Razvoj sustava za internetsku prodaju sportske opreme

Šalamon, David

Undergraduate thesis / Završni rad

2022

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Polytechnic of Međimurje in Čakovec / Međimursko veleučilište u Čakovcu**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:110:239511

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-05-20**

Repository / Repozitorij:

Polytechnic of Međimurje in Čakovec Repository - Polytechnic of Međimurje Undergraduate and Graduate Theses Repository

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
STRUČNI STUDIJ RAČUNARSTVO

David Šalamon

RAZVOJ SUSTAVA ZA INTERNETSKU PRODAJU SPORTSKE OPREME

ZAVRŠNI RAD

Čakovec 2022.

MEĐIMURSKO VELEUČILIŠTE U ČAKOVCU
STRUČNI STUDIJ RAČUNARSTVO

David Šalamon

RAZVOJ SUSTAVA ZA INTERNETSKU PRODAJU SPORTSKE OPREME

DEVELOPMENT OF E-COMMERCE FOR SPORTS EQUIPMENT

ZAVRŠNI RAD

Mentorica:
dr. sc. Sanja Brekalo

Čakovec 2022.

ZA REFERADU

ZAHVALA

Ovim putem želim zahvaliti svojoj mentorici dr. sc. Sanji Brekalo na pomoći i savjetima tijekom izrade završnoga rada. Zahvaljujem svojoj obitelji, prijateljima i svima ostalima koji su mi pružili podršku tijekom cjelokupnog studija.

David Šalamon

SAŽETAK

U radu opisana je izrada sustava za internetsku prodaju sportske opreme korištenjem MVC softverske arhitekture. Opisane su tehnologije koje se izvršavaju na mrežnom pregledniku i mrežnom poslužitelju. Glavna tehnologija za izradu ovog sustava programski je jezik PHP. Uz programski jezik PHP koristi se *Laravel framework* koji omogućava brz i efikasan razvoj sustava i bolju preglednost koda zahvaljujući MVC arhitekturi samog *frameworka*. Tehnologije kao što je HTML koriste se za strukturiranje podataka na mrežnom pregledniku, a CSS i *bootstrap* su za uljepšavanje i pozicioniranje strukturiranih podataka. *JavaScript* kao programski jezik i *jQuery* kao biblioteka *JavaScripta* koriste se za dodavanje raznih funkcionalnosti u samu aplikaciju. Svaki sustav za internetsku prodaju ima mogućnost plaćanja preko interneta. U izrađenoj aplikaciji, nakon što korisnik odabere proizvode, ima mogućnost plaćanja *Stripe* sustavom, jednim od najsigurnijih i najpopularnijih sustava za plaćanje preko interneta. U ovom radu koristi se *Mongo* baza podataka koja je nerelacijskog tipa. Sve slike iz aplikacije spremaju se na Amazonov mrežni servis S3. Slanje mailova vrši se preko *mailgun* davatelja usluga e-pošte. Razvojno okruženje koje se koristilo za izradu sustava je *PHPStorm* koji koristi alate za testiranje i otkrivanje grešaka, omogućava ekstenzije za bolju preglednost koda i ostale korisne ekstenzije koje pomažu i ubrzavaju razvoj sustava. Sustav koji je testiran i spreman za produkciju sprema se na *Cloud Hosting* korištenjem *DigitalOceana*.

Sustav za internetsku prodaju sportske opreme omogućava laku preglednost proizvoda. Korisnik može staviti proizvode u košaricu i platiti preko *Stripe* sustava.

Aplikacija je sustav za mrežno upravljanje sadržajem te podržava dvije razine upravljanja s različitim pravima, a to su korisnik i administrator. Korisnik može pregledavati, stavljati u košaricu i kupovati proizvode. Administrator dodaje, uređuje i briše proizvode i cijeli sadržaj mrežne stranice te upravlja narudžbama.

Ključne riječi: *Html, CSS, PHP, Laravel, JavaScript, AWS, DigitalOcean*

SADRŽAJ

1. UVOD 7
2. KORIŠTENE TEHNOLOGIJE 8
2.1. HTML 8
2.2. CSS 9
2.3. PHP 9
2.4. JavaScript i jQuery 11
2.5. MongoDB 12
2.6. Amazon Web Services S3 13
3. RAZVOJNO OKRUŽENJE 13
3.1. PHPStorm 13
3.2. Composer 13
3.3. Stripe 14
3.4. Mailgun 14
4. LARAVEL PROGRAMSKI OKVIR 15
4.1. Instalacija 15
4.2. MVC arhitektura 16
4.3. PHP Artisan 17
4.4. Analiza kostura aplikacije 18
4.5. Eloquent ORM 20
5. IZRADA APLIKACIJE 21
5.1. Konfiguracija 21
5.2. Uloge 22
5.2.1. Korisnik/gost 22
5.2.2. Administrator 22
5.3. Internetska trgovina 23
5.3.1. Početna stranica 23
5.3.2. Detalji proizvoda 24
5.3.3. Filtriranje proizvoda 25
5.3.4. Košarica 26
5.3.5. Plaćanje 27
5.3.6. Uspješna narudžba 28
5.4. Administracijsko sučelje 29
5.4.1. Brendovi 29

5.4.2. Kategorije 30
5.4.3. Proizvodi 31
5.4.4. Varijacije 32
5.4.5. Opcije varijacija 32
5.4.6. Vrijednosti opcija varijacija 32
5.4.7. Atributi 32
5.4.8. Vrijednosti atributa 33
5.4.9. Narudžbe 33
5.5. Postavljanje aplikacije na Cloud servis DigitalOceana 34
6. ZAKLJUČAK 35
7. LITERATURA 36
Popis slika/fotografija 37
Popis tablica 38
Popis kôdova 38

1. UVOD

Razvojem interneta raste broj tehnologija i programskih jezika. Danas se razvijaju nove tehnologije kojima želimo unaprijediti i ubrzati razvojni proces. Svaka mrežna aplikacija izrađena je u nekoliko odabranih tehnologija i programskih jezika. HTML se koristi za strukturu podataka, CSS za uređivanje i pozicioniranje tih podataka dok PHP i JavaScript daju dinamiku stranici.

Kako je aplikacija robusnija i teža za održavanje, koriste se i druge tehnologije koje mogu pomoći njezinu bržem razvoju. Zato se uz osnovne tehnologije dodaje *Laravel* programski okvir koji ubrzava razvoj i daje bolju preglednost koda svojom MVC arhitekturom.

2. KORIŠTENE TEHNOLOGIJE

U izradi mrežne aplikacije koriste se dvije vrste tehnologija: one koje se izvršavaju u mrežnom pregledniku i one koje se izvršavaju na serveru. Obje skupine imaju svoje prednosti kao i ograničenja te se koriste zajedno kako bi mrežna aplikacija izvršavala svoju zadaću.

2.1. HTML

HTML (engl. *HyperText Markup Language)* je prezentacijski jezik kojim se definira struktura mrežne stranice, a zamišljen je tako da stranica izgleda isto na svakom mrežnom pregledniku bez obzira na operacijski sustav. HTML elementi prikazuju se pomoću HTML oznaka (engl. *tags*) koje mrežni preglednik izvršava kod učitavanja stranice iz datoteka s ekstenzijom ".html" ili ".htm". Većina elemenata ima početnu i završnu oznaku kojima je jedina razlika "/" prije imena završne oznake, a jedan od njih je button. Kôd 1. Postoje elementi koji nemaju završnu oznaku, a neki od njih su: img, link, br i hr. Element može sadržavati atribute koji mrežnom pregledniku daju dodatne informacije o samom elementu. Atributi se zadaju pomoću ključ-vrijednosti (engl. *key-value*) parova kojima vrijednost proizvoljno najčešće zadaje korisnik. Neki od mogućih atributa su: *id*, *class*, *colspan*, *style* i *type*. Više HTML stranica međusobno je povezano hipervezama (engl. *hyperlink*) koje se definiraju „a" oznakom dok su odlomci odvojeni paragrafom, tj. „p" oznakom.

```
<button type="submit" name="submit" class="btn btn-primary">DODAJ</button>
```

Kôd 1. Gumb u formi s atributom klase

Izvor: Autor

Prvo službeno izdanje HTML objavljeno je 1993. godine sa svega dvadesetak oznaka te se u toj fazi na internetu nije mogla prikazati ni slika. Dvije godine kasnije objavljena je specifikacija za HTML 2 koja je temeljena na najpopularnijim HTML elementima koje su samovoljno implementirali kreatori mrežnih preglednika. S HTML-om 3 došao je prevelik broj zahtjeva i novih oznaka te je zbog sporosti ipak odbačen. Objavom HTML-a 4 1998. godine dolazi do velike promjene kako bi se smanjio broj elemenata za uređivanje sadržaja te se uvodi novi jezik koji će preuzeti tu ulogu. Zadnja verzija, HTML 5, objavljena je 22. siječnja 2008. i zamišljena je kao konačna verzija HTML-a koja će se razvijati dalje prema zahtjevima. U njoj je dodan velik broj novih elemenata, posebno onih vezanih za multimediju. S HTML-om 5 većina atributa koji se odnose na stil elementa smatra se zastarjelima. Zadnja podverzija, HTML 5.3., objavljena je 28. siječnja 2021. [1]

