Razvoj moderne online trgovine na Sylius e-commerce platformi

Slović, Kristina

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

RAZVOJ MODERNE ONLINE TRGOVINE NA SYLIUS E-COMMERCE PLATFORMI

Diplomski rad

Kristina Slović

Osijek, 2018.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 08.09.2018.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Kristina Slović
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 883 R, 27.09.2017.
OIB studenta:	38885630591
Mentor:	Doc.dr.sc. Ivica Lukić
Sumentor:	
Sumentor iz tvrtke:	Ivan Weiler
Predsjednik Povjerenstva:	Doc.dr.sc. Mirko Köhler
Član Povjerenstva:	Krešimir Vdovjak
Naslov diplomskog rada:	Razvoj moderne online trgovine na Sylius e-commerce platformi
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak diplomskog rada:	Objasniti pojam e-commerce te što predstavlja u pogledu interneta današnjice. Opisati Sylius e-commerce platformu, navesti njezine prednosti i mane u usporedbi s ostalim konkurentnim e-commerce platformama na tržištu. Opisati tehnologije koje se koriste za razvoj logike modernih E-commerce stranica. Objasniti arhitekturu Sylius platforme i njene sastavne komponente. Opisati obrasce dizajna korištene tijekom razvoja logike Sylius aplikacije kao što su Repository, Factory i Dependency Injection. Dokumentirati proces razvoja e-commerce rješenja iz backend perspektive od početnog planiranja, kreiranja programske logike i funkcionalnosti koje se koriste na stranici, izrade vlastitih ekstenzija i tehničke optimizacije stranice sve do postavljanja stranice u produkciju.
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	08.09.2018.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 19.09.2018.

Ime i prezime studenta:	Kristina Slović
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 883 R, 27.09.2017.
Ephorus podudaranje [%]:	3%

Ovom izjavom izjavljujem da je rad pod nazivom**: Razvoj moderne online trgovine na Sylius e-commerce platformi**

izrađen pod vodstvom mentora Doc.dr.sc. Ivica Lukić

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.

Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Cilj ovog rada je dati uvid u postojeće platforme za razvoj e-trgovina (engl. *e-commerce*) s posebnim osvrtom na Sylius platformu, njene prednosti i nedostatke, arhitekturu te proces kreiranja novih proširenja sustava koja će obuhvatiti dodatne zahtjeve klijenta. Kroz opis konfiguracije postavki i razvoja ekstenzija prikazan je najvažniji dio procesa razvoja moderne e-trgovine. Opisani proces razvoja odnosi se na postavljanje zahtjeva klijenata, konfiguraciju, dizajn željenih funkcionalnosti definiranih unutar zahtjeva, proces implementacije tih funkcionalnosti unutar Sylius ekstenzije te u konačnici opis korištenja ostvarenih ekstenzija.

Unutar sljedećeg poglavlja rada definira se sam pojam te povijesni razvoj e-trgovina kroz različite primjere platformi s osvrtom na one najpopularnije te s naglaskom na opis Sylius platforme. Treće poglavlje predstavlja idejno rješenje implementacije gdje se iz zahtjeva klijenata definiraju ekstenzije prikladnog dizajna funkcionalnosti. Četvrto poglavlje predstavlja konkretno programsko rješenje sustava koje se sastoji od opisa korištenih tehnologija, prikaza procesa konfiguracije i izrade te obrazaca dizajna koji su korišteni tijekom izrade ekstenzije. Na samom kraju nalazi se testiranje i korištenje ostvarenog sustava, odnosno primjeri korištenja razvijenih ekstenzija.

1.1. Zadatak diplomskog rada

Objasniti pojam e-trgovine te što predstavlja u pogledu interneta današnjice. Opisati Sylius platformu, navesti njezine prednosti i mane u usporedbi s ostalim konkurentnim platformama na tržištu. Opisati tehnologije koje se koriste za razvoj logike modernih e-trgovina. Objasniti arhitekturu Sylius platforme i njene sastavne komponente. Opisati obrasce dizajna korištene tijekom razvoja logike Sylius aplikacije kao što su *Repository*, *Factory* i *Dependency Injection*. Dokumentirati proces razvoja e-trgovine iz backend perspektive od početnog planiranja, kreiranja programske logike i funkcionalnosti koje se koriste na stranici, izrade vlastitih ekstenzija i tehničke optimizacije stranice sve do postavljanja stranice u produkciju.

2. RAZVOJ E-TRGOVINA I POSTOJEĆE PLATFORME ZA RAZVOJ

U širem smislu pojam e-trgovine se odnosi na izmjenu poslovnih informacija, održavanje poslovnih veza i provođenje poslovnih transakcija korištenjem telekomunikacijskih mreža. Glavne komponente e-trgovina su komunikacijski sustavi, upravljanje podatcima, transakcijski sustavi i sigurnost. U užem smislu e-trgovina predstavlja koncept prodaje i kupnje posredstvom interneta. Postojanje virtualnog tržišta omogućilo je brzu i fleksibilnu razmjenu dobara u par klikova s bilo koje lokacije u svijetu bez potrebe za okupacijom fizičkog prostora [1].

