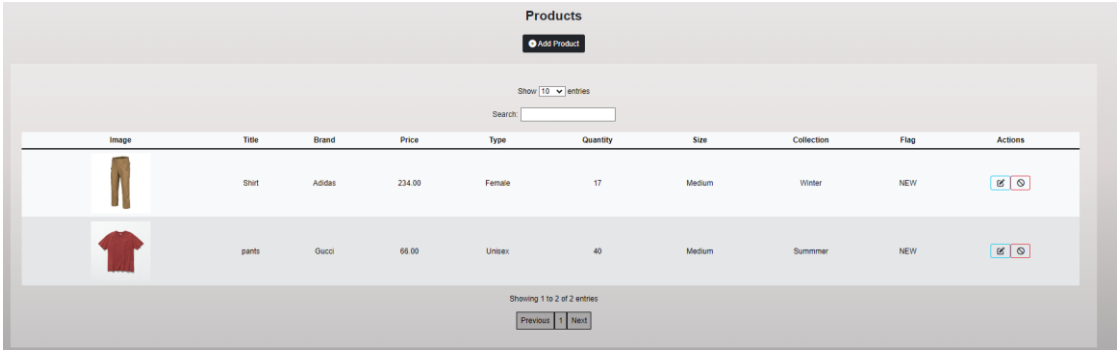

**Slika 5.22.** *Prikaz komentara kako ih admin vidi*

Osim toga, admini mogu i dodavati nove proizvode. Za to je potrebno odabrati *products* u izborniku u navbaru. Na vrhu stranice se nalazi gumb *Add product.* Klikom na njega, admin se preusmjerava na novu stranicu za unos podataka o proizvodu. Izgled stranice za dodavanje novog proizvoda se nalazi na slici 5.23. dok se proizvodi prikazuju kako na slici 5.24.

Ukoliko admin želi izmijeniti postojeći proizvod, to može napraviti odlaskom na *products* stranicu i klikom na gumb koji se nalazi na slici 5.25. biti će odveden na formu za uređivanje artikla koja je prikazana na slici 5.26.