2.2. CSS

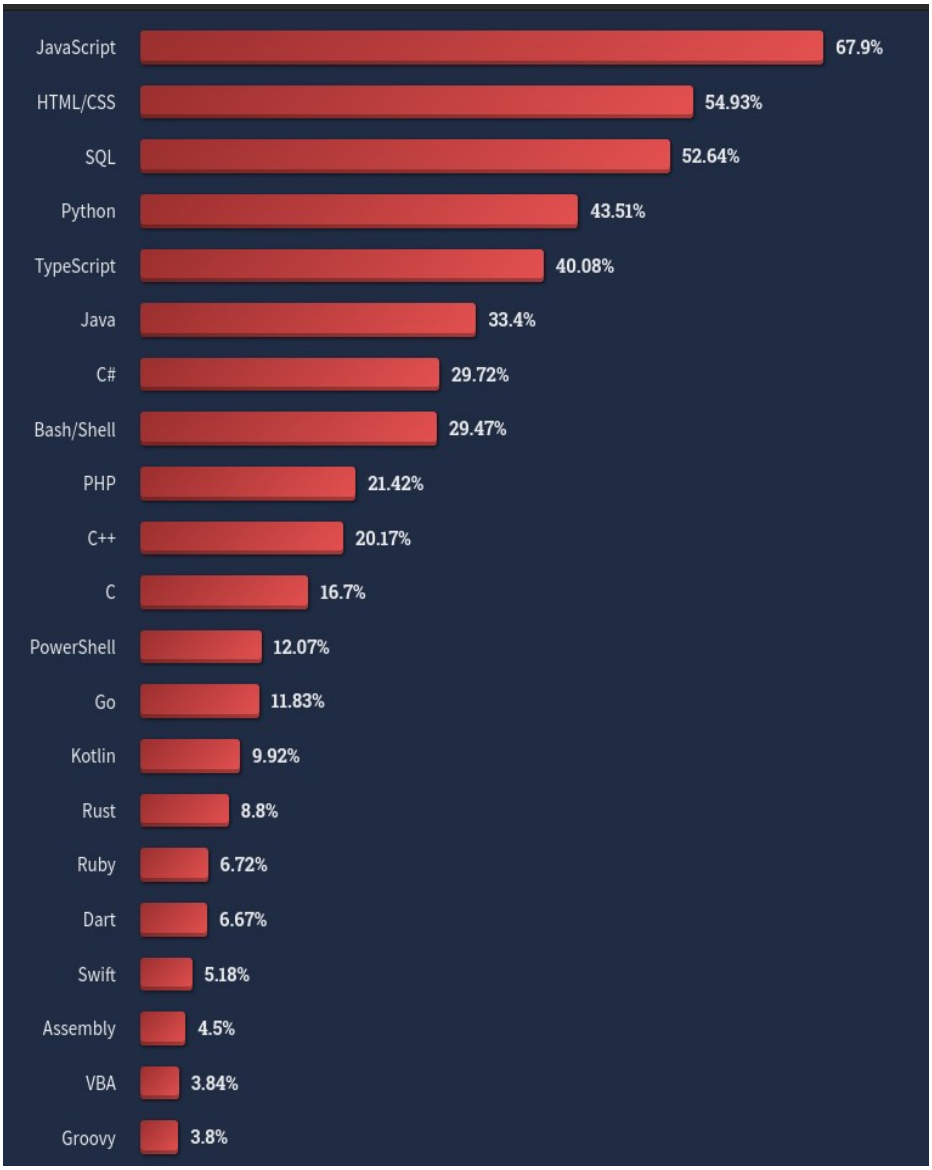
CSS (engl. *Cascading Style Sheets*) je jezik pomoću kojeg se mrežna stranica može uređivati. Pomoću CSS-a opisuje se kako bi elementi u HTML dokumentu trebali biti stilizirani. Njegova glavna karakteristika odvajanje je sadržaja stranice od dizajna raznih elemenata u HTML dokumentu te tako on ima glavnu ulogu u izradi kvalitetnoga korisničkog sučelja. [2]

2.3. PHP

PHP (engl. PHP: *Hypertext Processor*) osmislio je Rasmus Lerdorf 1994. godine. PHP je skriptni *open-source*, jezik koji se izvršava na poslužitelju. Nastao je iz potrebe za dinamičkim mrežnim stranicama, što se pomoću HTML-a nije moglo izraditi. Početne verzije bile su 1.0 i 2.0 te je uz eksponencijalni rast mrežnih stranica izrađena verzija 3.0 s podrškom za *Apache* poslužitelj, na kojem je u to vrijeme bilo implementirano 10 % mrežnih stranica.

Verzija 4.0 napravljena je na potpuno novoj strukturi, tzv. *Zend Engine*. *Zend Engine* služi kao kompajler, čime je u potpunosti poboljšana pouzdanost i struktura PHP-a. Glavno su svojstvo u novoj verziji klase, čime PHP postaje objektno orijentiran programski jezik, što je dodatno poboljšano u verziji 5.0 kojom su dodane razne funkcije za iznimke te ekstenzije za manipulaciju podacima kao što su XML (engl. *Extended Markup Language*) dokumenti. Verzija 6.0 nije

izrađena, već je dodana u kasnije verzije 5.x. Današnja stabilna verzija PHP-a je 8.1.7., a nova ažuriranja učestalo dolaze. PHP je jedan od najpopularnijih programskih jezika. Vrijedi istaknuti da je oko 80 % mrežnih stranica danas izrađeno u PHP-u. Na Slici 1. prikazani su podatci traženosti tehnologija prema korisnicima mrežne stranice *Stack Overflow*. [3, 4]



Slika 1. Prikaz najtraženijih tehnologija po ocjenama korisnika *Stack Owerflowa*

Izvor: https://survey.stackoverflow.co/2022/#most-popular-technologies-language-prof (17.6.2022.)

2.4. *JavaScript* i *jQuery*

Jedan od najpopularnijih programskih jezika za razvoj mrežnih stranica današnjice, koji je razvio Brendan Eich 1995. godine, *JavaScript*, kao i PHP, skriptni je jezik napravljen po uzoru na Javu. Razlika između *JavaScripta* i PHP-a je u tome da *JavaScript* nije *server-side*, već *client-side* jezik, što je velika prednost jer nema potrebe za učestalom komunikacijom sa serverom. Ako želimo da *Javascript* komunicira sa serverom, možemo koristiti node.js okruženje. Klikom na neki element stranica se ne osvježava, već sučelje ostaje isto te se promjena elemenata izvršava neovisno o drugim elementima. Kôd 2. koristi se za poboljšanje dinamičnosti mrežne stranice, što poboljšava iskustvo korisnika. Pomoću *JavaScripta* mogu se kreirati animacije, dijalozi i interaktivne karte te je dostupno mnogo biblioteka koje sadrže razne komponente koje se mogu uključivati u projekte.

Popularnost *JavaScripta* rezultirala je velikim brojem programskih okvira (engl. *frameworks*) i biblioteka (engl. *libraries*) koje su kreirali pojedinci ili kompanije. Jedna od najpopularnijih *JavaScript* biblioteka je *jQuery* čiji je slogan „Piši manje, napravi više” (engl. *write less, do more*). Kao što i sam slogan kaže, *jQuery* je biblioteka koja na brz i jednostavan način omogućuje manipulaciju DOM elementima, upravljanje događajima, animacijama i AJAX-om. AJAX (engl. *Asynchronous JavaScript And XML*) omogućuje asinkronu razmjenu podataka s mrežnim serverom bez potrebe da se cijela stranica ponovno preuzima sa servera. Korištenjem *jQueryja* drastično se olakšava korištenje AJAX-a te ujedno automatski pruža kompatibilnost i za starije verzije mrežnog preglednika koje su koristile *ActiveXObject* objekt. Za manipulaciju DOM elementima koristi se znak „$” sa selektorom, kao i u CSS-u, i neka od funkcija za manipulaciju. [5, 6, 7]

```
var imageElement = $('img.dimension_image');

imageElement.load(function () {
    var imageHeight = imageElement[0].naturalHeight;
    var rowElement = $('.cnf-height-total-container');
    var heightElement = $('.cnf-height-total');
    var percent = (imageHeight - 250) / imageHeight * 100 / 2 - 3;
```

Kôd 2. Element slike kod učitavanja

Izvor: Autor

2.5. *MongoDB*

MongoDB je *NoSQL* baza podataka otvorenog koda koja upotrebljava dokumentno-orijentirani model podataka i nestrukturirani jezik upita. Temelji se na konceptu zbirki i dokumenata. Softverska tvrtka 10gen 2007. godine počinje razvijati *Mongo* bazu podataka. Godine 2009. tvrtka 10gen prebacuje model baze na javno dostupan otvoreni kod. Nekoliko godina kasnije mijenjaju naziv u *MongoDB*. Danas je jedna od vodećih *NoSQL* baza podataka razvijena za visoke performanse, dostupnost i jednostavnu skalabilnost. Dokumenti se pohranjuju u bazu podataka u JSON obliku. Tablica 1. prikazuje razlike između *SQL* i *NoSQL* baze podataka. Zbirka je skupina *MongoDB* dokumenata i ekvivalent je tablici u *SQL* bazi podataka. Dokumenti unutar zbirke ne trebaju imati isti skup polja ili strukturu; svaki se dokument sastoji od barem jednog atributa čija vrijednost može biti broj, znakovni niz, Boolean, objekt i datum. [8, 9]

Tablica 1. Razlika između *SQL* i *NoSQL* baze podataka

Relacijske baze podataka	*NoSQL* baze podataka
Relacijska baza podataka	Baza podataka orijentirana na dokumente, bez relacija
Temeljena na tablicama, retcima i stupcima	Kolekcija temeljena na parovima ključ-vrijednost (engl. *key-value*), organizirana u polja i dokumente
Redak	Dokument
Spajanje tablica	Ugrađeni dokument
Ranjiva na *SQL* injekcije	Nije ranjiva na *SQL* injekcije

Izvor: https://intellipaat.com/blog/what-is-mongodb/#Dif_RDBMS (17.6.2022.)

2.6. *Amazon Web Services S3*

Amazonov mrežni servis S3 usluga služi skladištenju objekata u oblaku. Nudi skalabilnost, brzu dostupnost podataka, sigurnost i performanse. Jednostavan je za korištenje i lako se povezuje bilo kojom aplikacijom. Može optimizirati, smanjiti troškove i povećati organiziranost same aplikacije.

