

Razvoj web trgovine "House of Wine"

Ljubović, Denis

Undergraduate thesis / Završni rad

2021

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **The Polytechnic of Rijeka / Veleučilište u Rijeci**

Permanent link / Trajna poveznica: <https://um.nsk.hr/um:nbn:hr:125:681579>

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

Download date / Datum preuzimanja: **2024-12-27**



Repository / Repozitorij:

[Polytechnic of Rijeka Digital Repository - DR PolyRi](#)



VELEUČILIŠTE U RIJECI

Denis Ljubović

RAZVOJ WEB TRGOVINE „HOUSE OF WINE“ (završni rad)

Rijeka, 2021.

VELEUČILIŠTE U RIJECI
Poslovni odjel
Preddiplomski stručni studij Informatika

RAZVOJ WEB TRGOVINE „HOUSE OF WINE“
(završni rad)

MENTOR

Izv. prof. dr. sc. Alen Jakupović, prof. v.š.

STUDENT

Denis Ljubović

MBS: 2422000016/15

Rijeka, lipanj 2021.

VEUČILIŠTE U RIJECI
Poslovni odjel
Rijeka, 15.3.2021.

ZADATAK
za završni rad

Pristupnik Denis Ljubović, MBS: 2422000016/15.

Studentu preddiplomskog stručnog studija Informatika izdaje se zadatak za završni rad –
tema završnog rada pod nazivom:

RAZVOJ WEB TRGOVINE „HOUSE OF WINE“

Sadržaj zadatka:

Opisati računalne programe korištene u razvoju web trgovine. Prikazati HTTP protokol. Detaljno opisati razvoj serverske strane web trgovine s naglaskom na relacijsku bazu podataka i korištene računalne jezike SQL i PHP. Detaljno opisati razvoj klijentske strane web trgovine s naglaskom na računalne jezike HTML, CSS i JavaScript. Prikazati i objasniti važnije algoritme implementirane u sklopu web trgovine. Prikazati uporabu razvijene web trgovine, te pojasniti njezin responzivni dizajn.

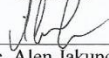
Preporuka:

Rad obraditi sukladno odredbama Pravilnika o završnom radu Veleučilišta u Rijeci.

Zadano: 15.3.2021.

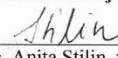
Predati do: 15.9.2021.

Mentor:



(izv.prof.dr.sc. Alen Jakupović, prof.v.š.)

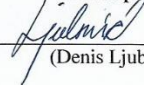
Pročelnik odjela:



(dr.sc. Anita Stilin, v. pred.)

Zadatak primio dana: 15.3.2021.

Pristupnik:



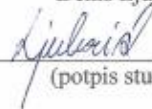
(Denis Ljubović)

Dostavlja se:
- mentoru
- pristupniku

IZJAVA

Izjavljujem da sam završni rad pod naslovom „Razvoj web trgovine „House of Wine““ izradio samostalno pod nadzorom i uz stručnu pomoć mentora prof. dr. sc. Alena Jakupovića.

Denis Ljubović



(potpis studenta)

Sažetak

Kroz ovaj završni rad bit će opisana izrada web stranice, to jest web trgovine (*engl. web shop*) pod nazivom „House of Wine“, koju otvaramo pomoću web preglednika (*engl. web browser*). Cilj web trgovine je olakšati i ubrzati kupovinu proizvoda koji se nude, bez potrebe fizičkog dolaska u trgovinu, te zapis i praćenje tih kupnji. Web trgovina funkcionira tako što kupac odabire proizvod ili proizvode koje želi kupiti, te bira ponuđene opcije dostave i plaćanja i tako u nekoliko koraka putem računala ostvaruje kupnju. Navedena web stranica napravljena je uz pomoć sljedećih tehnologija i programskih jezika: HTML, CSS, JavaScript, PHP i SQL uz relacijsku bazu podataka, Apache server i program XAMPP. Navedene tehnologije bit će detaljnije objašnjenje kroz sam završni rad, kao i sama funkcionalnost web trgovine zajedno sa svim njezinim značajkama.

Ključne riječi: Završni rad, web trgovina, programski jezik, HTML, CSS, JavaScript, PHP, SQL, relacijska baza podataka, Apache server.

Sadržaj

1. Uvod	1
2. Korišteni računalni programi.....	2
2.1. Microsoft Visual Studio Code	2
2.2. XAMPP	3
2.3. Web preglednik.....	4
3. HTTP protokol.....	5
3.1. GET i POST metode.....	6
3.1.1. GET	6
3.1.2. POST	8
4. Razvoj serverske strane web stranice	9
4.1. Definicija baze podataka	9
4.1.1. DMBS (Sustav za upravljanje bazama podataka)	9
4.2. Relacijska baza podataka.....	10
4.2.1. Tablica	12
4.2.2. Ključevi	12
4.2.3. Veze tablica	13
4.3. SQL.....	14
4.4. PHP	16
4.4.1. PHP sintaksa.....	17
4.4.2. PHP super-globalne varijable	18
4.4.3. \$_GET i \$_POST	19
4.4.4. PHP sesije.....	20

4.4.5. PHP funkcije.....	21
4.4.6. Spajanje na bazu podataka.....	25
5. Razvoj klijentske strane web stranice.....	27
5.1. HTML.....	27
5.1.1. HTML DOM model	30
5.1.2. Header.....	31
5.1.3. Navigation	32
5.1.4. Main.....	32
5.1.5. Footer.....	33
5.2. CSS	34
5.2.1. CSS sintaksa	35
5.2.2. CSS selektori	36
5.2.3. Dodavanje CSS datoteke u HTML dokument.....	40
5.3. JavaScript	42
5.3.1. JavaScript sintaksa.....	43
5.3.2. JSON.....	45
5.3.3. JavaScript – HTML DOM metode	46
5.3.4. JavaScript funkcije	48
6. Primjeri kôda iz projekta	51
6.1. Algoritam.....	51
6.1.1. PHP i SQL algoritmi	52
6.1.2. JavaScript algoritam	57
6.2. HTML i CSS primjeri kôda.....	58
7. Web trgovina „House of Wine“	62

7.1. Struktura stranice.....	62
7.1.1. Datoteka header.php	63
7.1.2. Datoteka footer.php	64
7.2. Putanja korisnika kroz kreiranje narudžbe	64
7.2.1. Naslovna (početna) stranica.....	65
7.2.2. Stranica za registraciju.....	65
7.2.3. Kreiranje narudžbe	67
7.2.4. Pregled detalja proizvoda	69
7.2.5. Košarica	70
7.2.6. Završetak kupnje.....	71
7.2.7. Potvrda kupnje	73
7.3. Putanja korisnika kroz profil	75
7.3.1. Prijava na račun	75
7.3.2. Profil	76
7.3.3. Izmjena osobnih podataka na profilu	77
8. Responzivni dizajn	79
8.1. HTML i CSS kôd za responzivan dizajn.....	80
8.2. Primjer responzivnosti na projektu.....	81
9. Zaključak	83
10. Popis korištenih kratica i anagrama.....	84
11. Literatura	85
12. Popis slika.....	87

1. Uvod

U ovom radu opisat ćemo cjelokupni proces izrade web trgovine pod nazivom „House of Wine“. Kroz projekt ćemo upoznati i razne tehnologije, kao i terminologiju vezanu za taj dio informatičke znanosti. Inspiracija za ovu temu za završni rad došla je od osobnog interesa za izradu web stranica, kao i za same tehnologije koje se koriste pri izradi istog. Sama tema vina, sireva i opreme odabrana je jer nema puno web trgovina koje nude takvu kombinaciju proizvoda. Stranica je u cijelosti izrađena bez tzv. okvira¹ (*engl. framework*). Kôd je napisan u programu Visual Studio Code, izdavatelja Microsoft, a pokretanje cijelog projekta izvršava se lokalno pomoću računalnog programa XAMPP i Apache servera. Sastoji se od serverske² (*engl. backend*) i klijentske³ (*engl. frontend*) strane/dijela. Za klijentsku stranu i izgled korisničkog sučelja (*engl. user interface*) korištene su tehnologije HTML (*engl. Hyper Text Markup Language*), CSS (*engl. Cascading Style Sheet*) i JavaScript. S druge strane, za izradu serverske strane, korišten je PHP (*engl. PHP: Hypertext Preprocessor*) u kombinaciji sa SQL-om (*engl. Structured Query Language*), za upravljanje dinamičnim podacima unutar MariaDB relacijske baze podataka izrađene pomoću programa phpMyAdmin, koji se prikazuju na klijentskoj strani. Sve navedene tehnologije i bit će detaljnije objašnjene kroz primjere u drugim poglavljima završnog rada kao i sama funkcionalnost, te povezivanje serverske i klijentske strane.

¹ Okvir ili *engl. framework* je platforma, okruženje za razvoj web aplikacija (Framework Definition: <https://techterms.com/definition/framework>, 2021.)

² Serverska strana ili *engl. backend* je dio programa koji nije vidljiv korisniku, to je sloj programa u kojem se nalaze podaci (Backend Definition: <https://techterms.com/definition/backend>, 2021.)

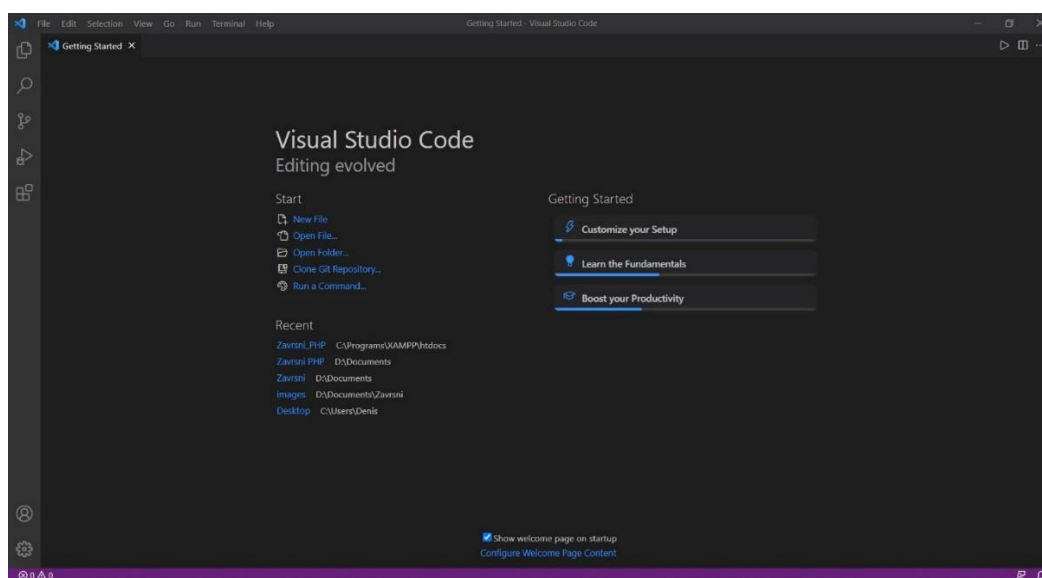
³ Klijentska strana ili *engl. frontend* je dio programa pomoću kojeg korisnik vrši interakciju, vidljiv je i često je sinonim za korisničko sučelje (Frontend Definition: <https://techterms.com/definition/frontend>, 2021.)

2. Korišteni računalni programi

2.1. Microsoft Visual Studio Code

Visual Studio Code je besplatni uređivač kôda⁴ (engl. *code/text editor*) tvrtke Microsoft u kojem je pisan sav kôd za navedenu web stranicu.

Slika 1: Sučelje programa Microsoft Visual Studio Code



Izvor: izradio autor

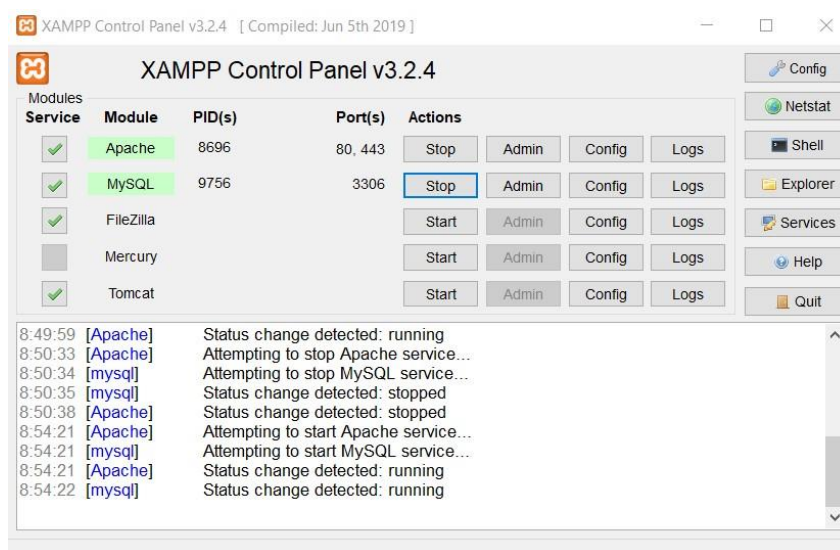
Visual Studio Code je program bogat različitim funkcionalnostima i mogućnostima uređivanja sučelja, a moguće ga je u velikim razmjerima prilagoditi prema korisnikovim željama. Također, program sadrži opciju različitih ekstenzija za različite programske jezike, koje dodatno obogaćuju funkcionalnosti i izgled samog sučelja.

⁴ Uređivač kôda je bilo koji računalni program kojim možemo pisati i mijenjati tekst (Text Editor Definition: <https://techterms.com/definition/texteditor>, 2021.)

2.2. XAMPP

XAMPP je besplatni softver otvorenog tipa⁵ (*engl. free open-source*) i višeplatformsko⁶ (*engl. cross-platform*) web server rješenje razvijeno od strane Apache Friends-a. Sastoji se od Apache HTTP (*engl. Hyper Text Transfer Protocol*) servera i MariaDB baze podataka, te prevoditelja skripti pisanih u PHP i Perl programskim jezicima.

Slika 2: Korisničko sučelje programa XAMPP



Izvor: izradio autor

U ovom završnom radu konkretno služi za lokalno⁷ pokretanje PHP kôda na Apache serveru, te MySQL baze podataka.

⁵ Softver otvorenog tipa odnosi se na program čiji je kôd javno dostupan

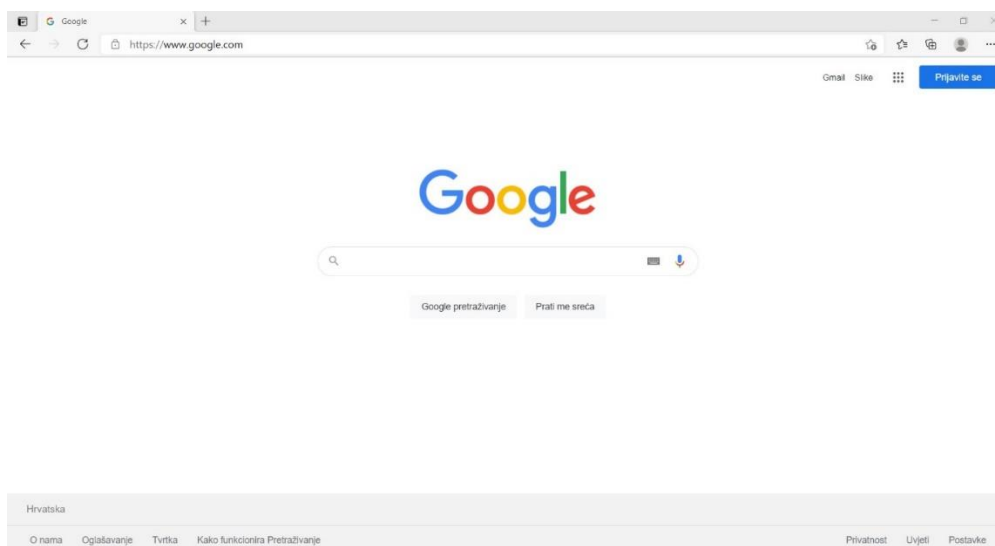
⁶ Višeplatformski znači da ga je moguće koristiti na više različitih uređaja sa različitim sustavima

⁷ Lokalno znači da se pokreće unutar sustava jednog računala, nije dostupno preko javnog servera

2.3. Web preglednik

Za pregled web stranice potreban je web preglednik poput Firefox-a, Google Chrome-a, Microsoft Edge-a, Safarija i slično. Općenito, web preglednik je aplikacija (software) za pristup World Wide Webu⁸, a funkcionira na način da korisnik potražuje web stranicu s određenog web mjesta, a pretraživač „uzima“ potrebni sadržaj sa web servera i prikazuje zatražene podatke na korisnikovom uređaju.

Slika 3: Web preglednik Microsoft Edge



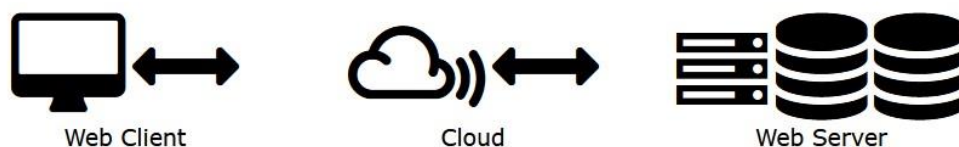
Izvor: izradio autor

⁸ WWW ili World Wide Web je dio Interneta, a sastoji se od web stranica koje čitamo preko web preglednika i služi se različitim protokolima i metodama za prijenos podataka, nastao je 1989., a uspostavio ga je Timothy Berners-Lee (WWW Definition: <https://techterms.com/definition/www>, 2021.)

3. HTTP protokol

HTTP je skraćenica za *engl. Hyper Text Transfer Protocol*, protokol zaslužan za komunikaciju računala klijenta i servera. Web stranice ne bi jednostavno bile moguće bez ovog protokola, koji djeluje na način, da se između klijenta i servera šalju zahtjevi (*engl. requests*) i odgovori (*engl. responses*).

Slika 4: Shematski prikaz slanja podatak kroz HTTP protokol



Izvor: https://www.w3schools.com/whatis/whatis_http.asp

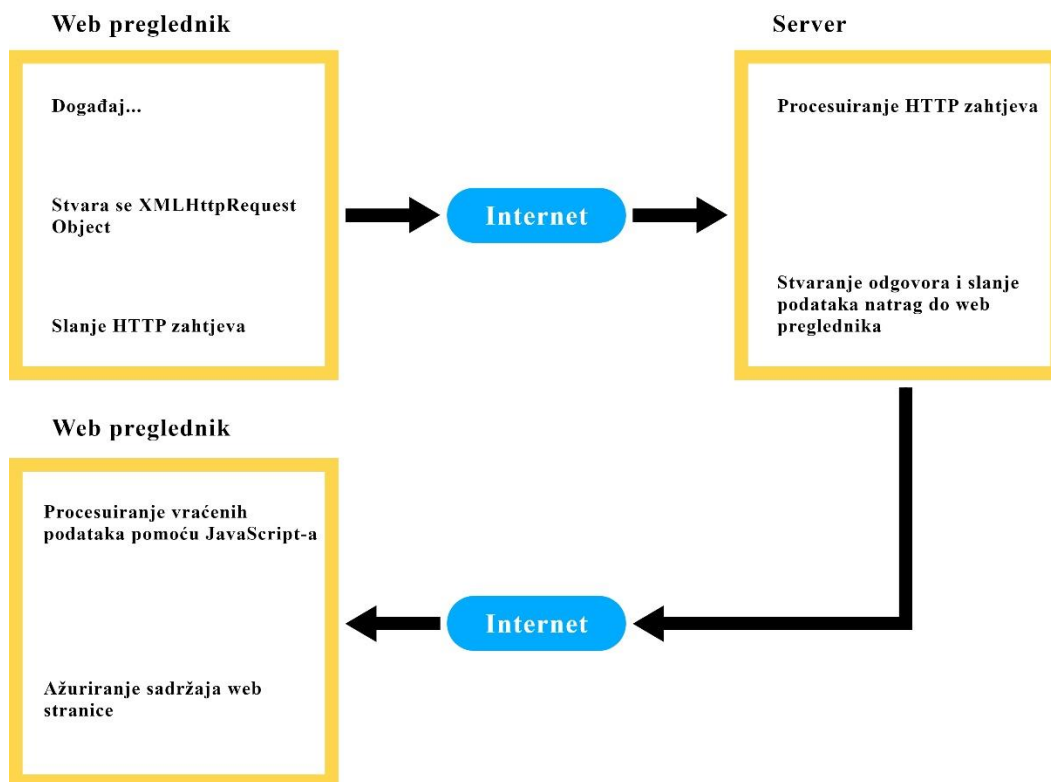
Način na koji ovaj protokol radi je sljedeći: Klijent (pomoću web preglednika) šalje HTTP zahtjev web stranici, nakon čega web server zaprima taj zahtjev i pokreće program koji procesuiraju taj zahtjev, zatim rezultat (*engl. output*) vraća korisniku kroz web preglednik, a klijent na sučelju web preglednika vidi taj odgovor koji je server poslao.

Svi web preglednici imaju ugrađen *XMLHttpRequest Object (XHR)*, JavaScript objekt koji služi za prijenos datoteka između web preglednika i web servera, a često se upotrebljava kako bi zatražio i primio podatke u svrhu modificiranja web stranice. XHR se ne koristi isključivo za HTTP protokol, isto kao što i podaci s kojima radi mogu biti različitih tipova poput HTML-a, CSS-a, XML-a, JSON-a, itd. XMR ažurira web stranice bez ponovnog učitavanja (*engl. reload*), potražuje i prima podatke sa servera nakon ažuriranja stranica i šalje podatke serveru.

Jedan ciklus izgleda tako da web preglednik traži HTML stranicu, zatim mu server vrati tu HTML datoteku, pa nakon toga potraži CSS datoteku koja oblikuje taj HTML i server ju vraća opet u taj web preglednik. Osim navedenog, datoteke koje se potražuju i koje server vraća mogu biti

različitih tipova poput JPG slika ili JavaScript kôda. (What is HTTP?: https://www.w3schools.com/whatis/whatis_http.asp, 2021.)

Slika 5: Prikaz ciklusa potražnje i vraćanja HTTP zahtjeva



Izvor: izradio autor prema shemi sa stranice: https://www.w3schools.com/whatis/whatis_http.asp

3.1. GET i POST metode

3.1.1. GET

GET metodu koristimo kada potražujemo podatke iz specifičnog izvora. To je jedna od najčešće korištenih metoda. Informacije koje šaljemo putem GET metode su vidljive svima jer se

ti podaci prosljeđuju kroz URL⁹ web stranice. GET metodu nikada ne bi trebali upotrebljavati za slanje osjetljivih ili privatnih podataka, kao što su npr. lozinke. Metoda GET također ima ograničenje u broju znakova od oko 2000 znakova. (HTTP Request Methods: https://www.w3schools.com/tags/ref_httpmethods.asp, PHP Form Handling: https://www.w3schools.com/PHP/php_forms.asp, 2021.)

Slika 6: Prosljeđivanje podataka kroz URL preko GET metode



Izvor: izradio autor

Na primjeru vidimo kako nakon što smo odabrali filter vina gumbom *CRNA*, u URL-u stranice se prosljeđio podatak *filter=red*. Nakon toga program provjerava vrijednost varijable *filter*, koja se sprema u globalnu varijablu `$_GET['filter']` (kasnije o tome u poglavlju o PHP super-globalnim varijablama). Ako je vrijednost jednaka *red*, program filtrira samo crna vina. Napomenimo da se engleska riječ *red* u prijevodu na hrvatski jezik koristi za crno vino. Slika ispod pokazuje dio kôda koji provjerava vrijednost globalne varijable `$_GET`. (PHP Superglobal - `$_GET`: https://www.w3schools.com/PHP/php_superglobals_get.asp, 2021.)

⁹ URL je akronim za *engl. Uniform Resource Locator*, a to je adresa specifične web stranice ili datoteke na Internetu. Sastoji se od prefiksa (*http://*), imena servera ili IP adrese i lokacije do datoteke (URL: <https://techterms.com/definition/url>, 2021.)

Slika 7: Primjer kôda gdje se koristi \$_GET varijabla

```
727     if (strpos($_GET['filter'], 'red') !== false) {
728         if ($filters == 0) {
729             $productQuery .= " AND ptype LIKE '%crno%'";
730             $filters = 1;
731         }
    }
```

Izvor: izradio autor

3.1.2. POST

Metoda POST, s druge strane, koristi se za slanje podataka serveru, a podaci koji se šalju spremljeni su unutar tijela HTTP zahtjeva i nevidljivi su drugima. POST metoda nema ograničenje, a zahtjevi koje šalje nikada se ne spremaju u povijest (*engl. history*) pretraživača. Ovu metodu koristimo i za neke druge naprednije funkcionalnosti pri podizanju datoteka na server. Za prikupljanje podataka POST metodom koristimo super-globalnu varijablu `$_POST` (više o tome u PHP poglavlju o super-globalnim varijablama). (PHP Superglobal - `$_POST`: https://www.w3schools.com/Php/php_superglobals_post.asp, 2021.)

