Dizajn i razvoj Internet trgovine

Grubač, Karlo

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Karlo Grubač

DIZAJN I RAZVOJ INTERNET TRGOVINE

ZAVRŠNI RAD

Varaždin, 2019.

SVEUČILIŠTE U ZAGREBU

FAKULTET ORGANIZACIJE I INFORMATIKE

V A R A Ž D I N

Karlo Grubač

Matični broj: 35918/07–R

Studij: Primjena informacijske tehnologije u poslovanju

DIZAJN I RAZVOJ INTERNET TRGOVINE

ZAVRŠNI RAD

Mentor/Mentorica:

Prof. dr. sc. Dragutin Kermek

Varaždin, srpanj 2019.

Karlo Grubač

Izjava o izvornosti

Izjavljujem da je moj završni rad izvorni rezultat mojeg rada te da se u izradi istoga nisam koristio drugim izvorima osim onima koji su u njemu navedeni. Za izradu rada su korištene etički prikladne i prihvatljive metode i tehnike rada.

Autor/Autorica potvrdio/potvrdila prihvaćanjem odredbi u sustavu FOI-radovi

Sažetak

Tema rada je dizajn i razvoj internet trgovine, sastoji se od uvodnog dijela i četiri velike cjeline. U prvom dijelu se nalazi uvod u SQL, dizajn, postupci kreiranja i model baze podataka kako bi se dobila slika pozadine aplikacije i lakše razumjela njena funkcionalnost. Zatim slijede tehnologije koje su korištene tijekom izrade, za strukturu aplikacije koristi se HTML, funkcionalnost omogućuje programski jezik PHP, dizajn stranice napravljen je s CSS-om i na kraju dolazi JavaScript koji daje dinamiku. Slijede ekranski prikazi Internet trgovine i dijelovi koda koji se nalaze u pozadini detaljno objasnjeni. Rad je koncipiran na način da polazi od pozadine sustava i ide prema površini i samom dizajnu. Na kraju se nalazi popis literature i slika.

1. Sadržaj

2. Uvod

U današnjem svijetu putem interneta može se kupiti gotovo sve. Svaka ozbiljnija tvrtka koja se bavi djelatnošću vezanom uz prodaju posjeduje internet trgovinu (eng. *Web shop*). Putem interneta se može kupiti gotovo sve što tvrtke nude u svojim poslovnim objektima, počevši od namještaja, odjeće, informatičke opreme pa sve do prehrambenih namjernica.

Internet trgovina se sastoji od vizualnog dijela, odnosno samog sučelja web aplikacije i njezinih formi. U pozadini se nalaze poslovni procesi koji omogućuju realizaciju zahtjeva korisnika. Internet trgovina kao i svaka internet aplikacija u pozadini mora imati dobro modeliranu bazu podataka. Korištenjem HTML-a i CSS-a izrađuju se sučelja, odnosno ekrani koji se prikazuju korisniku. Kada imamo bazu podataka u pozadini, HTML i CSS na površini potrebno je omogućiti njihovu komunikaciju i usklađen rad, to se realizira nekim od programskih jezika što će u ovom slučaju biti PHP. Ukoliko se želi napraviti projekt dinamičnijim koristiti će se *JavaScript* tehnologiju.

Tema internet trgovine u ovome radu biti će prodaja izvornih Hrvatskih gastronomskih proizvoda. Prolaziti će se kroz sve faze izgradnje i dizajna internet trgovine, detaljno će biti objašnjenje sve tehnologije koje su korištene i prikazati će se programska rješenja nekih složenijih poslovnih procesa uz napomene koja mogu biti alternativna rješenja. Uz sam tehnički dio izrade projekta pojasniti će se funkcionalnost, poslovni procesi i mogućnosti koje internet trgovina nudi korisnicima i administratorima.

3. Baza podataka

„Pod pojmom baze podataka danas se podrazumijeva kompjutorizirana baza podataka. Podaci su smješteni na disku u obliku koji nije razumljiv krajnjem korisniku te se za potrebe korištenja i rada s tim podacima koristi neki sustav za upravljanje bazama podataka. Može se slobodno reći da baza podataka nije ništa drugo nego skup povezanih, organiziranih podataka te da je korisnici u pravilu doživljavaju kao skup tablica.“ [1, str. 1].

U ovom poglavlju biti će objašnjene tehnologije, programski jezik i ostali alati kao i procesi izgradnje baze podataka.

3.1. Structured Query Language (SQL)

Strukturirani upitni jezik (eng. *Structured Query Language - SQL*) je standardizirani i najkorišteniji jezik za rad s bazama podataka koji omogućuje kreiranje baza podataka, manipuliranje podacima te dohvaćanje podataka iz baze. Prilikom korištenja SQL naredbi potrebno je poštivati određena pravila. Sintaksa predstavlja ispravan odabir riječi u SQL naredbi. SQL naredbe se najčešće koriste za dohvaćanje, umetanje, ažuriranje i brisanje podataka.[1]

SELECT je naredba za dohvaćanje podataka s obzirom na unijete parametre. Na slijedećem primjeru može se vidjeti naredba koja dohvaća ime i prezime korisnika iz tablice korisnici, pod uvjetom da je korisnik muškarac. [1]

```
SELECT ime, prezime FROM korisnici WHERE spol = 'M'
```

Nakon što se tablica kreira ona je potpuno prazna te je potrebno unijeti podatke. Za unošenje podataka u tablicu koristi se *INSERT* naredba. [1] Slijedeći primjer rezultirati će unošenjem novog retka u tablicu *korisnici*.

```
INSERT INTO korisnici (ime, prezime, spol) VALUES ('Pero', 'Perić', 'M')
```

Realno je očekivati da se prilikom upisivanja redova nije znala vrijednost nekog stupca ili je moguće da se vrijednost nekog stupca promijenila. Pod pojmom ažuriranja smatra se modificiranje podataka koji već postoje u bazi podataka. Slijedeći primjer prikazuje ažuriranje samo jednog polja u retku. [1]

```
UPDATE korisnici SET ime = 'Petar' WHERE prezime = 'Perić'
```

Jedna od karakteristika baza podataka je sposobnost pohrane velike količine podataka. Ponekad baza podataka može rasti prebrzo te je potrebno obrisati određene podatke iz same baze. Naredba *DELETE* omogućuje brisanje redova iz tablice. Slijedeći primjer rezultirati će brisanjem jednog retka u tablici. [1]

```
DELETE FROM korisnici WHERE prezime = 'Perić'
```

Pogledaju li se prethodno kreirani upiti, može se vidjeti kako se rijetko u odgovoru želi dohvatiti baš sve redove iz tablice. Kako razvojni programeri nemaju vremena sami tražiti redove koji zadovoljavaju neki uvjet, za filtriranje redova koristi se klauzula *WHERE*. Redovi koje se dobiju u odgovoru u pravilu nisu ispisani prema nekom unaprijed zadanom poretku, obično se ispisuju u onom slijedu u kojem su bili upisivani. Ako se dobiveni rezultat želi sortirati koristi se klauzula *ORDER BY*. Kod dohvaćanja podataka često se javlja potreba za grupiranjem podataka, to je postupak dijeljenja podataka u logičke grupe nad kojima se obično provode određeni izračuni. Grupe se kreiraju korištenjem klauzule *GROUP BY*. [1]

3.2. Dizajn baze podataka

„Dizajniranje baze podataka je tema kojoj je posvećen veliki broj knjiga što dovoljno govori o njenoj važnosti. Prilikom dizajniranja baze podataka prvo je potrebno dobro razmisliti koje podatke bi trebalo spremati u bazu. Kako bi se prilikom dizajniranja baze podataka obuhvatio što veći broj podataka koji će biti potrebni, vrlo je važno poznavati detalje svih poslovnih procesa koji se namjeravaju implementirati. Loše dizajnirana baza podataka rezultira određenim problemima kao što je postojanje redundantnih podataka te se manifestiraju anomalije dodavanja, brisanja i mijenjanja. Anomalija brisanja ne dozvoljava brisanje podataka iz baze ako se pritom ne obriše i neke druge povezane podatke. Anomalija dodavanja je suprotna anomaliji brisanja utoliko što ne dozvoljava upisivanje podataka, ako se u isto vrijeme ne upisuje i neke druge povezane podatke. Anomalija mijenjanja označava situaciju kad se isti podatak mora mijenjati na više mjesta.“ [1, str. 13]

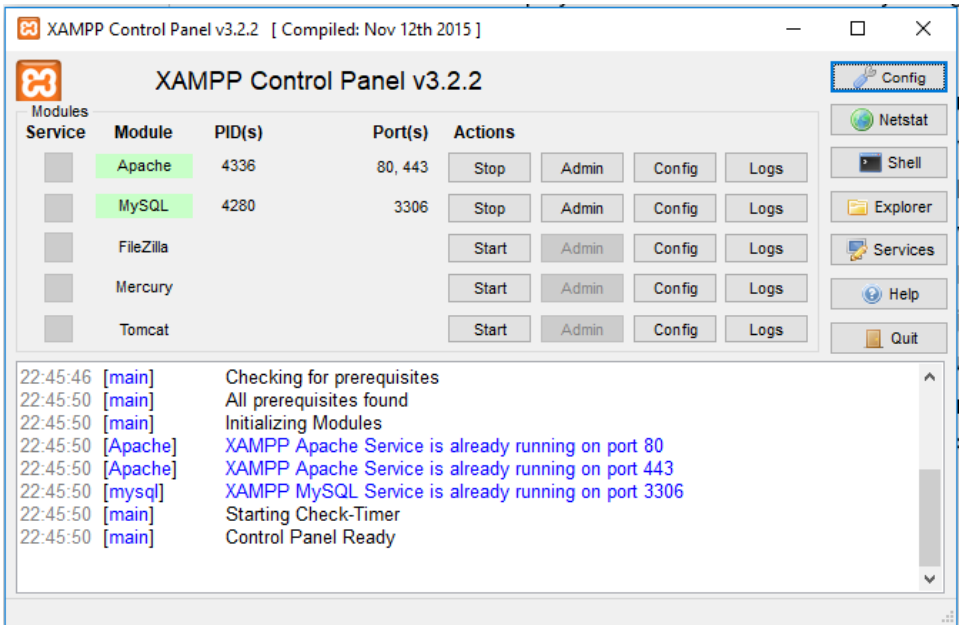
„Poželjno svojstvo baze podataka je da su podaci sadržani u bazi konzistentni i korektni, odnosno ispravni i točni. Nekorektni podaci mogu prouzrokovati niz problema. Pod pojmom konzistentnosti podrazumijeva se da podaci koji se unose u bazu zadovoljavaju

određena ograničenja, odnosno da su u skladu s definiranim ograničenjima. Unos ne konzistentnih podataka u bazu može se spriječiti na dva načina, postavljanjem ograničenja za unos nad poljima u bazi podataka ili postavljanjem ograničenja na formama za unos podataka.“ [1, str. 25]

3.3. Kreiranje baze podataka

Nakon što su definirani poslovni procesi i na temelju toga određeno koji su podaci potrebni, potrebno je kreirati bazu podataka. Baza podataka može se kreirati na mnoštvo raznih načina i korištenjem različitih alata. Jedan od najpoznatijih alata je Microsoft Access. To je program za upravljanje relacijskim bazama podataka koji nudi mnoštvo formi i sučelja koji pojednostavljuju čitav proces. Drugi način je korištenje alata *phpMyAdmin*, koji je ujedno i korišten u izradi projekta. Navedeni alat korišten je zbog svoje kompatibilnosti s programskim jezikom PHP.