3. RAZVOJNO OKRUŽENJE

3.1. PHPStorm

PHPStorm je integrirana razvojna okolina (engl. *Integrated Development Environment*) ili IDE za razvoj mrežnih stranica ili mrežnih aplikacija. Primarno služi za rad u PHP-u kao i za rad u svakom od njegovih programskih okvira (engl. *frameworks)* te zatim HTML-u, CSS-u i *JavaScriptu*. Neke od glavnih značajki naglašavanje su grešaka, automatsko nadopunjavanje, uređivanje koda, alati za ispravljanje grešaka. PHPStorm razvojno okruženje se plaća, ali postoji i besplatna verzija u trajanju od mjesec dana s nešto manje pogodnosti i dodataka.

3.2. *Composer*

Composer je alat za upravljanje ovisnim paketima u aplikacijama napisanim u PHP programskom jeziku. Pomoću alata *Composer* moguće je instalirati i ažurirati sve pakete koji su navedeni kao potrebni u aplikaciji koju programer razvija. U tom slučaju instalirati paket znači preuzeti (engl. *download*) sve datoteke određenog paketa i spremiti ih u određeni direktorij unutar same aplikacije. *Composer* može preuzeti točno određenu verziju ovisnog paketa prema specifikaciji koju sami postavimo. Kako bi *Composer* radio, na operacijskom sustavu potrebno je imati instaliran PHP interpreter. [10]

3.3. *Stripe*

S obzirom na to da se rad fokusira primarno za prodaju proizvoda, potrebno je imati sustav naplaćivanja, a jedan od najpopularnijih sustava za naplatu na internetu je upravo *Stripe*. Podržava više od 100 valuta uključujući i hrvatsku kunu te gotovo sve poznate vrste kartica kao što su Visa, MasterCard i AmericanExpress. Nakon registracije na službenoj stranici dobiva se pristup kontrolnoj ploči (engl. *Dashboard*) na kojoj se vide sve njegove mogućnosti. Glavna mu je mogućnost pregled svih transakcija u određenu periodu. *Developers* stranica služi za programere u kojoj se nalaze API zahtjevi, logovi koji prikazuju sve uspjele ili neuspjele zahtjeve, javni i tajni ključ kojim se povezuju API-ji, itd.

Stripe u svom sustavu ima dva načina rada: testni i produkcijski. U testnom načinu rada koriste se testne kartice koje nemaju limite pa se sve provedene transakcije u testnom načinu rada ne mogu stvarno isplatiti. Testno okruženje potpuno je isto kao i produkcijsko te omogućuje programeru da testira sve potrebne dijelove sustava prije nego što stranicu i sam sustav za plaćanje prebaci u produkcijski način rada. Kada je *Stripe* u produkcijskom okruženju, karticama se provjerava dopušteni limit, sigurnosni kôd i datum isteka. Iznose zaprimljene u produkcijskom načinu rada moguće je isplaćivati automatski u određenim periodima ili ručno po želji vlasnika *Stripe* računa [11].

3.4. *Mailgun*

Mailgun je usluga dostave e-pošte za slanje, primanje i praćenje e-pošte. Pomoću *Mailgun* platforme može se vidjeti, primati i slati e-mail pošta. U aplikaciji se koristi SMTP server, ali *Mailgun* ima i mogućnost korištenja njegovog API-ja koji omogućuje slanje elektroničke pošte u stvarnom vremenu, praćenje i dijagnostiku problema i još mnogo funkcija.

4. *LARAVEL* PROGRAMSKI OKVIR

Laravel je besplatno PHP razvojno okruženje otvorena koda namijenjeno razvoju mrežnih aplikacija korištenjem MVC strukture. U kratkom vremenu zadobio je pažnju razvojnih programera i od 2015. godine proglašen je jednim od najpopularnijih PHP razvojnih okruženja diljem svijeta, ponajprije zbog svoje jednostavnosti, jasne strukture i vrlo detaljne dokumentacije.

4.1. Instalacija

Za instalaciju *Laravela* potrebni su sljedeći uvjeti:

- *BCMath PHP Extension*
- *Ctype PHP Extension*
- *Fileinfo PHP Extension*
- *JSON PHP Extension*
- *Mbstring PHP Extension*
- *OpenSSL PHP Extension*
- *PDO PHP Extension*
- *Tokenizer PHP Extension*
- *XML PHP Extension.*

Laravel koristi *Composer* za upravljanje svojim zavisnostima. U naredbenom retku uz pomoć *Composera* dohvaćamo instalaciju *Laravela* te se automatski provodi njegova instalacija: ***composer global require laravel/installer.***

Nakon što se instalacija završi, stvaranje novog projekta provodi se korištenjem naredbe *laravel new* <naziv mape>. [12]

4.2. MVC arhitektura

PHP razvojno okruženje pomaže u bržem razvoju mrežne aplikacije, daje kvalitetniji odnosno stabilniji kostur aplikacije, smanjuje količinu ponavljajućih dijelova koda i uvelike olakšava kasnije mijenjanje i održavanje aplikacije. Glavna ideja na kojoj se temelji i koja to omogućava je model-pogled-kontroler (engl. *Model-View-Controller*) koji je u zadnjih nekoliko godina najčešće korišten obrazac u području programiranja.

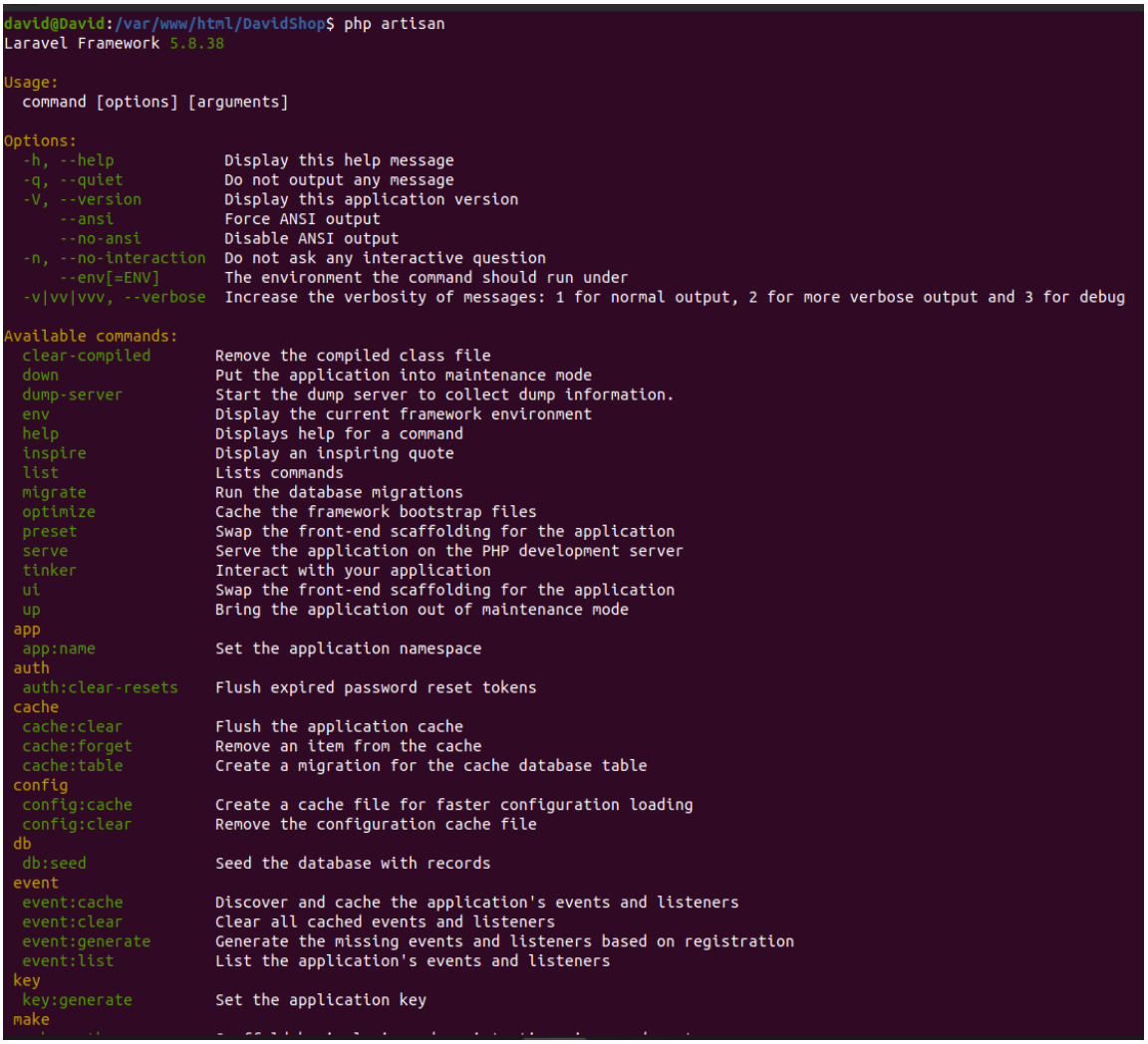
Sloj modela dio je aplikacije gdje bi se trebala nalaziti sva poslovna logika. Modeli su zaslužen za dohvat podataka iz baze podataka i njihovo oblikovanje u korisne koncepte u aplikaciji. Modeli se brinu i o relacijama jednog modela s drugim modelima.

Sloj pogleda dio je u aplikaciji gdje se nalazi većina HTML koda. U *Laravelu* pogledi se mogu nasljeđivati i ubacivati u druge poglede. Kako se linkovi *JavaScripta* i CSS-a ne bi ponavljali na svakom pogledu, definira se glavni predložak (engl. *Layout*) koji je na gotovo svim stranicama naslijeđen. U pogledima se može koristiti i PHP kôd za razne uvjete grananja, za petlje, kao i za varijable koje su proslijeđene na pogled.