Slika 8: Primjer kôda gdje se koriste \$_POST varijable

```
20     if (isset($_POST['register'])) {
21
22         //postavljanje vrijednosti varijabli koje unosi korisnik
23         $email = $_POST['email'];
24         $password = $_POST['password'];
25         $confirmPassword = $_POST['confirmPassword'];
26         $fname = $_POST['fname'];
27         $lname = $_POST['lname'];
28         $contact = $_POST['contact'];
29         $adress = $_POST['adress'];
30         $city = $_POST['city'];
31         $zip = $_POST['zip'];
    }
```

Izvor: izradio autor

Na primjeru vidimo kako se POST metoda koristi za spremanje podataka o kupcu (koje kupac unosi) u varijable. Ova *if* naredba izbora provjerava da li je korisnik kliknuo gumb za registraciju i pomoću POST metode sprema podatke u određene varijable koje se dalje šalju na server itd. (PHP Form, Handling: https://www.w3schools.com/PHP/php_forms.asp, 2021.)

4. Razvoj serverske strane web stranice

4.1. Definicija baze podataka

Baza podataka je organizirana i uređena cjelina elektroničkih podataka koji su povezani i pripremljeni tako da ih se može jednostavno koristiti. Unutar baze podataka se mogu obavljati razne akcije nad podacima poput spremanja, brisanja, sortiranja i uspoređivanja. Podaci koji su spremljeni u bazu imaju minimalnu redundanciju (zalihost)¹⁰. (Kaluža, 2008., 16.)

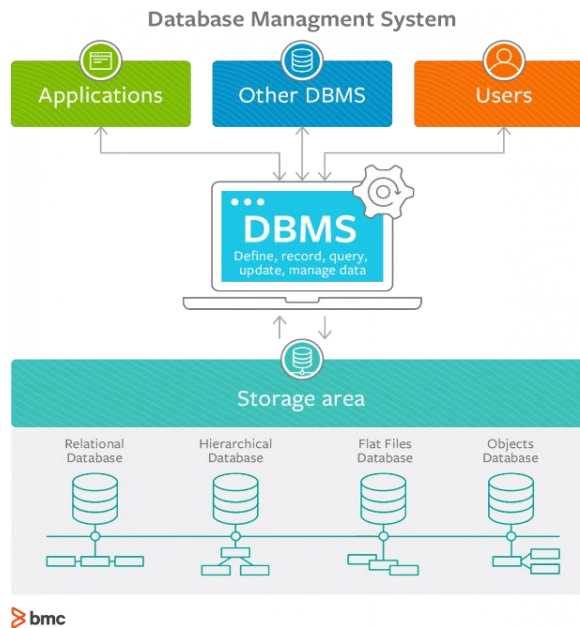
Razlikujemo tri osnovna modela baza podataka a to su hijerarhijski, mrežni i relacijski. U ovom završnom radu korištene su relacijske baze podataka, koje su danas gotovo isključivo u uporabi. Preciznije, korištena je baza podataka MariaDB. (https://e-u.hr/dok/udzbenik/31_210.pdf, 2021.)

4.1.1. DMBS (Sustav za upravljanje bazama podataka)

Sustav za upravljanje bazama podataka (SUBP) ili *engl. Database Management System* (DBMS) je program koji upravlja bazom podataka, točnije kontrolira pristup, kreiranje i manipulaciju podacima unutar baze, a neovisan je o računalnoj platformi ili računalnom programu koji ga koristi. (Kaluža, 2008., 16.)

¹⁰ Redundancija znači nepotrebno ponavljanje podataka u bazi (Kaluža, 2008., 54.)

Slika 9: DMBS shema

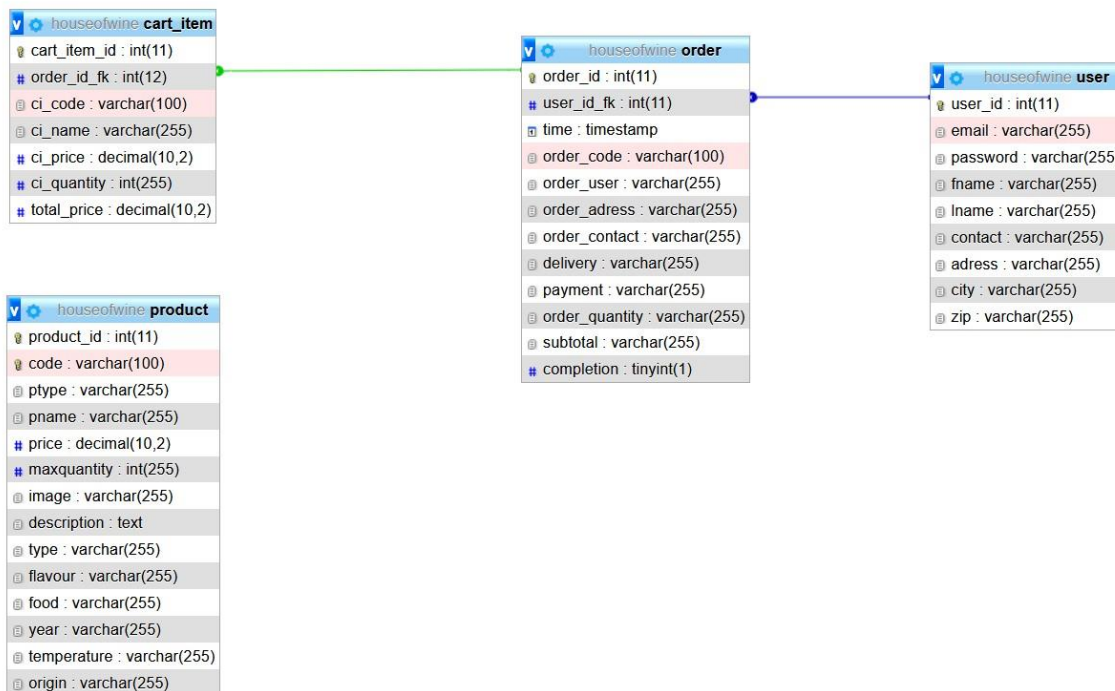


Izvor: <https://www.bmc.com/blogs/dbms-database-management-systems/>

4.2. Relacijska baza podataka

Relacijska baza podataka je skup tablica, odnosno relacija koje su međusobno povezane. Jedna takva relacija naziva se relacijskom shemom i sastavljena je od naziva te relacije i popisa njenih obilježja ili atributa koji su dio te relacije. Ta baza podataka u cijelosti, naziva se relacijska shema baze podataka, a sadržava relacijske sheme svih relacija. Relacijskom bazom upravlja RDBMS, odnosno sustav za upravljanje relacijskim bazama podataka, koji pohranjuje podatke u obliku međusobno povezanih tablica sačinjenih od entiteta i atributa. (https://e-u.hr/dok/udzbenik/31_210.pdf, 2021.)

Slika 10: Relacije među tablicama baze podataka "houseofwine"



Izvor: izradio autor

Relacije, relacijske baze podataka ovog završnog rada pod nazivom *houseofwine*, prikazane su iznad na slici broj 10. Unutar tablice *user* (korisnik) spremljeni su svi podaci o registriranim korisnicima na web stranici. *User* (korisnik) je entitet, a podaci unutar tablice su atributi tog entiteta. Svaka tablica je unikatno identificirana pomoću primarnog ključa, a za drugu tablicu se veže pomoću vanjskog ključa. Spomenuti ključevi su detaljnije objašnjeni u poglavlju ključevi, a podaci unutar tablice u poglavlju tablica. Unutar tablice *order* (narudžbe) spremljeni su podaci o narudžbama koje izvršavaju korisnici. Dvije tablice *order* i *user* su povezane preko vanjskog ključa *user_id_fk* vezom *jedan prema više (1:M)*, što znači da jedna narudžba može pripadati samo jednom korisniku (kupcu), a jedan korisnik može napraviti jednu ili više narudžbi. Detaljnije objašnjene veza nalazi se u poglavlju o vezama unutar relacijske baze podataka. Važno je napomenuti, da baza podataka sa slike broj 10 ima jednu specifičnost, a to je da tablica *cart_item* i tablica *product* nisu povezane, iako sadrže zajedničke attribute i po normama upravljanja bazama podataka bi trebale biti vezane kako ne bi došlo do redundancije. Međutim, nisu povezane iz

razloga što bi se tako podaci o proizvodima nakon što ih korisnik doda u košaricu mogli promijeniti i time ugroziti poslovanje, jer bi korisnik tako tijekom dodavanja proizvoda i tijekom potvrde kupnje mogao dobiti različite informacije o proizvodu. Iz tog razloga, podaci se u trenutku dodavanja proizvoda kopiraju iz tablice *product* u tablicu *cart_item* pomoću SQL naredbi.

4.2.1. Tablica

Tablica (*engl. table*) je temeljni objekt relacijske baze podataka unutar koje su spremljeni podaci. Tablica se sastoji od redaka ili slogova, a svaki redak se naziva relacija (*engl. relation*) i stupaca ili atributa koji pojmovno odgovaraju podatkovnom polju (*engl. field*). Osnovne karakteristike svake tablice su: nepostojanje dva jednaka retka i dva jednaka stupca, te irelevantnost redosljeda redaka i stupaca. (https://e-u.hr/dok/udzbenik/31_210.pdf, 2021.)

Slika 11: Prikaz tablice "user" iz baze podataka ovog projekta

STUPAC - POLJE
↓

REDAK - SLOG →

user_id	email	password	fname	lname	contact	adress	city	zip
47	siskomencetic@email.com	\$2y\$10\$1uqp3h6T.HwauFlugbnLpOD44hWyJ5K6Ta6foTEOc...	Šiško	Menčetić	+385999999999	Augusta Šenoa 1/B	Zagreb	10000
73	john.doe@email.com	\$2y\$10\$y4/a!htwYlIE0VcB3wFF7uoYDW209Y0/3nSC7X.RVX...	John	Doe	+385919638547	Zagrebačka 10	Rijeka	51000

Izvor: izradio autor

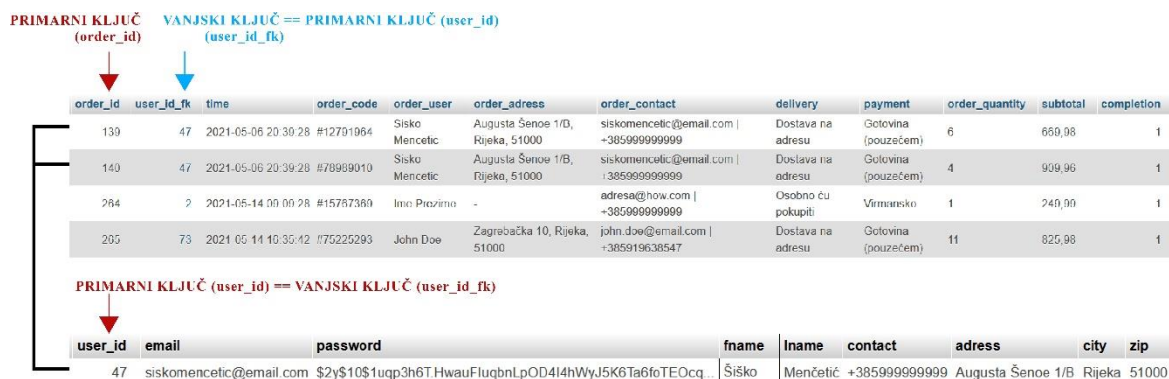
Slika broj 11 prikazuje tablicu *user* i njene retke i stupce. Istovrsni objekti (korisnici) prikazani su u tablici redcima, koji su opisani stupcima ili poljima (email, ime, prezime).

4.2.2. Ključevi

Svaka tablica unutar relacijske baze podataka ima polje ili skup polja kojima se može jednoznačno definirati redak (slog) tablice. To polje naziva se primarni ključ (*engl. primary key*). Primarni ključ, ključan je i za međusobno povezivanje tablica jer to je podatak na koji se referiramo kada povezujemo više tablica i tada primarni ključ jedne tablice postaje vanjski ključ (*engl. foreign key*) druge tablice. Vanjski ključ, za razliku od primarnog, može biti ponavljan u više tablica. Ako pogledamo tablicu u primjeru na slici 10, možemo zaključiti da tablica *order* uz svoj primarni ključ

order_id ima i vanjski ključ *user_id_fk*, jer je u vezi sa tablicom *user*. Ta veza nam omogućuje precizno određivanje kojem korisniku pripada koja narudžba. (https://e-u.hr/dok/udzbenik/31_210.pdf, 2021.)

Slika 12: Primjer primarnog i vanjskog ključa u tablici



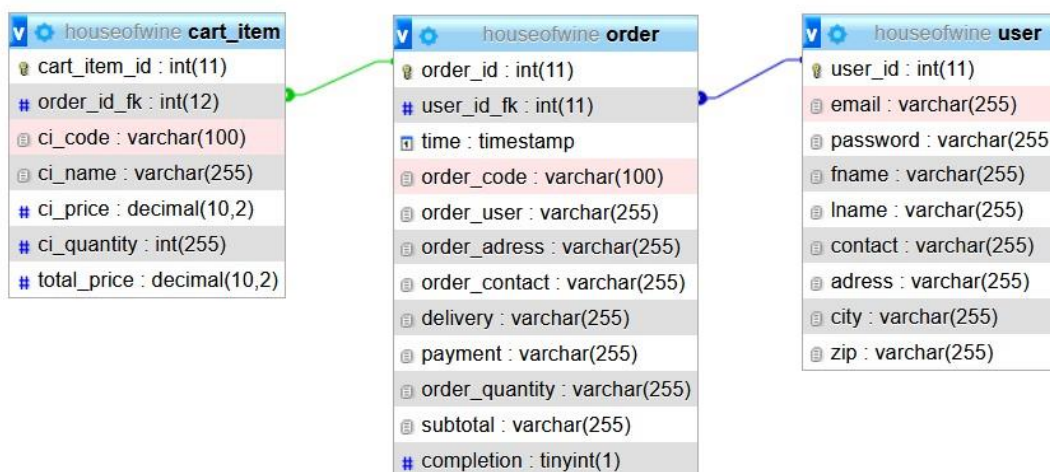
Izvor: izradio autor

4.2.3. Veze tablica

Na slici broj 12 vidimo kako su podaci dviju tablica *user* i *order* međusobno povezani vanjskim ključem *user_id_fk*, kojem pripada vrijednost primarnog ključa *user_id* tablice *order*. Relacija (*engl. relationship*) je veza ili odnos dvije relacijske tablice unutar relacijske baze podataka putem vrijednosti primarnog ključa. Bitno je napomenuti da se prilikom povezivanja tablica mora poštovati pravilo referencijalnog integriteta, koje nalaže, da vanjski ključ povezane tablice mora odgovarati primarnom ključu osnovne tablice. Veze kojima su tablice povezane dijele se na tri vrste:

1. Jedan prema jedan (*engl. one-to-one*) – 1:1
2. Jedan prema više (*engl. one-to-many*) – 1:M
3. Više prema više (*engl. many-to-many*) – M:M

Slika 13: Veze u tablici



Izvor: izradio autor

Slika broj 13 prikazuje odnos između tri tablice unutar baze podataka *houseofwine*. Već prethodno spomenute tablice, *user* i *order* povezane su vezom jedan prema više (1:M), točnije jedan korisnik može napraviti više narudžbi, a jedna narudžba može pripadati samo jednom korisniku. Tablice *order* i *cart_item*, također su povezane vezom jedan prema više (1:M), jer se u jednoj narudžbi može nalaziti samo jedan isti proizvod, dok se jedan isti proizvod može naći kroz više različitih narudžbi. Ovo vezivanje tablica nam bez sumnje govori koji korisnik je naručio koji proizvod i kroz koju narudžbu je to učinio. Možemo zamijetiti da je vanjski ključ koji referencira na primarni ključ osnovne tablice u tipu veze jedan prema više u tablici koja je na strani znaka više, to jest osnovna tablica je ona koja ima znak jedan u toj relaciji.

4.3. SQL

Strukturirani upitni jezik (*engl. Structured Query Language*) ili skraćeno SQL je upitni jezik za obradu podataka unutar baze podataka. Za ovaj projekt korištene su SQL naredbe unutar samog kôda u Visual Studio Code-u, kao i automatski generirani upiti unutar aplikacije myPhpAdmin za jednostavno upravljanje bazom podataka. (https://e-u.hr/dok/udzbenik/31_210.pdf, 2021.)

SQL koristi sintaksu standardiziranih pristupnih jezika za manipulaciju podacima, a to su DD¹¹, DDL¹², DML¹³ i DQL¹⁴ (Kaluža, 2008., 18.)

„Cilj standardiziranog pristupnog jezika je stvoriti pravila kojih se treba pridržavati kod stvaranja pitanja računalu za dobivanjem određenih podataka koji mogu dati određenu informaciju, a koja ima za posljedicu provođenje određene aktivnosti.“ (Kaluža, 2008., 23.)

Nastao je 70-ih godina pod imenom SEQUEL, a služio je za upravljanje i pronalaženje podataka. Tijekom godina se razvijao i dobivao različite inačice, a tijekom 90-ih objavljuju se i standardi za SQL koji su danas implementirani u svim vodećim DBMS sustavima. (Kaluža, 2008., 23.)

Neke od najpoznatijih i najkorištenijih SQL naredbi su SELECT (služi za odabir određenih polja iz tablice), FROM (za odabir tablice), WHERE (za postavljanje uvjeta o podatku kojeg tražimo), INSERT INTO (umetanje u tablicu), UPDATE (ažuriranje/izmjena podataka u tablici), DELETE (brisanje redaka ili podataka u tablici), ORDER BY (služi za određivanje redosljeda podataka prema vrijednosti odabranog atributa). Jedan primjer SQL naredbe iz ovog završnog rada možemo vidjeti na slici 14.

Slika 14: Primjer SQL kôda

```
100      $userQuery = "SELECT * FROM user WHERE email='$email' ";
```

Izvor: izradio autor

Naredba iznad, koja se sprema se u PHP varijablu *\$userQuery*, služi za odabir svih podataka (znak „*“ u SQL jeziku znači sve) iz tablice *user* gdje je atribut *email* jednak PHP varijabli *\$email*, tj. konkretno u ovom slučaju to je podatak koji korisnik unosi, a program odabire podatke iz redaka koji odgovaraju tom uvjetu. U doslovnom prijevodu, ova SQL naredba znači: „odaberi sve podatke

¹¹ *Engl. Data Dictionary* je riječnik podataka. (Kaluža, 2008., 18.)

¹² *Engl. Data Definition Language* je jezik za stvaranje tipova podataka. (Kaluža, 2008., 19.)

¹³ *Engl. Data Manipulation Language* je jezik za manipuliranje podacima. (Kaluža, 2008., 20.)

¹⁴ *Engl. Data Query Language* je podatkovni upitni jezik. (Kaluža, 2008., 21.)

iz tablice *user* gdje je atribut *email* jednak spremljenoj vrijednosti varijable *\$email*¹⁵. Na slici broj 15 možemo vidjeti još jedan primjer SQL kôda.

Slika 15: SQL kôd u programu phpMyAdmin

Showing rows 0 - 3 (4 total, Query took 0.0021 seconds.) [price: 45.00... - 220.00...]

```
SELECT product_id, code, pname, price, maxquantity FROM `product` WHERE maxquantity < 20 ORDER BY price ASC
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Options

	product_id	code	pname	price	maxquantity
<input type="checkbox"/> Edit Copy Delete	11	0.22408519519873446	Grasecco Extra Brut 0,75 l	45.00	15
<input type="checkbox"/> Edit Copy Delete	12	0.8618292695487191	Korlat Cabernet Sauvignon vrhunsko vino 0,75 l	99.99	5
<input type="checkbox"/> Edit Copy Delete	3	0.07756795834523436	Dekanter Veliki	100.00	5
<input type="checkbox"/> Edit Copy Delete	9	0.6412519595038629	Polica za vinske čaše	220.00	5

Izvor: izradio autor

Ovaj put, na primjeru slike broj 15, vidimo sučelje aplikacije phpMyAdmin u čijem smo *text editoru* upisali SQL naredbu „SELECT product_id, code, pname, price, maxquantity FROM `product` WHERE maxquantity < 20 ORDER BY price ASC“. Ta naredba prikazuje odabrane attribute i njihove vrijednosti unutar tablice *product*, a koji zadovoljavaju uvjet da je atribut *maxquantity* (tip podatka *integer*¹⁵) manji od 20 i slažu podatke u redosljedu prema vrijednosti atributa *price* od najmanjeg prema najvećem (uzlazno).

4.4. PHP

PHP je rekurzivni akronim za *engl. PHP: Hypertext Preprocessor*, a to je globalno korišteni programski jezik, posebice za razvoj web stranica i aplikacija (*engl. web development*) i može se „ugraditi“ unutar HTML kôda. U pravilu se koristi za serversku stranu programa, ali je funkcionalan i u drugim dijelovima programa. Lako ga je ubaciti u HTML kôd pomoću oznaka

¹⁵ *Integer* je naziv za tip podatka varijable koja je neki cijeli broj.

(*engl. tag*) `<?php ?>` svaki put kada nam trebaju procesne instrukcije PHP-a. Te oznake nam omogućavaju „skokove“ unutar i van PHP kôda unutar samog HTML dokumenta. (What is PHP?: <https://www.php.net/manual/en/intro-what-is.php>, 2021.)

4.4.1. PHP sintaksa

PHP kôd kao što smo već i rekli pišemo unutar `<?php ?>` oznaka. Varijable unutar PHP-a deklariramo na način da ispred naziva varijable stavimo znak \$, pa recimo ovako izgleda jedna PHP varijabla: `$varijabla`. Ona može biti broj ili tekstualnog tipa (*engl. string*). Za razliku od nekih programskih jezika, u PHP-u ne moramo definirati tip podatka već program sam prepoznaje o kojem se tipu radi.

Slika 16: Primjer PHP kôda unutar HTML elemenata

```
208 <div class="quantity-title">
209     Broj proizvoda
210 </div>
211 <div class="quantity-value">
212     <?php echo $totalQuantity; ?>
213 </div>
214 </div>
215 <div class="price-overview">
216     <div class="price-title">
217         Ukupna cijena
218     </div>
219     <div class="price-value">
220         <?php echo number_format($totalPrice, 2, ',', ' ') . " kn"; ?>
221     </div>
```

Izvor: izradio autor

Na slici je prikazan dio HTML i PHP kôda iz završnog rada. Može se vidjeti kako se PHP kôd na liniji 212 i liniji 220 „ubacio“ unutar HTML oznaka `<div></div>` kako bi se ispisale PHP varijable `$totalQuantity` i `$totalPrice`. Varijable se ispisuju pomoću ključne riječi `echo`¹⁶ koja u PHP-u služi za ispis. Također u 220. liniji kôda koristimo PHP ugrađenu funkciju (*engl. built-in function*) `number_format` koja oblikuje izgled broja na način da unosom argumenata određujemo kojim će se znakom odvajati decimale broja ili na koliko decimala se zapisuje decimalni zapis (u ovom slučaju 2, jer je riječ o cijeni). Naprimjer: broj koji je unutar tablice u bazi podataka zapisan

¹⁶ `echo` u PHP-u je poput ključne riječi `printf` u C jeziku ili `cout` u C++.

kao 20.575 će se korisniku prikazivat kao 20,58 jer ga zaokružujemo na dvije decimale i umjesto točke koristimo zarez za odvajanje.

4.4.2. PHP super-globalne varijable

To su varijable u PHP-u koje su predefinirane, a naziv su dobile po tome što su uvijek dostupne, neovisno gdje se nalazile, u kojoj funkciji, klasi ili datoteci. Neke od tih varijabli su `$_SERVER`, `$_SESSION`, `$_COOKIE`, `$_POST`, `$_GET` i one su korištene i u ovom projektu. (PHP Global Variables – Superglobals: https://www.w3schools.com/PHP/php_superglobals.asp, 2021.)

Slika 17: Super-globalna varijabla `$_SERVER`