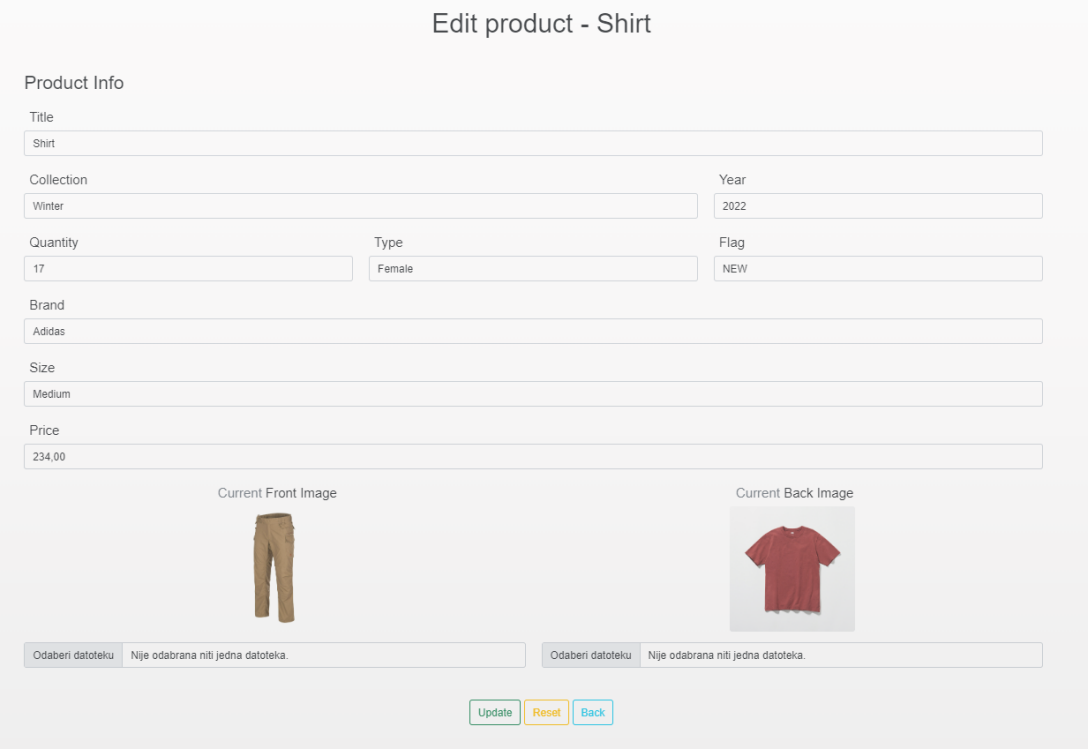



**Slika 5.23.** *Prikaz forme za dodavanje proizvoda*



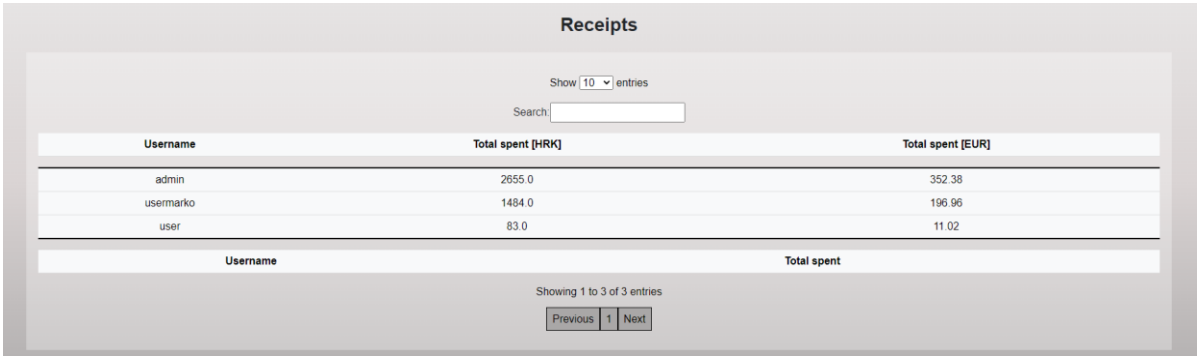

**Slika 5.24.** *Prikaz svih proizvoda kako ih admin vidi*

**Slika 5.25.** *Prikaz gumba uređivanje proizvoda*




**Slika 5.26.** *Prikaz forme za uređivanje postojećeg proizvoda*

Također, admini imaju pristup analitici korisnika gdje se u tablici prikazuje ukupna potrošnja za svakog korisnika. Prikaz te tablice nalazi se na slici 5.27.




**Slika 5.27.** *Prikaz analitike korisnika*

# 6. ZAKLJUČAK

Ovim završnim radom napravljena je web trgovina za prodaju odjeće. Sama ideja izrade ovakve aplikacije je došla za vrijeme pandemije kada su svi bili zatvoreni u kući. Postoje tri moguća korisnika, korisnik-administrator, korisnik-gost te registrirani korisnik. Administrator ima najveća prava, uvid u sve registrirane korisnike, proizvode i marke. Ima pravo dodavati i brisati nove proizvode te uklanjati korisnike. Registrirani korisnik je običan korisnik koji ima pravo kupovati proizvode i postavljati vlastite komentare za određene proizvode. Također, registrirani korisnici prilikom kupnje veće od 100HRK sakupljaju bodove koji se kasnije koriste kako bi im se omogućila pogodnost popusta pri kupnji. Korisnik gost ima najmanja prava, ne može kupovati artikle ili komentirati već samo vidjeti koji se proizvodi prodaju Ukoliko se odluči za kupnju proizvoda morati će se registrirati. Trenutna verzija web stranice omogućuje prodaju samo registriranim korisnicima i ne nudi prevelik izbor načina plaćanja. Također, korisnici mogu dodati samo 1 istovrsni proizvod. Postoji mogućnost unapređenja web stranice dodavanjem više načina plaćanja, davanja gostima mogućnost kupnje, stvaranja kodova za popuste i omogućavanja dodavanja više istovrsnih proizvoda.

# LITERATURA


[1] https://corporate.aboutyou.de/en/about-us 12.6.2022.

[2] https://corporate.zalando.com/en/about-us/who-we-are 12.6.2022.

[3] https://answear.hr/a/o-answear-hr 12.6.2022.

[4] https://ipsecsystems.com/web-application-development-services 13.6.2022.

[5] https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/c201_polaznik.pdf 13.6.2022.

[6] http://www.webtech.com.hr/html.php 13.6.2022.

[7] https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/c220_polaznik.pdf 14.6.2022.

[8]https://www.mojwebdizajn.net/edukacija/web-dizajn/prezentacije/uvod-u-css.php#slide-3 4.6.2022.

[9] https://www.webtech.com.hr/css.php 14.6.2022.

[10] https://tesla.carnet.hr/mod/book/view.php?id=5397&chapterid=835 16.6.2022.

[11] https://www.w3schools.com/bootstrap/bootstrap_get_started.asp 26.6.2022.

[12] https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript 26.6.2022.

[13] https://skillcrush.com/blog/javascript/ 27.6.2022.

[14]https://www.srce.unizg.hr/files/srce/docs/edu/osnovni-tecajevi/c501_polaznik.pdf 8.6.2022.

[15] https://www.w3schools.com/js/js_whereto.asp 28.6.2022.

[16] https://en.wikipedia.org/wiki/Ajax_(programming) 28.6.2022.

[17] https://stackify.com/return-ajax-response-asynchronous-javascript-call/ 1.7.2022.

[18]https://www.researchgate.net/figure/The-comparison-between-a-classic-web-application-model-and-an-Ajax-web-application-model_fig3_241415122 1.7.2022.

[19] https://hr.wikipedia.org/wiki/Python_(programski_jezik) 31.8.2022.

[20] https://www.postgresql.org/about/ 13.9.2022.

[21] https://www.educative.io/edpresso/what-is-visual-studio-code 12.6.2022.

# SAŽETAK

Cilj ovog završnog rada je izraditi web aplikaciju za prodaju odjevnih predmeta. Na početku su opisane tehnologije koje su korištene za izradu ove web aplikacije. Pokretanje aplikacije je opisano u samom radu. Postoje tri uloge korisnika, korisnik-gost, registrirani korisnik i korisnik-administrator. Administrator ima uvid u razne stvari poput popisa svih proizvoda, svih korisnika, brendova te analitike korisnika te ima prava dodavanja, uređivanja i brisanja istih. Korisnik gost nema prava kupovine i komentiranja već samo uvida u proizvode. Registrirani korisnik ima pravo kupovine proizvoda, komentiranja te ima omogućenu pogodnost sakupljanja bodova koji se koriste ta popust.

**Ključne riječi:** baza podataka, django, odjeća, web trgovina

# ABSTRACT

**Title:** Web application for selling clothes

The goal of this thesis is to create a web application for selling clothes. At the beginning of this work, the technologies used to create this web application were described. Starting the applications is described in the paper itself. There are three user roles, guest user, registered user and administrator user. The administrator has insight into various things such as the list of all products, all users, brands and user analytics and has the right to add, edit and delete them. The guest user does not have the right to buy or comment, but only to view the products. The registered user has the right to buy products, comment and has the convenience of collecting points that can be used for discounts.

Keywords: clothing, database, django, web store, web application,