Sloj kontrolera odražava poslovnu logiku aplikacije, odnosno pravila i funkcije koje ostvaruju programsku logiku. Odgovoran je za prihvaćanje i upravljanje unosom podataka, odnosno zahtjeva korisnika (GET, POST) i pretvara ih u odgovarajuće zahtjeve koje šalje modelu i pogledu. Sadrži sve kontrole aplikacija i određuje koje se promjene trebaju dogoditi u modelu i pogledu pa je na taj način direktno odgovoran za tijek kontrole programa.

4.3. PHP *Artisan*

Artisan je sučelje naredbenog retka koje koristi *Laravel*. Zaslužno je za mnogo korisnih naredbi koje ubrzavaju rad na aplikaciji. Nakon što se pozicioniramo unutar direktorija *Laravel* projekta naredbom PHP *Artisan* list, ispisat će se sve naredbe koje se mogu koristiti s *Artisanom*, kao što je vidljivo na Slici 2. Svaka naredba u sebi sadrži pomoćni prozor (engl. *help screen*) koji prikazuje i opisuje „prirodu“ određene naredbe. Ukoliko nam zatreba pomoć u vezi određene naredbe, napisat ćemo ključnu riječ *help* prije naredbe o kojoj želimo dobiti dodatne informacije (npr. *php artisan help migrate*).



Slika 2. Sve opcije PHP *Artisana*

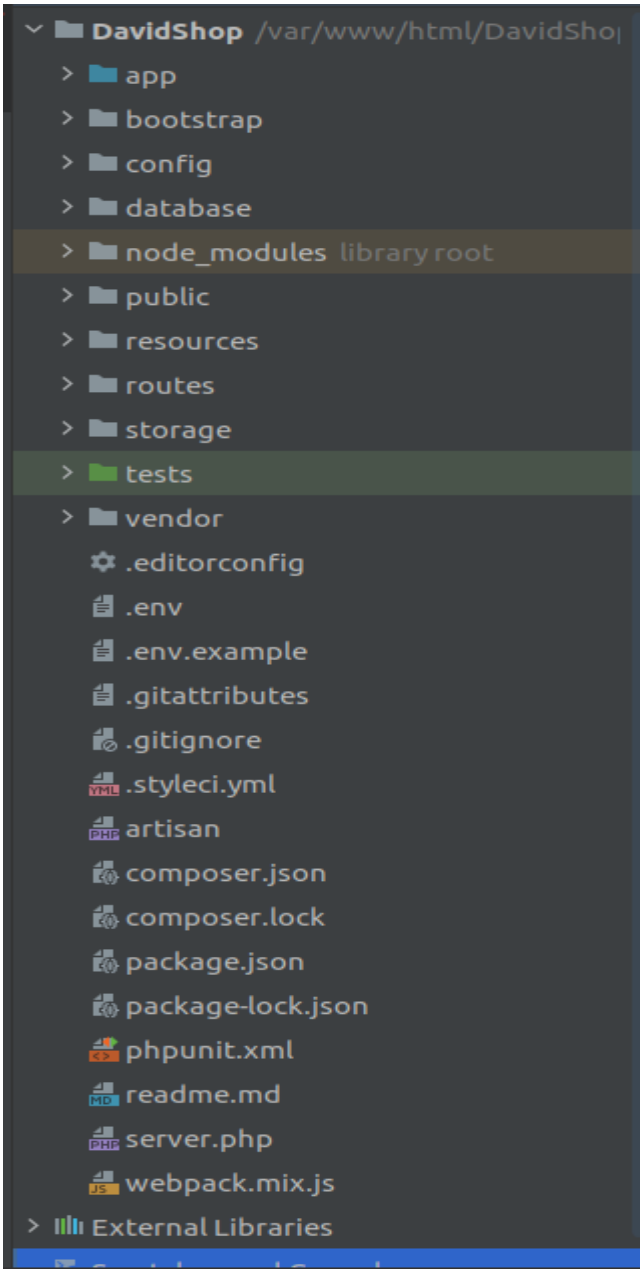
Izvor: Autor

4.4. Analiza kostura aplikacije

Osnovna struktura novog *Laravel* projekta sadrži sljedeće direktorije:

- *App* – u njemu se nalazi jezgra aplikacije, a to su kontroleri, modeli i *middleware* mehanizam (kontrola zahtjeva prema serveru).
- *Bootstrap* – direktorij koji služi za povezivanje ključnih komponenti cijelog *Laravel* programskog okvira (engl. *frameworka).* Ovaj direktorij sadržava i *cache* direktorij koji sprema podatke podmemorije.
- *Config* – direktorij koji sadržava konfiguracijske datoteke, a od značajnijih datoteka su app.php gdje se nalaze osnovne postavke aplikacije, te database.php gdje se nalaze postavke paze podataka.
- *Database* – u sebi sadrži migracije.
- *node_modules* – nije u osnovnoj strukturi *Laravel* aplikacije, nastaje pokretanjem *npm install* i u sebi sadrži module za *frontend* dio aplikacije.
- *Public* – u sebi sadrži index.php datoteku koja je početna točka (engl. *Entry point*). Kad korisnik prvi put posjeti mrežnu aplikaciju, kôd se prvo pokreće u toj datoteci.
- *Resouces* – direktorij sadrži sve predloške mrežne aplikacije.
- *Routes* – ovaj direktorij sadrži sve rute i tokove aplikacije, glavna datoteka je web.php.
- Storage – direktorij koji služi za lokalno spremanje podataka (sesije, podmemorija aplikacije, *blade* predlošci itd.).
- *tests* – unutar tog direktorija nalaze se testne skripte aplikacije.
- *vendor* – sadrži sve datoteke potrebne za rad *Composera*.

Nakon svih direktorija prikazanih na Slici 3. nalazi se jedna bitna datoteka, a to je .env koja služi kao konfiguracijska datoteka naše mrežne aplikacije. Unutar .env datoteke nalaze se bitne stavke kao što su ključ za kriptiranje aplikacije, podatci za spajanje baze podataka i upravljački program (engl. *driver)* za upravljanje e-mailovima.



Slika 3. Struktura direktorija i datoteka u novoinstaliranoj *Laravel* aplikaciji

Izvor: Autor

4.5. *Eloquent* ORM

Eloquent ORM (objektno-relacijsko mapiranje) omogućuje nam izbjegavanje pisanja dugih i često nepreglednih *SQL* upita. *Eloquent* također omogućava da se jedan model (klasa) poveže s jednom tablicom u bazi podataka. *Laravel* slijedi konvencije pisanja koda. Nije nužno da slijede jedan nakon drugog, ali je korisno i ubrzava izradu mrežne aplikacije. Model bi uvijek trebao biti u jednini, a naziv tablice trebao bi biti u množini. *Eloquent* je usko povezan i s migracijama. Ukratko, migracije predstavljaju uzorke tablica u slučaju izrađene aplikacije kolekcije. S obzirom na to da se u aplikaciji koristi *MongoDB* autora *Jenssegers*, u svakom modelu mora se postaviti model za *MongoDB* i naslijediti klasu s modelom koji je zadan za *MongoDB*. Unutar svakog modela postavlja se naziv kolekcije kako bi *Laravel* prepoznao da se radi o kolekciji, a ne o tablicama.[13]

5. IZRADA APLIKACIJE

Nakon uspješne instalacije *Laravel* aplikacije potrebno je napraviti nekoliko koraka kako bi aplikacija mogla funkcionirati. Upotrijebljen je paket laravel-*mongodb* autora *Jenssegers* za rad s *MongoDB* bazom podataka. [14]

Aplikacija ima mogućnost registracije, prijave, pretraživanja proizvoda, filtriranja proizvoda, detaljnog prikaza i naručivanja proizvoda, dodavanja kategorija i proizvoda te upravljanja narudžbama.

5.1. Konfiguracija

U radu je korištena *MongoDB* baza podataka koja nije podržana u *Laravel* programskom okviru (engl. f*ramework)* te je potrebno koristiti Jenssegersov paket. Potrebno je pokrenuti *Composer* naredbu *composer require jenssegers/mongodb* kako bi se instalirao paket u aplikaciji. U config/app.php datoteci dodaje se klasa davatelja usluga (engl. *Service provider*) Jenssegers\ Mongodb\MongodbServiceProvider::class. Kako bi se povezala baza podataka s aplikacijom, u config/database.php treba se dodati *driver mongodb* s potrebnim podatcima o bazi podataka, kako je prikazano na Slici 4.

```
'mongodb' => [
    'driver' => 'mongodb',
    'host' => env( key: 'DB_HOST', default: 'localhost'),
    'port' => env( key: 'DB_PORT', default: '27017'),
    'database' => env( key: 'DB_DATABASE'),
    'username' => env( key: 'DB_USERNAME'),
    'password' => env( key: 'DB_PASSWORD'),
    'options' => [
        'database' => env( key: 'DB_AUTHENTICATION_DATABASE', default: 'admin'),
    ],
],
```

Slika 4. Postavljanje *mongodb drivera* u config/database.php datoteci

Izvor: Autor

5.2. Uloge

U ovoj aplikaciji postoje dvije uloge, a to su:

- korisnik/gost
- administrator

Gost je uloga u kojoj korisnik ima pristup svim stranicama osim administrativnom sučelju. Može pregledavati proizvode, dodavati u košaricu i napraviti narudžbu (kupiti proizvod).

Administrator je uloga koja ima pristup svemu unutar aplikacije uključujući i administrativno sučelje gdje ima pregled korisnika, njihovih narudžbi, podataka mrežne trgovine, a isto tako može mijenjati, brisati i dodavati svaku opciju.

5.2.1. Korisnik/gost

Korisniku je omogućeno pretraživanje proizvoda, filtriranje po opcijama proizvoda, atributima proizvoda, brendovima i rasponu cijena. Proizvode je moguće dodati u košaricu i izvršiti kupovinu proizvoda koji se nalaze u košarici.