```
27 <?php
28
29     if (basename($_SERVER['PHP_SELF']) == 'index.php') {
30         echo "Početna - House of Wine";
31     } else if (basename($_SERVER['PHP_SELF']) == 'podrum.php') {
32         echo "Vinski podrum - House of Wine";
```

Izvor: izradio autor

Na slici vidimo primjer korištenja `$_SERVER` globalne varijable koja sadrži informacije o zaglavlju stranice ili lokacijama skripti. U ovom slučaju preko super-globalne varijable potražujemo naziv PHP datoteke u kojoj se trenutno nalazimo kako bi ispisali korisniku naziv stranice na kojoj se trenutno nalazi u zaglavlju. (PHP Superglobal - `$_SERVER`: https://www.w3schools.com/Php/php_superglobals_server.asp)

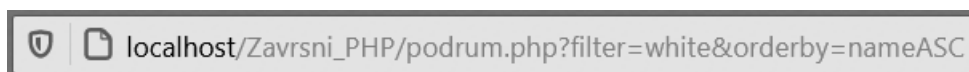
Varijabla `$_COOKIE` nam služi za spremanje kolačića (*engl. cookie*). Kolačići su male datoteke u koje server sprema određene podatke na korisnikovo računalo. Uz pomoć PHP jezika, kolačići se mogu stvarati i uništavati. (PHP Cookies: https://www.w3schools.com/php/php_cookies.asp, 2021.)

4.4.3. \$_GET i \$_POST

Metode GET i POST postavljamo i koristimo pomoću super-globalnih varijabli `$_GET` i `$_POST`. I jedna i druga metoda stvaraju polja od ključa i vrijednosti koja se s tim ključem uparuje. Jedina razlika je što se polje varijabli `$_GET` varijable šalje kroz URL stranice, a `$_POST` kroz HTTP protokol.

`$_GET` je PHP super-globalna varijabla koja služi za prikupljanje podataka nakon potvrđivanja *form-a*¹⁷ kojemu je pridodan atribut *method* (metoda) s vrijednošću *get*. Ova varijabla također može i prikupiti podatke prosljeđene u URL-u. (PHP Form Handling: https://www.w3schools.com/PHP/php_forms.asp, 2021.)

Slika 18: Prosljeđivanje podataka kroz URL



Izvor: izradio autor

Na slici vidimo da su kroz URL prosljeđene varijable *filter* i *orderby*. One u URL-u dolaze nakon naziva datoteke *podrum.php* i odvajaju se od naziva znakom „?“, a međusobno se odvajaju znakom „&“. U nastavku program provjerava pomoću varijable `$_GET` da li su postavljene varijable u URL-u i koja je njihova vrijednost.

`$_POST` je također PHP super-globalna varijabla koja kupi podatke nakon što su potvrđeni putem HTML *form-a*, a ovaj put *method* ima vrijednost *post*. (PHP Superglobal - `$_POST`: https://www.w3schools.com/Php/php_superglobals_post.asp, 2021.)

¹⁷ *Form* se odnosi na HTML element koji nam služi za prikupljanje unesenih podataka koje prosljeđujemo serveru GET i POST metodama (HTML Forms: https://www.w3schools.com/html/html_forms.asp, 2021.)

Slika 19: Prikaz HTML form elementa sa metodom POST

```
19 <h4>Registrirajte se</h4>  
20 <form action="register.php?actionregister" method="post">
```

Izvor: izradio autor

Na slici vidimo kako izgleda jedan HTML *form* element koji u sebi sadrži *post* metodu. Inače to je dio kôda za registraciju korisnika, jer podatke o korisniku prosljeđujemo putem *POST* metode, zbog osjetljivosti sadržaja.

4.4.4. PHP sesije

Sesije (*engl. session*) u PHP-u su način spremanja podataka unutar varijable, koje se kasnije mogu koristiti kroz ostale web stranice. Za razliku od kolačića, sesije se ne spremaju na računalo korisnika. Sesija funkcionira slično kao aplikacija na računalu, kada ju trebate otvarate ju i činite nekakve radnje i nakon toga ju zatvarate. Jedina razlika je ta što sesija nakon što ju zatvorite prestaje postojati i ne zna tko ju koristi, ona služi samo za spremanje varijabli unutar trenutnog zbivanja. Ona sadržava informaciju o jednom korisniku i dostupna je na svim stranicama jedne aplikacije. (PHP Sessions: https://www.w3schools.com/php/php_sessions.asp, 2021.)

Sesija počinje funkcijom *session_start()*, a postavlja se unutar PHP globalne varijable *\$_SESSION*. Varijabla *\$_SESSION* je ustvari *array*¹⁸ koji sadrži sesijske varijable dostupne za trenutno stanje. Pogledajmo primjer na slici ispod. (*\$_SESSION*: <https://www.php.net/manual/en/reserved.variables.session.php>, 2021.)

Slika 20: Poziv funkcije *session_start()*

```
includes > dbconnect.php > ...  
1 <?php  
2  
3 session_start();
```

Izvor: izradio autor

¹⁸ U programiranju, tip podatka koji se sprema kao polje s više vrijednosti

U datoteci *dbconnect.php* započinjemo sesiju koja traje sve dok je korisnik prijavljen na stranicu. Kada se korisnik prijavi inicijaliziramo globalnu varijablu u `$_SESSION['loggedIn']` i postavljamo njenu vrijednost na *true*, koja sprema podataka o tome da li je korisnik prijavljen na stranicu. U drugu globalnu varijablu `$_SESSION['userId']` spremamo podatak o *id* atributu tog korisnika iz naše baze podataka.

Slika 21: Inicijalizacija super-globalnih varijabla `$_SESSION`

```
103
104     $_SESSION['loggedIn'] = true;
105     $_SESSION['userId'] = $userRow['user_id'];
```

Izvor: izradio autor

Nakon što se korisnik odjavi iz programa, tada se i sesija uništava, funkcijom `session_destroy()`, a varijabla `$_SESSION['loggedIn']` se uništava pozivom funkcije `unset()`.

Slika 22: Poziv funkcija `session_destroy()` i `unset()`

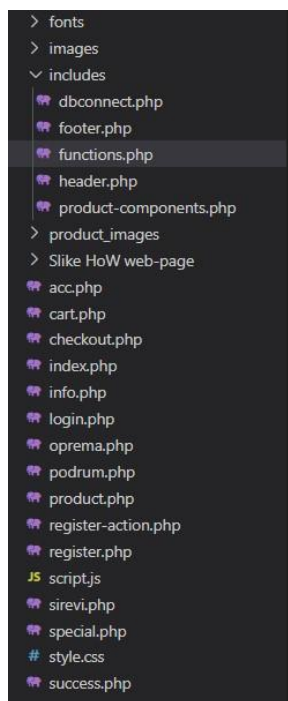
```
223     }
224     session_destroy();
225     unset($_SESSION['loggedIn']);
226     header('Location: index.php');
227 }
```

Izvor: izradio autor

4.4.5. PHP funkcije

PHP kôd se može pozivati i iz drugih PHP datoteka, uključivanjem datoteke i pozivanjem potrebne funkcije što možemo vidjeti kroz primjere ispod.

Slika 23: Struktura projekta u sučelju Visual Studio Code-a



Izvor: izradio autor

Slika broj 23 prikazuje strukturu web stranice u sučelju Visual Studio Code-a. Za ovaj primjer bitne su nam datoteke *functions.php*, unutar mape *includes*, te datoteke *header.php* i *index.php*.

Slika 24: Inicijalizacija funkcije `cartItems()`

```
822 function cartItems() {
823
824     $cart_count = 0;
825     if (empty($_SESSION["shopping_cart"])) {
826
827         $_SESSION['counter'] = 0;
828     }
829
830     else if (!empty($_SESSION["shopping_cart"])) {
831
832         $cart_count = $_SESSION['counter'];
833     }
834     echo "<div class=\"cart-count\">" . $cart_count . "</div>";
835 }
```

Izvor: izradio autor

Slika broj 24 prikazuje inicijalizaciju funkcije `cartItems()` unutar datoteke `functions.php`, koja provjerava da li je super-globalna varijabla `$_SESSION['shopping_cart']` prazna ili u njoj postoji zapis podataka i ako postoji da se u varijablu `$cart_count` spremi vrijednost druge globalne varijable `$_SESSION['counter']`, koja ustvari, na ovoj web stranici nosi podatak o broju proizvoda koje je jedan korisnik dodao u košaricu. Na kraju funkcija, već prethodno spomenutom naredbom `echo`, ispisuje broj koji se sprema u varijablu `$cart_count`. U slučaju da niti jednog proizvoda nema u košarici, ta vrijednost će biti 0 jer je varijabla `$cart_count` inicijalizirana na vrijednost 0 na početku funkcije. Na sljedećoj slici pratimo pozivanje spomenute funkcije.

Slika 25: Poziv funkcije `cartItems()`

```
108 <div>
109     <?php
110
111     cartItems();
112     ?>
113 <a href="cart.php">
```

Izvor: izradio autor

Slika broj 26 prikazuje dio kôda iz datoteke *header.php* unutar koje je na određenom mjestu HTML kôda pozvana funkcija *cartItems()*. A na slici broj 26 ispod, vidimo način na koji se jedne datoteke PHP-a pozivaju unutar druge datoteke PHP-a.

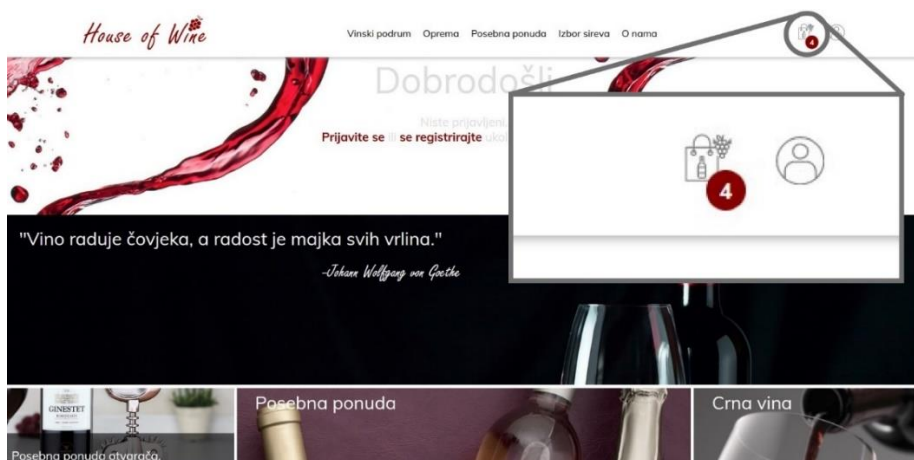
Slika 26: Ubacivanje datoteka *header.php* i *functions.php*

```
1  <?php
2
3  include ("includes/functions.php");
4  include ("includes/header.php");
5  ?>
```

Izvor: izradio autor

Da pojasnimo, slika broj 26 prikazuje dio kôda datoteke *index.php* (početna stranica web stranice „House of Wine“). Ako pogledamo 4. i 5. liniju kôda, vidimo da one sadrže PHP naredbu *include* kojom se datoteke *functions.php* i *header.php* ubacuju u datoteku *index.php* i time sve što je sadržano u tim datotekama, sada se, može koristiti i u datoteci *index.php*, pa tako i funkcija *cartItems()*, koja se, naime pojavljuje u svim ostalim datotekama ove web stranice koje pozivaju datoteku *header.php* jer ona poziva funkciju *cartItems()*.

Slika 27: Rezultat funkcije *cartItems()*



Izvor: izradio autor

Slika broj 27 prikazuje rezultat kôda opisanog na prethodnim slikama, tj. stranicu *indeks.php*, u kojoj se pozvana datoteka *header.php*, koja poziva funkciju *cartItem()*, a broj „4“ koji vidimo na slici pored košarice je broj koji ispisuje navedena funkcija. Više primjera, ujedno i složenijih, PHP kôda i njegove funkcionalnosti možemo pronaći u kasnijem poglavlju web trgovina „House of Wine“.

4.4.6. Spajanje na bazu podataka

Spajanje na SQL bazu podataka vezanu uz ovaj projekt izvršavamo pomoću PHP kôda, a to činimo na sljedeći način:

Slika 28: Spajanje na bazu podataka *houseofwine*

```
includes > dbconnect.php > ...
1  <?php
2
3  session_start();
4  //parametri za povezaivanje s bazom
5
6  $dbHost = "localhost";
7  $dbUser = "root";
8  $dbPassword = "";
9  $dbName = "houseofwine";
10
11 //povezaivanje s bazom
12 $connection = mysqli_connect($dbHost, $dbUser, $dbPassword, $dbName);
13
14 //provjera da li postoji veza
15 if ($connection === false) {
16     die("GREŠKA: Spajanje na bazu nije uspjelo!" . mysqli_connect_error());
17 }
18 ?>
```

Izvor: izradio autor

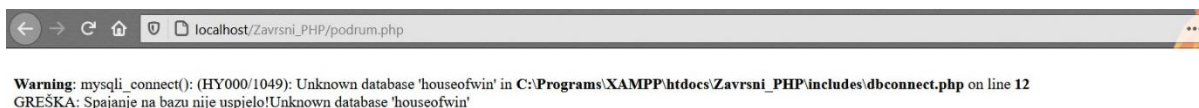
Slika broj 28 prikazuje dio kôda iz datoteke *dbconnect.php*, a ta datoteka sadrži kôd potreban za povezivanje s bazom podataka *houseofwine*. Cijeli kôd se nalazi unutar već prethodno spomenutih PHP oznaka (*<?php ?>*). Funkcija *session_start()* u 3. liniji kôda kreira PHP sesiju (*engl. session*) ili nastavlja trenutnu koja je prosljeđena putem GET ili POST metode. Ova funkcija vraća podatak tipa *boolean*¹⁹ što znači da funkcija vraća vrijednost *true* ako je sesija uspješno

¹⁹ Tip podatka koji vraća rezultat *true* ili *false* (1 ili 0).

pokrenuta ili u drugom slučaju vrijednost *false*. Sesije možemo spremati u PHP super-globalne varijable *\$_SESSION*. (session_start: <https://www.php.net/manual/en/function.session-start.php>, 2021.)

Nakon toga, inicijalizirane su varijable *\$dbHost*, *\$dbUser*, *\$dbPassword* i *\$dbName* u koje spremamo podatke o nazivu *host-a*²⁰, ime korisnika baze, lozinku potrebnu za spajanje na bazu i ime baze podataka na koju se povezujemo. Potom, inicijaliziramo varijablu *\$connection*, u koju spremamo funkciju *mysqli_connect()* tipa *boolean*, koja služi za otvaranje veze s bazom, a argumenti koje prima su 4 varijable objašnjene gore. Ako je povezivanje s bazom uspješno, tada navedena funkcija vraća vrijednost *true* i povezivanje na bazu je uspješno, a u protivnom vraća vrijednost *false*. U ovom kôdu još imamo i *if* naredbu²¹ koja provjerava da li je funkcija vratila *false*, i ako je poziva se funkcija *die()* koja zaustavlja izvršavanje daljnjeg kôda, tj. „gasi“ program uz poruku i funkciju *mysqli_connect_error()* koja ispisuje o kakvoj grešci je riječ, kao na slici broj 29, gdje je umjesto imena baze *houseofwine* upisano *houseofwin* (bez zadnjeg slova „e“).

Slika 29: Ispis SQL greške



Izvor: izradio autor

Datoteka *dbconnect.php* ubačena je u datoteku *functions* naredbom *include*, kako bi se kasnije u kôdu mogla koristiti varijabla *\$connection* za manipulaciju podacima kroz SQL naredbe.

²⁰ *Engl. host* je računalo koje posjeduje sve datoteke potrebne za web stranicu. (Web Host Definition: <https://techterms.com/definition/webhost>, 2021.)

²¹ Služi za provjeru stanja u programu, koristeći se logičkim operatorima (*i*, *ili*).

5. Razvoj klijentske strane web stranice

5.1. HTML

HTML je skraćenica za *engl. Hyper Text Markup Language*, to je univerzalni jezik za stvaranje strukture web stranice, kojeg može čitati bilo koja računalna platforma, a dijeli se na elemente koji se slažu u stranici i međusobno su odvojeni oznakama (*engl. tag*). (HTML Introduction: https://www.w3schools.com/html/html_intro.asp, 2021.)

Prvu inačicu HTML-a napisao je Tim Berners-Lee 1993. godine, koji je još 4 godine prije izumio WWW (World Wide Web). (A Brief History of HTML: https://www.washington.edu/accesscomputing/webd2/student/unit1/module3/html_history.html, 2021.)

U ovom završnom radu korištena je posljednja inačica HTML-a, a to je HTML5, koji je 2012. godine postao standard. HTML datoteku možemo prepoznati po ekstenziji *.html*, a možemo ju otvoriti pomoću web preglednika, npr. Google Chrome. Na slici ispod vidimo kako izgleda jednostavni HTML dokument.

Slika 30: HTML kôd