PhpMyAdmin je besplatni softverski alat napisan u PHP-u, namijenjen upravljanju administracijom baze podataka preko weba. Ovaj alat omogućuje korištenje operacija za upravljanje bazom podataka, tablicama, stupcima, indeksima, korisnicima, dozvolama koristeći korisničko sučelje, a ima mogućnost izravnog izvršavanja SQL upita. Da bi se koristio *PhpMyAdmin* potrebno je instalirati *XAMPP* na lokalnom računalu. *XAMPP* je besplatan paket s mnoštvom platformi internet poslužitelja otvorenog koda koji je razvila tvrtka *Apache Friends*. Sastoji se od Apache *HTTP* poslužitelja, *MariaDB* baze podataka i skripti pisanih na programskim jezicima PHP i Perl. Budući da većina implementiranih internet poslužitelja koristi iste komponente kao i *XAMPP*, to omogućuje prelazak s lokalnog testnog poslužitelja na produkcijskog poslužitelja. [2]

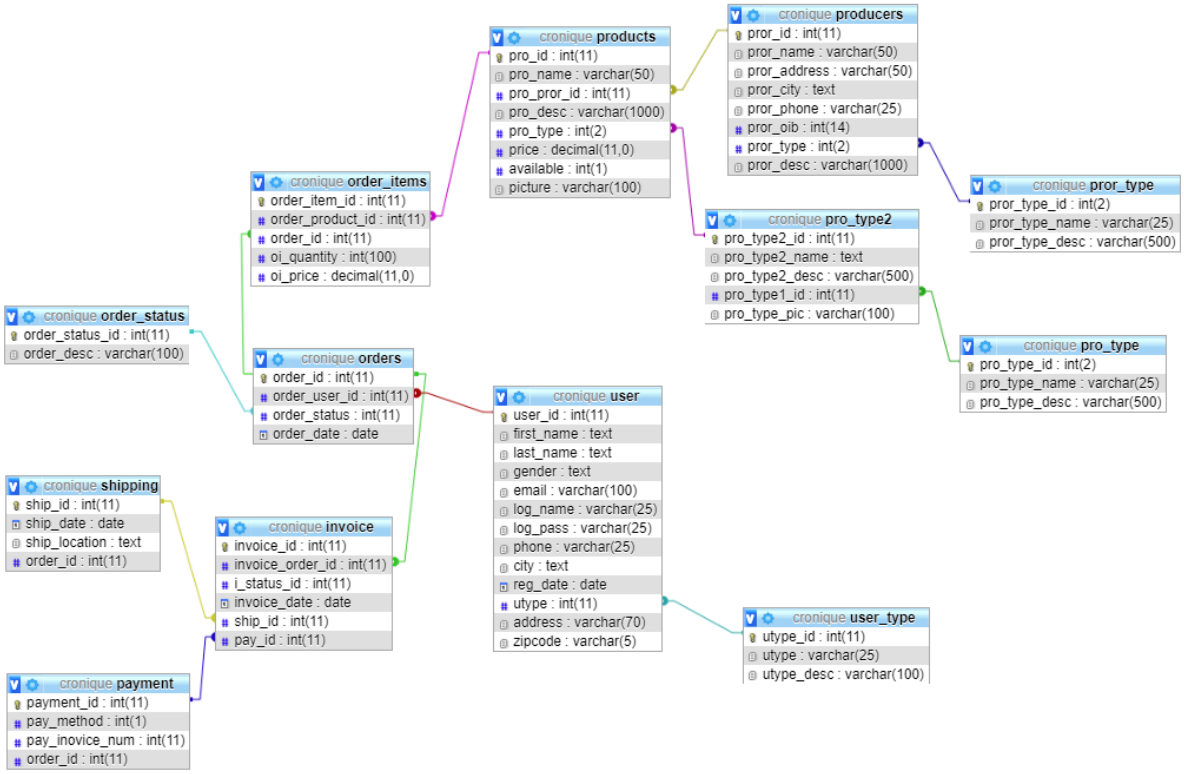


Slika 1: XAMPP korisničko sučelje

Kako bi se moglo pristupiti i koristiti sve funkcionalnosti potrebne za rad na internet aplikaciji prije svakog korištenja potrebno je provjeriti da li su pokrenuti moduli za Apache server i *MySQL* bazu podataka. Nakon što su obavljene sve prethodno definirane radnje može se pristupiti *PhpMyAdmin-u* na dva načina, u oba slučaja se pristupa preko internet preglednika upisivanjem adrese u URL.

3.4. Model baze podataka

Relacijski model baze podataka sastoji se od skupa tablica i relacija između njih. Osnovni koncepti modela podataka su entiteti, atributi, veze i ograničenja. Entitet je koncept ili objekt koji promatramo, može biti stvarni ili apstraktni predmet ili događaj o kojemu informacijski sustav prikuplja podatke. Sastoji se od niza dodijeljenih atributa. Atribut je obilježje entiteta, odnosno identificira, kvantificira, izražava kvalitetu ili stanje entiteta. Veza je odnos između dva entiteta. Ograničenja su granice sudjelovanja entiteta u vezi, ona opisuju koliko entiteta jednog tipa može biti u vezi s entitetima drugog tipa. [1]



Slika 2: Model baze podataka

Model baze podataka internet trgovine sastoji se od slijedećih tablica*: user, products, producers, order_items, orders, invoice, user_type, pro_type2, pro_type, pror_type, order_status, shipping i payment*. U tablici *user* nalaze se osnovni podaci o svim registriranim korisnicima kao što su ime, prezime, spol, korisničko ime, lozinka i ostalo. Atribut *utype* je vanjski ključ na tablicu *user_type* gdje se nalaze podaci o tipu korisnika. Korisnici su podijeljeni u tri kategorije, to su administratori, registrirani i ne registrirani korisnici. Tablica *producers* sadrži podatke o proizvođačima, vanjski ključ je atribut *pror_type* koji se referencira na tablicu *pror_type*. U tablici *pror_type* definirani su tipovi proizvođača, ovisno o tome za kakvu djelatnost je proizvođač registriran. U tablici *products* nalaze se svi proizvodi, što uključuje naziv, opis, dostupnost, sliku i cijenu proizvoda. Uz prethodno navedene atribute tablica ima dva vanjska ključa, to su *pro_pror_id* i *pro_type*. Atribut *pro_pror_id* se referencira na tablicu *producers* vezom jedan na prema više, što znači da proizvod može imati samo jednog proizvođača, dok proizvođač može imati više proizvoda. Atribut *pro_type* popunjava se vrijednostima iz tablice *pro_type2* gdje su definirani tipovi proizvoda. Entiteti u tablici *orders* predstavljaju narudžbe koje su preko vanjskog ključa *order_user_id* povezane s tablicom *user*. Svaki entitet predstavlja jednu narudžbu koju je određeni korisnik započeo ili realizirao. Proizvodi koji su vezani za određenu narudžbu nalaze se u tablici *order_itmes*. Tablica *order_items* osim samih proizvoda koje dohvaća iz tablice *products* preko vanjskog ključa

order_product_id sadrži i podatke o količini i cijeni proizvoda koji se nalaze u košarici. Nakon što korisnici završe s odabirom proizvoda, podaci o njihovoj kupnji spremaju se u tablicu *invoice*. Tablica invoice se popunjava posljednja u poslovnom procesu na način da se preko vanjskog ključa *invoice_order_id* referencira na tablicu *orders*. Osim te veze, preko vanjskih ključeva *ship_id* i *pay_id* povezana je s tablicama *payment* i *shipping.* U tablici *payment* nalaze se relevantni podaci o načinu plaćanja, te ovisno o tome i podaci o kreditnoj kartici korisnika. Podaci o lokaciji na koju će se izvršiti dostava naručenih proizvoda nalaze se u tablici *shipping*.

4. Tehnologije

U ovom će poglavlju biti definirane sve tehnologije koje su korištene tijekom izrade projekta.

4.1. HTML5

HyperText Markup Language (HTML) je prezentacijski jezik za izradu web stranica. Jednostavan je za uporabu i lako se uči, zbog toga je vrlo popularan i opće prihvaćen. HTML5 je sljedeća generacija HTML-a koja zamjenjuje sve prethodne inačice i podržava nove značajke potrebne modernim internet aplikacijama. Poput svojih prethodnika, HTML5 je projektiran da bude višeplatformski. Ne mora se koristiti Windows, Mac OS X, Linux ili neki drugi operacijski sustav da bi se moglo iskoristiti prednosti HTML-a 5, jedina stvar koja je potrebna je moderan operacijski sustav. Najnovije verzije preglednika, kao i preglednici na mobilnim uređajima pružaju izvrsnu podršku za HTML5. Unatoč tome projektiran je da bude , što je više moguće, kompatibilan unatrag sa postojećim preglednicima. [3]

4.1.1. Povijest

HTML se prvi puta pojavio krajem 1991. godine kao dokument pod nazivom HTML oznake (eng. *tags*), zasluge pripadaju Tim Berners-Lee-u. Prva verzija HTML jezika objavljena je 1993. godine i dobila je naziv HTML 2.0, radilo se o relativno jednostavnom dizajnu koji se sastojao od dvadeset elemenata. Koliko se radilo o ograničenom jeziku govori i podataka da u HTML dokument nije bilo moguće dodati sliku. World Wide Web Consortium (W3C) je u ožujku 1995. godine objavio verziju HTML 3.0, koja je imala nekoliko većih noviteta. Osim mogućnosti da se kreiraju tablice, prihvaćene su mnoge oznake koje su bile podržane u najpoznatijim internet preglednicima. Takve promjene rezultirale su nastankom više istih oznaka za pojedine funkcije. U prosincu 1997. godine W3C ugasio je radnu grupu za HTML i objavio HTML 4.0. Iako se nastavilo s prihvaćanjem standarda od strane internet preglednika, istovremeno se proglašavalo suvišnim neke od prethodnih standarda. U prosincu 1999. godine predstavljena je verzija HTML 4.01., koja je sadržava neke manje promjene u specifikaciji. [3]

HTML5 je nova inačica koja se pojavila nakon HTML-a 4.01, nastala je u suradnji World Wide Web Consortium-a (W3C) i Web Hypertext Application Technology Working Group (WHATWG). WHATWG je prvenstveno radio s internet aplikacijama i formama, dok je W3C radio s XHTML-om 2.0. Nakon 2006. godine ove dvije kompanije odlučile su se udružiti i razviti novu verziju HTML-a, do tada radile su odvojeno. HTML5 donio je nove mogućnosti koje njegovi prethodnici nisu imali, tu se prvenstveno misli na mogućnost reprodukcije videa na

stranicama bez korištenja drugih programa. Osim toga, omogućeno je upravljanje pomoću tipkovnice, opcije za bilo koju vrstu manipulacije, opcija povuci i ispusti (eng. *drag and drop*), platno (eng. *canvas*) i mnoštvo novih elemenata. [3]

4.1.2. Struktura HTML dokumenta

HTML dokument sastoji se od elemenata i oznaka, a može imati i atribute kojima se definiraju svojstva elemenata. Svaka HTML oznaka počinje znakom manje od (<), a završava znakom veće od (>). Dodavanjem kose crte (/) prije završnog znaka veće od (>) kreira se zatvarajuća HTML oznaka. Element *<!DOCTYPE>* nalazi se na samom početku HTML dokumenta i njime se definira inačica koja se koristi za izradu HTML dokumenta. Početak HTML dokumenta označava se elementom *<html>*. Elementi *<head>* i *<body>* nalaze se unutar *<html>* elementa. Element *<head>* predstavlja zaglavlje u kojem se nalazi naslov stranice, te se specificiraju jezične značajke HTML dokumenta i dodaju stilska obilježja stranice. Stilska obilježja mogu biti direktno ugrađena ili dodana kao referenca na vanjsku CSS datoteku. Skripte kreirane u *JavaScript* jeziku i pozivi raznih biblioteka često se nalaze unutar zaglavlja. U elementu *<body>* kreira se sadržaj HTML dokumenta. Primjer standardnog HTML elementa izgleda ovako:

```
<html></html>
```

Osim standardnih postoje i samozatvarajući HTML dokumenti, kod takvih elemenata nema zatvarajuće oznake. U nastavku slijedi primjer koda jednostavnog HTML dokumenta:

```
<!DOCTYPE hmtl>
<html>
	<head>
		<title>Naziv stranice</title>
	</head>
	<body>
		<h1>Naslov</h1>
		<p>Sadržaj stranice</p>
	</body>
</html>
```

4.2. CSS

Cascading Style Sheets (CSS) je stilski jezik koji se koristi za opisivanje prezentacije dokumenata pisanog u HTML-u ili XML-u (uključujući XML dijalekte kao što su SVG, MathML ili XHTML), što uključuje boje, izgled, fontove i drugo. CSS opisuje kako bi se elementi trebali prikazivati na zaslonu ili na drugim medijima, te dopušta prilagodbu različitim tipovima uređaja. CSS je neovisan o HTML-u i može se koristiti s bilo kojim drugim XML-baziranim jezikom oznaka. Odvajanje HTML-a od CSS-a olakšava održavanje, dijeljenje stilskih uputa između stranica i prilagođavanje stranica različitim okolinama, to se naziva odvajanje strukture od prezentacije. [5]