5.2.2. Administrator

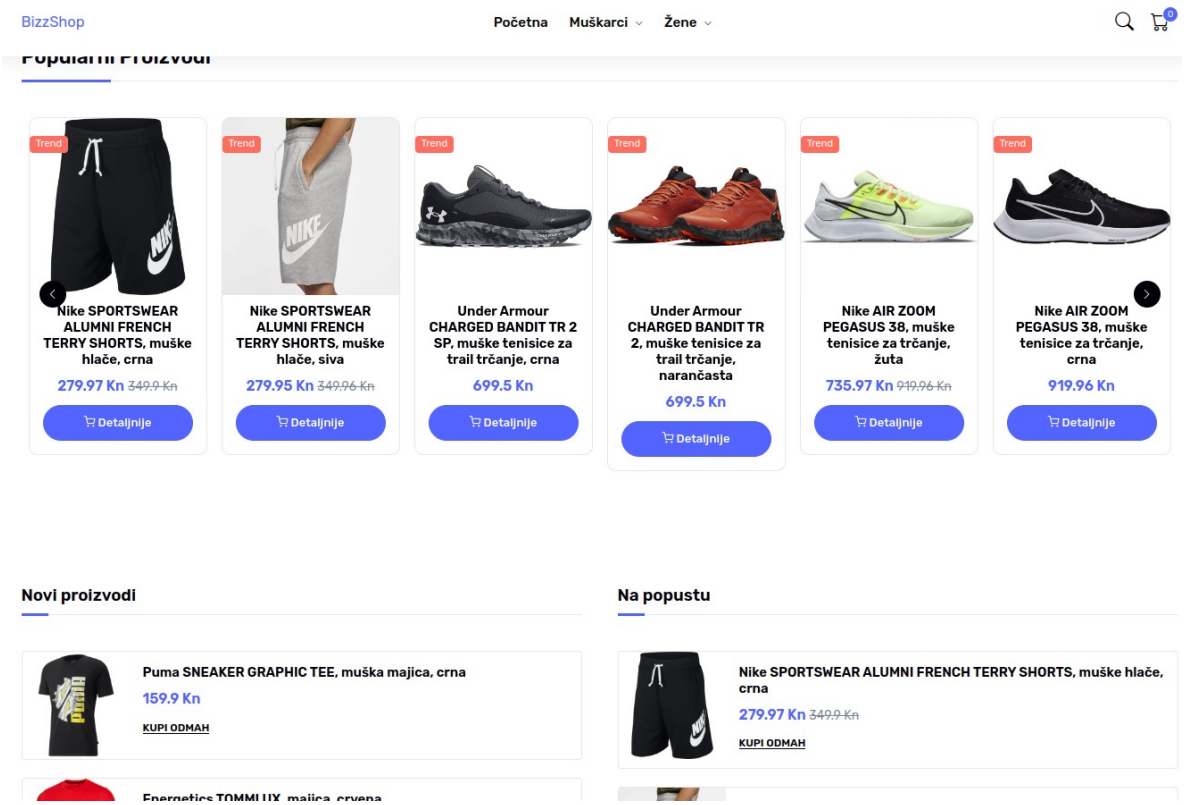
Administrator ima pristup svim dijelovima sustava, uključujući i administrativno sučelje. Pristupom u administrativno sučelje administrator je u mogućnosti kreirati kategorije, potkategorije, proizvode, upravljati postojećim proizvodima, kategorijama i potkategorijama, pregledom svih narudžbi korisnika i otkazivanjem narudžbi.

5.3. Internetska trgovina

Internetska trgovina glavni je modul koji korisnik vidi nakon učitavanja stranice. U ovom je modulu omogućen prikaz proizvoda, pretraživanje i kupovina proizvoda, uređivanje profila i sl.

5.3.1. Početna stranica

Početna stranica glavna je stranica koju korisnik vidi kada pristupi mrežnoj stranici, vidljivo na Slici 5. Na početnoj su stranici prikazane kategorije. Na naslovnici se nalaze proizvodi koje je administrator posljednje dodao. Proizvode s početne stranice moguće je detaljnije vidjeti klikom na gumb “detaljnije” kako bi korisnik mogao odabrati opciju proizvoda i dodati u košaricu.

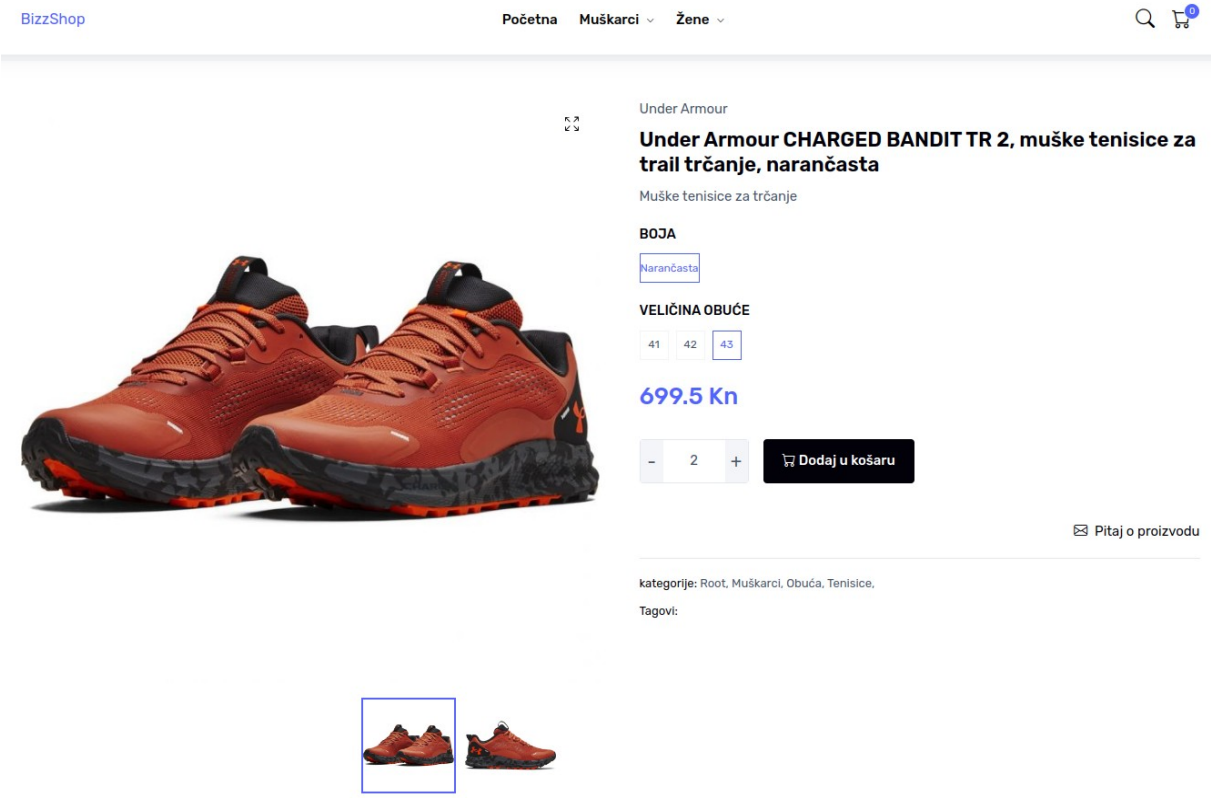


Slika 5. Početna stranica

Izvor: Autor

5.3.2. Detalji proizvoda

Nakon što korisnik klikne na gumb “detaljnije”, otvara se nova stranica u kojoj su prikazani svi podatci odabranoga proizvoda, što je vidljivo na Slici 6. Uz vidljive detalja proizvoda, ako korisnik želi dodati proizvod u košaricu, mora odabrati željenu opciju proizvoda i količinu.