```
1  <!DOCTYPE html>
2
3  <html>
4      <head>
5          <title>Naslov stranice</title>
6      </head>
7      <body>
8          <h1>Naslov paragrafa</h1>
9          <p>Paragraf</p>
10     </body>
11 </html>
```

Izvor: izradio autor

Deklaracija u 1. liniji kôda `<!DOCTYPE html>` definira da je taj dokument HTML5 dokument. Nakon toga ide `<html>` oznaka unutar koje ide sav HTML kôd te stranice, a završava u liniji 11 `</html>`. To je sintaksa pisanja HTML kôda, tj. otvaranje i zatvaranje oznaka unutar kojih unosimo sadržaj i ugnježđujemo druge oznake ili ubacujemo druge tipove kôda, kao što smo imali u primjeru s PHP-om. Nadalje, `<head>` oznaku koristimo za umetanje meta podataka²² o stranici, `<title>` oznaku koristimo za definiciju naslova HTML stranice (u web pregledniku prikazuje se kao naslov kartice, *engl. tab*), `<body>` označava tijelo stranice i ujedno je i dio u kojem su svi vidljivi sadržaji web stranice poput naslova, podnaslova, slika, poveznica, tablica, itd. Element `<h1>` služi za pisanje velikih naslova, dok `<p>` služi za umetanje paragrafa unutar tijela stranice. Ovo je samo par osnovnih elemenata koji se koriste prilikom izrade web stranice, čisto da bi razumjeli kako izgleda HTML dokument, a više oznaka i načina strukturiranja HTML dokumenta ćemo vidjeti na konkretnim primjerima web stranice „House of Wine“ u kasnijim poglavljima, a na slici ispod vidimo jedan takav primjer.

Slika 31: Primjer HTML kôda na projektu

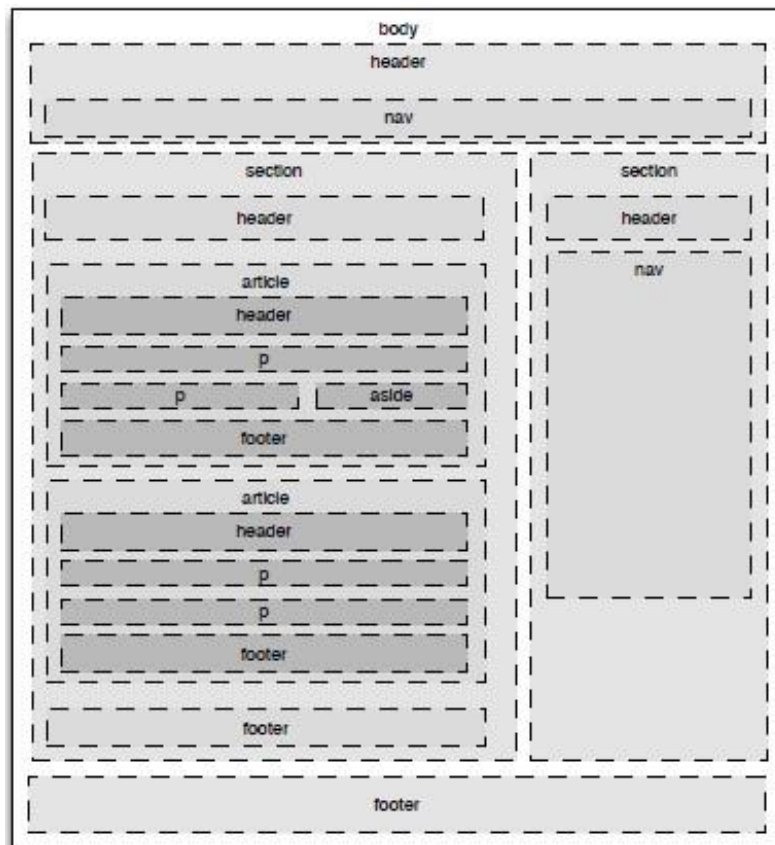
```
1      <footer>
2          <div class="message-container">
3              <div class="drive"></div>
4              <div class="age"></div>
5              <div class="message">Uživajte odgovorno!</div>
6          </div>
7          <div class="footer-container">
8              <div class="contact">
9                  <h2>Kontaktirajte nas</h2>
10                 <ul>
11                     <li>+385 99 399 9383</li>
12                     <li>info@houseofwine.hr</li>
13                     <li>prodaja@houseofwine.hr</li>
14                 </ul>
15             </div>
16             <div class="media">
17                 <h2 class="text">Društvene mreže</h2>
18                 <a href="#" class="media-links"></a>
19                 <a href="#" class="media-links"></a>
20             </div>
21             <div class="location">
22                 <h2>Gdje se nalazimo</h2>
23                 <ul>
24                     <li>Adresa, Ulica, Grad</li>
25                     <li>Poštanski broj</li>
26                     <li>Država</li>
27                 </ul>
28             </div>
```

Izvor: izradio autor

²² Meta podaci nisu uvijek prikazani na stranici, ali ih računalo i web preglednik razumiju (HTML `<meta>` Tag: https://www.w3schools.com/tags/tag_meta.asp, 2021.)

Na slici broj 31 vidimo još jedan primjer HTML kôda, ovaj put malo složeniji iz ovog završnog rada, na kojem su prikazane još neke mogućnosti HTML-a kao dodavanje CSS klasa i atributa svakom elementu ili određivanja nekih drugih elemenata poput veličine slike i dodavanja poveznica koje vode na druge stranice.

Slika 32: Struktura web stranice



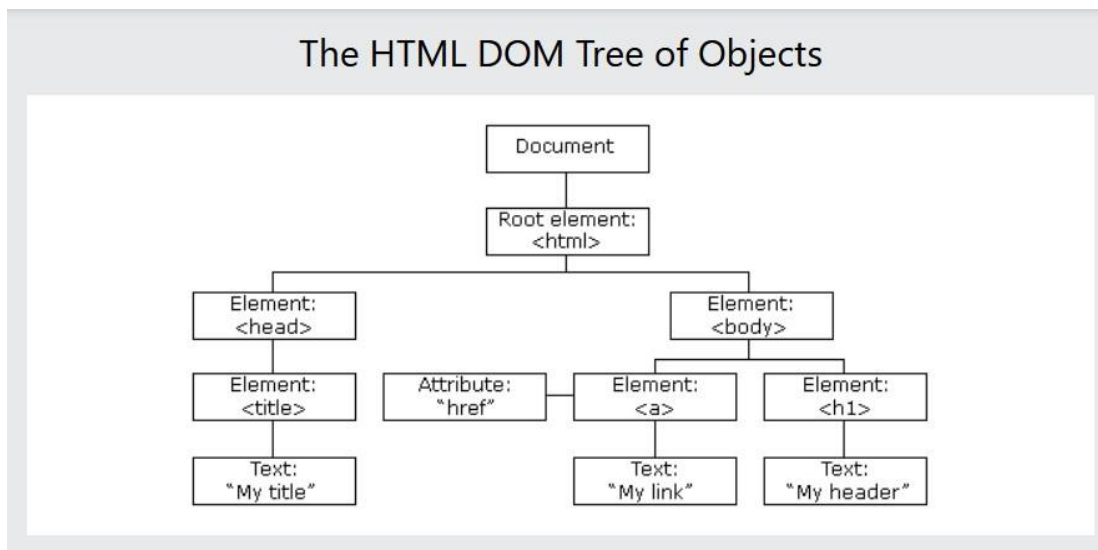
Izvor: Hogan, 2010., 28.

Slika broj 32 prikazuje strukturu web stranice. Više o tome u idućem poglavlju *HTML DOM model*.

5.1.1. HTML DOM model

HTML elementima je moguće manipulirati putem JavaScript-a ili CSS-a (koji će dodatno biti pojašnjeni u ovom završnom radu), ali prvo moramo moći odrediti čime unutar HTML dokumenta želimo manipulirati. Da bi pristupili HTML elementu, tj. da bi ga pronašli u mnoštvu drugih moramo ga moći identificirati na neki način. Postoji nekoliko načina da bi se to učinilo: putem svojstva *id*, putem klase, putem samog imena elementa, itd. HTML DOM ili HTML Document Object Model je model koji definira HTML elemente kao objekte, kao i sva svojstva, metode i događaje HTML elemenata. HTML DOM je ujedno i API²³ za JavaScript, pomoću kojeg možemo mijenjati, dodavati ili brisati HTML attribute i elemente, kao i CSS stilove, te reagirati na događaje (*engl. event*) unutar HTML dokumenta. HTML DOM model možemo gledati kao razgranatu strukturu u obliku krošnje, slika broj 33. (What is the HTML DOM?: https://www.w3schools.com/whatis/whatis_html5dom.asp, 2021.)

Slika 33: HTML DOM model



Izvor: https://www.w3schools.com/whatis/whatis_html5dom.asp

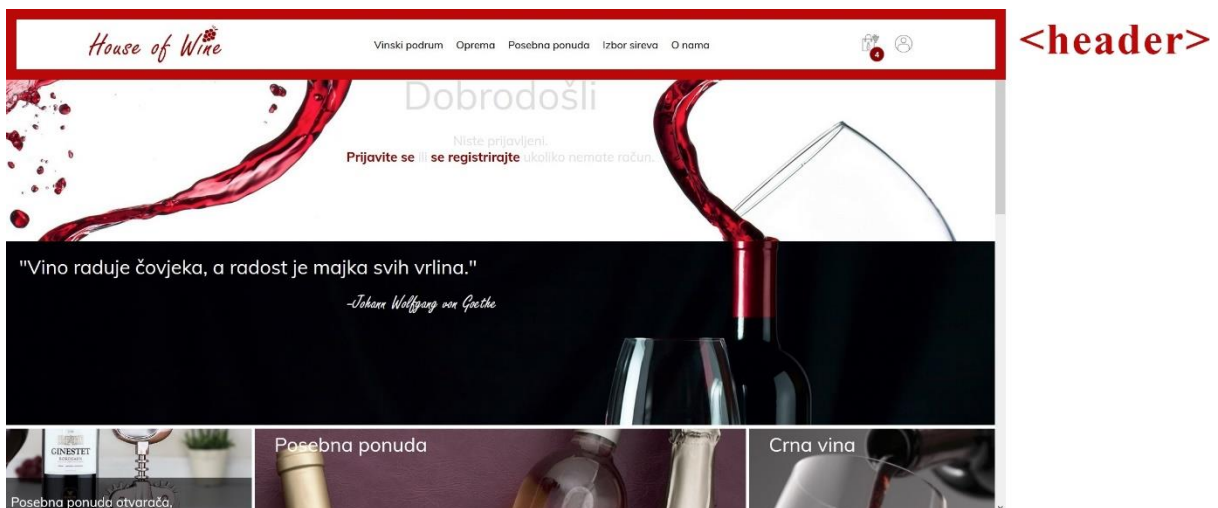
²³ API je akronim za *engl. Application Programming Interface*, to je aplikacijsko-programsko sučelje za web, služi za proširivanje mogućnosti web preglednika ili web servera ako se radi o serverskom API-u (Web APIs – Introduction: https://www.w3schools.com/js/js_api_intro.asp, 2021.)

Ovakav model nam pomaže uočiti gdje što „pripada“ unutar samog HTML dokumenta i kakve atribute sadrži. Kasnije u CSS primjerima ćemo vidjeti kako se CSS kôdom može utjecati na izgled i ponašanje HTML elemenata upravo preko „odabiranja“ elementa putem njihovog imena, klase ili atributa koji se vežu uz taj element.

5.1.2. Header

Header je dio HTML dokumenta koji može sadržavati bilo što, od logotipa stranice, preko tražilice (*engl. search bar*), pa sve do izbornika ili će biti jednostavan i samo sadržavati naslov stranice. Označava se oznakom `<header>`. Može ih biti i više u jednoj stranici, a sve ovisi o developeru²⁴ i o načinu na koji slaže stranicu. U praksi se koristi za pisanje dijela stranice koji se ponavlja kroz ostale stranice, kao što je npr. navigacijski dio stranice i mjesto gdje stoji logotip.

Slika 34: Header element na sučelju web stranice



Izvor: izradio autor

²⁴ Osoba, programer, koji razvija računalni program.

5.1.3. Navigation

Navigation je engleska riječ za navigaciju, od čega i naziv za oznaku `<nav>`, koja u HTML dokumentu, u praksi služi za stavljanje navigacije po stranici (poveznice na druge stranice). Na slici ispod vidimo *header* stranice unutar kojeg je smješten i *navigation*.

Slika 35: Navigation element na sučelju stranice

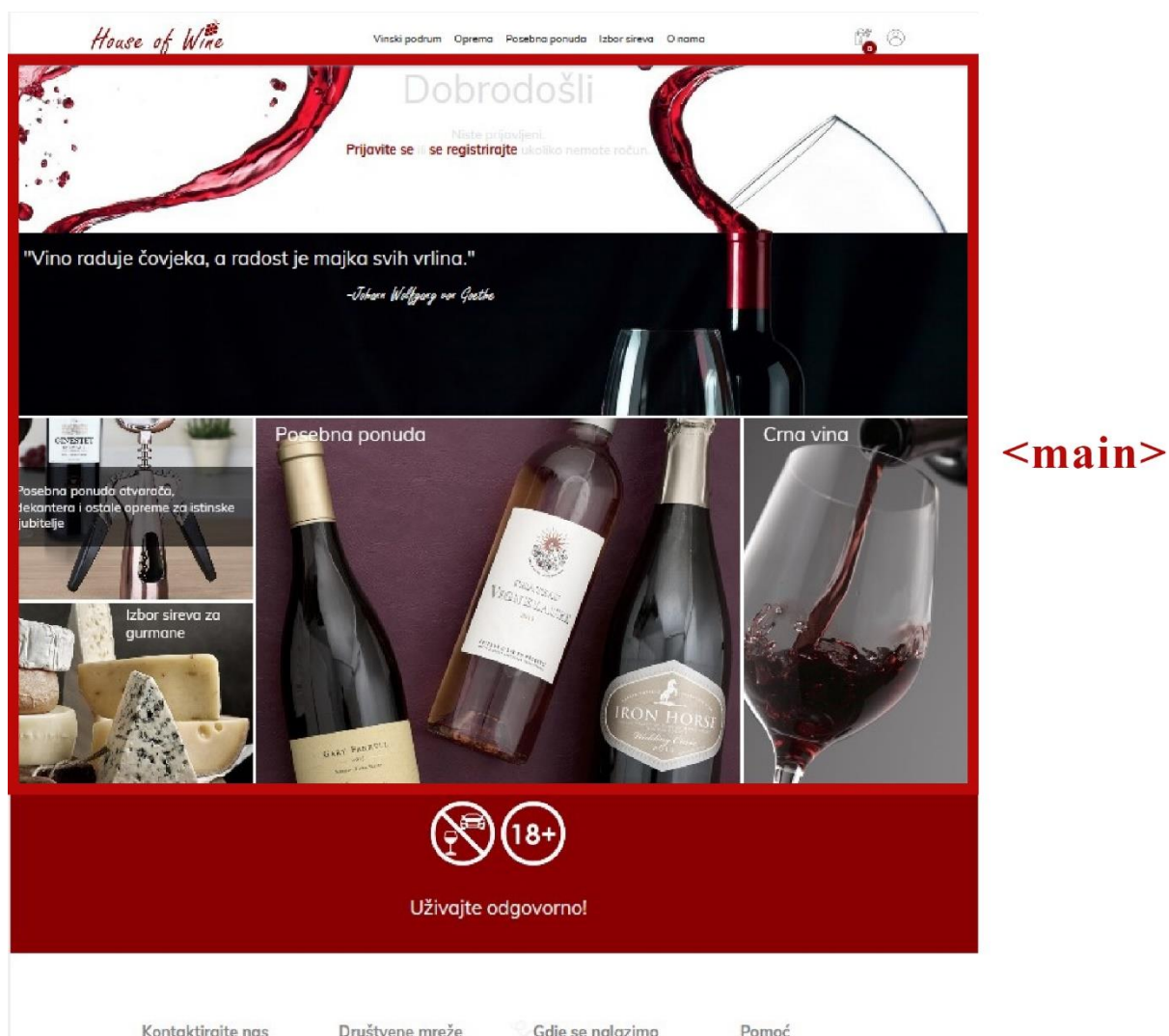


Izvor: izradio autor

5.1.4. Main

Main predstavlja glavni dio dokumenta i sadrži jedinstvene dijelove koji su specifični samo za tu stranicu. Označava se oznakom `<main>` i ne bi trebao sadržavati nikakve ponavljajuće dijelove kao što su navigacija, *header*, logotip i slično. Također, ne bi trebao biti ugniježđen unutar ponavljajućih elemenata, niti bi se trebao pojaviti više od jedan puta unutar jednog HTML dokumenta.

Slika 36: Main element na sučelju stranice

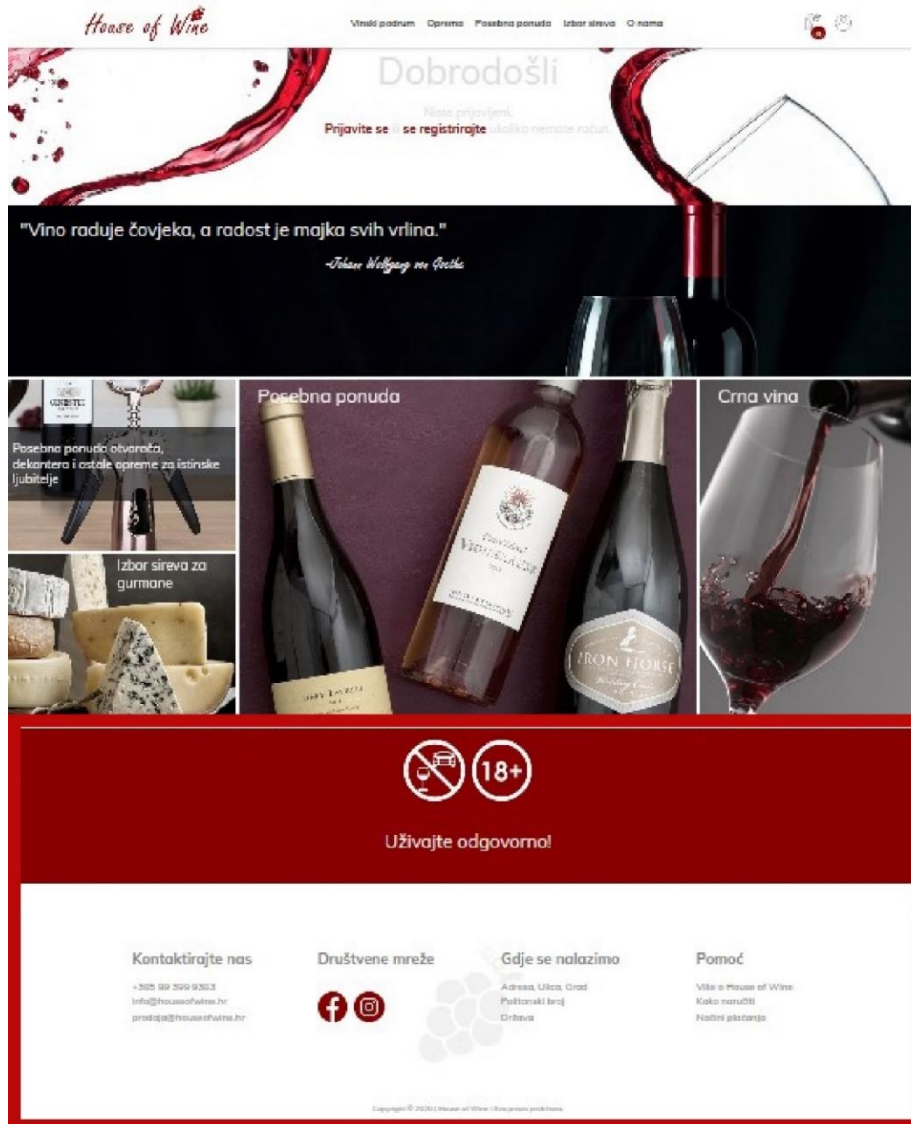


Izvor: izradio autor

5.1.5. Footer

Element *footer* u HTML-u se uobičajeno koristi za prikaz informacija na dnu stranice, koje se također, kao i *header*, ponavljaju preko više stranica. Unutar *footer-a*, najčešće nalazimo informacije o vlasniku web stranica, tko polaže prava na stranicu, kada je ona izrađena i sl. Također ih može biti više na jednoj stranici, a mogu sadržavati raznorazne informacije.

Slika 37: Footer element na sučelju stranice



<footer>

Izvor: izradio autor

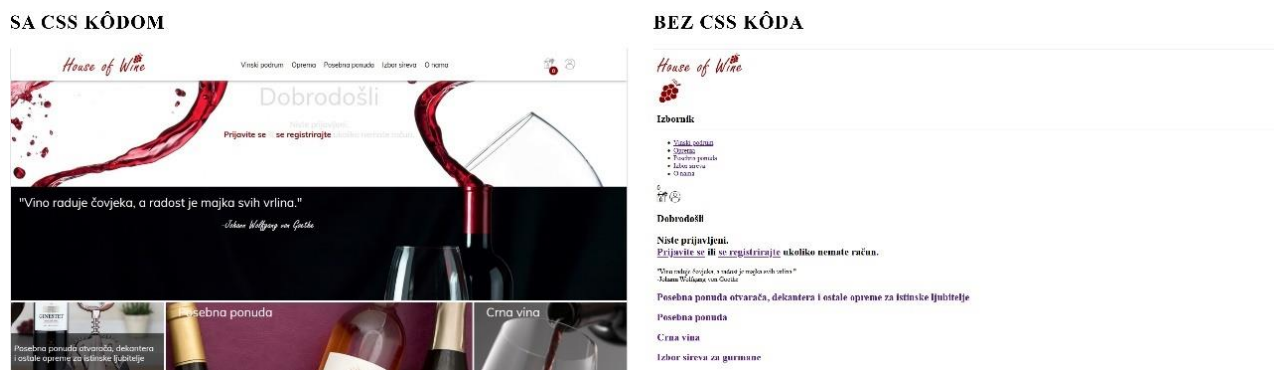
5.2. CSS

CSS je akronim za *engl. Cascading Style Sheet*, a to je jezik za dodavanje stilova unutar web stranice, točnije on opisuje način na koji se HTML elementi slažu i prikazuju na web stranici. Jedna CSS datoteka može kontrolirati stilove više stranica, stoga je vrlo praktičan i koristan. CSS nam omogućuje kontroliranje boja i slaganje elemenata u rasporedu (*engl. layout*), razmak između

elementa, boje fonta, veličine naslova, prikaz slika i sve to je moguće prilagoditi za različite računalne uređaje. Riječ *Cascading* u skraćenici CSS znači da se stilovi koje smo dodali za element „roditelja“, primjenjuje i na sve „dječje“ elemente unutar tog elementa. Ukratko, ako smo unutar elementa `<p>` (paragraf) dodali nekakav podnaslov tipa `<h6>`, i za paragraf `<p>` odredili boju fonta ljubičastu, tada će i taj podnaslov imati ljubičastu boju fonta, osim u slučaju ako za taj drugi element ne specificiramo neke druge karakteristike. (CSS Introduction: https://www.w3schools.com/Css/css_intro.asp, 2021.)

Da bi se najbolje uočila razlika web stranice sa CSS kôdom i bez njega, najbolje je promotriti HTML dokument unutar kojeg je uključena CSS datoteka i HTML dokument koji nema CSS datoteku. Tu razliku možemo vidjeti na slici broj 38.

Slika 38: Prikaz stranice bez CSS-a i sa CSS-om



Izvor: izradio autor

5.2.1. CSS sintaksa

CSS sintaksa se sastoji od selektora i bloka deklaracija. Selektor određuje koji dio HTML dokumenta (koje elemente) ćemo mijenjati, a blok deklaracija sadrži jednu ili više deklaraciju odvojenih znakom točka-zarez „;“. Svaka deklaracija sastoji se od naziva atributa i vrijednosti koja mu se pridodaje. Tijelo deklaracije počinje i završava vitičastim zagradama „{“ i „}“. (CSS Syntax: https://www.w3schools.com/Css/css_syntax.asp, 2021.)

Slika 39: Primjer CSS funkcije iz projekta

```
680 .content2 h2 {
681     font-family: Century Gothic;
682     font-size: 30px;
683     font-weight: normal;
684     color: white;
685     margin: 5px 0px 0px 30px;
686 }
```

Izvor: izradio autor

Na slici broj 39 prikazan je dio CSS kôda ovog završnog rada. U 680. liniji `.content2 h2` je selektor, a `{` je mjesto gdje se otvara blok deklaracija. Na liniji 682 `font-size` je atribut koji služi za određivanje veličine fonta (teksta), a `30px` je vrijednost tog atributa izražena u pikselima²⁵ (*engl. pixel*). 682. linija završava znakom `;`, što znači da se u sljedećem redu može upisati novi atribut i tako sve do 686. linije gdje se blok deklaracija zatvara znakom `}`.

5.2.2. CSS selektori

CSS selektori (od *engl. select*, što znači odabir) nam služe za pronalazak/odabir HTML elemenata na koje želimo primijeniti određeni stil. Postoji 5 kategorija CSS selektora:

1. Jednostavni
2. Kombinatorni
3. Pseudo – klase
4. Pseudo – elementi
5. Atributni

²⁵ Piksel je najmanja jedinica digitalne slike ili grafike koja može biti prikazana na digitalnom uređaju s ekranom. (What Does Pixel Mean?: <https://www.techopedia.com/definition/24012/pixel>, 2021.)

Jednostavni CSS selektori su bazirani na imenu, klasi ili *id-u* HTML elementa. Na slici ispod vidimo sva tri navedena primjera.

Slika 40: Primjeri CSS selektora

1. KLASNI SELEKTOR

```
52 .navbar-container {
53   display: flex;
54   justify-content: space-around;
55   align-items: center;
56   background-color: white;
57   box-shadow: -1px 1px 6px #999;
58   position: fixed;
59   width: 100%;
60   padding: 0 20px;
61   z-index: 1000;
62 }
```

2. IMENSKI SELEKTOR

```
38 body {
39   margin: 0;
40   padding: 0;
41   overflow-x: hidden;
42 }
```

3. ID SELEKTOR

```
1657 #confirm-changes-button {
1658   background-color: lightgreen;
1659   border-color: lightgreen;
1660 }
```

Izvor: izradio autor

Primijetimo da je sintaksa za klasni selektor sljedeća: stavljamo točku „.“, a iza točke bez odvajanja naziv klase. Imenski selektor funkcionira na način da ga nazovemo onako kako se zove oznaka unutar HTML dokumenta, u ovom primjeru *body*, a može biti bilo koji HTML element, npr. `<div>`, `<p>`, ``, `<footer>`, itd. Posljednji, *id* selektor deklariramo tako što stavljamo znak ljestvi „#“ ispred naziva, a naziv odgovara *id* atributu HTML oznake. (CSS Selectors: https://www.w3schools.com/Css/css_selectors.asp, 2021.)

Kombinatorni selektori su selektori koji kojima se ostvaruje veza između dva jednostavna selektora uz pomoć kombinatora (*engl. combinator*). Npr. jedan takav selektor je tzv. nasljedni selektor koji je u sintaksi razmak (tipka *space* na tipkovnici). (CSS Combinators: https://www.w3schools.com/Css/css_combinators.asp, 2021.)

Slika 41: Kombinatorni CSS selektor

```
1180  .delivery-price input {
1181      position: absolute;
1182      float: right;
1183  }
```

Izvor: izradio autor

Slika broj 41 prikazuje kombinirani selektor koji se sastoji od klasnog selektora *.delivery-price* i imenskog selektora *input*, koji su odvojeni razmakom. Ovaj primjer znači da se zadani kôd primjenjuje na sve *<input>* HTML elemente koji su ugniježđeni unutar HTML elementa klase *delivery-price*.

Sljedeće na redu su pseudo – klase, koje se koriste za definiranje posebnih stanja elemenata, npr.: promjena izgleda elementa kada korisnik prelazi pokazivačem miša preko tog elementa (*hover*).

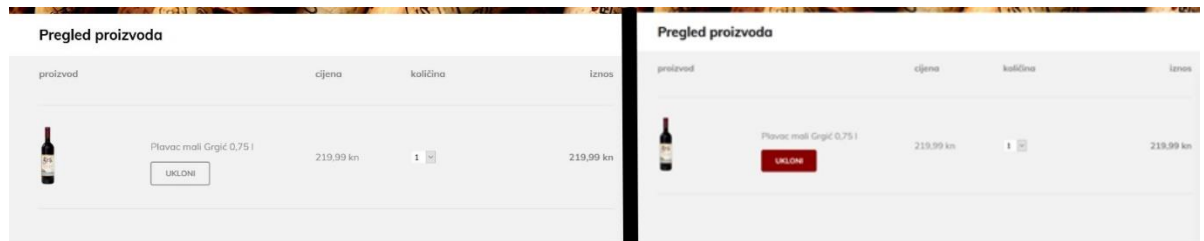
Slika 42: Pseudo-klasni selektor

```
1328  .remove-btn: hover {
1329      background-color: ■ rgb(138, 0, 0);
1330      border: solid ■ rgb(138, 0, 0) 2px;
1331      color: ■ white;
1332      cursor: pointer;
1333      transition: all ease-in-out 250ms;
1334  }
```

Izvor: izradio autor

Na slici broj 42 vidimo sintaksu definiranja pseudo-klase; upisujemo ime klase na koju želimo utjecati kada prelazimo preko elementa koji je te klase u HTML dokumentu i dodajemo dvotočku „:“ i ključnu riječ *hover*. Rezultat ovog selektora je prikazan na slici broj 43.

Slika 43: Rezultat selektora sa slike 42



Izvor: izradio autor

Lijeva strana slike prikazuje gumb za uklanjanje proizvoda u „prirodnom“ stanju, a kada smo preko njega prešli pokazivačem miša, on postaje crven a slova unutar njega bijela, što je rezultat kôda na slici broj 42.

Pseudo – elementi su vrsta selektora koji služi za stiliziranje određenog dijela elementa, npr.: promjena izgleda prvog slova u riječi ili prvog reda teksta, dodavanje sadržaja prije ili nakon nekog elementa, itd.

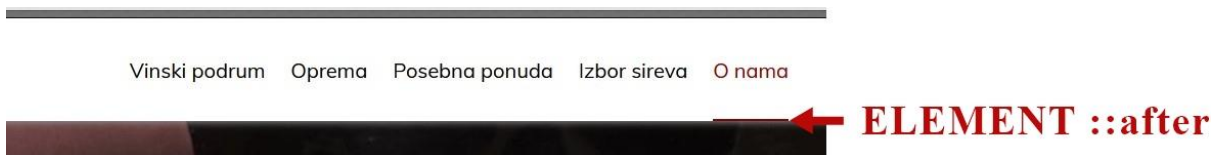
Slika 44: Pseudo-element

```
216 #active-link::after {  
217     width: 100%;  
218     left: 0%;  
219 }
```

Izvor: izradio autor

Sintaksa ovog selektora je da uz ime prvog selektora dodajemo dvije dvotočke „:“ i zatim ključnu riječ *after*, kojom definiramo utjecaj na sadržaj koji se pojavljuje u HTML DOM-u iza elementa s *id-om active-link*. U ovom primjeru na slici broj 44 smo odredili da je širina elementa s *id-om active-link* jednaka 100%, točnije element se širi cijelom dužinom elementa unutar kojeg je ugniježđen, kako to izgleda vidimo na slici ispod.

Slika 45: Prikaz elementa `::after` na primjeru projekta



Izvor: izradio autor

Posljednja vrsta selektora koju ćemo opisati su atributni selektori, a to su oni koji imaju specifičan atribut ili vrijednost atributa.

Slika 46: Atributni selektor

```
a[target] {  
  background-color: yellow;  
}
```

Izvor: https://www.w3schools.com/Css/css_attribute_selectors.asp

Sintaksa je takva da pišemo selektor elementa, u ovom slučaju imenski selektor *a*, te iza njega bez razmaka pišemo unutar uglatih zagrada „[]“ ime atributa, ovdje *target*, a rezultat toga je da će svi HTML elementi tipa *a* koji unutar sebe sadrže atribut *target* imati žutu pozadinsku boju (engl. *yellow*). (HTML Styles – CSS: https://www.w3schools.com/html/html_css.asp, 2021.)

Također postoji i tzv. univerzalni selektor, a on se označava znakom zvjezdice „*“ i služi za odabiranje svih HTML elemenata te web stranice, tj. pravila koja pišemo unutar bloka deklaracije iza tog znaka primjenjuje se na cijelu HTML datoteku.

5.2.3. Dodavanje CSS datoteke u HTML dokument

Kako bi primijenili CSS attribute na HTML datoteku, moramo toj HTML datoteci specificirati tu CSS datoteku ili kôd i „reći“ mu to je stil koji pripada toj stranici. CSS datoteku ili sam kôd moguće je dodati u HTML dokument na 3 načina. 1. način je „linijski“ (engl. *inline*), tj.

dodavanjem atributa *style* u HTML element i zatim upisivanjem željenog CSS kôda u taj atribut. Sintaksu tog načina vidimo na slici ispod.

Slika 47: Inline dodavanje CSS-a

```
58 <div style="background-color: rgba(129, 129, 129, .1)">  
59 <div class="last-order-details" style="font-family: Century Gothic; color: rgb(129, 129, 129)">
```

Izvor: izradio autor

2. način je unutarnji (*engl. internal*), a koristi se za definiranje stila za jednu cijelu HTML stranicu. Definira se unutar `<head>` elementa HTML stranice dodavanjem `<style>` elementa. Sintaksu unutarnjeg načina ubacivanja CSS kôda vidimo na slici broj 48.

Slika 48: Internal dodavanje CSS-a

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
body {background-color: powderblue;}  
h1 {color: blue;}  
p {color: red;}  
</style>  
</head>  
<body>  
  
<h1>This is a heading</h1>  
<p>This is a paragraph.</p>  
  
</body>  
</html>
```

Izvor: izradio autor

3. i posljednji način je vanjski (*engl. external*), gotovo sav CSS ovog završnog rada ubačen je na taj način. CSS kôd je u ovom slučaju odvojen od HTML datoteke, u zasebnoj je datoteci ekstenzije `.css`, a dodaje se na način da unutar `<head>` elemenata dokumenta ubacimo referencu na tu CSS datoteku pomoću `<link>` elementa. To u praksi izgleda kao na slici broj 49.