CSS1 je jezik koji ljudi mogu čitati i pisati te izraziti stil na uobičajeni način terminologije, kasnije je zamijenjen s CSS2. CSS2.1 uvodi nove mogućnosti kao što su novi selektori, apsolutno, relativno i fiksno pozicioniranje elemenata, slojevitost elemenata, koncept tipova medija, te podržava apsolutno pozicioniranje, brojanje, prekid stranice i sl. Pogreške u CSS2 ispravljene su u verziji CSS2.1 koja izbacuje slabo podržane ili nepotpuno interoperabilne osobine. CSS3 podijeljen je u više dokumenata koji se zovu moduli. Svaki modul dodaje funkcionalnost i zamjenjuje dio CSS2.1 specifikacije.[4]

Postoje tri tipa stilskih listova, a to su preglednički, autorski i korisnički. Kod pregledničkih stilskih listova elementi imaju svoje predefinirane osobine. Problem koji se može pojaviti korištenjem različitih preglednika je da jedan element nema iste osobine kod različitih preglednika. Kod autorskih stilskih listova osobine elemenata obično definira dizajner. Korisnički stilski list omogućuje korisniku da može definirati vlastite osobine za određene elemente, kao što su veći font, druga boja, kontrast i sl. Za jednu osobinu nekog elementa autorski nadjačava preglednički, a korisnički nadjačava autorski. [4]

Organizacija W3C (World Wide Web Consortium) propisuje uporabu jedinica boja. Jedinice boje navode se u deklaracijskom bloku elementa s odgovarajućim svojstvom, a vrijednost se može iskazati predodređenim nazivom boje i brojevnim izrazom. Za jedinicu boje kod predodređenim naziva boja koriste se ključne riječi na engleskom jeziku. Zbog ograničenja koja je imao hardver 1996. godine, CSS1 koristio je svega šesnaest boja. Daljnjim razvojem tehnologije broj korištenih boja narastao je na sto četrdeset i sedam boja koje su podržavali svi važni preglednici. Brojevne jedinice zasnivaju se na dva modela boja, to su RGB i HSL. Model RGB koristi primarne boje na način da im zadaje intenzitet svjetlosti. Primarne boje su crvena, zelena i plava. Sekundarne boje mogu se dobiti miješanjem samo dviju primarnih boja, to su cijan, žuta i magenta. Definirani raspon intenziteta svjetlosti za svaku boju je osam bitova, od 0 do 255, u heksadekadskom sustavu od #00 do #FF. Njihovim miješanjem dobiva se paleta s 16.777.216 boja. Jedinice boje u RGB modelu mogu se koristiti na tri načina, to su heksadekadskim brojevima, dekadskim vrijednostima i postotkom intenziteta određene boje. Kao dopuna RGB modelu zbog nekih njegovih ograničenja uveden je HSL model. HSL model

umjesto intenziteta svjetlosti boje definira nijansom, zasićenjem i osvijetljenošću. Nijansa predstavlja stupanj na krugu boja, zasićenje se predstavlja postotkom, a osvjetljenje se opisuje intenzitetom svjetlosti koja se reflektira od površine i iskazuje u postocima. [6]

4.3. Bootstrap

Bootstrap je trenutno najpopularniji okvir (eng. *framework*) za izradu korisničkih sučelja internet aplikacija i stranica. Razvili su ga dvojica web programera Mark Otto i Jacob Thronton, obojica su zaposlenici popularne kompanije Twitter. *Bootstrap* je potpuno besplatan za preuzimanje, vrlo lako se implementira i koristi te programeru štedi mnogo vremena pri izradi projekta. [7]

Nakon preuzimanja, u paketu se mogu pronaći tri datoteke koje se odnose na CSS, *JavaScript* i fontove. Unutar CSS datoteke nalazi se nekoliko inačica kao što su *bootstrap.css*, *bootstrap.min.css*, *bootstrap-theme.css*, *bootstrap.css.map*, *bootstrap.min.css.map* i *bootstrap-theme.css.map*. Potrebno je izabrati jednu od inačica te se preporuča da to bude s inačicom css dok ekstenzije s inačicom map nisu neophodne. Unutar JS datoteke nalaze se dvije *JavaScript* komponente, to su bootstrap.js i bootstrap.min.js. One proširuju funkcionalnost postojećih elemenata korisničkog sučelja. Datoteka s fontovima sadrži mnoštvo besplatnih ikona koje korisnik može koristiti na svojoj Internet stranici ili aplikaciji. Komponente koje *Bootstrap* nudi su raznolike i svestrane, odnosno sadrže sve što jednoj treba jednoj internet stranici ili aplikaciji. Vrlo važna značajka ovog alata je to što je kompatibilan sa svim internet pretraživačima kao što su Internet Explorer, Chrome, Safari, Opera i drugi. Sadržaj je prilagodljiv svim veličinama ekrana te je dinamičan, iz čega se može zaključiti da je potpuno kompatibilan i za mobilne telefone. [7]

Način korištenja je vrlo jednostavan, korisničko sučelje kreirano isključivo korištenjem *Bootstrap* biblioteka izgledati će lijepo i pregledno te neće biti potrebne posebne dorade ukoliko se radi o internet aplikaciji koja naglasak stavlja na funkcionalnost. Želi li se koristiti Bootstrap, točnije jednu od njegovih biblioteka za *JavaScript* potrebno je u zaglavlju HTML stranice unutar oznaka <*head*> napisati slijedeće.

```
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.1.3/js/bootstrap.min.js">
```

Nakon što je definiran poziv *Bootstrap* biblioteke potrebno je pronaći predložak koji u potpunosti odgovara zahtjevima programera za izradu korisničkog sučelja. Kada je pronađen željeni predložak, generirani HTML kod se kopira u skriptu. Ukoliko niti jedan od predložaka

ne zadovoljava potrebe korisnika definirane u korisničkom zahtjevu predlaže se pronalazak najsličnijeg predloške te njegova modifikacija. Za provođenje modifikacija potrebno je znanje iz prethodno definiranih tehnologija HTML-a i CSS-a. Potreba za modifikacijom najčešće se javlja pri izradi internet stranica gdje korisnik želi postići jedinstvenost njegove stranice.

4.4. JavaScript

„JavaScript je izvorno bio zamišljen kao skriptno sučelje između Web stranice učitane klijentski preglednik i aplikacije na poslužitelju. Od predstavljanja 1995. godine, JavaScript je postao ključna komponenta razvoja za web te je pronašao primjenu i drugdje.“ [8, str 1]

Funkcionira bez prethodne pripreme i u većini preglednika, što znači da instalacija i konfiguriranje putanja do raznih biblioteka nisu potrebni. Za minimalnu funkcionalnost potrebno je dodati blok skripte i početi raditi. *JavaScript* blokovi dodaju se u zaglavlje dokumenta, ali ih se također može uključiti u tijelo dokumenta, ili čak i u jedan i u drugi. Međutim, dodavanje skripte u tijelo dokumenta ne smatra se dobrom tehnikom jer otežava pronalaženje skripte kada ju kasnije treba mijenjati. [8]

Zanimljiv podatak je da od svibnja 2017. godine 94,5% od deset milijuna najpopularnijih internet stranica koristi *JavaScript*. Najčešća upotreba *JavaScript-a* je dodavanje funkcionalnosti HTML stranicama na strani klijenta, što znači da se kod neće izvršavati na poslužitelju već isključivo na klijentskoj strani. Takvim se pristupom postiže da se kod izvršava lokalno u korisnikovom pregledniku, čime se povećava općeniti odaziv aplikacije na korisničke akcije. *JavaScript* kod također može otkriti radnje korisnika koje sam HTML ne može, kao što su pojedinačni pritisci na tipke. Primjena *JavaScript-a* ogleda se i u korištenju interaktivnih sadržaja, na primjer igre, reprodukcije zvuka i videozapisa. Koristi se i za animaciju elemenata stranice, njihovo premještanje, izmjenu veličine, sakrivanje i prikazivanje, itd.

4.5. PHP

PHP je programski jezik namijenjen za izradu dinamičnih Web stranica. PHP je postao popularan i za generiranje XML dokumenata, grafike, animacija, PDF datoteka i još mnogo toga. Radi na svim većim operacijskim sustavima, od inačica *Unix-a* uključujući *Linux*, *FreeBSD*, *Ubuntu*, *Debian* i *Solaris* do Windowsa i Mac OS X-a. Može se koristiti na svim vodećim Web poslužiteljima, uključujući Apache, Microsoft IIS i Netscape/iPanet. Jedna od PHP-ovih najznačajnijih osobina je njegova široka podrška za baze podataka. PHP podržava sve veće baze podataka, uključujući *MySQL*, *PostgreSQL*, *Oracle*, *Sybase*, *MS_SQL*, *DB2* i baze podataka kompatibilne s *ODBC-om*. S PHP-om je izrada Web stranica s dinamičnim

sadržajem iz baze podataka izuzetno jednostavna. Rasmus Lerdorf prvi je put koncipirao PHP 1994. godine, no PHP koji se danas koristi poprilično se razlikuje od početne inačice. [9]

4.5.1. Osnove jezika

„Leksička struktura programskog jezika je skup temeljnih pravila koja upravljaju načinom na koji pišete programe na tom jeziku. To je sintaksa najniže razine koja određuje stvari kao što su format imena varijabli , znakovi koji se koriste za komentare i način međusobnog razdvajanja iskaza programa. Imena klasa i funkcija koje definira korisnik, kao i ugrađene konstrukcije i ključne riječi kao što su *echo*, *while*, *class*, itd. ne razlikuju velika i mala slova. Stoga su sljedeća tri reda ekvivalentna:

```
echo("test");
ECHO("test");
Echo("test");
```

Varijable razlikuju velika i mala slova. To znači da su *$test*, *$TEST* i *$Test* tri različite varijable.“ [9, str. 15]

„Iskaz (engl. *statement*) je kolekcija PHP koda koja nešto radi. To može biti jednostavno pridruživanje varijable ili složena petlja s više izlaznih točaka. PHP koristi točku-zarez za razdvajanje jednostavnih iskaza. Složenom iskazu u kojem se koriste vitičaste zagrade za označavanje bloka koda, kao što je uvjet ili petlja, nakon zatvaranja vitičaste zagrade nije potrebna točka-zarez. U PHP programu bjeline nisu bitne, što znači da se iskaz može raširiti preko velikog broja redova ili staviti više iskaza u jedan red.“ [9, str. 16]

Komentari pružaju objašnjenje osobama koje čitaju kod, dok ih PHP zanemaruje prilikom izvršavanja. Komentari će prvenstveno koristiti osobi koja nije pisala kod radi lakšeg snalaženja i shvaćanja koncepcije rada razvojnog programera, ali mogu biti i od iznimne koristi samom autoru koda. Autor koda nakon nekoliko mjeseci može potpuno zaboraviti vlastiti kod te u slučaju da su potrebne određene dorade komentari će mu biti od iznimne važnosti. Dobra je navika pisati komentare koji su dovoljno kratki da količinom ne nadilaze sam kod, no dovoljno opširni da ih se može koristiti za objašnjenje pojedinih dijelova. Očite stvari se ne bi trebale komentirati kako se ne bi izgubili komentari koji opisuju bitne dijelove. Primjer nepotrebnog komentara izgleda ovako:

```
$x = 1;          // sprema 1 u varijablu $x
```

Identifikatori se u PHP-u koriste za imenovanje varijabli, funkcija, konstanti i klasa i on nije ništa drugo nego ime. Prvi znak identifikatora mora biti ASCII slovo (veliko ili malo), znak podvlake (_) ili bilo koji od znakova između ASCII 0x7F i ASCII 0Xff. Nakon početnog znaka valjani su ti znakovi i znamenke od nula do devet. Imena varijabli uvijek počinju znakom dolara ($) i u njima se razlikuju velika i mala slova. Konstanta je identifikator za jednostavnu vrijednost, a zadaju se pomoću funkcije *define()*. [9]