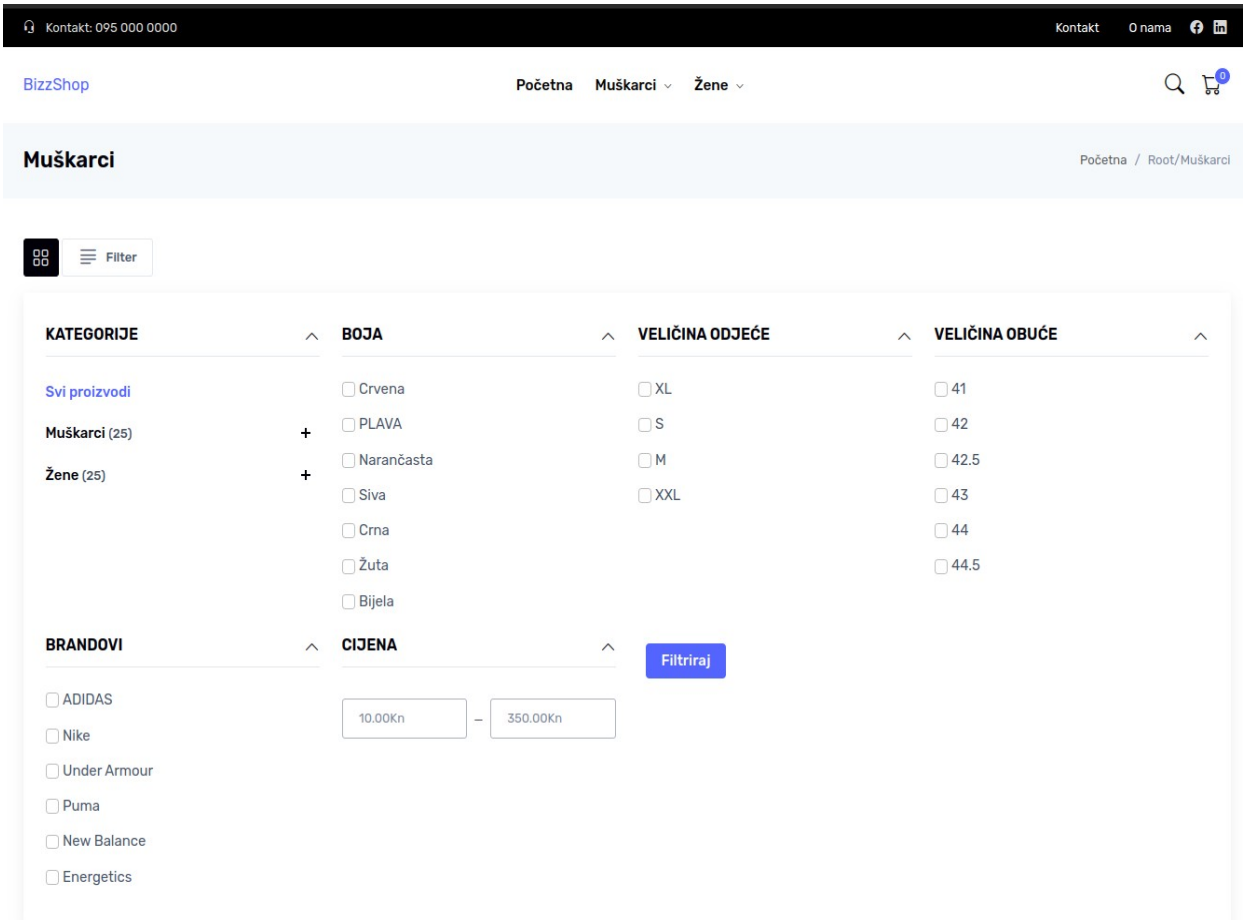


Slika 6. Prikaz detaljnih podataka proizvoda

Izvor: Autor

5.3.3. Filtriranje proizvoda

Nakon što korisnik odabere željenu kategoriju na početnoj stranici, aplikacija ga preusmjeri na stranicu sa svim proizvodima koji su u odabranoj kategoriji. Na toj stranici dobije se filter navigacija prikazan na Slici 7. kojom korisnik može odabrati željene potkategorije, raspon cijena, brendova i opcija proizvoda. Osim filtriranja proizvoda korisnik može pretraživati proizvode prema upisanoj riječi klikom na ikonu tražilica.

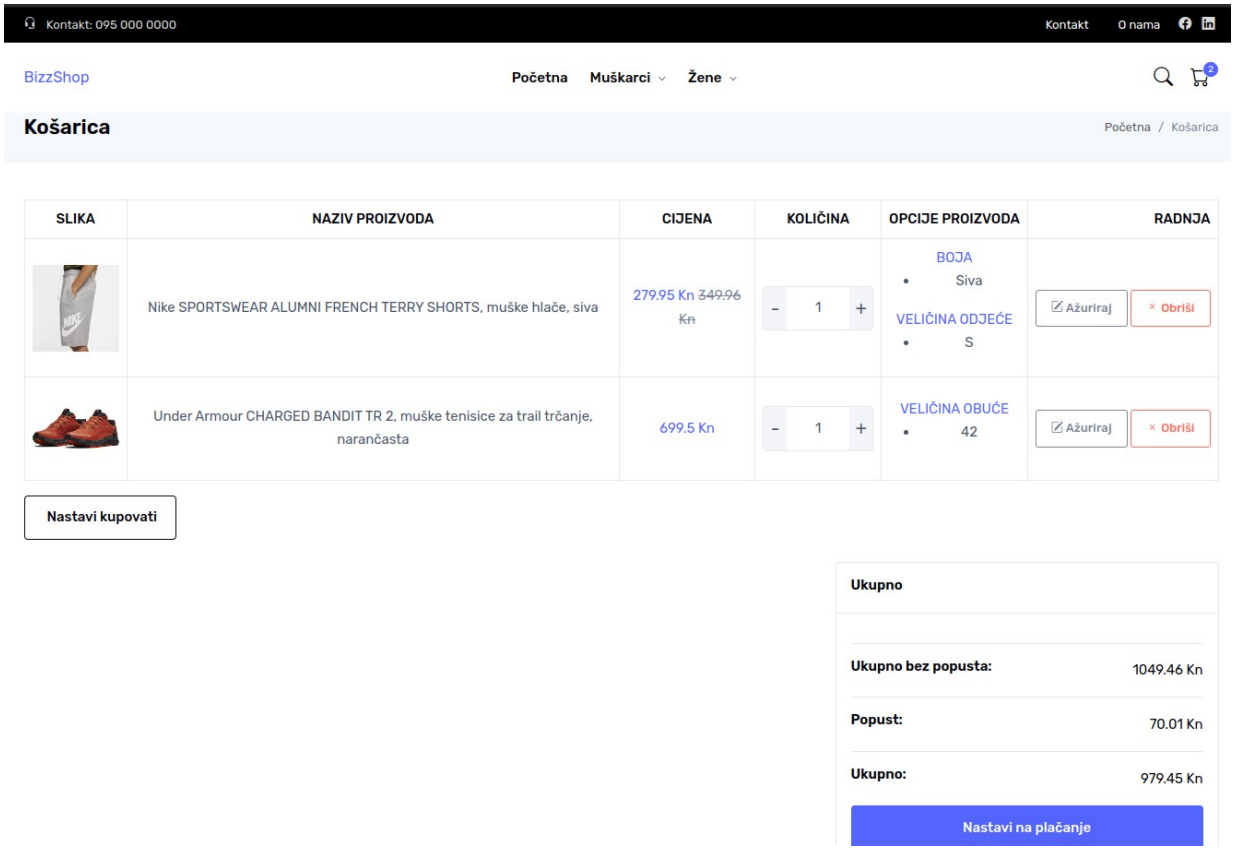


Slika 7. Prikaz filtriranja proizvoda

Izvor: Autor

5.3.4. Košarica

Korisnik može dodavati proizvode u košaricu, vidjeti detalje dodanih proizvoda, povećavati količinu svakog proizvoda i imati uvid u ukupan iznos svih proizvoda (Slika 8.) Kada korisnik pritisne na gumb „nastavi na plaćanje“, aplikacija ga preusmjeri na stranicu za plaćanje.

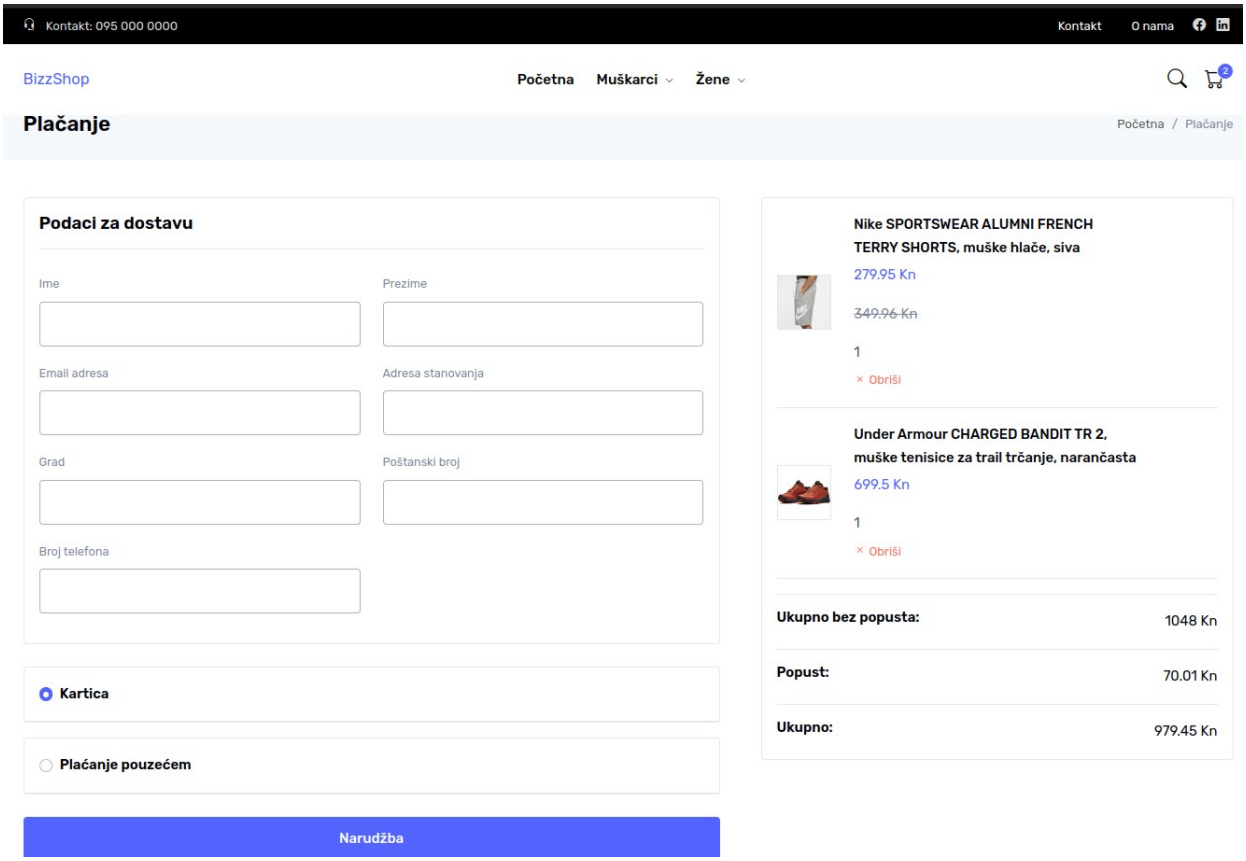


Slika 8. Pregled košarice

Izvor: Autor

5.3.5. Plaćanje

U košarici korisnik plaća sve proizvode klikom na gumb „nastavi na plaćanje”. Na stranici za plaćanje prikazuju se proizvodi koji su odabrani u košarici i forma za dostavu koju je potrebno ispuniti kako bi se plaćanje uspješno izvršilo (Slika 9.) Korisnik ima dvije mogućnosti kod plaćanja: klikom na oznaku kartica gdje upisuje podatke kartice i plaća automatski klikom na gumb “narudžba” ili klikom na oznaku plaćanje pouzećem gdje korisnik plaća gotovinom kod primopredaje paketa. Nakon uspješne narudžbe korisnik će biti preusmjeren na stranicu s porukom uspješne transakcije, a ako podatci nisu valjani, korisnika se vraća nazad na stranicu za narudžbu.