Slika 49: external dodavanje CSS-a

```
53     </title>
54     <meta name="viewport" content="width=device-width, user-scalable=no">
55     <link rel="stylesheet" type="text/css" href="style.css">
56     <link rel="icon" href="images/How_Logo_Alt.png">
57 </head>
```

Izvor: izradio autor

Unutar `<link>` elementa dodajemo atribut *rel*, on određuje vezu između trenutnog i vezanog dokumenta, zatim specificiramo tip datoteke atributom *type* koji ima vrijednost *text/css*, i zatim referencu na tu CSS datoteku pomoću atributa *href* kojemu dodajemo vrijednost imena te CSS datoteke zajedno sa njegovom ekstenzijom *.css* i ako je ta datoteka u drugoj mapi, moramo postaviti i određenu putanju do te datoteke. Važno je napomenuti da CSS datoteka ne smije sadržavati nikakav HTML kôd i mora biti spremljena sa ekstenzijom²⁶ (*engl. extension*) *.css*.

5.3. JavaScript

JavaScript je programski jezik za web, on je ujedno i najpopularniji programski jezik na svijetu, a osnovna svrha mu je ažuriranje i izmjena elemenata unutar HTML i CSS datoteka. JavaScript može manipulirati s više vrsta podataka kao što su brojevi, znakovi (*engl. string*), objekti, polja i funkcije. JavaScript datoteku možemo prepoznati po ekstenziji *.js*. Ubacivanje JavaScript kôda vršimo na načine prikazane na slici broj 50. (What is JavaScript: https://www.w3schools.com/whatis/whatis_js.asp, 2021.)

²⁶ Ekstenzija se odnosi na dio iza naziva bilo koje datoteke koji se odvaja točkom od datoteke. On nam govori koji je tip datoteke.

Slika 50: Primjer JavaScript kôda

```
47     </footer>
48     <script src="script.js"></script>
49 </body>
50 </html>
```

```
4 <head>
5 <script>
6
7     if ( window.history.replaceState ) {
8     window.history.replaceState( null, null, window.location.href );
9     }
10
11     function setScroll () {
12     window.onbeforeunload = function(e) {
13     localStorage.setItem('scrollpos', window.scrollY);
14     };
15     }
16
17     function getScroll () {
18
19     document.addEventListener("DOMContentLoaded", function(event) {
20     var scrollpos = localStorage.getItem('scrollpos');
21     if (scrollpos) window.scrollTo(0, scrollpos);
22     });
23     }
24 </script>
```

Izvor: izradio autor

5.3.1. JavaScript sintaksa

Varijable se u JavaScriptu deklariraju i inicijaliziraju kao što je prikazano na slici broj 51. Ključna riječ *var* označava da se radi o varijabli, a *x*, *y* i *z* su nazivi tih varijabli. Varijablama *x* i *y* dodane su vrijednost 5 i 6, a u varijablu *z* sprema se njihov zbroj, 11. Oni su tipa podatka *integer* (*int*). Varijable se još u JavaScriptu mogu označavati i sa ključnom riječi *const* za konstantu (slika broj 51) ili *let*. (What is JavaScript?: https://www.w3schools.com/whatis/whatis_js.asp, 2021.)

Slika 51: Varijable tipa int u JavaScript jeziku

```
var x = 5;
var y = 6;
var z = x + y;
```

Izvor: https://www.w3schools.com/whatis/whatis_js.asp

Na slici broj 52 vidimo da varijabla može sadržavati i tip podatka *string*²⁷.

²⁷ *String* je tip podatka u koji možemo spremiti bilo koji znak, brojevni, slovo ili simbol. Sa stringom ne možemo izvoditi matematičke operacije.

Slika 52: JavaScript varijabla tipa string

```
var car = "Fiat";
```

Izvor: https://www.w3schools.com/whatis/whatis_js.asp

Nadalje, varijabla može biti i tipa objekt (*engl. object*), pa joj tada pridružujemo više atributa i vrijednosti. Slična sintaksa kao i kod objekta je polje (*engl. array*). Tada jednoj varijabli pridružujemo set vrijednosti koji se nalaze na određenom polju. Ova dva primjera vidimo na slici broj 53.

Slika 53: Objekt i polje u JavaScript-u

OBJEKT

```
var car = {type:"Fiat", model:"500", color:"white"};
```

POLJE

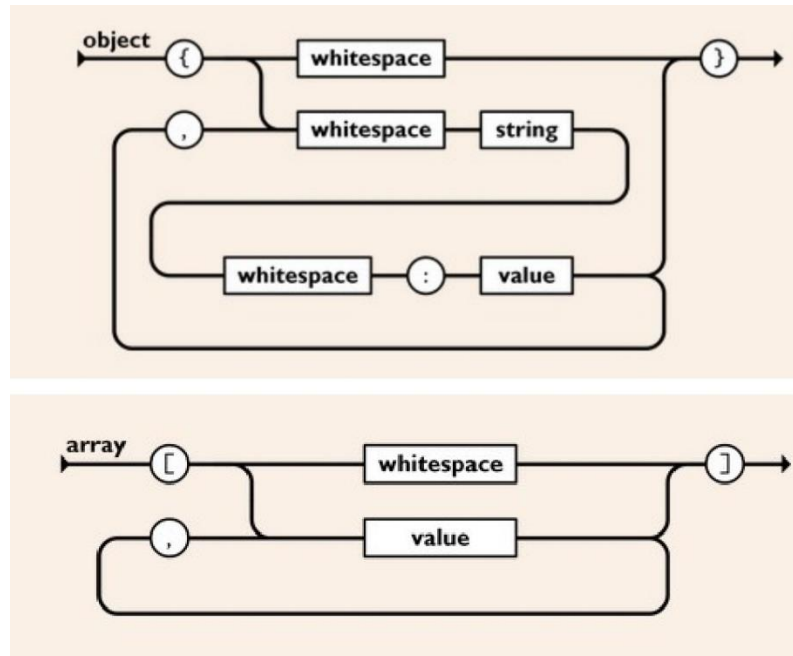
```
var cars = ["Saab", "Volvo", "BMW"];
```

Izvor: https://www.w3schools.com/whatis/whatis_js.asp

U JavaScript-u, atributi i podaci objekta pišu se unutar vitičastih zagrada i međusobno se odvajaju zarezom, dok se podaci u tipu podatka polje pišu između uglatih zagrada i isto odvajaju zarezom. Valja spomenuti da se podaci tipa *string* pišu u zagradama, dvostrukim ili jednostrukim (" " ili ' '), a podaci tipa *integer* ili primjerice *float*²⁸ se pišu bez zagrada, tako program prepoznaje da se radi o brojevnom podatku.

²⁸ Float je tip podatka za spremanje decimalnih brojeva

Slika 54: Shematski prikaz pisanja JavaScript objekta i polja



Izvor: <https://www.json.org/json-en.html>

5.3.2. JSON

JSON (JavaScript Object Notation) je otvoreni standard za čitljivu razmjenu podataka. Glavna prednost mu je što je lagan za čitanje i pisanje, a računalo ga jednostavno pretvara i generira. Utemeljen je na JavaScript standardu ECMA-262. U JSON strukturi mogu se koristiti različiti tipovi podataka poput brojeva, *string-ova*, *boolean-a*, nizova i objekata. (Introducing JSON: <https://www.json.org/json-en.html>, 2021.)

U ovom radu JSON nam je bitan zbog svoje podrške za PHP jezik, a očituje se u ove dvije funkcije: `json_encode()` i `json_decode()`.

`Json_encode()` je funkcija koja prima podatak tipa *array* (polje), a vraća reprezentaciju argumenata kao tip podatka *string* u JSON formatu ili u slučaju greške vraća rezultat *false*. Primjer u ovom završnom radu vidimo kod postavljanja „kolačića“ (*engl. cookie*). (Introducing JSON: <https://www.json.org/json-en.html>, 2021.)

Slika 55: Postavljanje kolačića u PHP jeziku

```
219     setcookie($cookieName, json_encode($userCart), time() + (10 * 365 * 24 * 60 * 60));  
220  
221     setcookie($counterCookieName, json_encode($cartItemsCount), time() + (10 * 365 * 24 * 60 * 60));  
222
```

Izvor: izradio autor

Da objasnimo sliku, u varijable `$userCart` i `$cartItemsCount` spremaju se podaci tipa *array*. Da bi mogli koristiti funkciju `setcookie()`, koja služi da stvaranje kolačića moramo joj proslijediti argument tipa *string*, zbog toga koristimo `json_encode()` i sada JSON pretvara to polje u *stringove* koje možemo spremiti u kolačić.

Na sličan način radi i funkcija `json_decode()`, samo što ona na osnovu nekakvog *stringa* koji smo joj prosljedili formira odgovarajući niz polja i vraća ga ili u slučaju greške vraća rezultat `NULL`²⁹.

Slika 56: `json_decode` funkcija

```
544  
545     $cartRestore = json_decode($_COOKIE[$cookieName], true);  
546     $counterRestore = json_decode($_COOKIE[$counterCookieName], true);  
547
```

Izvor: izradio autor

5.3.3. JavaScript – HTML DOM metode

JavaScript - HTML DOM metode su radnje koje možemo činiti nad HTML elementima, tj. kako možemo mijenjati njihova svojstva pomoću kôda JavaScript-a. HTML DOM-u se može pristupiti i drugim programskim jezicima, ali ovdje ćemo se koncentrirati na JavaScript. (JavaScript – HTML DOM Methods: https://www.w3schools.com/js/js_html_dom_methods.asp, 2021.)

²⁹ `NULL` je vrijednost podatka koja nije upisana, odnosno, kada polje za podatak ostane prazno ima vrijednost `NULL`. Napomena: `NULL` i `0` nije isto. (SQL `NULL` Values: https://www.w3schools.com/sql/sql_null_values.asp, 2021.)

Metoda *getElementById* je način da pristupimo HTML elementu koristeći njegovo svojstvo *id*.

Slika 57: Pristup elementima preko id atributa

```
10 const fName = document.getElementById('fname');
11 const lName = document.getElementById('lname');
12 const email = document.getElementById('email');
13 const contact = document.getElementById('contact');
14 const adress = document.getElementById('adress');
15 const city = document.getElementById('city');
16 const zip = document.getElementById('zip');
```

Izvor: izradio autor

Na slici broj 57 vidimo primjere deklaracija varijabli preko *id* svojstva elementa. Dakle sintaksa je sljedeća, pišemo ključnu riječ *document*, zatim točku i onda pozivamo metodu *getElementById* i u zaglavlje metode upisujemo *id* elementa kojeg želimo odabrati. Na slici ispod vidimo te HTML elemente.

Slika 58: HTML elementi s id atributima

```
32 <label for="fname">Ime</label>
33 <label for="lname">Prezime</label>
34 <input type="text" id="fname" name="fname" placeholder="Ime" value="<?php echo $_SESSION['fname']; ?>" disabled>
35 <input type="text" id="lname" name="lname" placeholder="Prezime" value="<?php echo $_SESSION['lname']; ?>" disabled>
36 <label for="email">E-mail adresa</label>
37 <label for="contact">Kontakt</label>
38 <input type="text" id="email" name="email" placeholder="E-mail" value="<?php echo $_SESSION['email']; ?>" disabled>
39 <input type="text" id="contact" name="contact" placeholder="Kontakt" value="<?php echo $_SESSION['contact']; ?>" disabled>
40 <label for="adress">Adresa</label>
41 <label for="city">Grad</label>
42 <input type="text" id="adress" name="adress" placeholder="Adresa" value="<?php echo $_SESSION['adress']; ?>" disabled>
43 <input type="text" id="city" name="city" placeholder="Grad" value="<?php echo $_SESSION['city']; ?>" disabled>
44 <label for="zip">Poštanski broj</label>
45 <label for="emptyinput"></label>
46 <input type="text" id="zip" name="zip" placeholder="Poštanski broj" value="<?php echo $_SESSION['zip']; ?>" disabled>
```

Izvor: izradio autor

Slika broj 58 prikazuje HTML elemente, čije *id* attribute možemo pronaći u prethodno spomenutom JavaScript kôdu. Kod metode *getElementById*, jasno je o kojem se točno elementu radi, budući da se u jednoj HTML datoteci pojavljuje samo jedan element s istim *id* atributom. Osim te metode, važno je spomenuti i metodu *getElementsByClassName*, koja funkcionira na sličan način kao i prva spomenuta metoda, samo što se element traži pomoću vrijednosti atributa klase.

Slika 59: Pristup HTML dokumentima preko klase

```
1  const menuButton = document.getElementsByClassName('menu-button')[0];
2  const menuBar = document.getElementsByClassName('bar');
3  const navLinks = document.getElementsByClassName('nav-links')[0];
4  const navbarContainer = document.getElementsByClassName('navbar-container')[0];
```

Izvor: izradio autor

Na primjeru broj 59 možemo primijetiti jednu razliku, a to je da kada tražimo element po atributu klase moramo odrediti i koji je to element u dokumentu, pomoću odabiranja tog polja sintaksom `[0]`. Unutar uglatih zagrada ide broj polja elementa kojeg odabiremo, po onom redu po kojem se slažu unutar HTML dokumenta, a to činimo iz razloga što više HTML dokumenata može imati istu klasu. U 2. liniji koda vidimo da polje nije određeno, što znači da se u konstantu `menuBar` spremaju svi elementi te klase.

5.3.4. JavaScript funkcije

JavaScript funkcije su komadi kôda (blokovi) koji nešto izvršavaju, najčešće nad HTML elementima. Funkcija se definira na način da upišemo ključnu riječ *function*, nakon čega ide razmak i ime funkcije, te obične zagrade unutar kojih idu parametri funkcije (*parametar1*, *parametar2*, ...). Važno je napomenuti da funkcija ne mora imati definirane parametre, kao i u drugim programskim jezicima. Kod koji se izvršava unutar funkcije ograđen je vitičastim zgradama `{}`. Pogledajmo primjer na slici broj 60.

Slika 60: JavaScript funkcija enableEditing()

```
47 function enableEditing () {
48     userUpdate.style.opacity = 0;
49     if (confirmChangesButton.hidden == true) {
50         confirmChangesButton.hidden = false;
51         editButton.style.backgroundColor = "rgb(200, 0, 0)";
52         editButton.style.borderColor = "rgb(200, 0, 0)";
53         editButton.firstChild.data = "DOVRŠI BEZ SPREMANJA";
54         userPersonalData.style.border = "solid red 2px";
55         dataChange.style.opacity = "100%";
56         fName.disabled = false;
57         lName.disabled = false;
58         email.disabled = false;
59         contact.disabled = false;
60         adress.disabled = false;
61         city.disabled = false;
62         zip.disabled = false;
63     }
64
65     else {
66         confirmChangesButton.hidden = true;
67         editButton.style.backgroundColor = "rgb(138, 0, 0)";
68         editButton.style.borderColor = "rgb(138, 0, 0)";
69         editButton.firstChild.data = "PROMJENA PODATAKA";
70         userPersonalData.style.border = "none";
71         dataChange.style.opacity = "0";
72         fName.disabled = true;
73         lName.disabled = true;
74         email.disabled = true;
75         contact.disabled = true;
76         adress.disabled = true;
77         city.disabled = true;
78         zip.disabled = true;
79     }
80 }
```

Izvor: izradio autor

Funkcija se izvršava u onom trenutku kada se pozove unutar dokumenta. Poziv funkcije se izvršava tako da napišemo ime funkcije na mjestu gdje ju želimo izvršiti, sa svim njenim parametrima, ako ih ima.

Slika 61: Poziv funkcije pomoću onclick eventa

```
48 <div class="action-container">
49     <button id="edit-button" onclick="enableEditing();" form="none">IZMJENA PODATAKA</button>
50     <button type="submit" id="confirm-changes-button" name="confirm_changes" hidden>POTVRDI IZMJENE</button>
51 </div>
```

Izvor: izradio autor

Funkciju na slici pozivamo pomoću JavaScript *eventa onclick*. Što znači da kada pritisnemo gumb *IZMJENA PODATAKA* poziva se i izvršava navedena funkcija. Rezultat objašnjene funkcije pogledajmo na slici ispod.

Slika 62: Rezultat funkcije *enableEditing()*

The screenshot shows a user profile page for 'Šiško Menčetić'. The page has a header with the logo 'House of Wine' and navigation links. Below the header, the user's name is displayed. The main content area contains a form for editing user data. The form is titled 'IZMJENA PODATAKA' and has two columns of fields. The left column contains: 'Ime' (Šiško), 'E-mail adresa' (siskomencetic@email.com), 'Adresa' (Augusta Senoe 1/B), and 'Poštanski broj' (51000). The right column contains: 'Prezime' (Menčetić), 'Kontakt' (+385999999999), and 'Grad' (Rijeka). A red button labeled 'DOKRAJ BEZ SPREMANJA' is positioned to the right of the 'Prezime' field, and a green button labeled 'POTVRDI IZMJENE' is positioned to the right of the 'Kontakt' field. Below the form, the text 'Moj narudžbe' is visible.

Izvor: izradio autor

6. Primjeri kôda iz projekta

U ovom poglavlju proći ćemo kroz primjere kôda svih jezika koji su se koristili za ovaj projekt. Sav navedeni kôd je pisan samostalno, uz pomoć *engl. tutorial*-a sa weba (*W3Schools*, *YouTube* i *Stack Overflow*).

6.1. Algoritam

Algoritam je definiran kao konačan slijed naredbi za rješavanje određenog zadatka, konkretnije u računalnoj terminologiji, to je niz naredbi koje zadajemo računalu za rješavanje problema. Algoritam je definiran još davno u matematici, kroz osnovna pravila kod rješavanja matematičkih problema poput jednadžbi. Algoritmi imaju nekoliko svojstava, tj. algoritam mora imati određena svojstva kako bi bio algoritam. On mora biti konačan (nakon određenog broja koraka daje rezultat), diskretan (znači da mora izvoditi diskretne radnje koje vode rješenju, svaki korak mora biti jednostavan i razumljiv, kako bi riješio složen problem), determinantan (za iste uvijete uvijek će vratiti iste vrijednosti), imati svojstvo masovnosti (nije nužno, ali je poželjno da bude primjenjiv na veći broj ulaznih vrijednosti, upotrebljiv na više različitih, ali sličnih problema) i efikasan (u što kraćem roku rješava što više problema). (Pojam algoritma: <https://mooc.carnet.hr/mod/book/tool/print/index.php?id=26661>, 2021.)

U računalstvu postoji tzv. problem „beskonačne petlje“ (*engl. infinite loop*), a veže se uz svojstvo konačnosti algoritma. To se događa kada se algoritam ponavlja i vrti u krug, zbog određenih vrijednosti koje vraća. Npr. ako imamo *for* petlju³⁰ unutar koje smo definirali varijablu na inicijalnu vrijednost 0 i prosljedili programu naredbu da se radnje unutar te petlje ponavljaju sve dok je ta varijabla jednaka 0, a ona je uvijek jednaka 0, barem dok joj ne definiramo neku drugu vrijednost i tako ostvarujemo izlaz iz te beskonačnosti.

³⁰ Petlja u računalstvu označava dio kôda koji se izvršava sve dok se ne zadovolji postavljeni uvjet za prekid tog bloka naredbi.

Slika 63: Algoritam koji se izvršava pomoću for petlje

```
<?php
$maxQuantity = $product["maxquantity"];

for ($i = 1; $i <= $maxQuantity; $i++) { ?>
    <option <?php if ($product["quantity"] == $i) echo "selected"; ?> value=<?php echo $i; ?><?php echo $i; ?></option>
}
?>
```

Izvor: izradio autor

6.1.1. PHP i SQL algoritmi

PHP i SQL algoritmi su stavljeni u jedno poglavlje, jer su često korišteni u kombinaciji, tj. preko PHP jezika se provode akcije nad relacijskom bazom podataka.

Slika 64: Algoritam za registraciju korisnika

```
19
20 if (isset($_POST['register'])) {
21
22     //postavljanje vrijednosti varijabli koje unosi korisnik
23     $email = $_POST['email'];
24     $password = $_POST['password'];
25     $confirmPassword = $_POST['confirmPassword'];
26     $fname = $_POST['fname'];
27     $lname = $_POST['lname'];
28     $contact = $_POST['contact'];
29     $adress = $_POST['adress'];
30     $city = $_POST['city'];
31     $zip = $_POST['zip'];
32
33     //razne provjere tocnosti unosa podataka
34     if (empty($email)) {
35         $error .= "emailEmpty";
36         $validMail = false;
37     }
38
39     if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
40         $error .= "emailInvalid";
41         $validMail = false;
42     }
43
44     if (empty($password)) {
45         $error .= "passwordEmpty";
46     }
47
48     if (strlen(trim($password)) < 6) {
49         $error .= "passwordInvalid";
50     }
51
52     if ($password != $confirmPassword) {
53         $error .= "passwordMismatch";
54     }
55
56     if (empty($fname)) {
57         $error .= "fnameEmpty";
58     }
59
60     if (empty($lname)) {
61         $error .= "lnameEmpty";
62     }
63 }
```

Izvor: izradio autor

U prvom primjeru vidimo *if* naredbu, koja provjerava je li korisnik kliknuo gumb *register* pomoću metode *POST*, i ako je svi podaci koje je upisivao u predviđena polja se ponovno kroz *\$_POST* varijable upisuju u pripadajuće varijable od linije 23 do linije 31. Nakon toga slijedi provjera točnosti upisa podataka. Npr. funkcija *empty*, provjerava da li je varijabla prazna, jer je obavezno da korisnik ne ostavi prazno polje pri unosu podataka. U tom slučaju se u varijablu tipa *string* nadopunjuje (*engl. append*) opis te greške, kako bi kasnije mogli ispisati korisniku odgovarajuću poruku. Drugi primjer je funkcija *filter_var* koja provjerava točnost oblika e-mail adrese (mora biti u formatu: *tekst@tekst.tekst*) pomoću filtera *FILTER_VALIDATE_MAIL*. Uskličnik „!“ ispred funkcije *filter_var* unutar naredbe *if* ima značenje „ne“, tj. izvrši taj kôd ako e-mail nema pravilan format. Varijabla *\$validMail*, koju pratimo u sljedećem koraku, se u slučaju greške postavlja na vrijednost *false*.

Slika 65: Algoritam za provjeru da li korisnik već postoji

```
79
80     if ($validMail === true) {
81         //varijable koje služe za provjeru da li korisnik sa unesenim e-mailom već postoji u bazi
82         $userQuery = "SELECT email FROM user WHERE email = '$email' LIMIT 1";
83         $userResult = mysqli_query($connection, $userQuery);
84         $userRow = mysqli_fetch_assoc($userResult);
85
86         if ($userRow) {
87             $error .= "emailExisting";
88         }
89     }
90 }
91
```

Izvor: izradio autor

Na slici broj 65 vidimo provjeru da li postoji već korisnik s takvim e-mailom, jer e-mail je unikatan podatak (ne mogu biti dva korisnika s istim e-mailom). Provjeravamo je li varijabla *\$validMail* postavljena na vrijednost *true*, ako je, tada se u varijablu *\$userQuery* sprema SQL upit *SELECT* za odabiranje atributa *email* iz tablice *user* i zatim se u varijablu *\$userResult* sprema funkcija *mysqli_query*, koja izvršava SQL upit u našoj bazi podataka, a funkcija *mysqli_fetch_assoc* dohvaća redove iz tablice koji odgovaraju SQL upitu i sprema ih u polje, koje mi spremamo u varijablu *\$userRow*. Nakon toga ide provjera da li takvo polje postoji, i ako da

ispisuje se poruka korisniku da je takav e-mail već registriran u bazi. Nakon toga idemo na sljedeći dio algoritma koji služi za unos podataka u bazu i još neke provjere.