4.5.2. Tipovi podataka

PHP pruža osam tipova vrijednosti i tipova podataka (eng. *data type*). Četiri od njih su skalarni tipovi (tipovi jednostavnih vrijednosti): cjelobrojne vrijednosti, brojevi s pokretnim zarezom, nizovi znakova i logičke vrijednosti. Dva su složeni (zbirni) tipovi: polja i objekti. Preostala dva su posebni tipovi: resurs i NULL. Cjelobrojne vrijednosti (engl. *integers*) su cijeli brojevi, raspon prihvatljivih vrijednosti ovisi o specifikacijama platforme. Brojevi s pokretnim zarezom (engl. *floating point numbers*) koji se često nazivaju realnim brojevima predstavljaju numeričke vrijednosti s decimalnim znamenkama. Nizovi znakova (engl. *strings*) su vrlo često korišteni u Web aplikacijama te sadrže podršku na temeljnoj razini za rad s njima i mogu biti proizvodne duljine. Znakovi su izdvojeni jednostrukim ili dvostrukim navodnicima. Logička vrijednost (engl. *boolean*) govori nam je li nešto istinito ili ne. Kao i većina programskih jezika, PHP definira neke vrijednosti kao istinite a neke kao lažne. Istinitost i lažnost najčešće se koriste za određivanje ishoda uvjeta. Polja (engl. *array*) sadrži grupu vrijednosti koje se mogu identificirati prema poziciji gdje nula označava prvu poziciju ili prema nekom imenu za identifikaciju znakova. Tako se razlikuju indeksirani i asocijativni nizovi. PHP podržava i objektno-orijentirano programiranje te promiče čisti modularni dizajn, pojednostavljuje uklanjanje pogrešaka, održavanje i olakšava ponovnu upotrebu koda. Klase (engl. *classes*) su građevni elementi objektno-orijentiranog dizajna. Klasa je definicija strukture koja sadrži svojstva (varijable) i metode (funkcije). Samo je jedna vrijednost tipa podatka *NULL*. Ta vrijednost dostupna je preko ključne riječi *NULL* koja ne razlikuje velika i mala slova. Vrijednost *NULL* je varijabla koja nema vrijednosti. [9]

4.5.3. Varijable

Varijable u PHP-u su identifikatori sa znakom dolara ($) kao prefiksom. Varijabla može sadržavati vrijednost bilo kojeg tipa. Tip varijable ne provjerava se ni u vrijeme izvođenja ni u vrijeme prevođenja te se uvijek može zamijeniti drugom vrijednošću drugog tipa.

```
$x = "tekst";
$x = 10;
```

```
$x = array("tekst", 10, "tekst2")
```

U PHP-u ne postoji eksplicitna sintaksa za deklariranje varijabli. Nakon što se varijabli zada vrijednost, ona je stvorena. Drugim riječima, zadavanje vrijednosti varijabli funkcionira kao deklaracija. Djelokrug varijable, kojeg zadaje mjesto deklaracije varijable, određuje koji dijelovi programa mogu pristupiti varijabli. Četiri su tipa djelokruga varijable u PHP-u: lokalni, globalni, statički i parametar funkcije. Varijabla koja je deklarirana u nekoj funkciji za tu je funkciju lokalna. To znači da je vidljiva samo kodu u toj funkciji i nije dostupna izvan nje. Varijable deklarirane izvan funkcije su globalne. To znači da im se može pristupiti iz bilo kojeg dijela programa. Međutim, podrazumijeva se da nisu dostupne unutar funkcija. Statička varijabla zadržava svoju vrijednost između poziva funkcije, no vidljiva je samo unutar te funkcije. Varijabla se deklarira kao statička pomoću ključne riječi *static*. Parametri funkcije su lokalni, što znači da su dostupni samo unutar svoje funkcije. [9]

4.5.4. Izrazi i operatori

„Izraz (engl. *expression*) je dio PHP koda koji je moguće evaluirati da bi se stvorila vrijednost. Najjednostavniji su izrazi doslovne vrijednosti i varijable. Operator (engl. *operator*) uzima neke vrijednosti i nad njima provodi određene operacije. Poredak po kojem se operatori u izrazu evaluiraju ovisi o njihovom prvenstvu. Asocijativnost definira poredak po kojem se evaluiraju operatori iste razine prvenstva.“ [9, str. 34]

„Aritmetički operatori su operatori koji se prepoznaju iz svakodnevne uporabe i najčešće su korišteni. Većina aritmetičkih operatora je binarna, osim aritmetičke negacije i aritmetičke potvrde koje su unarne. Aritmetički operatori su: zbrajanje, oduzimanje, množenje, dijeljenje, modul, aritmetička negacija i aritmetička potvrda. Operator modula pretvara oba operanda u cjelobrojne vrijednosti i vraća ostatak dijeljenja prvog operanda drugim. Operator aritmetičke negacije vraća operand pomnožen s -1, što znači da mu mijenja predznak. Operator aritmetičke potvrde vraća operand pomnožen s +1, što nema nikakvog učinka.“ [9, str. 38]

Rad s nizovima znakova ključan je dio PHP aplikacija i ima poseban operator za nastavljanje nizova (.) koji dodaje desni operand lijevom i vraća niz znakova koji je nastao. Povećanje i smanjenje vrijednosti varijable za jedan je među najuobičajenijim operacijama u programiranju. Kratice za te uobičajene operacije su unarni operatori samo povećanja (++) i samo smanjenja (--). Ovi su operatori jedinstveni po tome što rade samo sa varijablama. Operatori usporedbe uspoređuju operande, a rezultat je istina (engl. *true*) ili laž (engl. *false*). Operatori usporedbe su: jednakost (==), identičnosti (===), nejednakost (!= ili <>), ne identičnost (!==), veće od (>), veće od ili jednako (>=), manje od (<) i manje od ili jednako (<=).

Razlika između operatora jednakosti i identičnosti je u tome što jednakost provjerava da li su operandi jednaki dok operator identičnosti provjerava da li su operandi jednaki i istog tipa.

Logički operatori pružaju načine za gradnju složenih logičkih izraza. Ponašaju se prema svojim operandima kao prema logičkim vrijednostima i vraćaju logičku vrijednost. Logički operatori su: logičko I (&&, and), logičko ILI (||, or), logičko ekskluzivno ILI (xor) i logička negacija (!). Rezultat operacije logičkog I je istina ako i samo ako su oba operanda istina, u drugim slučajevima rezultat je laž. Rezultat operacije logičkog ILI je istina ako je bilo koji od operanda istina, u drugim slučajevima rezultat je laž. Rezultat operacije logičkog ekskluzivnog ILI je istina ako je jedan od operanda istina, ali ne i oba. [9]

4.5.5. Iskazi za kontrolu toka

„PHP podržava niz tradicionalnih programerskih konstrukcija za kontrolu toka izvršavanja programa. Uvjetni iskazi, kao što su *if/else* i *switch*, dopuštaju programu da izvrši različite dijelove koda, ili nijedan, ovisno o nekom uvjetu. Petlje, kao što su *while* i *for*, podržavaju ponovno izvršavanje određenih odlomaka koda.“ [9, str. 47]

Iskaz *if* provjerava istinitost izraza i, ako je izraz istina, evaluira ga. Ključna riječ *else* koristi se kako bi se odredio alternativni iskaz koji će se izvršiti ako je izraz lažan. Iskaz *if/else* izgleda ovako:

```
If (izraz) {
      Iskaz
}
else  {
       iskaz
}
```

Često se koriste lanci iskaza *if* pa PHP za njih pruža jednostavniju sintaksu, iskaz *elseif*. [9]

Iskazu *switch* daje se izraz i on uspoređuje njegovu vrijednost sa svim slučajevima u preklopniku. Izvršava sve iskaze koji odgovaraju, sve do prve ključne riječi *break* na koju naiđe. Ako nijedan ne odgovara, a dana je ključna riječ *default*, svi iskazi nakon ključne riječi *default* se izvršavaju, sve do prve ključne riječi *break*. [9] Iskaza *switch* izgleda ovako:

```
switch ($varijabla){
case 'vrijednost1':
       iskaz
```

```
        brake;
case 'vrijednost2':
        iskaz
        brake;
          default:
        iskaz
        brake;
                }
```

„Iskaz *while* je najjednostavniji oblik petlje. Ako izraz evaluira u *true*, iskaz se izvršava i izraz se ponovno evaluira (ako je još uvijek *true*, tijelo petlje se ponovno izvršava, i tako dalje). Petlja se napušta kada izraz više nije točan, odnosno evaluira se u *false*. Pomoću ključne riječi *break* može se ranije izaći iz petlje. Ukoliko se želi osigurati da se tijelo petlje izvede bar jednom, koristi se petlja *do/while*. Iskaz *do/while* se najčešće koristi za napuštanje bloka koda kad dođe do stanja pogreške.“ [9, str. 51-53] Iskaz *while* izgleda ovako:

```
while (izraz) {
      iskaz
}
```

Iskaz *for* sličan je iskazu *while*, no dodaje izraze inicijalizacije i upravljanja brojačem, često je kraći i pregledniji od petlje *while*. Iskaz u for petlji može se ostaviti prazan, čime se daje znak da u toj fazi ne treba učiniti ništa. U tom slučaju petlja postaje beskonačna i neće prestati ispisivati sadržaj iskaza. U petljama for, baš kao i petljama *while*, može se upotrijebiti ključna riječ *break* kako bi se završila petlja. Petlja *for* izgleda ovako:

```
for (izraz1; izraz2; izraz3){
            iskaz;
            …;
}
```

Iskaz *foreach* dopušta da se prolazi kroz elemente u polju. [9]

„PHP pruža dvije konstrukcije za učitavanje koda i HTML-a iz drugih modula, to su ključne riječi *requier* i *include*. Oni učitavaju datoteku dok PHP izvodi skriptu, radi s uvjetima i petljama, te prijavljuju pogrešku ako ne mogu pronaći datoteku koja se učitava. Osnovna razlika među njima je u tome što je pokušaj da se zatraži nepostojeća datoteka krupna

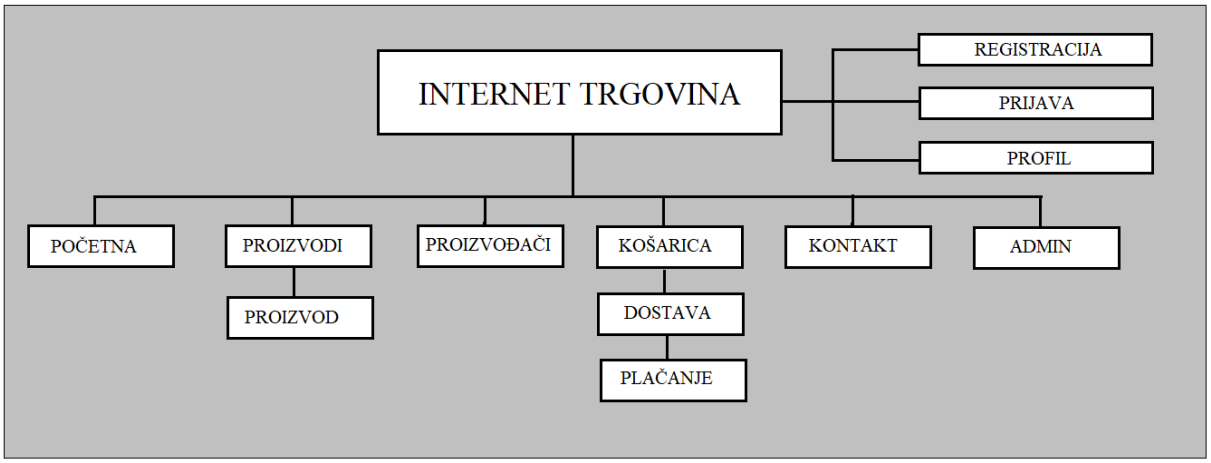
pogreška, dok pokušaj da se takva datoteka uključi daje upozorenje no ne zaustavlja izvršenje skripte. Uobičajena upotreba include je za odvajanje sadržaja specifičnog za stranicu od općenitog dizajna stranice, kao što su zaglavlje i podnožje. Konstrukcija require prikladnija je za učitavanje biblioteka koda, kod kojih se stranica ne može prikazati ako se biblioteka ne učita.“ [9, str. 57]

5. Ekranski pregledi internet trgovine

Internet trgovina „Cronique“ namijenjena je prodaji proizvoda isključivo Hrvatskih proizvođača. Sastoji se od modula kojima je pristup ograničen ovisno o korisničkim ulogama korisnika.

5.1. Moduli

Moduli su oblici (sheme) prema kojima se strukturiraju i organiziraju elementi unutar internet trgovine. Pristup modulima je ograničen, ovisno o tipu korisnika koji je prijavljen. Zaglavlje i podnožje internet aplikacije vidljivi su bez obzira u kojem se modulu korisnik nalazi. Zaglavlje se sastoji od dva gumba (eng. *button*), to su „Prijava“ i „Registracija“. Nakon što se korisnik prijavi gumb „Registracija“ će se promijeniti u gumb „Profil.“. Prilikom pritiska na gumb „Registracija“ otvara se forma koju su novi korisnici obvezni popuniti. Ukoliko je registracija uspješno završena ili korisnik već posjeduje korisnički račun, pritiskom na gumb „Prijava“ otvara se forma u koju je potrebno upisati korisničko ime i lozinku.