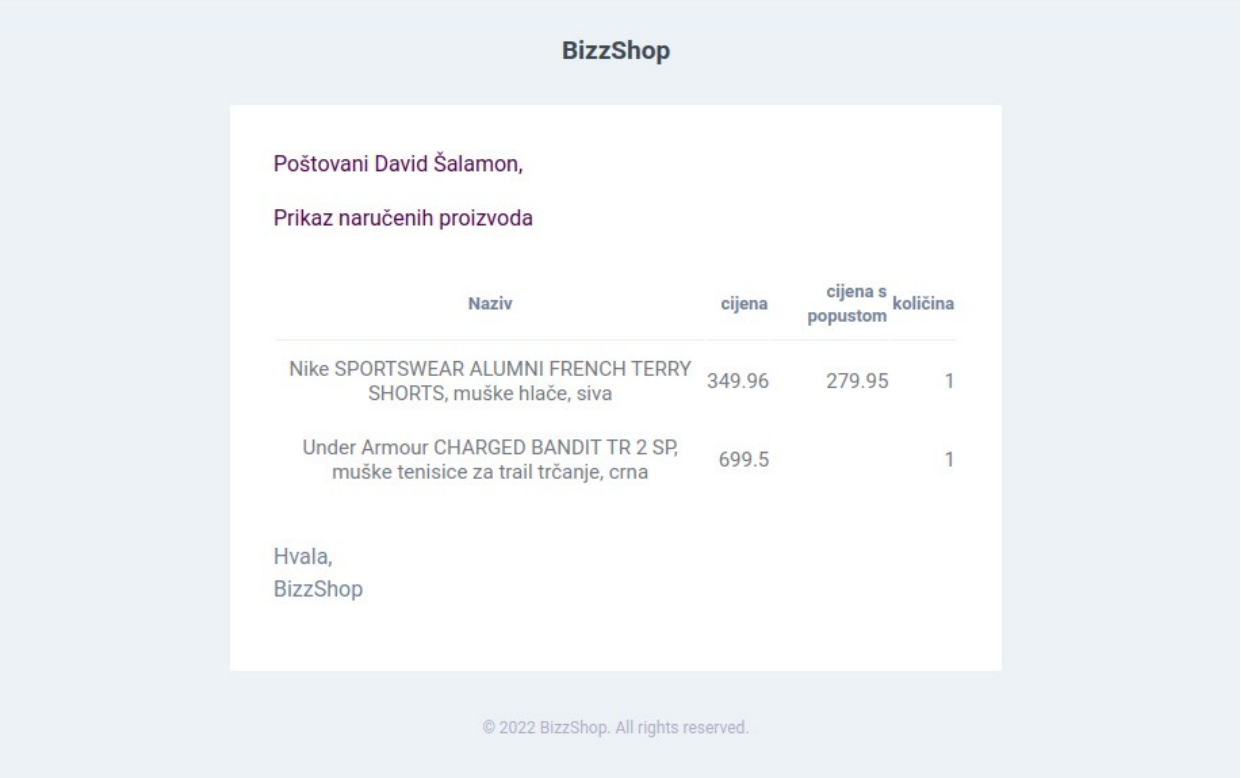


Slika 9. Prikaz stranice za narudžbu

Izvor: Autor

5.3.6. Uspješna narudžba

Nakon uspješne narudžbe korisniku se šalje mejl narudžbe u kojoj se nalazi tablica naručenih proizvoda, popust i ukupna cijena (Slika 10.).



Slika 10. Prikaz narudžbe korisnika u elektroničkoj pošti

Izvor: Autor

5.4. Administracijsko sučelje

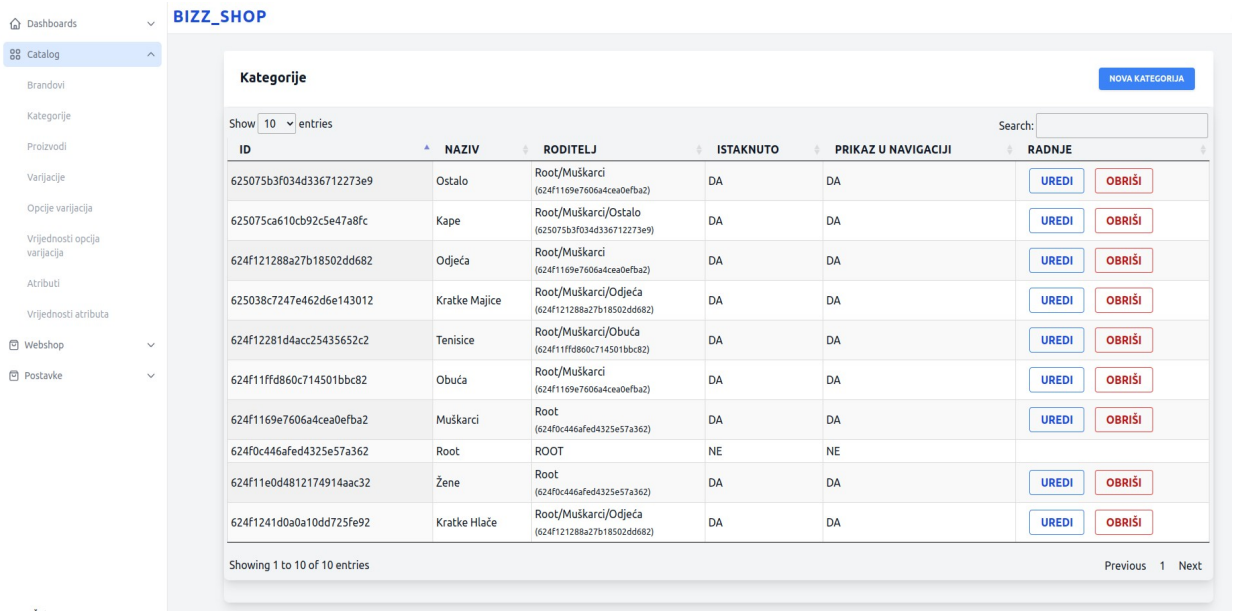
Administrator u url-u mora upisati /admin kako bi pristupio formi za prijavu u administrativno sučelje. Nakon uspješne prijave korisnika s ulogom administrator u glavnoj navigaciji korisnik ima pristup svim funkcijama za promjenu sadržaja u sustavu.

5.4.1. Brendovi

Odabirom “brendovi” u padajućem izborniku “katalog” korisnik može vidjeti popis brendova, dodavati novi brend ili urediti postojeći.

5.4.2. Kategorije

Odabirom kategorije u padajućem izborniku “katalog” korisnik može vidjeti popis kategorija kako je prikazano na Slici 11., dodati novu kategoriju ili urediti postojeću. Kod stvaranja nove ili uređenja postojeće kategorije korisnik može odabrati nadređenu kategoriju kako bi stvorio potkategoriju.

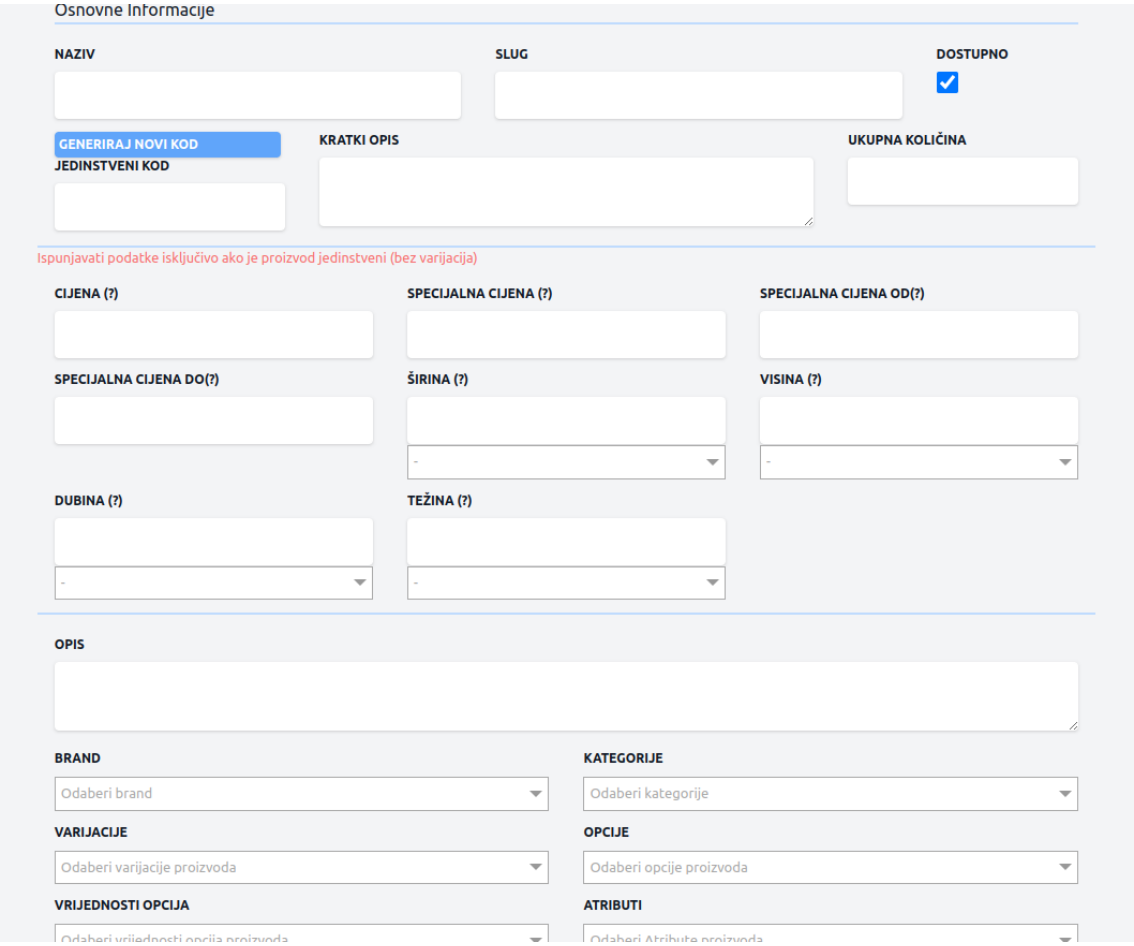


Slika 11. Prikaz kategorija

Izvor: Autor

5.4.3. Proizvodi

Unutar sučelja administrator može upravljati svim proizvodima u internetskoj trgovini. Klikom na “novi proizvod” administrator se preusmjerava na formu gdje upisuje sve potrebne podatke o proizvodu, povezuje varijacije, opcije, atribute i ostale vrijednosti koje pripadaju proizvodu (Slika 12.) Klikom na “uredi” administrator može ažurirati podatke proizvoda ili jednostavno imati uvid u detaljnije informacije samoga proizvoda.