Slika 66: Algoritam za unos u bazu podataka i dodatne provjere

```
91
92     if (empty($error)) {
93
94         $passwordEncrypted = password_hash($password, PASSWORD_DEFAULT);
95         $sqlQuery = "INSERT INTO user (email, password, fname, lname, contact, address, city, zip)
96         VALUES ('$email', '$passwordEncrypted', '$fname', '$lname', '$contact', '$address', '$city', '$zip')";
97
98         mysqli_query($connection, $sqlQuery);
99
100        $userQuery = "SELECT * FROM user WHERE email='$email' ";
101        $userResult = mysqli_query($connection, $userQuery);
102        $userRow = mysqli_fetch_assoc($userResult);
103
104        $_SESSION['loggedIn'] = true;
105        $_SESSION['userId'] = $userRow['user_id'];
106        $_SESSION['email'] = $userRow['email'];
107        $_SESSION['fname'] = $userRow['fname'];
108        $_SESSION['lname'] = $userRow['lname'];
109        $_SESSION['contact'] = $userRow['contact'];
110        $_SESSION['address'] = $userRow['address'];
111        $_SESSION['city'] = $userRow['city'];
112        $_SESSION['zip'] = $userRow['zip'];
113        $_SESSION['success'] = "Uspješno ste se registrirali i prijavili.";
114
115        if (isset($_SESSION["shopping_cart"])) {
116            $userSelectQuery = "SELECT user_id FROM user WHERE email = '$email' LIMIT 1";
117            $userSelectResult = mysqli_query($connection, $userSelectQuery);
118            $userSelectRow = mysqli_fetch_assoc($userSelectResult);
119
120            $_SESSION['userId'] = $userSelectRow['user_id'];
121
122            $userId = $_SESSION['userId'];
123            $guestId = $_SESSION['guestId'];
124
125            $orderUpdateQuery = "UPDATE order SET user_id_fk = '$userId' WHERE user_id_fk = '$guestId'";
126            mysqli_query($connection, $orderUpdateQuery);
127
128            $guestDeleteQuery = "DELETE FROM user WHERE user_id = '$guestId'";
129            mysqli_query($connection, $orderUpdateQuery);
130        }
131        header('Location: index.php');
132    }
133 }
134
```

Izvor: izradio autor

Dio kôda na slici broj 66 provjerava da li je *string* *\$error* prazan, i ako je, znači da nije bilo ni jedne greške prilikom unosa podataka. Sljedeći korak je enkripcija³¹ lozinke pomoću funkcije *password_hash* koja izvršava enkripciju proslijeđene varijable odabranim algoritmom, u ovom slučaju *PASSWORD_DEFAULT*. Nakon toga se upisuje SQL naredba u varijablu *\$sqlQuery* za umetanje novog retka podataka u tablicu *user* i upisuju se svi podaci koje je korisnik upisao. Nakon što se izvrši upis podataka, podaci tog istog korisnika se odabiru iz tablice i njihove vrijednosti se

³¹ Enkripcija je način šifriranja podataka putem odgovarajućih ključeva, samo se pomoću ključa podaci mogu dekriptirati (obnuti proces).

upisuju u odgovarajuće super-globalne varijable `$_SESSION`, kako bi smo ih mogli koristiti u svim dijelovima stranice. Za kraj se još provjerava da li je korisnik prije registracije imao proizvode u košarici. U slučaju da je tada se podaci o košarici spremaju u novi redak tablice `order`, a stari podaci (dok nije bio registriran, već prijavljen kao gost), se brišu. Taj dio kôda osigurava da korisnik ne mora ponovno stavljati proizvode u košaricu, nego samo nastavlja kupnju. Zadnji red algoritma nas vodi pomoću funkcije `header` natrag na početnu stranicu.

Slika 67: Poruke greški na korisničkom sučelju

Vinski podrum Oprema Posebna ponuda Izbor sireva O nama

Registrirajte se

neispravanoblikmaila
Nispravna e-mail adresa.

Lozinka
Niste unijeli lozinku

Potvrdite lozinku

Ime
Niste unijeli ime.

Prezime
Niste unijeli prezime

Kontakt broj
Niste unijeli kontakt broj.

Adresa
Niste unijeli adresu.

Grad
Niste unijeli grad.

Poštanski broj
Niste unijeli poštanski broj.

REGISTRACIJA

Izvor: izradio autor

Slika broj 67 prikazuje i kako se objašnjeni kôd prikazuje korisniku u web pregledniku, a slika ispod provjere unutar same HTML datoteke.

Slika 68: Provjera podataka unutar HTML datoteke

```
18 <div class="register-container">
19 <h4>Registrirajte se</h4>
20 <form action="register.php?action=register" method="post">
21 <?php if (strpos($error, 'emailExisting') !== false) : ?>
22 <p class="error-text existing-account">Račun s ovim e-mailom već je registriran.</p>
23 <?php endif ?>
24 <input type="text" id="email" name="email" placeholder="E-mail adresa" value="<?php echo $email; ?>"><br>
25 <?php if (strpos($error, 'emailEmpty') !== false) : ?>
26 <p class="error-text">Niste unijeli e-mail adresu.</p>
27 <?php endif ?>
28 <?php if (strpos($error, 'emailInvalid') !== false && strpos($error, 'emailEmpty') === false) : ?>
29 <p class="error-text">Neispravna e-mail adresa.</p>
30 <?php endif ?>
31 <input type="password" id="password" name="password" placeholder="Lozinka" value="<?php echo $password; ?>"><br>
32 <?php if (strpos($error, 'passwordEmpty') !== false) : ?>
33 <p class="error-text">Niste unijeli lozinku.</p>
34 <?php endif ?>
```

Izvor: izradio autor

Pogledajmo još jedan primjer PHP algoritma na slici broj 69.

Slika 69: Algoritam za spremanje proizvoda u bazu podataka

```
590
591 else if (isset($_SESSION['loggedIn'])) {
592
593     $randGuestId = rand(1000000000000000, 9999999999999999);
594     $guestName = "Guest" . "#" . $randGuestId;
595
596     $userQuery = "INSERT INTO user (fname) VALUES ('$guestName')";
597     mysqli_query($connection, $userQuery);
598
599     $userSelectQuery = "SELECT user_id FROM user WHERE fname = '$guestName' LIMIT 1";
600     $userSelectResult = mysqli_query($connection, $userSelectQuery);
601     $userSelectRow = mysqli_fetch_assoc($userSelectResult);
602
603     $userId = $userSelectRow['user_id'];
604     $_SESSION['guestId'] = $userId;
605 }
606
607 $randCode = rand(10000000, 99999999);
608 $orderCode = "#" . $randCode;
609
610 $orderQuery = "INSERT INTO `order` (user_id_fk, order_code, completion) VALUES ('$userId', '$orderCode', 0)";
611 mysqli_query($connection, $orderQuery);
612
613 $orderSelectQuery = "SELECT order_id FROM `order` WHERE user_id_fk = '$userId' AND completion = 0 LIMIT 1";
614 $orderSelectResult = mysqli_query($connection, $orderSelectQuery);
615 $orderSelectRow = mysqli_fetch_assoc($orderSelectResult);
616
617 $orderId = $orderSelectRow['order_id'];
618 $ciQuantity = 1;
619 $ciTotal = $ciQuantity * $price;
620
621 $cartItemSelectQuery = "SELECT * FROM cart_item WHERE order_id_fk = '$orderCode' AND ci_code = '$code' LIMIT 1";
622 $cartItemSelectResult = mysqli_query($connection, $cartItemSelectQuery);
623 $cartItemSelectRow = mysqli_fetch_assoc($cartItemSelectResult);
624
625 if (!$cartItemSelectRow) {
626
627     $cartItemQuery = "INSERT INTO cart_item (order_id_fk, ci_code, ci_name, ci_price, ci_quantity, total_price) VAL
628     mysqli_query($connection, $cartItemQuery);
629 }
630
631 }
```

Izvor: izradio autor

To je algoritam koji se pokreće kada korisnik doda proizvod u košaricu i tada se izvršavaju razne SQL naredbe za umetanje podataka o proizvodima iz košarice u tablicu baze podataka.

Sintaksa je vrlo slična kao i u prethodno prikazanom algoritmu, stoga nećemo detaljno objašnjavati liniju po liniju.

6.1.2. JavaScript algoritam

U ovom dijelu rada bit će opisan JavaScript algoritam koji izvršava „izvlačenje“ izbornika za navigaciju stranicom u responzivnom pregledu stranice, koji je opširnije opisan u poglavlju o responzivnom dizajnu.

Slika 70: Funkcija `addEventListener`

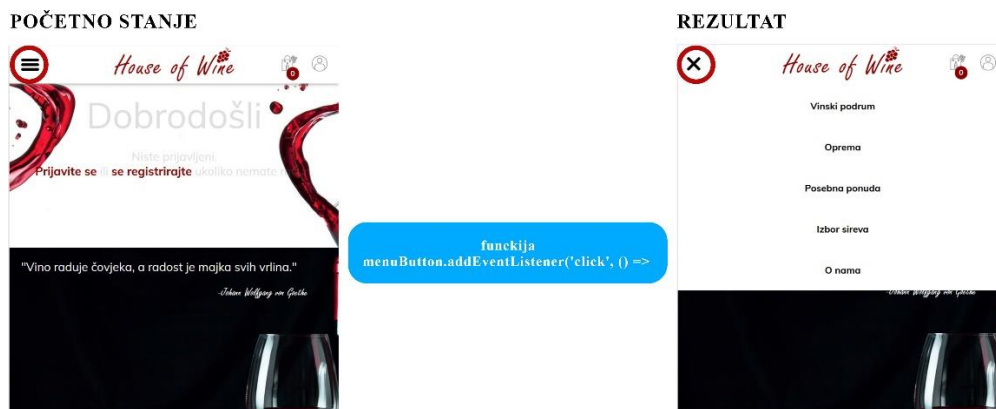
```
18
19  menuButton.addEventListener('click', () => {
20      navlinks.classList.toggle('active');
21      menuButton.classList.toggle('active');
22
23      for (var i = 0; i < menuBar.length; i++) {
24          if (i == 0) {
25              if (menuBar[i].style.transform == "rotate(45deg) translateY(225%)")
26                  menuBar[i].style.transform = "rotate(0deg) translateY(0%)";
27              else
28                  menuBar[i].style.transform = "rotate(45deg) translateY(225%)";
29          }
30
31          else if (i == 1) {
32              if (menuBar[i].style.display == "none")
33                  menuBar[i].style.display = "initial";
34              else
35                  menuBar[i].style.display = "none";
36          }
37
38          else if (i == 2) {
39              if (menuBar[i].style.transform == "rotate(-45deg) translateY(-225%)")
40                  menuBar[i].style.transform = "rotate(0deg) translateY(0%)";
41              else
42                  menuBar[i].style.transform = "rotate(-45deg) translateY(-225%)";
43          }
44      }
45  })
46
```

Izvor: izradio autor

Funkcija se zasniva na događaju ili *eventu* koji nastaje prilikom klika miša na element *menuButton*. Nakon što smo kliknuli na taj gumb „uključuju“ se klase s atributom *active* iz CSS-a i izvršavaju se promjene nad elementima. Pomoću *for* petlje unutar koje je *if* naredba određujemo na kojem elementu se izvršavaju promjene, jer svaki od 3 elementa ima različit cilj. Ovim

elementima *menuBar* činimo izmjene u CSS kôdu pomoću ključne riječi *style* u JavaScript-u. Rezultat koda vidimo na slici broj 71.

Slika 71: Rezultat *addEventListener* funkcije



Izvor: izradio autor

6.2. HTML i CSS primjeri kôda

Kao i za PHP i SQL, HTML i CSS su usko povezani, pa ćemo primjere kôda gledati kroz spoj te dvije tehnologije. Promotrimo dio HTML kôda na slici u datoteci *cart.php*, u kojoj prikazujemo web stranicu košarice (unutar kojeg je ugrađen i PHP, ali njega ćemo ignorirati u ovom dijelu, samo je sadržajno bitan).

Slika 72: HTML kôd

```
108 <main>
109 <div class="heading-cart">
110 <h2>Moja košarica</h2>
111 </div>
112 <div class="heading-small">
113 <h2>Pregled proizvoda</h2>
114 </div>
115 <div class="cart-container">
116 <?php
117
118 if (isset($_SESSION["shopping_cart"])) {
119     $totalPrice = 0;
120     $totalQuantity = 0;
121 }>
122 <div class="cart-left-container">
123 <div class="cart-products-container">
124 <div class="title-bar-container">
125 <div class="title-name">
126     proizvod
127 </div>
128 <div class="title-price">
129     cijena
130 </div>
131 <div class="title-quantity">
132     količina
133 </div>
134 <div class="title-total">
135     iznos
136 </div>
137 </div>
138 <?php
139
140 foreach ($_SESSION["shopping_cart"] as $product) {
141 }>
142 <div class="info-bar-container">
143 <div class="product-img-name-remove">
144 <div class="product-img">
145 <a href="product.php?code=<?php echo $product["code"] . "&ptype=" . $pr
146 </div>
147 <div class="product-name-remove">
148 <div class="product-name">
149 <a href="product.php?code=<?php echo $product["code"] . "&ptype="
150 </div>
151 <div class="product-remove">
152 <form action="cart.php?actionproductRemove" method="post">
```

Izvor: izradio autor

Cijeli kôd je prevelik da bi stao na jednu sliku, ali ovdje možemo uočiti kako se dijelovi stranice koju ćemo kasnije vidjeti na slici 74 slažu unutar jedne HTML datoteke. Točnije kako se elementi različitih klasa ugnježđuju jedan u drugi i tako tvore već prije objašnjeni HTML DOM. Međutim, to je samo struktura dokumenta, da bi posložili elemente na način na koji želimo dodajemo im i CSS kôd iz CSS datoteke *style.css*.

Slika 73: CSS kôd

```
969
970 .cart-products-container {
971     display: grid;
972     grid-template-rows: 1fr 8fr;
973     grid-template-columns: 1fr;
974     grid-template-areas:
975         "title-bar-container"
976         "info-bar-container";
977     grid-gap: 20px;
978 }
979
980 .cart-subtotal-container {
981     display: grid;
982     grid-area: "cart-subtotal-container";
983     grid-template-rows: 1fr 4fr;
984     grid-template-columns: 1fr;
985     grid-template-areas:
986         "subtotal-title"
987         "subtotal-info";
988     background-color: transparent;
989     /*border: solid rgb(0, 0, 0) 2px;*/
990     color: #rgb(129, 129, 129);
991     max-height: 220px;
992     min-height: 220px;
993     border-radius: 0px;
994 }
995
996 .buy-button {
997     display: grid;
998     grid-area: "buy-button";
999     align-self: end;
1000     justify-items: center;
1001     align-items: center;
1002     padding: 0 5% 0 5%;
1003     margin-bottom: 35px;
1004 }
1005
1006 .buy-button form {
1007     width: 100%;
1008     height: 100%;
1009     padding-bottom: 6%;
1010 }
1011
```

Izvor: izradio autor

Ovdje je također prikazan samo dio kôda koji je pisan po onim pravilima koje smo spomenuli u poglavlju o CSS-u. Tu vidimo brojne opcije i bogatstvo koje nam pruža CSS3. A na slici ispod vidimo rezultat kombinacije HTML i CSS-a na ovom projektu.




Slika 74: Prikaz košarice (cart.php)

House of Wine

Vinski podrum Oprema Posebna ponuda Izbor sireva O nama

Moja košarica

Pregled proizvoda

proizvod	cijena	količina	iznos
 Luris Rosé Bicklisti 0,75 l <input type="button" value="UKLONI"/>	32,99 kn	1	32,99 kn
 Korlat Cabernet Sauvignon vrhunsko vino 0,75 l <input type="button" value="UKLONI"/>	99,99 kn	1	99,99 kn
 Dekanter Veliki <input type="button" value="UKLONI"/>	100,00 kn	1	100,00 kn

Pregled košarice

Broj proizvoda 3
Ukupna cijena 232,98 kn

Košarica ukupno* 232,98 kn

NA PLAĆANJE

Izvor: izradio autor

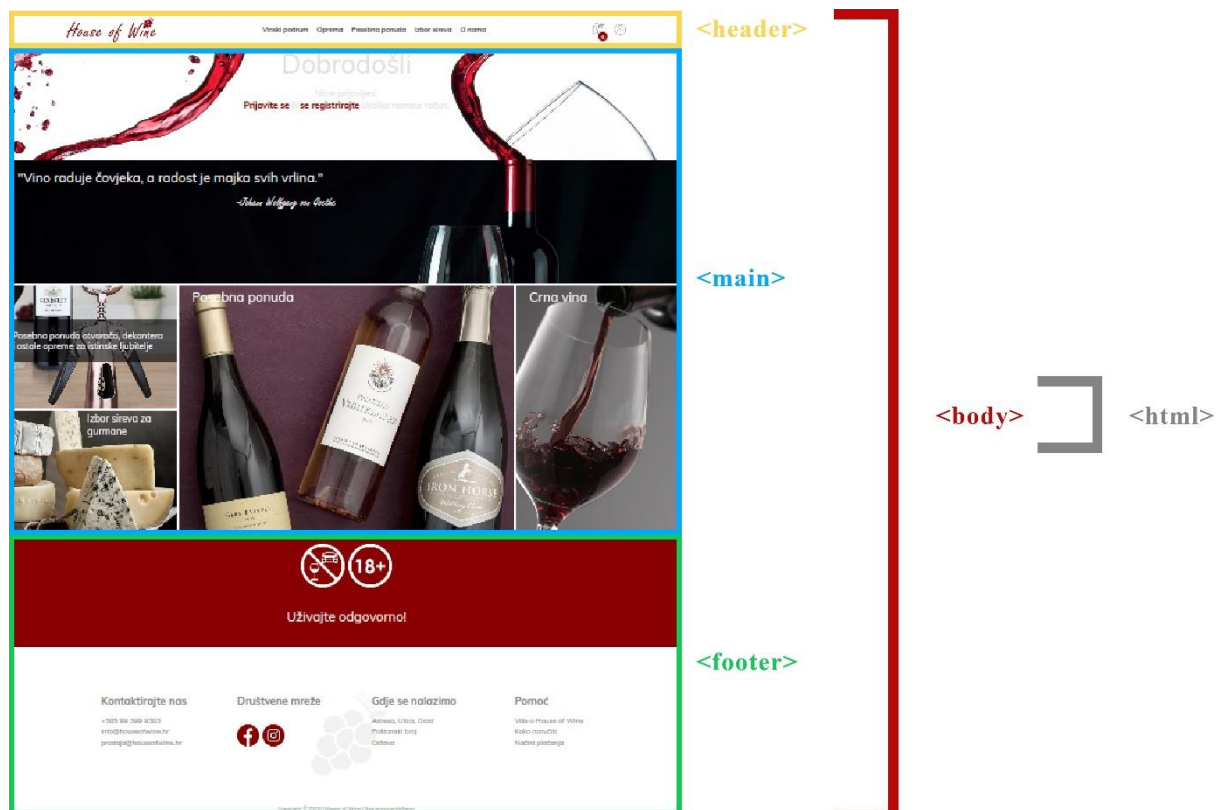
7. Web trgovina „House of Wine“

U ovom poglavlju je objašnjena funkcionalnost cijelog projekta, kao i primjeri sučelja kroz koje korisnik prolazi tijekom posjete stranici. Stranica je testirana i otvarana uz nekoliko web preglednika na više uređaja. Za testiranje su korištena računala s Windows 10 operacijskim sustavom i pametni telefon s Android 10 operacijskim sustavom. A od web preglednika, upotrebljavani su Google Chrome, Firefox, Microsoft Edge i Brave.

7.1. Struktura stranice

Na primjeru ćemo proučiti naslovnu stranicu web trgovine „House of Wine“, kako bi bolje razumjeli samu strukturu dokumenta, pa tako i ostalih dokumenata koji su dio ove trgovine.

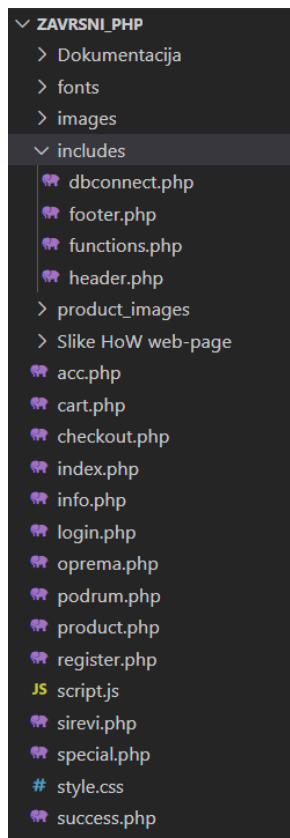
Slika 75: Raspored početne stranice



Izvor: izradio autor

Na slici broj 75 je bitno primijetiti da ova web stranica ima dva elementa koji se ponavljaju i na ostalim stranicama, a to su elementi *header* i *footer*. Zbog toga, ta dva elementa su spremljena kao odvojene datoteke, kako bi se mogle pozivati na drugim stranicama bez nepotrebnog ponavljanja kôda.

Slika 76: Struktura i poredak datoteka unutar projekta



Izvor: izradio autor

7.1.1. Datoteka header.php

Datoteka *header.php* je datoteka unutar koje je spremljen *header* dio web stranice, ali također i *head*, kao i svi pripadajući meta-podaci koji se vežu uz pojedine dijelove stranice. Unutar *header*-a nalazi se i *navigation* koji nam služi kao navigacija za kretanje kroz ostale web stranice, kao što su ponuda proizvoda, podaci o stranici, košarica, profil, itd. Na početku svake stranice

koristimo PHP funkciju `include ("includes/header.php")` kako bi uključili `header` datoteku iz mape `includes` u te dokumente.

Slika 77: Uključivanje header datoteke

```
1 <?php
2
3 include ("includes/functions.php");
4 include ("includes/header.php");
5 ?>
```

Izvor: izradio autor

7.1.2. Datoteka footer.php

Slično kao i `header`, `footer` element se također ponavlja kroz više stranica i spremljen je u datoteku `footer.php`. Za razliku od `header`-a, ta se datoteka poziva na kraju svake stranice, također PHP funkcijom `include ("includes/footer.php")`. Ona sadrži i referencu za JavaScript datoteku koja je vezana uz projekt.

Slika 78: Uključivanje footer datoteke

```
59
60 <?php
61
62 include ("includes/footer.php");
63 ?>
```

Izvor: izradio autor

7.2. Putanja korisnika kroz kreiranje narudžbe

U ovom dijelu završnog rada bit će opisano kako se korisnik kreće kroz pojedine dijelove stranice i koje se to sve radnje može izvršavati na stranici.

7.2.1. Naslovna (početna) stranica

Kada korisnik otvori web lokaciju „House of Wine“ dolazi na njenu početnu stranicu (*index.php*). U gornjem dijelu stranice nalazi se izbornik, kao i tekst dobrodošlice i ponuđene opcije za prijavu ili registraciju na stranicu.

Slika 79: Naslovna stranica web stranice "House of Wine"



Izvor: izradio autor

Nakon toga korisnik klikom miša na vezu *registrirajte se* odlazi na stranicu za registraciju.

7.2.2. Stranica za registraciju

Nakon što je korisnik došao na stranicu za registraciju prikazuje mu se sljedeće.

Slika 80: Stranica za registraciju

House of Wine

Vinski podrum Oprema Posebna ponuda Izbor sireva O nama

Registracija

Nemate račun kod nas?

Registrirajte se

E-mail adresa

Lozinka

Potvrdite lozinku

Ime

Izvor: izradio autor

Korisnik zatim upisuje tražene podatke u HTML *form*-u prikazanu na slici i klikom na gumb *REGISTRACIJA* se POST metodom putem HTTP-a šalju serveru podaci za registraciju.

Slika 81: Forma za registraciju korisnika

Registrirajte se

anaanic@email.com

Lozinka mora sadržavati barem 6 znakova.