Slika 3: Moduli

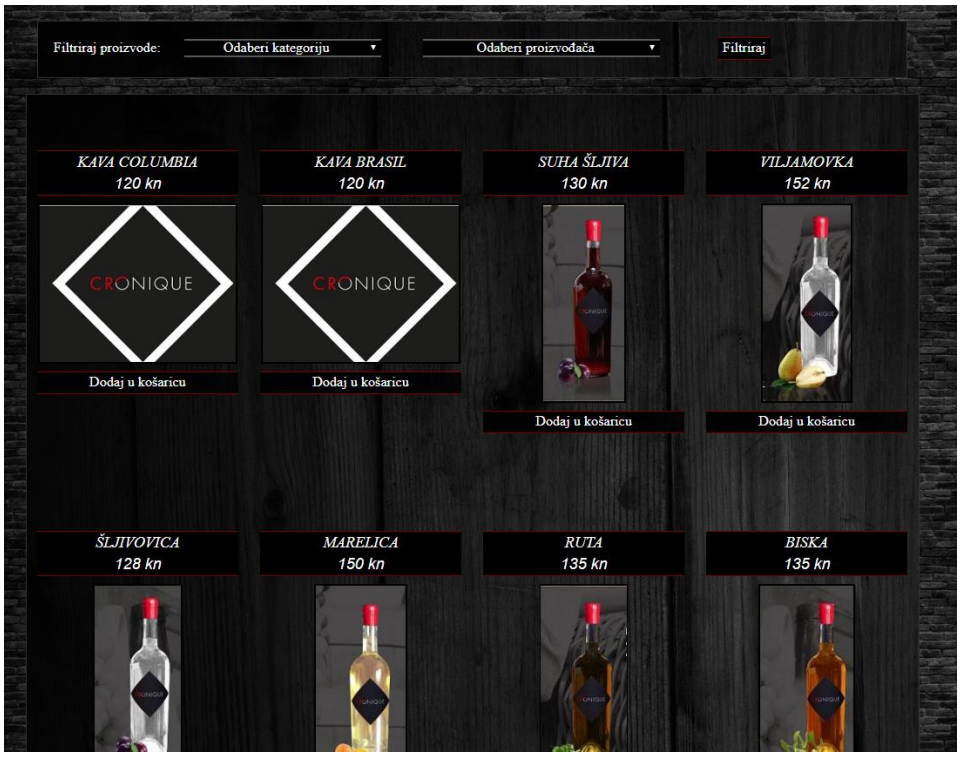
5.1.1. Modul „Početna“

Modul „Početna“ je ujedno i prvi ekranski pregled koji se prikazuje kada korisnik posjeti internet trgovinu. Vidljiv je svim tipovima korisnika i izgleda isto bez obzira da li se korisnik prijavio u sustav. Na vrhu se nalazi kratak tekst koji opisuje brand internet trgovine, kao i njegovu ulogu i ciljeve. Nakon uvodnog dijela dolazi dio koji se sastoji od kategorija proizvoda, te njihovih slika i kratkih opisa.

Slika 4: Modul "Početna"

5.1.2. Modul „Proizvodi“

Modul „Proizvodi“ je najveći i ujedno najvažniji modul u sklopu internet trgovine. Na vrhu modula nalazi se sučelje za filtriranje proizvoda. Proizvodi se mogu filtrirati prema kategoriji i proizvođaču. U slučaju da korisnik ne koristi sučelje za filtriranje proizvoda prikazati će se proizvodi svih kategorija i proizvođača. U nastavku se ispisuju kartice s proizvodima, raspoređene su po četiri u jednom retku. Kartica proizvoda sastoji se od naziva, cijene, slike i ovisno o tipu korisnika prikazuje se gumb „Dodaj u košaricu“.



Slika 5: Modul "Proizvodi"

Pritiskom na sliku proizvoda sustav će nas automatski preusmjeriti na modul „Proizvod“. Modul „Proizvod“ korisniku prikazuje više podataka o samom proizvodu. U naslovu se nalazi naziv proizvoda, zatim slijede podaci o proizvođaču, cijeni, dostupnosti proizvoda i opis. Prikazana je slika proizvoda kao i gumb „Dodaj u košaricu.“

Slika 6: Modul "Proizvod"

5.1.3. Modul „Košarica“

Nakon što korisnik odabere opciju za dodavanje proizvoda u košaricu sustav ga automatski preusmjerava u modul „Košarica“. U tom se modulu nalaze svi proizvodi koje je korisnik prethodno odabrao. Uz same nazive odabranih proizvoda korisniku može vidjeti cijenu i količinu, kao i ukupnu cijenu narudžbe. Polje količina moguće je mijenjati prema potrebi, a ono će se automatski popuniti s vrijednosti „1“ prilikom odabira proizvoda. Ukoliko korisnik odluči promijeniti polje količina, nakon što upiše željenu vrijednost potrebno je odabrati opciju „Ažuriraj košaricu“, kako bi se promijenili podaci u košarici. Uz svaki proizvod nalazi se opcija za brisanje istog s popisa narudžbe. Uz mogućnost ažuriranja košarice nalazi se gumb „Isprazni košaricu“ s mogućnošću brisanja svih proizvoda iz košarice, te gumb „Dovrši narudžbu“ koji automatski preusmjerava pod modul „Dostava“.

Slika 7: Modul "Košarica"

Modul „Dostava“ sastoji se od polja za unos adrese na koju će dostava biti isporučena i gumba „Nastavi s plaćanjem“ koji automatski preusmjerava na modul „Plaćanje“.

Slika 8: Modul "Dostava"

Modul „Plaćanje“ korisniku nudi dvije mogućnosti, to su plaćanje karticom i plaćanje pouzećem. Ukoliko korisnik odabere plaćanje karticom otvoriti će se pod izbornik s poljem za unos broja kartice i mogućnošću završetka plaćanja. Odabirom plaćanja pouzećem sustav će prikazati poruku „Izabrali ste plaćanje pouzećem!“, te će korisnik također imati mogućnost završetka narudžbe.

Slika 9: Modul "Plaćanje"

5.1.4. Modul „Admin“

Modul „Admin“ namijenjen je isključivo administratoru Internet trgovine, te kao takav nije vidljiv ostalim korisnicima. Podijeljen je na tri dijela, dio vezan uz proizvođače, proizvode i korisnike. Uz proizvođače i korisnike vezana su dva pod modula, za dodavanje novih proizvođača, odnosno korisnika i ažuriranje postojećih. Kod dijela za vezanog za administriranje proizvoda postoje opcije za dodavanje novih proizvoda, ažuriranje postojećih i dodavanje novih kategorija proizvoda. U svim slučajevima klikom na gumb „Dodaj“ otvara se nova forma u koju je potrebno unijeti sve potrebne podatke. Odabirom gumba „Ažuriraj“ otvara se padajući izbornik gdje administrator treba odabrati željeni zapis, nakon toga otvara mu se popunjena forma sa relevantnim podacima za izabrani zapis.

Slika 10: Modul "Admin"

5.1.5. Ostali moduli

Modul „Registracija“ nalazi se u zaglavlju stranice i vidljiv je isključivo korisniku koji nije prijavljen u sustav. Klikom na gumb „Registracija“ u tijelu stranice otvara se forma koju korisnik mora popuniti. Forma se sastoji od osnovnih informacija o korisniku kao što su ime, prezime, spol, adresa i ostalo. Nakon što je korisnik popunio sve podatke potrebno je kliknuti na gumb „Registriraj se“. Ukoliko je registracija uspješno obavljena sustav će korisnika automatski preusmjeriti na modul za prijavu.

Slika 11: Modul "Registracija"

Modul „Prijava“ namijenjen je registriranim korisnicima kako bi se mogli prijaviti u sustav i obavljati kupnju na internet trgovini. Gumb za prijavu nalazi se u zaglavlju stranice, a nakon što se korisnik uspješno prijavi u sustav naziv gumba promijeni se u „Odjava“. Klikom na gumb „Odjava“ korisnik će se odjaviti iz sustava. Modul „Prijava“ sastoji se od forma koja ima dva polja za unos, a to su korisničko ime i lozinka koje je korisnik definirao prilikom registracije.

Ukoliko se pokuša prijaviti korisnik koji se prethodno nije registrirao ili je upisao neispravno korisničko ime ili lozinku, sustav će javiti grešku.

Slika 12: Modul "Prijava"

Modul „Profil“ nalazi se na istom mjestu kao i modul „Registracija“. Za razlika od modula za registraciju koji je vidljiv isključivo ne prijavljenim korisnicima, ovaj modul vidljiv je isključivo korisnicima koji su prijavljeni u sustav. Klikom na gumb „Profil“ u tijelu stranice pojaviti će se lista s osnovnim podacima o korisniku i gumb „Prikaži narudžbe“. Odabirom opcije za prikaz narudžbe u nastavku stranice ispisati će se sve narudžbe prijavljenog korisnika. Korisnik može vidjeti datum, jedinstveni broj i trenutni status narudžbe.

Slika 13: Modul "Profil"

Modul „Kontakt“ nalazi se u glavnom izborniku i vidljiv je svim tipovima korisnika. Sastoji se od popisa kontakt informacije putem kojih korisnici mogu kontaktirati administraciju internet

trgovine. Internet trgovina nudi nekoliko mogućnosti komunikacije s korisnicima, putem e-mail pošte, telefona i mobitela.

Slika 14: Modul "Kontakt"

5.2. Korisničke uloge

Unutar internet trgovine postoje korisničke uloge koje se razlikuju po ovlastima. Ovlasti mogu biti vezane uz mogućnost korištenja određenih funkcionalnosti i pristupu određenim dijelovima aplikacije. Korisničke uloge podijeljene su prema tipu korisnika, tako se razlikuje administrator, registrirani korisnik i ne registrirani korisnik.

Ne registrirani korisnik može pristupiti internet trgovini ali ne može koristiti gotovo nikakve funkcionalnosti prije registracije. Osim početne stranice i kontakta, može vidjeti popis svih proizvoda ali nema mogućnost dodavanja proizvoda u košaricu.

Registrirani korisnik nakon uspješne prijave ima znatno veće mogućnosti. Uz sve mogućnosti koje ima i ne registrirani korisnik, može koristiti i sve funkcionalnosti internet trgovine. Dozvoljen mu je pristup slijedećim modulima: „Profil“, „Početna“, „Proizvodi“, „Proizvođači“, „Košarica“ i „Kontakt“. Unutar modula „Proizvodi“ ima mogućnost dodavanja proizvoda u košaricu. U modulu „Košarica“ može vidjeti proizvode koje je izabrao te nastaviti s kupnjom.

Administrator ima pristup modulu „Admin“, te svim modulima kao i registrirani korisnik. U modulu „Admin“ može dodavati i ažurirati proizvođače, dodavati i ažurirati proizvode, dodavati kategorije proizvoda, dodavati i ažurirati korisnike. Uloga administratora prvenstveno je namijenjena osobi koja će raditi na održavanju, unaprjeđivanju i testiranju same aplikacije.

6. Opis funkcionalnosti Internet trgovine

U ovom dijelu rada biti će prikazani dijelovi koda radi lakšeg razumijevanja funkcionalnosti Internet trgovine i njezinih mogućnosti.

6.1. Povezivanje s bazom podataka

Ukoliko se podaci u Internet aplikaciji žele prikazivati iz baze ili se želi mijenjati i unositi podatke potrebno je imati konekciju s bazom podataka. Kako se pri izradi same aplikacije kreira više skripti u kojima se izvršavaju prethodno definirane radnje, idealno rješenje je kreirati funkciju koja u sebi sadrži kod koji omogućuje povezivanje s bazom podataka. Funkciju će se zatim pozivati u određenim skriptama prema potrebi.

U projektu se nalazi skripta pod nazivom *bp.php* u kojoj se nalaze dvije funkcije *otvoriBP* i *zatvoriBP*. Prije samih funkcija kreirana je globalna varijabla pod nazivom *bp* i dodijeljena joj je vrijednost *null*. Funkcija se kreira na način da se pozove PHP ugrađena funkcija *function*, zatim slijedi naziv funkcije i parametri koje funkcija prima unutar oblih zagrada. Unutar vitičastih zagrada nalazi se kod koji će se izvršiti pozivanjem funkcije.