Slika 12. Prikaz forme za novi proizvod

Izvor: Autor

5.4.4. Varijacije

Ukoliko proizvod nije jedinstven, stvaraju se nove varijacije. Korisnik upisuje sve potrebne podatke o varijaciji, povezuje opcije, atribute i ostale vrijednosti koje pripadaju varijaciji i spaja ih s nadređenim proizvodom. Klikom na “uredi” administrator može ažurirati podatke varijacije ili jednostavno imati uvid u detaljnije informacije same varijacije.

5.4.5. Opcije varijacija

Opcije neke varijacije ili proizvoda mogu biti boja, veličina itd. Odabirom opcije u padajućem izborniku “katalog” korisnik može vidjeti popis opcija varijacija, dodavati novu opciju ili urediti postojeću uz povezivanje svih vrijednosti koje pripadaju toj opciji.

5.4.6. Vrijednosti opcija varijacija

Vrijednosti opcija su vrijednosti koje predstavljaju tu opciju, a mogu biti npr. crna, bijela, XL, M itd. Odabirom vrijednosti opcija varijacija u padajućem izborniku “katalog” korisnik može vidjeti popis vrijednosti opcija varijacija, dodavati novu vrijednost ili urediti postojeću.

5.4.7. Atributi

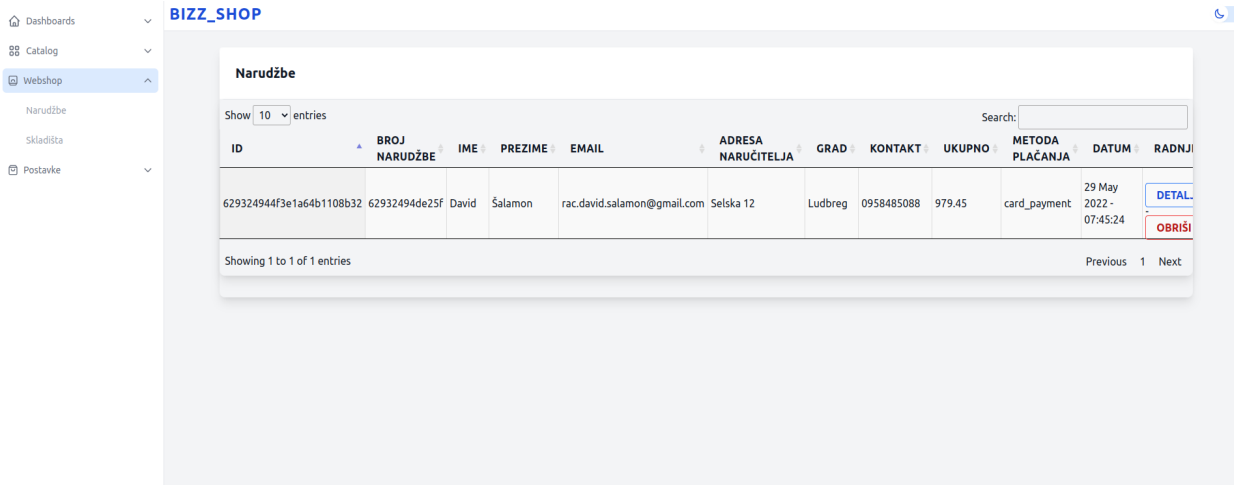
Atributi neke varijacije ili proizvoda mogu biti npr. vrsta materijala, veličina ekrana itd. Odabirom atributa u padajućem izborniku “katalog” korisnik može vidjeti popis atributa, dodavati novi atribut ili urediti postojeći uz povezivanje svih vrijednosti atributa koje pripadaju tom atributu.

5.4.8. Vrijednosti atributa

Vrijednosti atributa varijacije ili proizvoda mogu biti npr. pamuk, plastika, 5.1 itd. Odabirom vrijednosti atributa u padajućem izborniku "katalog" korisnik može vidjeti popis vrijednosti atributa, dodavati novu vrijednost ili urediti postojeću.

5.4.9. Narudžbe

Odabirom narudžbe u padajućem izborniku *webshop* administrator može vidjeti popis svih narudžbi i podatke korisnika koji je stvorio narudžbu (Slika 13.) Klikom na "detaljno" administrator ima uvid u proizvode koji su naručeni, vrstu plaćanja, količinu svakog proizvoda i ukupnu cijenu. Nakon poslane pošiljke administrator ima opciju obrisati narudžbu.



Slika 13. Prikaz tablice narudžbi

Izvor: Autor

5.5. Postavljanje aplikacije na *Cloud* servis *DigitalOceana*

Posljednji je korak izrađenu mrežnu trgovinu s lokalnog servera premjestiti na *Cloud* servis kako bi bio dostupan na internetu te kako bi se njime moglo upravljati putem C panela u „oblaku“. Za *Cloud Hosting* odabran je *DigitalOcean*. Kompletna stranica je putem *Filezillae* premještena u *droplet DigitalOceana* te je stvoreno novo okruženje (engl. *Environment*) na kojemu se aplikacija pokreće. [15]

6. ZAKLJUČAK

U prošlosti je svaka trgovina, da bi mogla funkcionirati, zahtijevala određeni prostor i prodavače. U današnje vrijeme sve je više internetskih aplikacija za prodaju. Mrežne aplikacije jeftinije su od standardnog načina rada. Također, sam proces kupnje proizvoda često može biti puno brži.

Korištenjem modernih tehnologija kao što su PHP i *Laravel* programsko okruženje moguće je izraditi internetsku trgovinu koja će sama naplaćivati kupljene proizvode. *Stripe* sustav za plaćanje korištenjem *JavaScript* biblioteke i službenog API-ja omogućuje sigurno provođenje transakcija i bilježi ih u *Stripe* administratorskom sučelju. *Stripe* sustav za plaćanje pruža mnogo mogućnosti koje se mogu dodatno proširiti, kao što su mjesečne pretplate ili obročno plaćanje.

Cijela aplikacija osmišljena je tako da se može u bilo kojemu trenutku nadograđivati ovisno o potrebama kupaca i klijenata. Smatram da je za daljnji razvoj potrebno integrirati skladišne sustave kako bismo dobili prijenos količina i ostalih podataka u realnom vremenu.

7. LITERATURA

[1] HTML, https://developer.mozilla.org/en-US/docs/Web/HTML (17.6.2022.)

[2] CSS , https://developer.mozilla.org/en-US/docs/Web/CSS (20.6.2022.)

[3] PHP , http://www.phpbbhq.com/developmentofphp.php (20.6.2022.)

[4] PHP, https://www.w3schools.com/php/DEFAULT.asp (21.6.2022.)

[5] JavaScript, https://developer.mozilla.org/en-US/docs/Web/JavaScript (21.6.2022.)

[6] jQuery, https://jquery.com/ (21.6.2022.)

[7] jQuery, https://www.w3schools.com/jquery/(22. 6. 2022.)

[8] MongoDB, https://www.mongodb.com/developer-tools (22.6.2022.)

[9] MongoDB Compass, https://www.mongodb.com/products/compass (22.6.2022.)

[10] Composer, https://getcomposer.org/doc/00-intro.md (23.6.2022.)

[11] https://stripe.com/docs, (23.6.2022.)

[12] https://laravel.com/docs/5.8, (24.6.2022.)

[13] https://laravel.com/docs/5.8/eloquent, (1.7.2022.)

[14] https://github.com/jenssegers/laravel-mongodb (4.7.2022.)

[15] https://docs.aws.amazon.com/index.html?nc2=h_ql_doc_do (4.7.2022.)

Popis slika/fotografija

Slika 1. Prikaz najtraženijih tehnologija po ocjenama korisnika *Stack Owerflowa* 11
Slika 2. Sve opcije *PHP Artisana* 18
Slika 3. Struktura direktorija i datoteka u novo instaliranoj *Laravel* aplikaciji 20
Slika 4. Postavljanje mongodb drivera u config/database.php datoteci 22
Slika 5. Početna stranica 24
Slika 6. Prikaz detaljnih podataka proizvoda 25
Slika 7. Prikaz filtriranja proizvoda 26
Slika 8. Pregled košarice 27
Slika 9. Prikaz stranice za narudžbu 28
Slika 10. Prikaz narudžbe korisnika u elektroničkoj pošti 29
Slika 11. Prikaz kategorija 31
Slika 12. Prikaz forme za novi proizvod 32
Slika 13. Prikaz tablice narudžbi 24

Popis tablica

Tablica 1. Razlika između *SQL* i *NoSQL* baze podataka 13

Popis kôdova

Kôd 1. Gumb u formi s atributom klase 9
Kôd 2. Element slike kod učitavanja 12