Ana

Anić

+385912345678

Ulica Petra Preradovića 12

Zagreb

10000

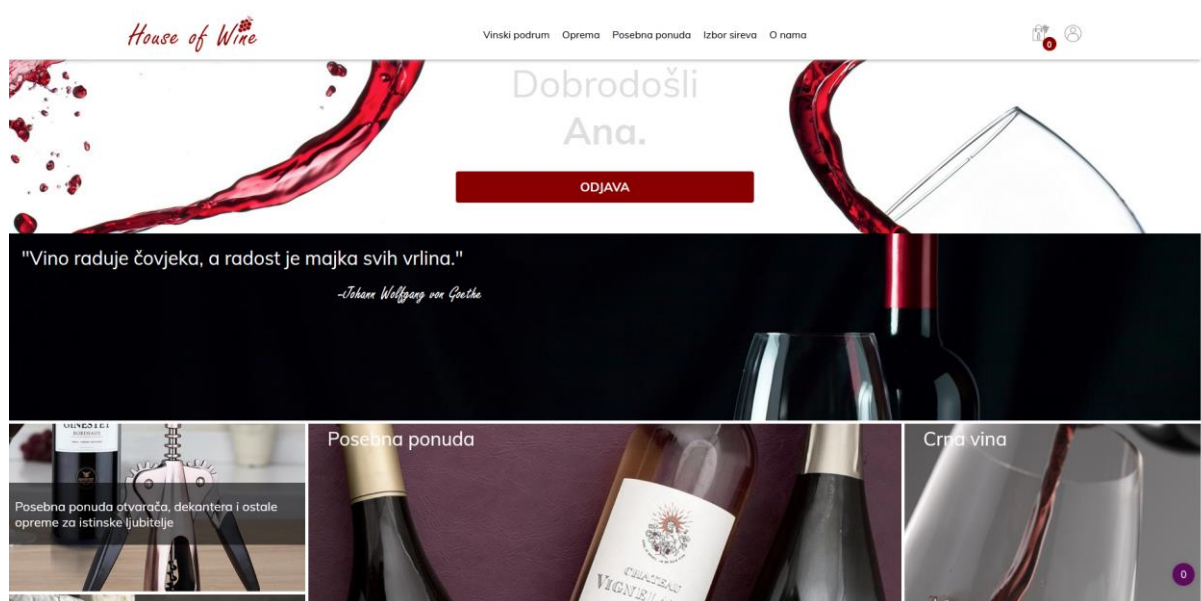
REGISTRACIJA

Već imate račun? [Prijavite se.](#)

Izvor: izradio autor

Korisnik upisuje podatke, ali lozinka ima samo 5 znakova, stoga program vraća poruku greške i korisnik ponovno upisuje lozinku. Bitno je napomenuti da se nakon greške podaci koji su upisani ne brišu kako korisnik ne bi morao ispočetka obavljati upis podataka. Nakon uspješne registracije program vraća korisnika na početnu stranicu, ovaj put u poruci dobrodošlice piše i ime korisnika, a pojavio se i gumb za odjavu. U bazu podataka su se spremili svi podaci koje je korisnik upisao.

Slika 82: Početna stranica kada je korisnik prijavljen



Izvor: izradio autor

Slika 83: Redak u tablici registriranog korisnika u bazi podataka

user_id	email	password	fname	lname	contact	adres	city	zip
277	anaanic@email.com	\$2y\$10\$6M2/1W7xNdxjv0UDDBoYOeebQx.LHDG2q08ixlGGb7...	Ana	Anić	+385912345678	Ulica Petra Preradovića 12	Zagreb	10000

Izvor: izradio autor

7.2.3. Kreiranje narudžbe

Sada korisnik odlazi na stranicu *Vinski podrum*, tj. *podrum.php* i traži od programa da sortira proizvode po cijeni od najmanje prema najvećoj, te stavlja u košaricu željene proizvode, a

ti proizvodi se odmah dodaju i u tablicu *cart_item* (slika 84), kao i narudžba, čiji se podaci registriraju u tablici *order* (slika 85).

Slika 84: Novi redak u tablici *order*

order_id	user_id_fk	time	order_code	order_user	order_adress	order_contact	delivery	payment	order_quantity	subtotal	completion
372	277	2021-05-30 15:50:48	#35034709								0

Izvor: izradio autor

Slika 85: Novi redci u tablici *cart_item*

cart_item_id	order_id_fk	ci_code	ci_name	ci_price	ci_quantity	total_price
557	372	0.41813073346217233	Iuris Rosé Biciklisti 0,75 l	32.99	1	32.99
558	372	0.8618292695487191	Korlat Cabernet Sauvignon vrhunsko vino 0,75 l	99.99	1	99.99

Izvor: izradio autor

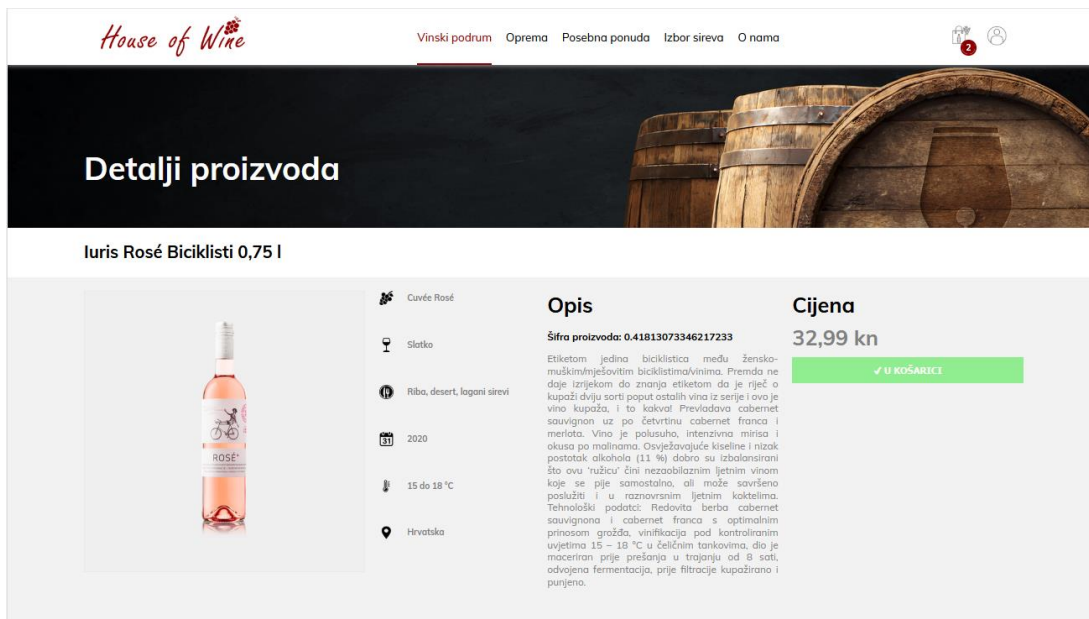
Slika 86: Prikaz proizvoda na sučelju

Izvor: izradio autor

Nakon dodavanja proizvoda u košaricu korisnik odlučuje još i pogledati detalje o proizvodu kojeg je dodao pa klikom miša na taj proizvod otvara novu stranicu, *product.php*.

7.2.4. Pregled detalja proizvoda

Slika 87: Prikaz stranice *product.php*



Izvor: izradio autor

Zatim, korisnik odlučuje da želi završiti kupnju i odlazi u košaricu (*cart.php*), klikom na ikonu košarice u desnom kutu *header*-a stranice.

7.2.5. Košarica



Slika 88: Košarica

House of Wine

Vinski podrum Oprema Posebna ponuda Izbor sireva O nama

Moja košarica

Pregled proizvoda

proizvod	cijena	količina	iznos
 Iuris Rosé Bicklisti 0,75 l UKLONI	32,99 kn	1	32,99 kn
 Korlat Cabernet Sauvignon vrhunsko vino 0,75 l UKLONI	99,99 kn	1	99,99 kn

Pregled košarice

Broj proizvoda 2
Ukupna cijena 132,98 kn

Košarica ukupno* 132,98 kn

[NA PLAĆANJE](#)

* Ukupna cijena košarice je iznos svih proizvoda u košarici (PDV uračunat u cijenu).

Izvor: izradio autor

Korisnik nakon toga mijenja količinu prvog proizvoda na 4, a drugi proizvod ipak odlučuje obrisati iz košarice. Te promjene se vide i u tablici `cart_item`.


Slika 89: Košarica nakon brisanja proizvoda i mijenjanja količine

House of Wine

Vinski podrum Oprema Posebna ponuda Izbor sireva O nama

Moja košarica

Pregled proizvoda

proizvod	cijena	količina	iznos
 Iuris Rosé Bicklisti 0,75 l UKLONI	32,99 kn	4	131,96 kn

Pregled košarice

Broj proizvoda 4
Ukupna cijena 131,96 kn

Košarica ukupno* 131,96 kn

[NA PLAĆANJE](#)

* Ukupna cijena košarice je iznos svih proizvoda u košarici (PDV uračunat u cijenu).

Izvor: izradio autor

Slika 90: Redak tablice *cart_item* nakon promjena

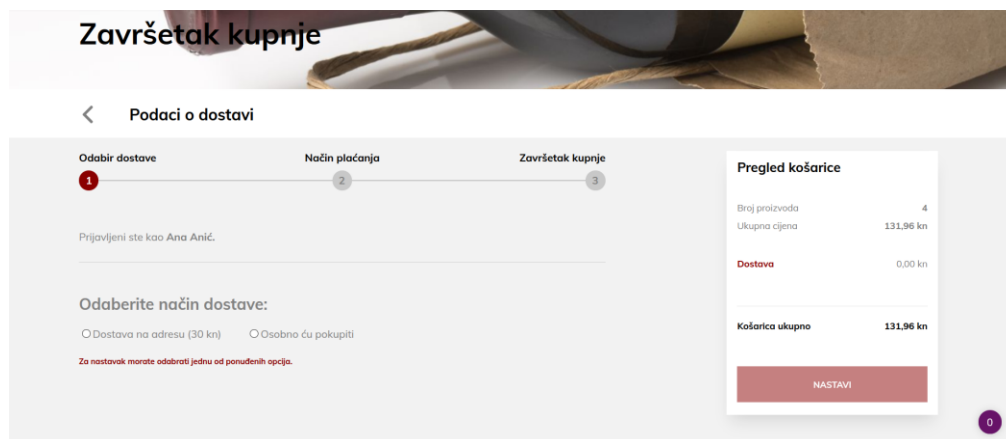
cart_item_id	order_id_fk	ci_code	ci_name	ci_price	ci_quantity	total_price
557	372	0.41813073346217233	Iuris Rosé Biciklisti 0,75 l	32.99	4	131.96

Izvor: izradio autor

Prvi proizvod je sada ažuriran u tablici *cart_item*, promijenjena mu je ukupna cijena, kao i količina, a drugi je obrisan iz tablice. Korisnik klikom na gumb *NA PLAĆANJE* nastavlja proces kupnje na stranici *checkout.php*.

7.2.6. Završetak kupnje

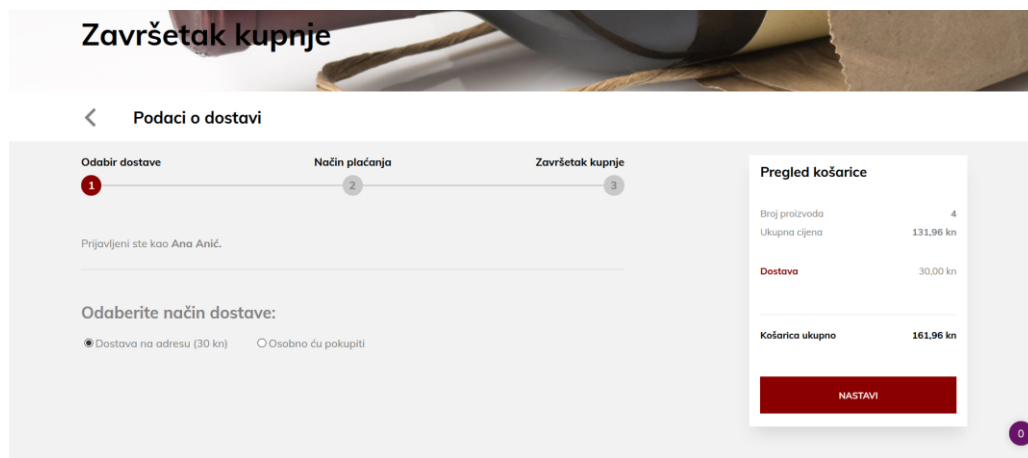
Slika 91: Sučelje datoteke *checkout.php* - bez odabira dostave



Izvor: izradio autor

Stranica *checkout.php* nas vodi kroz tri koraka, gdje korisnik prvo bira dostavu, nakon čega se ona primjenjuje na ukupnoj cijeni. Važno je napomenuti da se korisnik u bilo kojem koraku može vratiti korak nazad pomoću tipke na nazad, koja je prikazana ikonom strelice lijevo odmah uz naslov stranice.

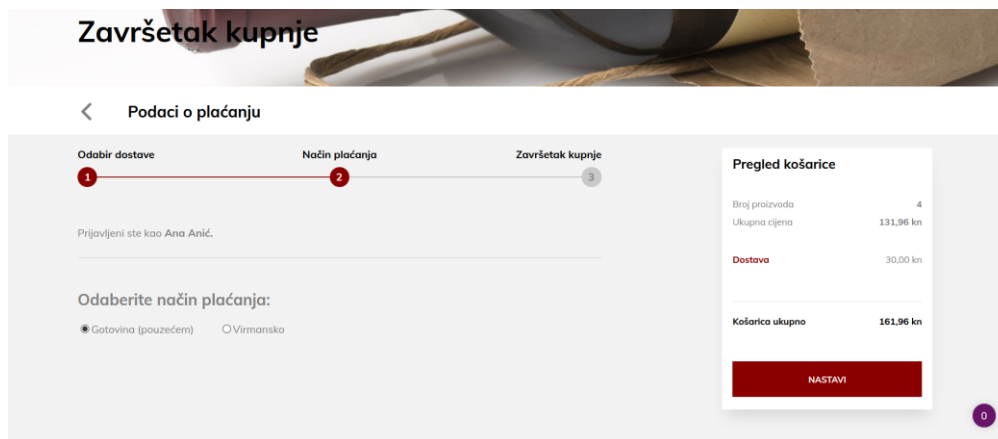
Slika 92: Sučelje datoteke checkout.php - dostava odabrana



Izvor: izradio autor

Nakon dostave, korisnik bira i način plaćanja:

Slika 93: Odabir plaćanja



Izvor: izradio autor

Nakon što je odabrao plaćanje, korisnik dalje upisuje podatke potrebne za dostavu (ako je korisnik prijavljen na stranicu, podaci se ispunjavaju automatski, a mogu se mijenjati po potrebi).

7.2.7. Potvrda kupnje

Slika 94: Stranica za upis podataka o kupcu i dostavi

Potvrda

Odabir dostave Način plaćanja Završetak kupnje


1 2 3

Prijavljeni ste kao Ana Anić.
Prije potvrde kupnje provjerite ispravnost podataka i sadržaj Vaše narudžbe.

* Obavezna polja

Podaci o kupcu	Podaci za dostavu
Ana	Ulica Petra Preradovića 12
Anić	Zagreb
anaanic@email.com	10000
+385912345678	

Sadržaj narudžbe

proizvod	cijena	količina	iznos
 Iuris Rosé Bliciklisti 0,75 l 0.41013073346217233	32,99 kn	4	131,96 kn

Pregled košarice

Broj proizvoda 4
Ukupna cijena 131,96 kn

Dostava 30,00 kn

Košarica ukupno 161,96 kn


POTVRDI KUPNJU

Izvor: izradio autor

Korisnik u ovom dijelu može vidjeti i sadržaj svoje narudžbe, nakon što je spreman za kupnju klikom miša na gumb **POTVRDI KUPNJU** završava proces kupnje, a program provjerava da li su uneseni podaci korektni na sličan način opisan već u dijelu o PHP algoritmu za registraciju.

Slika 95: Uspješno obavljena narudžba

Uspjeh! Vaša narudžba je zaprimljena.
Hvala na narudžbi i ukazanom povjerenju.



[Natrag na početnu stranicu](#)

Detalji Vaše narudžbe

Narudžba #35034709

Ime i prezime	Adresa	Kontakt	Način dostave	Način plaćanja	Količina proizvoda	Ukupna cijena	Datum i vrijeme
Ana Anić	Ulica Petra Preradovića 12, Zagreb, 10000	anaanic@email.com +385912345678	Dostava na adresu (30 kn)	Gotovina (pouzećem)	4	161,96 kn	2021-05-30 16:14:16

Sadržaj narudžbe

#	Naziv proizvoda	Šifra proizvoda	Cijena	Količina	Iznos
1	Iuris Rosé Bicklisti 0,75 l	0.41813073346217233	32,99 kn	4	131,96 kn

[NATRAG NA POČETNU STRANICU](#)

Vaš
House of Wine

Izvor: izradio autor

Nakon što je korisnik potvrdio narudžbu, detalji o narudžbi se ažuriraju u tablici *order*, a sesija košarice se briše i program je spreman za ponovnu kupnju. Detalji o kupnji se također korisniku prikazuju na ekranu uz poruku da je kupnja uspješno obavljena. Nakon toga kupac se može vratiti na početnu stranicu.

Slika 96: Promjene nastale u retku narudžbe nakon uspješne kupnje

order_id	user_id_fk	time	order_code	order_user	order_adress	order_contact	delivery	payment	order_quantity	subtotal	completion
372	277	2021-05-30 16:14:16	#35034709	Ana Anić	Ulica Petra Preradovića 12, Zagreb, 10000	anaanic@email.com +385912345678	Dostava na adresu (30 kn)	Gotovina (pouzećem)	4	161,96	1

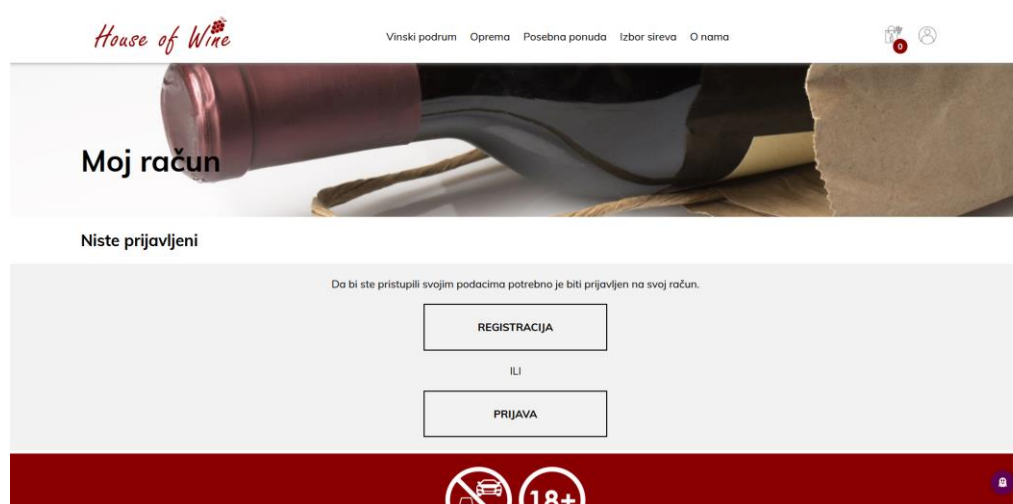
Izvor: izradio autor

7.3. Putanja korisnika kroz profil

7.3.1. Prijava na račun

Korisnik želi ući na svoj profil (*acc.php*) i pogledati svoje narudžbe. Ulaskom na profil jasno je da nije prijavljen i odabire opciju prijave na račun.

Slika 97: Profil kada korisnik nije prijavljen



Izvor: izradio autor

Slika 98: Forma za prijavu na stranicu

Prijavite se

E-mail adresa

Lozinka

PRIJAVA

Nemate račun kod nas? [Registrirajte se.](#)

Izvor: izradio autor

7.3.2. Profil

Korisnik se uspješno prijavljuje i otvara stranicu *acc.php* koja sadrži podatke o korisniku.

Slika 99: Prikaz profila korisnika

The screenshot shows the user profile page for 'House of Wine'. The user is identified as 'Korisnik: Šiško Menčetić'. The profile details are as follows:

Ime	Prezime
Šiško	Menčetić

E-mail adresa	Kontakt
siskomencetic@email.com	+385999999999

Adresa	Grad
Augusta Senec 1/B Rijeka	Rijeka

Poštanski broj
51000

There is a red button labeled 'IZMJENA PODATAKA' (Edit Data) next to the profile details.

Moje narudžbe

Narudžba #12791964

Ime i prezime	Adresa	Kontakt	Način dostave	Način plaćanja	Količina proizvoda	Ukupna cijena	Datum i vrijeme
Šiško Menčetić	Augusta Senec 1/B, Rijeka, 51000	siskomencetic@email.com +385999999999	Dostava na adresu	Gotovina (pouzećem)	6	669,98 kn	2021-05-06 20:39:28

Sadržaj narudžbe

#	Naziv proizvoda	Sifra proizvoda	Cijena	Količina	Iznos
1	Plavac mali Grgić 0,75 l	0.15522042769493574	219,99 kn	2	439,98 kn
2	Muškat žuti Podrum Strigova 0,75 l	0.40540353712197724	50,00 kn	4	200,00 kn

Narudžba #78989010

Ime i prezime	Adresa	Kontakt	Način dostave	Način plaćanja	Količina proizvoda	Ukupna cijena	Datum i vrijeme
Šiško Menčetić	Augusta Senec 1/B, Rijeka, 51000	siskomencetic@email.com +385999999999	Dostava na adresu	Gotovina (pouzećem)	4	909,96 kn	2021-05-06 20:39:28

Sadržaj narudžbe

#	Naziv proizvoda	Sifra proizvoda	Cijena	Količina	Iznos
1	Plavac mali Grgić 0,75 l	0.15522042769493574	219,99 kn	4	879,96 kn

Narudžba #58843485

Ime i prezime	Adresa	Kontakt	Način dostave	Način plaćanja	Količina proizvoda	Ukupna cijena	Datum i vrijeme
---------------	--------	---------	---------------	----------------	--------------------	---------------	-----------------

Izvor: izradio autor

Na stranici profila korisnik može listati kroz sve narudžbe koje je napravio, jer su spremljene u bazi podataka.

7.3.3. Izmjena osobnih podataka na profilu

Korisnik može jednostavno izmijeniti svoje podatke klikom miša na gumb *IZMJENA PODATAKA* i tada stranica dopušta akcije nad podacima pomoću JavaScript-a.

Slika 100: Profil je u načinu rada za izmjenu podataka

The screenshot shows the user profile page for 'Šiško Menčetić' in edit mode. The page header includes the logo 'House of Wine' and navigation links: 'Vinski podrum', 'Oprema', 'Posebna ponuda', 'Izbor sireva', and 'O nama'. The user's name 'Korisnik: Šiško Menčetić' is displayed. The main content area is a form titled 'IZMJENA PODATAKA' with two columns of input fields. The left column contains fields for 'Ime' (Šiško), 'E-mail adresa' (siskomencetic@email.com), 'Adresa' (Augusta Senec 1/B), and 'Poštanski broj' (51000). The right column contains fields for 'Prezime' (Menčetić), 'Kontakt' (+385999999999), and 'Grad' (Rijeka). Two buttons are visible on the right: a red 'DOKRŠI BEZ SPREMANJA' button and a green 'POTVRDI IZMJENE' button. Below the form, there is a link for 'Moje narudžbe'.

Izvor: izradio autor

Nakon što je promijenio podatak o adresi, korisnik klikom na gumb *POTVRDI IZMJENE* sprema izmjene i podaci u tablici *user* se ažuriraju. Ovaj dio kôda također provjerava ima li grešaka prilikom izmjene podataka i te podatke vraća na stare vrijednosti, a korisniku ispisuje o kakvoj se grešci radi. Sve slučajeve možemo promotriti kroz iduće tri slike.

Slika 101: Korisnik je uspješno proveo izmjenu podataka

The screenshot shows the user profile page for 'Šiško Menčetić' after a successful update. The page header is the same as in Slika 100. The main content area now displays a green message: 'Svi podaci su uspješno izmijenjeni.' Below this message is the same form as in Slika 100, but with updated values: 'Adresa' is 'Jelačićev trg 16' and 'Poštanski broj' is '51000'. The red 'DOKRŠI BEZ SPREMANJA' button is still present, but the green 'POTVRDI IZMJENE' button has been replaced by a red 'IZMJENA PODATAKA' button.

Izvor: izradio autor

Slika 102: Promjene korisnikovih podataka su napravljene na bazi podataka

user_id	email	password	fname	lname	contact	adress	city	zip
47	siskomencetic@email.com	\$2y\$10\$1uqp3h6T.HwauFluqbnLpOD4I4hWwYJ5K6Ta6foTEOcq...	Šiško	Menčetić	+385999999999	Jelačićev trg 16	Rijeka	51000

Izvor: izradio autor

Slika 103: Korisnik je napravio greške tijekom unosa

Korisnik: Šiško Menčetić

GREŠKE PRU UNOSU:
IME (prazno polje) E-MAIL: siskomencetic (neispravan e-mail)

Navedeni podaci su vraćeni na prethodne vrijednosti, a ostali su uspješno izmijenjeni.