U nastavku slijedi kod za ostvarivanje konekcije s bazom podataka:

```
$GLOBALS['bp'] = null;
function otvoriBP() {
 $GLOBALS['bp'] = new mysqli ('127.0.0.1', 'root', '', 'cronique') or
 die ("Neuspjelo spajanje na bazu");
      mysqli_set_charset($GLOBALS['bp'], 'utf8mb4');
}
function zatvoriBP() {
      mysqli_close($GLOBALS['bp']);
}
```

Unutar funkcije globalnoj varijabli *bp* pridružuje pridružujemo vrijednost koju vraća PHP ugrađena funkcije *new mysqli*. Funkcija prima četiri parametra, prva vrijednost je lokacija na kojoj se nalazi MySQL server, zatim korisničko ime i lozinka s kojom se spaja na serveru i zadnja vrijednost predstavlja naziv baze podataka. Ukoliko neki od parametara ne zadovoljava vrijednost sustav će ispisati poruku da spajanje na bazu nije uspjelo. PHP ugrađena funkcija

mysqli_set_charset se koristi radi ispravnog ispisivanja dijakritičkih znakova. Funkcija *zatvoriBP* poziva se nakon izvršenih radnji na bazi podataka koja prekida konekciju pomoću PHP ugrađene funkcije *mysqli_close*.

6.2. Prijava u sustav

Prijava u sustav za korisnika predstavlja jednostavnu radnju gdje u formi za prijavu unese korisničko ime i lozinku, zatim pritiskom na tipku „Prijavi se“ potvrdi unesene podatke. Ukoliko su podaci ispravni automatski bude preusmjeren na početnu stranicu i prijavljen u sustav ili u slučaju da podaci nisu ispravni dobije povratnu informaciju. U pozadini aplikacije navedena radnja ima nekoliko koraka i proteže se kroz tri PHP skripte. Dolaskom korisnika na formu za prijavu poziva se skripta *login_form.php* gdje se nalazi HTML kod kreiran uz pomoć bootstrap-a.

U nastavku slijedi kod koji prikazuje formu za prijavu korisnika:

```
<div class="row" id="navbar">
<div class="container" id= "login_form">
  <form method="post" action="login.php">
      <h1>PRIJAVA</h1>
            <div class="form-group row" id="kor_ime">
                  <label for="inputUsername" class="col-sm-2 form-control-
                  label" required">Korisničko ime</label>
                  <div class="col-sm-5">
                        <input type="username" class="form-control"
                        id="inputUsername" name="username"
                        placeholder="Korisničko ime"">
                  </div>
            </div>
            <div class="form-group row">
                  <label for="inputPassword" class="col-sm-2 form-control-
                  label" required>Lozinka</label>
                  <div class="col-sm-5">
                        <input type="password" class="form-control"
                        id="inputPassword" name="password"
                        placeholder="Lozinka">
                  </div>
            </div>
            <br><br>
            <input type="submit" value="Prijavi se" id="button">
            <br><br>
```

```
  </form>
</div>
</div>
<br><br><br><br><br><br>
```

Unutar skripte nalazi se forma koja podatke šalje POST metodom i izvršava se u skripti *login.php*. Forma se sastoji od dva polja za unos i gumba za potvrdu odnosno izvršavanje forme. Forma se nalazi unutar <*div*> blokova kojima je dodijeljen jedinstveni naziv i samim time im je u podnožju skripte pridružen CSS kod kojim je definiran izgled forme. Nakon što korisnik unese potrebne podatke i potvrdi formu sustav otvara skriptu *login.php*.

U nastavku slijedi prikaz skripte *login.php*:

```
<?php
	include 'header.php';
	if (isset($_POST['username']))  {
		$username = $_POST['username'];
		$password = $_POST['password'];

		otvoriBP();

		if (login($username,$password) == true) {
			header ("Location:index.php");
		}
	}
	else {
		echo '<p4><br><br>Korisnik ne postoji!</p4>';
	}
?>
```

Skripta prvo poziva skriptu *header.php* pomoću iskaza *include*. Iskaz *include* uzima sav kod koji se nalazi u navedenoj skripti, vrlo je korisna iz razloga što se često javlja potreba za korištenjem istog HTML ili PHP koda na različitim mjestima unutar internet aplikacije. Zatim dolazi uvjet u kojem se nalazi PHP ugrađena funkcija *isset()* koja provjerava da li postoji vrijednost u polju za unos korisničkog ime koja je poslana preko forme za unos iz prethodnog koraka. Ukoliko vrijednost postoji kreiraju se varijable *$username* i *$password*, u njih se spremaju vrijednosti koje je korisnik unio na formi. Poziva se funkcija *otvoriBP()* i samim time ostvaruje se konekcija s bazom podatak. Sljedeći uvjet poziva funkciju *login()* koja je kreirana

u skripti *bp.php* i prosljeđuju joj se dva parametra s vrijednošću prethodno kreiranih varijabli. Ukoliko funkcija vrati istinitu vrijednost korisnik će biti prijavljen u sustav i preusmjeren na početnu stranicu, ako funkcija vrati vrijednost laž ispisati će se poruka da korisnik ne postoji.

U nastavku slijedi prikaz funkcije *login():*

```
function login($username,$password) {
	$rj = $GLOBALS['bp']->query("SELECT user_id, first_name, last_name,
	utype FROM user WHERE log_name='$username' AND log_pass =
	'$password'");
	if ($rj->num_rows > 0) {
		list ($id, $fname, $lname, $utype) = mysqli_fetch_array($rj);

		session_start();

		$_SESSION['ak_id'] = $id;
		$_SESSION['ak'] = $username;
		$_SESSION['ak_name'] = $fname ." ". $lname;
		$_SESSION['utype'] = $utype;

		return true;
	}
	else {
		return false;
	}
}
```

Funkcija login prima dva parametra, korisničko ime i lozinku. U varijablu *$rj* sprema se vrijednost koju je vratio SQL upit. Upit vraća jedinstveni broj korisnika, ime, prezime i tip korisnika iz tablice *user* gdje je zadovoljen uvjet. Uvjet provjerava korisničko ime i lozinku koje je funkcija primila putem parametra i kao rezultat iz tablice će vratiti uvijek samo jedan redak u obliku niza. Kako bi se ti podaci mogli koristiti potrebno ih je spremiti u listu pomoću PHP ugrađene funkcije *list* i *mysqli_fetch_array* koja prolazi kroz članove niza i zapisuje ih u prethodno definiranu listu. Nakon toga se koristi PHP građena funkcija *session_start()* koja označava početak sesije u kojoj će podaci o korisniku biti spremljeni u super globalne varijable *$_SESSION*, čija je glavna karakteristika to da se mogu koristiti u bilo kojem dijelu koda gdje je pozvana skripta *bp.php*. Funkcija kao rješenje vraća vrijednost *true* ukoliko je korisnik unio točno korisničko ime i lozinku, u suprotnome SQL upit neće vratiti niti jedan redak i funkcija će vratiti vrijednost *false*.

6.3. Proces kupovine proizvoda

Korisnik dolaskom na modul „Proizvodi“ dobiva prikaz svih proizvoda koji se trenutno nalaze u bazi podataka u tablici *products.* U zaglavlju modula nalazi se filter o čijem odabiru ovisi koje će se kategorije proizvoda prikazati i od kojeg proizvođača. Dolaskom na modul, filteri su automatski postavljeni da prikazuju sve proizvode. Odabirom vrijednosti u nekom od filtera i pritiskom na tipku „Filtriraj“ stranica će se automatski osvježiti s novim vrijednostima i u filteru će ostati zapisana prethodno odabrana vrijednost. Aplikacija funkcionira na način da je proizvode moguće filtrirati samo po jednom od dva filtera.

U nastavku slijedi zaglavlje skripte *products.php*:

```
include 'header.php';

otvoriBP();

if(isset($_SESSION['utype'])){
    $utype = $_SESSION['utype'];
}
else {
    $utype = 0;
}
```

Prilikom pokretanja skripte *products.php* izvršava se provjera korisnika, odnosno tipa korisnika kako bi sustav u nastavku znao što prikazati korisniku u skladu s njegovim pravima. Provjera je kreirana pomoću uvjeta *if* i PHP ugrađene funkcije *isset()* koja provjera da li super globalna varijabla *utype* ima vrijednost. Kreira se lokalna varijabla *$utype* u koju se sprema vrijednost tipa korisnika iz super globalne varijable ako je uvjet ispunjen, u suprotnome poprima vrijednost nula i samim time najnižu razinu prava.

6.3.1. Filter

Filter se nalazi u zaglavlju modula „Proizvodi“, sastoji se od dva padajuća izbornika s mogućnošću odabira jedne vrijednosti i tipkom za potvrđivanje forme. Filter za odabir kategorije dohvaća podatke iz tablice *pro_type_2* koja se sastoji od jedinstvenog broja i naziva kategorije, dok se vrijednosti koje upit vrati kao rezultat spremaju u lokalnu varijablu *$kat*. Filter za odabir proizvođača dohvaća jedinstveni broj proizvođača i naziv iz tablice *producers*, a vrijednosti se spremaju u lokalnu varijablu *$pror*.

```
$kat = $GLOBALS['bp']->query("SELECT pro_type2_id, pro_type2_name FROM
pro_type2");
```

```
$pror = $GLOBALS['bp']->query("SELECT pror_id, pror_name FROM producers");
```

Unutar bloka s jedinstvenim nazivom nalazi se forma koja funkcionira na GET metodi i izvršava se u skripti u kojoj se nalazi.

U nastavku slijedi kod za prikaz filtera:

```
echo '
<div class = "filter">
      <div id="fnaslov"><h2>Filtriraj proizvode: </h2></div>
            <div id="fopcije">
                  <form method="GET name ="filter">
                        <select name="kat" id="fselect">
                        <option disabled selected>Odaberi
                        kategoriju</option>';
                        if (isset($_GET['kat'])){
                              $sel_kat = $_GET['kat'];
                              $sel_kat = $GLOBALS['bp']->query("SELECT
                              DISTINCT pro_type2_id, pro_type2_name FROM
                              pro_type2 WHERE pro_type2_id = $sel_kat");
                              while($sel = $sel_kat->fetch_assoc()){
                                    echo '<option
                                    value="'.$sel['pro_type2_id'].'"
                                    Selected disabled>
                                    '.$sel['pro_type2_name'].' </option>';
                              }
                        }
                        while($pod = $kat->fetch_assoc()){
                              echo '<option
                              value="'.$pod['pro_type2_id'].'">
                              '.$pod['pro_type2_name'].'</option>';
                        }
 echo '     </select>
            <select name="pror" id="fselect">
            <option disabled selected>Odaberi proizvođača</option>';
            if (isset($_GET['pror'])){
                  $sel_pror = $_GET['pror'];
                  $sel_pror = $GLOBALS['bp']->query("SELECT DISTINCT
                  pror_id, pror_name FROM producers WHERE pror_id =
                  $sel_pror");

                  while($sel = $sel_pror->fetch_assoc()){
                        echo '<option value="'.$sel['pror_id'].'" selected
                        disabled>'.$sel['pror_name'].'</option>';
                  }
```

```
            }
            while($pod = $pror->fetch_assoc()){
                  echo '<option value="'.$pod['pror_id'].'">
                  '.$pod['pror_name'].'</option>';
            }
echo '       </select>
             <input type="submit" value="Filtriraj" id="fbtn">
             </form>
      </div>
</div>
```

Uvjetom *if* ispituje se da li postoji super globalna varijabla *$_GET['kat']* koja se šalje GET metodom. Prilikom prvog pokretanja skripte ovaj uvjet neće biti ispunjen jer nije odabrana vrijednost u filteru za odabir kategorije, te će se kod nastaviti dalje izvršavati. Slijedi petlja *while* kojom će se u filteru za odabir kategorije ispisati sve kategorije proizvoda koje se nalaze u tablici *pro_type2*. Ukoliko je korisnik odabrao vrijednost u filteru, uvjet će biti ispunjen i vrijednost parametra će se spremiti u lokalnu varijablu $sel_kat. U tom slučaju upit za dohvat podataka će u uvjetu ispitivati da li je jedinstvena vrijednost kategorije proizvoda jednaka vrijednosti lokalne varijable *$sel_kat* i kao rezultat vratiti jedan redak. Ovakav proces rezultira ispisivanjem filtera na način da odabrana kategorija bude automatski odabrana i korisniku se neće nuditi mogućnost da ju ponovno odabere. Identično rješenje primijenjeno je i kod filtera za odabir proizvođača, dok se na kraju nalazi gumb koji potvrđuje forme.