Ime	Prezime
Šiško	Menčetić
E-mail adresa	Kontakt
siskomencetic@email.com	+385999999999
Adresa	Grad
Jelačićev trg 16	Rijeka
Poštanski broj	
51000	

IZMJENA PODATAKA

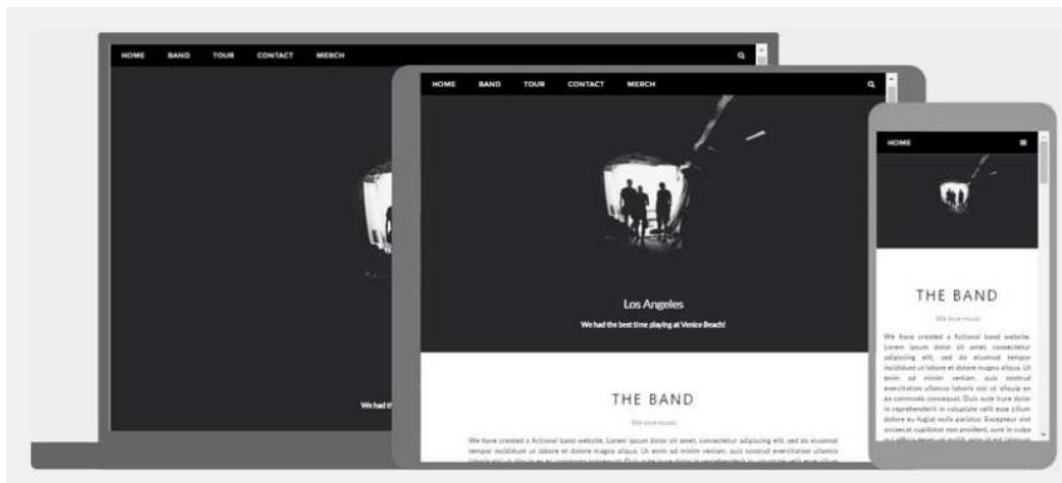
Izvor: izradio autor

8. Responzivni dizajn

Responzivni dizajn (*engl. responsive design*) odnosi na način slaganja prikaza web stranice, tako da je sadržaj vidljiv i koristan na svim uređajima i ekranima. Takav način dizajniranja web stranice je bitan ako web stranicu gledamo na recimo monitoru neke manje ili veće rezolucije, pa se tomu mora prilagoditi i sadržaj. Web stranicu možemo otvoriti i pomoću pametnih telefona ili tableta pa tako sadržaj koji je bez problema stao na veliki ekran, sada moramo prilagoditi manjem i užem ekranu, a da se sadržaj stranice znatno ne promijeni. Na slici broj 104 vidimo kako se izgled stranice mijenja na različitim uređajima. (HTML Responsive Web Design: https://www.w3schools.com/html/html_responsive.asp, 2021.)

Općenito, za dizajn različitih dijelova ove stranice korišteni su alati iz *Adobe* programskih paketa, a pod time se misli na ikonice i logotipe koji se pojavljuju na stranici.

Slika 104: Skica responzivnog dizajna



Izvor: https://www.w3schools.com/html/html_responsive.asp

8.1. HTML i CSS kôd za responzivan dizajn

Responzivnost na web stranicama postižemo raznim metodama. Npr., konkretno za ovaj projekt, u *header* datoteci je ubačen *meta* element HTML-a, koji daje instrukcije web pregledniku kako da kontrolira dimenzije stranice i koje operacije su dopuštene vezane uz skaliranje³² ekrana.

Slika 105: Prikaz viewport opcije u meta elementu

```
54 <meta name="viewport" content="width=device-width, user-scalable=no">
```

Izvor: izradio autor

Unutar ovog *viewport meta* elementa pregledniku je definirano na koji način da skalira stranicu, ovisno o veličini ekrana na kojoj je prikazana. Atributu *content* dodana je vrijednost, koja postavlja širinu stranice na onu koliko uređaj na kojem se gleda može maksimalno podržati, kao i da korisnik ne može koristiti opciju povećavanja ili smanjivanja ekrana (*engl. zoom*) na uređaju, a nije niti potrebno, jer je sav sadržaj prilagođen svim uređajima.

U CSS kôdu, responzivnost postižemo dodavanjem posebnog *@media* pravila. Ono služi kako bi se primijenili različiti stilovi, ovisno o tipu uređaja na kojem se prikazuje, tj. ovisno o veličini ekrana. (CSS @media Rule: https://www.w3schools.com/cssref/css3_pr_mediaquery.asp, 2021.)

Slika 106: Sintaksa @media pravila

```
2346  
2347 @media (max-width: 1000px) {  
2348
```

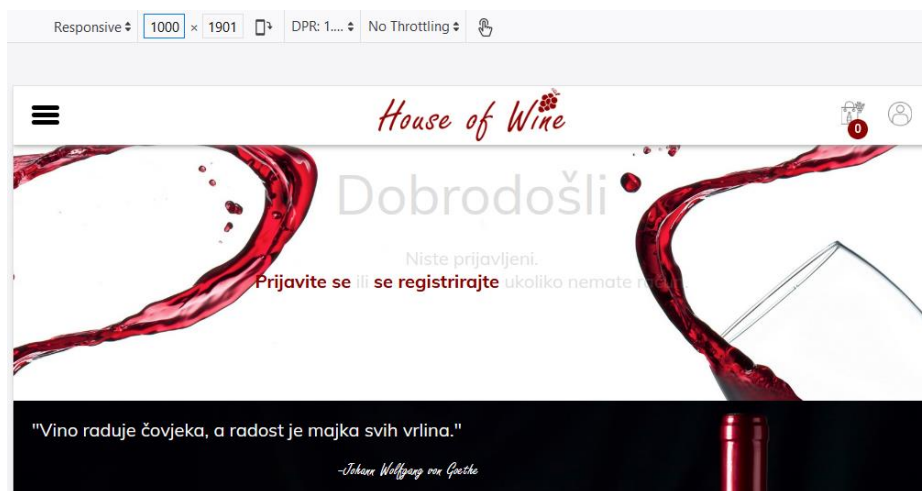
Izvor: izradio autor

Na slici broj 106 vidimo kako izgleda sintaksa *@media* pravila. U zagradama specificiramo atribut *max-width* kojim definiramo kada će program koristiti pravila koja smo

³² Skaliranje se odnosi na povećavanje i smanjivanje prikaza na ekranu pomoću opcija na uređaju, npr. gestama prstiju na pametnom telefonu.

upisali u blok naredbi unutar vitičastih zagrada `{}`. U slučaju na ovoj slici, kada je ekran manji od 1000px, aktivira se taj blok naredbi. Kako to izgleda na stranici vidimo na slici ispod.

Slika 107: Primjer stranice na ekranu od 1000px



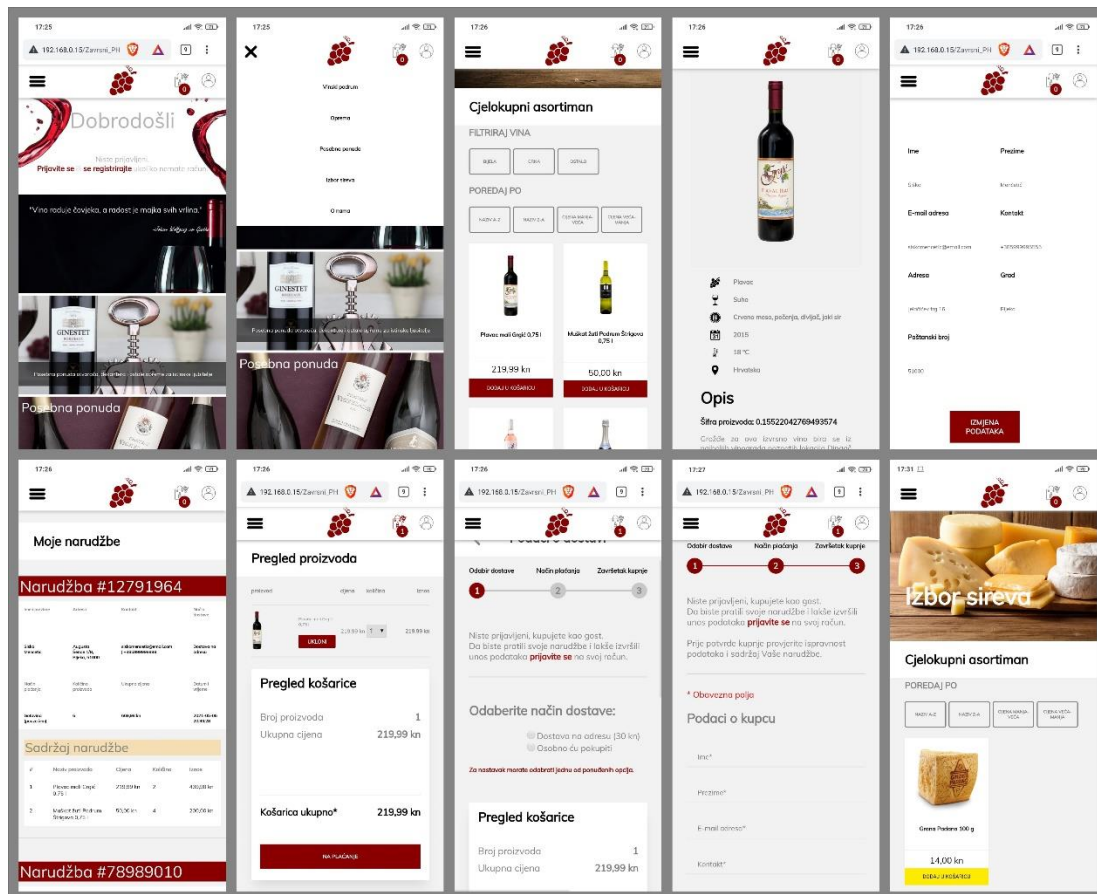
Izvor: izradio autor

8.2. Primjer responzivnosti na projektu

Ovdje ćemo samo pokazati nekoliko primjera kako se ova stranica prikazuje na mobilnom uređaju, kako bismo bolje razumjeli koje nam prednosti nudi responzivni dizajn. Slike ekrana su napravljene na uređaju s Android 10 operacijskim sustavom na web pregledniku Brave koji koristi Chromium³³ bazu. Na slici 108 prikazani su neki dijelovi i stranice iz ovog projekta.

³³ Chromium je vrsta kôda zaslužna za funkcionalnost web preglednika, napravljen je od strane Google-a, a koriste ga Google Chrome, Microsoft Edge, Opera i drugi.

Slika 108: Prikaz nekih dijelova web trgovine "House of Wine" na mobilnom uređaju



Izvor: izradio autor

9. Zaključak

Ovaj završni rad, izrađen je u periodu od par mjeseci, a u to ne ulazi samo pisanje kôda, nego i proučavanje literature i učenje o tehnologijama koje su se koristile. Najizazovniji dio ovog projekta bio je usklađivanje serverske i klijentske strane. Pod time mislim da je bilo potrebno često izmjenjivati već postojeće kôdove i prilagođavati ih novim potrebama. Na serverskoj strani bilo je izazovno i formiranje same baze podataka te izrada kvalitetnog korisničkog sučelja za ažuriranje tih podataka kroz to sučelje web stranice. U području dizajna, također je bilo izazovnih situacija, pogotovo pri izradi responzivnog dizajna za različite uređaje.

Kroz ovaj projekt sam stekao novo znanje i nadopunio staro, naučeno kroz kolegije na fakultetu. Možda najvažnije od svega je shvaćanje koliko je potrebno znanja i rada za izradu jedne cjelovite web trgovine. Ovdje nije u pitanju samo pisanje kôda, nego i testiranja kojima dolazimo do zaključaka gdje su greške, što može biti još bolje i zašto se nešto u programu događa ili ne.

U ovom projektu smo upoznali tehnologije i programske jezike koji su gotovo uvijek prisutni u razvoju web stranica. Od samo izgleda stranice, tj. klijentske strane pa sve do manipulacije podacima koji se nalaze na serverskoj strani. Nakon ovoga, zaključujem da je središte svog zbivanja, općenito u računalnim programima, algoritam. Oni su ti koji „vode“ program i funkcionalno koriste podatke kako bi olakšali brojne procese, poput, u ovom slučaju, obične kupnje.

Smatram, da bi se u budućnosti ova web trgovina mogla još nadograđivati, raznoraznim funkcijama, poput tražilice, atraktivnijeg sučelja i većeg broja opcija.

10. Popis korištenih kratica i anagrama

API	(Application Programming Interface) aplikacijsko-programsko sučelje za web
CSS	(Cascading Style Sheet) jezik za dodavanje stilova unutar web stranice
DBMS	(Database Managment System) sustav za upravljanje bazama podataka
DD	(Data Dictionary) riječnik podataka
DDL	(Data Definition Language) jezik za stvaranje tipova podataka
DML	(Data Manipulation Language) jezik za manipuliranje podacima
DOM	(Document Object Model) model koji definira HTML elemente kao objekte
DQL	(Data Query Language) podatkovni upitni jezik
HTML	(Hyper Text Markup Language) univerzalni jezik za stvaranje strukture web stranice
HTTP	(Hyper Text Transfer Protocol) protokol za komunikaciju računala klijenta i servera
IP	(Internet Protocol) skup pravila koji određuju kako se paketi šalju kroz mrežu
JPG	(Joint Photographic Group) format za spremanje slike
JSON	(JavaScript Object Notation) otvoreni standard za čitljivu razmjenu podataka
PHP	(PHP: Hypertext Preprocessor) programski jezik za razvoj web stranica i aplikacija
RDBMS	(Relational Database Managment System) sustav za upravljanje relacijskim bazama
SQL	(Structured Query Language) strukturirani upitni jezik za obradu podataka
SUPB	(Sustav za upravljanje bazama podataka) isto što i DBMS
URL	(Uniform Resource Locator) Adresa specifične web stranice ili datoteke na Internetu
WWW	(World Wide Web) dio Interneta koji se sastoji od web stranica
XHR	(XMLHttpRequest) JavaScript objekt za prijenos datoteka
XML	(Extensible Markup Language) jezik sličan HTML-u, ali bez predefiniраниh oznaka

11. Literatura

1. A Brief History of HTML,
https://www.washington.edu/accesscomputing/webd2/student/unit1/module3/html_history.html
(2021.)
2. Backend Definition, <https://techterms.com/definition/backend> (2021.)
3. CSS Attribute Selectors, https://www.w3schools.com/Css/css_attribute_selectors.asp (2021.)
4. CSS Combinators, https://www.w3schools.com/Css/css_combinators.asp (2021.)
5. CSS Introduction, https://www.w3schools.com/Css/css_intro.asp (2021.)
6. CSS Selectors, https://www.w3schools.com/Css/css_selectors.asp (2021.)
7. CSS Syntax, https://www.w3schools.com/Css/css_syntax.asp (2021.)
8. CSS @media Rule, https://www.w3schools.com/cssref/css3_pr_mediaquery.asp (2021.)
9. Framework Definition, <https://techterms.com/definition/framework> (2021.)
10. Frontend Definition, <https://techterms.com/definition/frontend> (2021.)
11. Hogan, B., HTML5 and CSS3, The Pragmatic Bookshelf, Dallas, 2011.
12. HTML Introduction, https://www.w3schools.com/html/html_intro.asp (2021.)
13. HTML Responsive Web Design, https://www.w3schools.com/cssref/css3_pr_mediaquery.asp
(2021.)
14. HTML Styles – CSS, https://www.w3schools.com/html/html_css.asp (2021.)
15. HTML <main> Tag, https://www.w3schools.com/tags/tag_main.asp (2021.)
16. HTTP Request Methods, https://www.w3schools.com/whatis/whatis_http.asp (2021.)
17. https://e-u.hr/dok/udzbenik/31_210.pdf (2021.)
18. Introducing JSON, <https://www.json.org/json-en.html> (2021.)
19. JavaScript Functions, https://www.w3schools.com/js/js_functions.asp (2021.)
20. JavaScript – HTML DOM Methods,
https://www.w3schools.com/js/js_html_dom_methods.asp (2021.)
21. Kaluža, M., Sustavi baza podataka, Veleučilište u Rijeci, Rijeka, 2008.
22. Lockhart, J., Modern PHP, O'Reilly Media, Sebastopol, 2015.
23. onclick Event, https://www.w3schools.com/jsref/event_onclick.asp (2021.)
24. PHP Cookies, https://www.w3schools.com/php/php_cookies.asp (2021.)

25. PHP Form Handling, https://www.w3schools.com/PHP/php_forms.asp (2021.)
26. PHP Sessions, https://www.w3schools.com/php/php_sessions.asp (2021.)
27. PHP Superglobal - \$_GET, https://www.w3schools.com/PHP/php_superglobals_get.asp (2021.)
28. PHP Superglobal - \$_POST, https://www.w3schools.com/Php/php_superglobals_post.asp (2021.)
29. PHP Superglobal - \$_SERVER, https://www.w3schools.com/Php/php_superglobals_server.asp (2021.)
30. Pojam algoritma, <https://mooc.carnet.hr/mod/book/tool/print/index.php?id=26661> (2021.)
31. session_start, <https://www.php.net/manual/en/function.session-start.php> (2021.)
32. SQL NULL Values, https://www.w3schools.com/sql/sql_null_values.asp (2021.)
33. Text Editor Definition, <https://techterms.com/definition/texteditor> (2021.)
34. URL, <https://techterms.com/definition/url> (2021.)
35. What is HTTP?, https://www.w3schools.com/whatis/whatis_http.asp (2021.)
36. What is JavaScript?, https://www.w3schools.com/whatis/whatis_js.asp (2021.)
37. What is PHP?, <https://www.php.net/manual/en/intro-whatism.php> (2021.)
38. What is the HTML DOM?, https://www.w3schools.com/whatis/whatis_html5dom.asp (2021.)
39. WWW Definition, <https://techterms.com/definition/www> (2021.)
40. \$_SESSION, <https://www.php.net/manual/en/reserved.variables.session.php> (2021.)

12. Popis slika

Slika 1: Sučelje programa Microsoft Visual Studio Code.....	2
Slika 2: Korisničko sučelje programa XAMPP	3
Slika 3: Web preglednik Microsoft Edge	4
Slika 4: Shematski prikaz slanja podataka kroz HTTP protokol.....	5
Slika 5: Prikaz ciklusa potražnje i vraćanja HTTP zahtjeva	6
Slika 6: Prosljeđivanje podataka kroz URL preko GET metode.....	7
Slika 7: Primjer kôda gdje se koristi \$_GET varijabla.....	8
Slika 8: Primjer kôda gdje se koriste \$_POST varijable	8
Slika 9: DMBS shema	10
Slika 10: Relacije među tablicama baze podatak "houseofwine"	11
Slika 11: Prikaz tablice "user" iz baze podataka ovog projekta	12
Slika 12: Primjer primarnog i vanjskog ključa u tablici.....	13
Slika 13: Veze u tablici.....	14
Slika 14: Primjer SQL kôda.....	15
Slika 15: SQL kôd u programu phpMyAdmin	16
Slika 16: Primjer PHP kôda unutar HTML elemenata.....	17
Slika 17: Super-globalna varijabla \$_SERVER.....	18
Slika 18: Prosljeđivanje podataka kroz URL	19
Slika 19: Prikaz HTML form elementa sa metodom POST.....	20
Slika 20: Poziv funkcije session_start()	20
Slika 21: Inicijalizacija super-globalnih varijabla \$_SESSION.....	21
Slika 22: Poziv funkcija session_destory() i unset()	21
Slika 23: Struktura projekta u sučelju Visual Studio Code-a	22
Slika 24: Inicijalizacija funkcije cartItems()	23
Slika 25: Poziv funkcije cartItems()	23
Slika 26: Ubacivanje datoteka header.php i functions.php	24
Slika 27: Rezultat funkcije cartItems().....	24
Slika 28: Spajanje na bazu podataka houseofwine.....	25

Slika 29: Ispis SQL greške	26
Slika 30: HTML kôd	27
Slika 31: Primjer HTML kôda na projektu.....	28
Slika 32: Struktura web stranice	29
Slika 33: HTML DOM model	30
Slika 34: Header element na sučelju web stranice	31
Slika 35: Navigation element na sučelju stranice.....	32
Slika 36: Main element na sučelju stranice	33
Slika 37: Footer element na sučelju stranice	34
Slika 38: Prikaz stranice bez CSS-a i sa CSS-om	35
Slika 39: Primjer CSS funkcije iz projekta.....	36
Slika 40: Primjeri CSS selektora	37
Slika 41: Kombinatorni CSS selektor.....	38
Slika 42: Pseudo-klasni selektor.....	38
Slika 43: Rezultat selektora sa slike 42	39
Slika 44: Pseudo-element	39
Slika 45: Prikaz elementa ::after na primjeru projekta	40
Slika 46: Atributni selektor.....	40
Slika 47: Inline dodavanje CSS-a.....	41
Slika 48: Internal dodavanje CSS-a.....	41
Slika 49: external dodavanje CSS-a	42
Slika 50: Primjer JavaScript kôda	43
Slika 51: Varijable tipa int u JavaScript jeziku	43
Slika 52: JavaScript varijabla tipa string	44
Slika 53: Objekt i polje u JavaScript-u.....	44
Slika 54: Shematski prikaz pisanja JavaScript objekta i polja	45
Slika 55: Postavljanje kolačića u PHP jeziku.....	46
Slika 56: json_decode funkcija.....	46
Slika 57: Pristup elementima preko id atributa	47
Slika 58: HTML elementi s id atributima.....	47

Slika 59: Pristup HTML dokumentima preko klase.....	48
Slika 60: JavaScript funkcija enableEditing()	49
Slika 61: Poziv funkcije pomoću onclick eventa	49
Slika 62: Rezultat funkcije enableEditing()	50
Slika 63: Algoritam koji se izvršava pomoću for petlje	52
Slika 64: Algoritam za registraciju korisnika.....	52
Slika 65: Algoritam za provjeru da li korisnik već postoji.....	53
Slika 66: Algoritam za unos u bazu podataka i dodatne provjere	54
Slika 67: Poruke greški na korisničkom sučelju.....	55
Slika 68: Provjera podataka unutar HTML datoteke.....	56
Slika 69: Algoritam za spremanje proizvoda u bazu podataka	56
Slika 70: Funkcija addEventListener.....	57
Slika 71: Rezultat addEventListener funkcije	58
Slika 72: HTML kôd	59
Slika 73: CSS kôd.....	60
Slika 74: Prikaz košarice (cart.php).....	61
Slika 75: Raspored početne stranice.....	62
Slika 76: Struktura i poredak datoteka unutar projekta	63
Slika 77: Uključivanje header datoteke	64
Slika 78: Uključivanje footer datoteke	64
Slika 79: Naslovna stranica web stranice "House of Wine".....	65
Slika 80: Stranica za registraciju	66
Slika 81: Forma za registraciju korisnika	66
Slika 82: Početna stranica kada je korisnik prijavljen.....	67
Slika 83: Redak u tablici registriranog korisnika u bazi podataka	67
Slika 84: Novi redak u tablici order	68
Slika 85: Novi redci u tablici cart_item.....	68
Slika 86: Prikaz proizvoda na sučelju.....	68
Slika 87: Prikaz stranice product.php.....	69
Slika 88: Košarica.....	70

Slika 89: Košarica nakon brisanja proizvoda i mijenjanja količine	70
Slika 90: Redak tablice cart_item nakon promjena	71
Slika 91: Sučelje datoteke checkout.php - bez odabira dostave	71
Slika 92: Sučelje datoteke checkout.php - dostava odabrana	72
Slika 93: Odabir plaćanja	72
Slika 94: Stranica za upis podataka o kupcu i dostavi.....	73
Slika 95: Uspješno obavljena narudžba.....	74
Slika 96: Promjene nastale u retku narudžbe nakon uspješne kupnje	74
Slika 97: Profil kada korisnik nije prijavljen.....	75
Slika 98: Forma za prijavu na stranicu	75
Slika 99: Prikaz profila korisnika	76
Slika 100: Profil je u načinu rada za izmjenu podataka	77
Slika 101: Korisnik je uspješno proveo izmjenu podataka	77
Slika 102: Promjene korisnikovih podataka su napravljene na bazi podataka.....	78
Slika 103: Korisnik je napravio greške tijekom unosa	78
Slika 104: Skica responzivnog dizajna.....	79
Slika 105: Prikaz viewport opcije u meta elementu	80
Slika 106: Sintaksa @media pravila.....	80
Slika 107: Primjer stranice na ekranu od 1000px.....	81
Slika 108: Prikaz nekih dijelova web trgovine "House of Wine" na mobilnom uređaju	82