6.3.2. Popis proizvoda

Proizvodi se ispisuju ovisno o tome da li je korisnik odabrao vrijednost u filteru. Prvi uvjet ispituje da li postoje super globalne varijabla *$_GET['kat']* i *$_GET['pror']*, ukoliko ne postoje uvjet neće biti ispunjen i izvršiti će se dio koda u iskazu bloka *else*.

U nastavku slijedi kod za prikaz ispisa proizvoda bez filtera:

```
while($pod = $filter->fetch_assoc()){
echo '
<div class = "products">
      <div class = "product" text-center>
      <form method="GET" action ="insert_cart.php">
            <a href="product.php?pro_id='.$pod['pro_id'].'"><div
            class="product-name" id="name">'.$pod['pro_name'].'</div></a>
            <div class="product-price" id="price">'.$pod['price'].'
            kn</div>
```

```
            <a href="product.php?pro_id='.$pod['pro_id'].'"><div
            class="product-picture"><img src="products/'.$pod['picture'].'"
            id="pic""></div></a>';
            if ($utype !== 0) {
echo '            <div class="kosarica"><input type = "submit"
                  id="kosarica" value="Dodaj u košaricu"></div>';
            }
echo '      <input type="hidden" value="'.$pod['pro_id'].'" name="pro_id">
            <input type="hidden" value="'.$pod['price'].'" name="price">
        </form>
        <br>
        </div>
</div>';
}
```

Proizvodi se dohvaćaju pomoću *while* petlje koja prolazi kroz sve retke u tablici *products* i ispisuju se unutar forme koja podatke šalje GET metodom i izvršava se u skripti *insert_cart.php*. Forma se nalazi unutar blokova *products* i *product*. Unutar forme nalazi se uvjet koji provjerava tip korisnika i ovisno o tome ispisuje se gumb koji potvrđuje formu odnosno dodaje proizvod u košaricu. Navedeni uvjet kreiran je kako bi se onemogućilo ne registriranom korisniku dodavanje proizvoda u košaricu.

U nastavku slijedi kod za ispis proizvoda s filterom:

```
if (isset($_GET['kat']) or isset($_GET['pror'])) {
      $kat = "pro_type = pro_type";
      $pror = "pro_pror_id = pro_pror_id";

      if (isset($_GET['kat'])) {
        $kat_id = $_GET['kat'];
        $kat = "pro_type = '$kat_id'";
      }
      if (isset($_GET['pror'])) {
        $pror_id = $_GET['pror'];
        $pror = "pro_pror_id = '$pror_id'";
      }
      $filter = $GLOBALS['bp']->query("SELECT pro_id, pro_name, picture,
      price FROM products WHERE $kat AND $pror");
      while($pod = $filter->fetch_assoc()){
echo '
      <div class = "products">
```

```
      <div class = "product" text-center>
            <form method="GET" action ="insert_cart.php">
                  <a href="product.php?pro_id='.$pod['pro_id'].'"><div
                  class="product-name"
                  id="name">'.$pod['pro_name'].'</div></a>
                  <div class="product-price" id="price">'.$pod['price'].'
                  kn</div>
                  <a href="product.php?pro_id='.$pod['pro_id'].'"><div
                  class="product-picture"><img
                  src="products/'.$pod['picture'].'" id="pic""></div></a>';
                  if ($utype !== 0) {
echo '                  <div class="kosarica"><input type = "submit"
                        id="kosarica" value="Dodaj u košaricu"></div>';
                  }
echo '            <input type = "hidden" value="'.$pod['pro_id'].'"
                  name="pro_id">
                  <input type = "hidden" value="'.$pod['price'].'"
                  name="price">
              </form>
              <br>
       </div>
       </div>';
      }
}
```

U slučaju da je korisnik odabrao vrijednost u filteru, uvjet će biti ispunjen i kreirati će se lokalne varijable *$kat* i *$pror* kojima se dodjeljuje vrijednost znakovnog niza koji će se kasnije koristiti u SQL upitu za dohvat podataka. Znakovni nizovi su zapravo dio SQL upita koji će se nalaziti u WHERE uvjetu, a kako se ne zna koji je točno filter odabran u lokalne varijable spremljen je znakovni niz koji neće promijeniti upit za dohvat podataka, odnosno i dalje će obuhvatiti sve retke u tablici. U nastavku se nalaze dva uvjeta koja ispituju postojanje super globalnih varijabli zasebno i time se utvrđuje kojem filteru je dodijeljena vrijednost. Ako super globalna varijabla *$_GET['kat']* postoji, uvjet će biti ispunjen. Zatim se kreira lokalna varijabla *$kat_id* koja prima vrijednost super globalne varijable *$_GET['kat']*. Lokalna varijabla *$kat* koja je prethodno kreirana prima novu vrijednost znakovnog niza koji će nakon ove promjene u WHERE uvjetu SQL upita koji slijedi vratiti isključivo proizvode s jedinstvenim brojem proizvođača koji je odabran u filteru. Cijeli postupak se ponavlja i za super globalnu varijablu *$_GET['pror']*, odnosno za proizvođače. SQL upit koji će dohvatiti podatke za ispis proizvoda sprema se u lokalnu varijablu *$filter*. Upit u WHERE uvjetu ima lokalne varijable *$kat* i *$pror* koje će omogućiti dohvat podataka prema prethodno odabranim vrijednostima u filteru.

6.3.3. Dodavanje proizvoda u košaricu

Nakon što korisnik pritisne gumb „Dodaj u košaricu“ iz forme koja se nalazi u skripti *products.php* i pokreće se skripta *insert_cart.php*. Forma GET metodom šalje dva parametra koja sadrže jedinstveni broj proizvoda i cijenu.

U nastavku slijedi kod funkcija za provjeru narudžbe i proizvoda:

```
include 'header.php';

$pro_id = $_GET['pro_id'];
$price = $_GET['price'];
$quan = 1;
$user_id = $_SESSION['ak_id'];
$order_status = 1;
$order_id = 0;

otvoriBP();

function provjera_order($user_id) {
	$rez = $GLOBALS['bp']->query("SELECT order_id FROM orders ord WHERE
	ord.order_user_id = $user_id AND order_status = 1");

	while ($pod = $rez->fetch_assoc()){
		$order_id = $pod['order_id'];
	}

	return $order_id;
}

function provjera_product ($pro_id, $order_id) {
	$rez = $GLOBALS['bp']->query("SELECT oi_quantity FROM order_items
	WHERE order_id = $order_id AND order_product_id = $pro_id");

	while ($pod = $rez->fetch_assoc()){
		$quan = $pod['oi_quantity'];
	}

	return $quan;
}
```

Vrijednosti tih parametara spremljenih u super globalne varijable koje skripta prima preko parametra spremaju se u lokalne varijable *$pro_id* i *$price*. Kreirane su lokalne varijable *$quan* koja označava količinu i dobiva vrijednost jedan, *$order_status* označava status narudžbe i dobiva vrijednost jedan i *$order_id* čija je vrijednost postavljena na nulu . Varijabla *$user_id* prima vrijednost iz super globalne varijable *$_SESSION['ak_id']* gdje se nalazi jedinstveni broj aktivnog korisnika koji je prijavljen u sustav. Poziva se funkcija *otvoriBP()* kako bi se mogli izvršavati upiti prema bazi podataka u nastavku i kreiraju se dvije nove funkcije za provjeru narudžbe i proizvoda.

Funkcija za provjeru narudžbe preko parametra prima vrijednost lokalne varijable *$user_id*, provjerava da li za navedenog korisnika u sustavu postoji otvorena ili ne dovršena narudžba. Provjera se izvršava u tablici *orders* gdje se nalaze sve dovršene i ne dovršene narudžbe korisnika. Ukoliko se u tablici nalazi zapis za korisnika sa statusom narudžbe jednako jedan, vrijednost jedinstvenog broj te narudžbe upisuje se u lokalnu varijablu *$order_id* koju funkcija ujedno vraća kao rješenje.

Funkcija za provjeru proizvoda preko parametra prima vrijednost jedinstvenog broja proizvoda i jedinstvenog broja narudžbe iz prethodno kreiranih lokalnih varijabli. Zatim u tablici *order_items* provjera količinu za proizvod na otvorenoj narudžbi. Rezultat upita sprema se u lokalnu varijablu *$quan* i vraća se kao rezultat funkcije.

U nastavku slijedi kod sa sadržajem skripte *insert_chart.php*:

```
$order_id = provjera_order($user_id);

if ($order_id == null) {
      $ins = $GLOBALS['bp']->query("INSERT INTO orders (order_user_id,
      order_status, order_date) VALUES ($user_id, $order_status, now())");
}

$order_id = provjera_order($user_id);
$quan = provjera_product($pro_id, $order_id);

if ($quan == null) {
      $ins = $GLOBALS['bp']->query("INSERT INTO order_items
      (order_product_id, order_id, oi_quantity, oi_price) VALUES ($pro_id,
      $order_id, 1, $price)");
}

else {
```

```
        $quan = $quan + 1;


        $upd = $GLOBALS['bp']->query("UPDATE order_items SET oi_quantity =
        $quan WHERE order_id = $order_id AND order_product_id = $pro_id");
}
zatvoriBP();
header("Location: cart.php");
```

U prethodnom dijelu kreirane su funkcije koje je potrebno pozvati kako bi se kod unutar bloka funkcije izvršio i vratio rezultat. Pozvana je funkcija za provjeru narudžbi i proslijeđen joj je parametar lokalne varijable *$user_id*, rezultat koji funkcija vraća spremljen je u lokalnu varijablu *$order_id*. Ispituje se da li lokalna varijabla *$order_id* ima vrijednost različitu od *null*, u slučaju da je tvrdnja istinita izvršava se SQL upit za unos podataka u *orders* tablicu. Ponovno se poziva funkcija za provjeru narudžbi i vrijednost se sprema u lokalnu varijablu *$order_id*. Poziva se funkcija za provjeru proizvoda koja prima dva parametra, jedinstveni broj proizvoda i jedinstveni broj narudžbe. Rezultat funkcije sprema se u lokalnu varijablu $quan i provjera se da li je vrijednost različita od null. U slučaju da je tvrdnja istinita izvršava se SQL naredba za unos podataka u tablicu *order_items* gdje se polju *oi_quantity* dodjeljuje vrijednost jedan. U slučaju da tvrdnja nije istinita, vrijednost lokalne varijable *$quan* se uvećava za jedan i izvršava se SQL upit koji ažurira polje *oi_quantity* u tablici *order_items*. Polje oi_quantity označava količinu naručenih proizvoda, dok prethodni uvjet osigurava funkcionalnost aplikacije u slučaju da korisnik u košaricu doda proizvod koji se ondje već nalazi. Na kraju skripte poziva se funkcija za zatvaranje veze s bazom podataka i automatski se preusmjerava na skriptu *cart.php*.

6.3.4. Košarica

Skripta *cart.php* se koristi za prikaz košarice, pružanje određene funkcionalnosti i pozivanje drugih PHP skripti. U zaglavlju uključuje skriptu *header.php* i poziva funkciju za ostvarivanje veze s bazom podataka. Kreirane su lokalne varijable *$price*, *$order_id*, *$id* i *$cart*. U lokalnu varijablu *$cart* spremljena je vrijednost SQL upita koji dohvaća naziv, cijenu, količinu, jedinstveni broj proizvoda i jedinstveni broj narudžbe.

U nastavku slijedi kod zaglavlja skripte *cart.php*:

```
<?php
include 'header.php';
otvoriBP();

$price = 0;
```

```
$order_id = 0;
$id = $_SESSION['ak_id'];
$cart = $GLOBALS['bp']->query(" SELECT pro.pro_name, pro.price,
ori.oi_quantity, pro.pro_id, ord.order_id FROM orders ord, order_items ori,
products pro WHERE pro.pro_id = ori.order_product_id AND ord.order_id =
ori.order_id AND ord.order_status = 1 AND ord.order_user_id = 11");
```

Sadržaj košarice nalazi se unutar forme koja podatke šalje GET metodom i izvršava se u samoj skripti. Podaci u košarici nalaze se u tablici koja u zaglavlju ima proizvod, cijenu i količinu. Sadržaj tablice se ispisuje pomoću *while* petlje unutar koje su kreirane lokalne varijable *$order_id* i *$pro_id* kojima su pridružene vrijednosti iz super globalnih varijabli dok se lokalna varijabla *$price* kalkulira. U kolonama se ispisuju nazivi proizvoda, količine iz vrijednosti koju vraća upit s početka skripte i cijena iz lokalne varijable koja je prethodno iskalkulirana. Posljednja kolona ispisuje se u obliku slike i predstavlja vezu na skriptu *cart_del.php* kojoj šalje parametre jedinstvenog broj proizvoda i jedinstvenog broja narudžbe preko URL-a stranice. Slijedi uvjet koji ispituje da li postoji super globalna varijabla koja se kreira ukoliko korisnik potvrdi formu pritiskom gumba pod nazivom „Ažuriraj košaricu“. Ukoliko je uvjet ispunjen izvršava se SQL naredba za ažuriranje polja količine u tablici *order_items* i automatski se ažurira. U posljednjem redu tablice nalaze se dvije kolone, u prvoj je tekst „Ukupno“ dok se u drugoj ispisuje lokalna varijabla *$price*. Zaglavlje skripte sadrži gumbe za ažuriranje košarice, dovršetak narudžbe i brisanje sadržaja košarice.

U nastavku slijedi kod sadržaja skripte *cart.php*:

```
echo '
<div class="cart">
      <div class = "cart-title" text-center>
            <h1>Košarica</h1>
      </div>
      <form method="GET">
            <div id = "cart-body">
                  <table id="table">
                        <tr>
                            <th>Proizvod</th>
                            <th id="price">Cijena/kn</th>
                            <th id="quantity">Količina</th>
                            <th>  </th>
                        </tr>';

                        while($pod = $cart->fetch_assoc()){
```

```
                        $price = $price + $pod['price'] *
                        $pod['oi_quantity'];
                        $order_id = $pod['order_id'];
                        $pro_id = $pod['pro_id'];

echo '
                        <tr>
                        <td>'.$pod['pro_name'].'</td>
                        <td id="price">'.$pod['price'].' kn</td>

                        <td id="quantity"><input type="text"
                        name="quan('.$pod['pro_id'].')"
                        value="'.$pod['oi_quantity'].'" style="width:
                        25px; color:black; text-align:center"> </td>
                        <td id="del2"><a href =
                        "cart_del.php?pro='.$pod['pro_id'].'&ord='.$po
                        d['order_id'].'"><img src="delete.png"
                        id="del"></a></td>
                        </tr>';

                        if (isset($_GET['submit'])) {

                        $quan = $_GET['quan('.$pod['pro_id'].')'];
                        $upd = $GLOBALS['bp']->query("UPDATE
                        order_items SET oi_quantity = $quan WHERE
                        order_product_id = $pro_id AND order_id =
                        $order_id");
                        header("Location: cart.php");
                        }
                    }
echo '
                        <tr id="line">
                              <td><div><p>Ukupno:</p></div></td>
                              <td><div id="price"><p>'.$price.'
                              kn</p></div></td>
                              <td><div id="price"><p></p></div></td>
                              <td><div id="price"><p></p></div></td>
                        </tr>
                  </table>
                <input type="submit" name="submit" value="Ažuriraj
                košaricu" id="btn">
                <p id="sub"><a id="sub2"
                href="cart_submit.php?ord='.$order_id.'">Dovrši
                narudžbu</a></p>
```

```
                <p id="del_all"><a id="del_all2"
                href="cart_del_all.php?ord='.$order_id.'"> Isprazni
                košaricu</a></p>
            </form>
        </div>
</div>
<br><br><br>';
```

Gumb za ažuriranje košarice ponovno pokreće skriptu i potvrđuje formu koja se proteže kroz čitavu skriptu. Gumb za brisanje sadržaja košarice kreiran je kao link na skriptu *cart_del_all.php* i šalje jedinstveni broj narudžbe u parametru. Navedeni se parametar sprema u lokalnu varijablu *$order_id* i poziva funkciju za povezivanje s bazom podataka. Izvršava upit za brisanje retka u tablici *order_itmes* čiji jedinstveni broj narudžbe odgovara vrijednost iz lokalne varijable.

U nastavku slijedi kod skripte *cart_del_all.php*:

```
<?php
include 'bp.php';
otvoriBP();


$order_id = $_GET['ord'];
$del = $GLOBALS['bp']->query("DELETE FROM order_items WHERE order_id =
$order_id");


header("Location: cart.php");
?>
```

6.3.5. Završetak kupovine

Nakon što korisnik u košarici odabere opciju za dovršetak kupovine sustav ga automatski preusmjerava na skriptu *cart_submit.php,* te joj se prosljeđuje parametar s vrijednosti jedinstvenog broja narudžbe. U navedenoj skripti nalaze se SQL upiti za upisivanje podataka u tablicu *invoice* gdje se statusu dodjeljuje vrijednost jedan i ažurira se status narudžbe u tablici *orders*. Nakon provedenih operacija automatski se preusmjerava na skriptu *shipment.php* kojoj se u parametru prosljeđuje vrijednost jedinstvenog broja narudžbe.

U nastavku slijedi kod skripte *cart_submit.php*:

```
<?php
include 'bp.php';
otvoriBP();
```

```
$order_id = $_GET['ord'];
$ins_invoice = $GLOBALS['bp']->query("INSERT INTO invoice(invoice_order_id,
i_status_id, invoice_date) VALUES ($order_id, 1, now())");
$upd_orders = $GLOBALS['bp']->query("UPDATE orders SET order_status = 2
WHERE order_id = $order_id");

header("Location: shipment.php?ord=$order_id");
?>
```

Skripta *shipment.php* sadrži formu za unos podatka o dostavi i tipku za nastavak plaćanja. Forma se izvršava u skripti *ship_submit.php* kojoj se GET metodom prosljeđuje vrijednost jedinstvenog broja narudžbe preko skrivenog polja za unos. U skripti su kreirane lokalne varijable *$order_id* i *$adress* koje primaju vrijednosti iz super globalnih varijabli te *$ship_id* kojoj se inicijalno dodjeljuje vrijednost nula. Zatim slijede SQL naredbe, prvo se upisuje redak u tablicu *shipping* gdje se kao datum unosa sprema trenutno vrijeme pozivanjem ugrađene funkcije *now()*, nakon toga se upitom dohvaća prethodno kreirani redak u tablici. U petlji *while* izvršava se SQL upit za ažuriranje polja *ship_id* u tablici *invoice* koji odgovara vrijednosti jedinstvenog broja narudžbe. U zaglavlju sustav automatski preusmjerava korisnika na skriptu *payment.php* kojoj se u parametru šalje vrijednost jedinstvenog broja narudžbe.

U nastavku slijedi kod skripte *ship_submit.php*:

```
<?php
include 'bp.php';
otvoriBP();

$order_id = $_GET['ord'];
$adress = $_GET['adress'];
$ship_id = 0;

$ins_ship = $GLOBALS['bp']->query("INSERT INTO shipping (ship_date,
ship_location, order_id) VALUES (now(), '$adress', $order_id)");

$sel_ship = $GLOBALS['bp']->query("SELECT ship_id FROM shipping WHERE
order_id = $order_id");

while($pod = $sel_ship->fetch_assoc()){
      $ship_id = $pod['ship_id'];
      $upd_invoice = $GLOBALS['bp']->query("UPDATE invoice SET ship_id =
      $ship_id WHERE invoice_order_id = $order_id");
}
```

```
header("Location: payment.php?ord=$order_id");
?>
```

Skripta *payment.php* sastoji se od dvije forme koje koriste GET metodu za slanje podataka i izvršavaju se u skripti *pay_submit.php*. Unutar forme se nalaze dva gumba kojima je dodijeljen jedinstveni naziv, nakon njih dolaze dva bloka koja su skrivena. U zaglavlju skripte nalazi se *javascript* kod koji se referencira na jedinstvene nazive koji su dodijeljeni gumbima. Blok *javascript* naredbi omogućuje prikaz bloka elemenata ovisno o tome koji gumb korisnik pritisne.

U nastavku slijedi JavaScript kod u skripti *payment.php*:

```
<script>
      $(document).ready(function(){
            $("#cart").click(function() {
                  $("#cart_body").toggle();
                  $("#cash").toggle();
            });

            $("#cash").click(function() {
                  $("#cash_body").toggle();
                  $("#cart").toggle();
            });
        });
 </script>
```

U skripti *pay_submit.php* kreirane su tri lokalne varijable i izvršava se tri SQL upita. Prvo se izvršava unos podataka u tablicu *payment*, zatim se iz navedene tablice dohvaća jedinstveni broj plaćanja. Vrijednost koju upit vrati upisuje se u lokalnu varijablu *$pay_id* i ažurira se polje jedinstvenog broja plaćanja u tablici *invoice*. Skripta preusmjerava korisnika na početnu stranice i time završava proces kupnje.

U nastavku slijedi kod skripte *pay_submit.php*:

```
<?php
include 'header.php';

$order_id = $_GET['ord'];
$method = $_GET['method'];
$cart_num = $_GET['cart_num'];
```

```
otvoriBP();


$pay_ins = $GLOBALS['bp']->query("INSERT INTO payment (pay_method,
pay_inovice_num, order_id) VALUES ('$method', '$cart_num', '$order_id')");


$pay_sel = $GLOBALS['bp']->query("SELECT payment_id FROM payment WHERE
order_id = '$order_id'");


while($pod = $pay_sel->fetch_assoc()){
      $pay_id = $pod['payment_id'];
}


$inv_upd = $GLOBALS['bp']->query("UPDATE invoice SET pay_id = '$pay_id'
WHERE invoice_order_id = '$order_id'");


zatvoriBP();


header("Location: index.php");
?>
```

7. Zaključak

Prilikom izrade rada cilj je bio dobiti funkcionalnu Internet trgovine bez korištenja gotovih rješenja kojih danas ima mnoštvo na internetu i uvelike skraćuju vrijeme izrade i održavanje iste. Dijelovi aplikacije nisu izrađeni u skladu s najboljom praksom ali prikazani su na način logičkog razmišljanja o poslovnim procesima i traženjem rješenja za iste korištenjem programskog jezika PHP. Jezik je korišten bez dodatnih okvira koji također mogu olakšati proces izrade i održavanja. Rad polazi od baze podataka koja je temelj čitave aplikacije i završava ekranskim pregledima i dijelovima koda, time je obuhvaćen cjelokupni proces izrade internet trgovine i njene funkcionalnosti. Bitno je naglasiti kako se određena poslovna rješenja putem programskog jezika mogu riješiti na mnoštvo različitih načina koji mogu biti kombinacija autorove mašte i znanja. Prilikom izrade važno je razmišljati o održavanju i nadogradnji dijelova koda i samim time može se mjeriti kvaliteta projekta. U radu su neke greške zamijećene u kasnoj fazi izrade i procijenjeno je kako je jednostavnije nastaviti nego raditi izmjene iz razloga što se pogreška nalazi na previše mjesta ili veže na sebe previše drugih dijelova aplikacije. Za izradu projekta jednako je važno poznavanje tehnologija koje će se koristiti kao i sam poslovni proces. Dobrim poznavanjem poslovnih procesa izbjegavaju se iznenadne i ne predviđene situacije u procesu izrade i lakše se kreiraju održiva rješenja, no prethodna tvrdnja uvelike ovisi o iskustvu osobe koja izrađuje aplikaciju. U radu se stavlja naglasak na učenje tehnologija i poslovnih procesa dok je kvaliteta same aplikacije kao takve manje bitna.

Popis literature

[1] Doc. dr. sc. Kornelije Rabuzin, Uvod u SQL, Varaždin 2011.

[2] phpMyAdmin: „Bringing MySQL to the web“, 2003-2019. [Na internetu]. Dostupno: https://www.phpmyadmin.net/ [pristupano 22.6.2019]

[3] Mark Pilgrim, HTML5 Spreman za upotrebu, Zagreb 2010.

[4] Prof. dr. sc. Dragutin Kermek: Izgradnja Web aplikacija . Dostupno: https://elfarchive1516.foi.hr/pluginfile.php/4589/mod_resource/content/1/predavanja/Kermek_IWA_03.pdf [pristupano 24.6.2019]

[5] Peter Gasston, Knjiga za CSS3: Vodič kroz budućnost web dizajna, Zagreb 2013.

[6] "Jedinice boje u jeziku CSS" (bez dat.). u Wikipedia, the Free Encyclopedia. [Na internetu]. Dostupno: https://hr.wikipedia.org/wiki/Jedinice_boje_u_jeziku_CSS [pristupano 23.6.2019]

[7] Bootstrap. [Na internetu]. Dostupno: https://getbootstrap.com/ [pristupano 23.6.2019]

[8] Shaley Powers, Naučite JavaScript, Zagreb 2010.

[9] Rasmus Lerdorf, Kevin Tatroe i Peter MacIntyre, Izrada dinamičkih Web stranica: Programiranje PHP, Zagreb 2015.

Popis slika