2.1. Razvoj e-trgovina i općenito o platformama za razvoj

E-trgovina se prvi put spominje 70-ih s pojavom sustava elektroničkog prijenosa sredstava. Međutim ovakvi sustavi bili su ograničeni na veće tvrtke. Nakon pojave elektroničke razmjene podataka (EDI), omogućene su razne transakcije i otvorena je prilika za širu upotrebu. EDI se odnosio na elektroničko slanje poslovnih dokumenata kao što su narudžbe i računi. Komercijalizacijom interneta i predstavljanjem web-a ranih 90-ih, E-trgovina se spominje u kontekstu elektroničke razmjene dobara, a od 1998. nastaju prvi sustavi u SAD-u i Europi koji se od tog trenutka ubrzano šire i unapređuju. Širenjem e-trgovina nastaju i prve platforme (Sl.2.1.) za razvoj [1]. Platforme omogućuju okruženje za brzu i fleksibilnu izradu aplikacija. Ovakve platforme implementiraju opći tijek programa poput procesa isplate, ali također ostavljaju mogućnost proširenja i prilagodbe funkcionalnosti vlastitim potrebama [2]. Arhitektura platformi temelji se na snažnom modelu arhitekture koji sadrži različite obrasce dizajna kao što je *Dependency Injection, Factory, Decorator i Publish Subscribe* model. Obrasci dizajna omogućuju platformama elegantna i iznova upotrebljiva rješenja učestalih problema tijekom razvoja softvera. Unutar idućih poglavlja opisane su neke od najpopularnijih platformi: Magento, Shopify, OROCommerce i Sylius.

Sl. 2.1. Razvoj e-trgovina

2.2. Magento

Mageno je platforma otvorenog koda za razvoj e-trgovina koja se temelji na Zend razvojnom okruženju i PHP programskom jeziku. Magento je predstavio Varien 2007. Na početku je tvrtka težila razviti novu granu osCommerce sustava, međutim zbog mnoštva zastarjelih funkcionalnosti su odlučili lansirati novi sustav Magento 1. Razvile su se brojne verzije Magenta kao što je Magento Community, Magneto Enterprise i Magento Mobile. U 2015. lansiran je ažurirani sustav s naprednijim funkcionalnostima i novijim razvojnim alatima i strukturama Magento 2 [3].

Glavne funkcionalnosti Magenta [4]:

- Marketinški, promocijski i konverzijski alati
- Optimizacija tražilica
- Upravljanje stranicom putem administratorskog sučelja
- Upravljanje katalogom proizvoda
- Pregled i sortiranje proizvoda
- Isplata i dostava
- Upravljanje narudžbama
- Korisnički računi
- Analitika i izvještavanje
- Mobilne e-trgovine

Magento je vrlo fleksibilna platforma kojom se mogu realizirati vrlo specifični zahtjevi u obliku modula kao temeljne jedinice arhitekture sustava, no kako bi se iskoristila sva njegova funkcionalnost potrebno je tehničko iskustvo, odnosno iskustvo u kodiranju. Podržani su razni načini plaćanja i dostave i veliki broj dodatnih aplikacija i tema. Hosting je uključen unutar Magenta, ali je i moguće angažirati vlastitog pružatelja hosting-a.

2.3. Shopify

U samim početcima Shopify je bio web trgovina, a tek poslije se razvio kao platforma za razvoj e-trgovina. Shopify je kao platforma za razvoj osnovan 2006. u Ottawi, a osnovali su ju Tobias Lütke, Daniel Weinand i Scott Lake. Korištenje samog Shopify-ja nije besplatno. Shopify je

izrađen u Ruby on Rails razvojnom okruženju, a primjeren je za korištenje i korisnicima slabijeg tehničkog znanja koji se ne žele upustiti u samo kodiranje. Kao i kod Magenta dostupan je velik broj dodatnih aplikacija i tema te vlastiti *hosting*. Shopify nudi svoj način plaćanja Shopify Payments, ali također podržava i druge načine plaćanja uz nadoknadu pri transakciji [4].

2.4. OROCommerce

B2B (*Business to business*) platforma otvorenog koda koju je osnovao ORO 2017. godine. ORO aplikacija je PHP web aplikacija koja koristi Symfony razvojno okruženje. Skalabilnosti, proširivost, portabilnosti i mogućnost prilagodbe glavne su karakteristike OROCommerce platforme. ORO nudi različite nadogradnje u obliku paketa koji su dostupni na ORO Marketplace-u. Također je moguća izrada vlastitih paketa u svrhu implementacije potrebnih funkcionalnosti. U glavne funkcionalnosti sustava spadaju kompleksna struktura korisničkih računa s različitim razinama pristupa, kontrole i dozvola, zaštita podataka i upravljanje suglasnostima, višestruke organizacije, web stranice i trgovine, CMS (sustav za upravljanje sadržajem), personalizirano upravljanje katalozima, višestruke i prilagodljive liste cijena, višestruke liste za kupnju, učinkovita interakcija na razini kupac - prodavač, segmentacija podataka i fleksibilan tijek procesa e-trgovine [5].

2.5. Sylius

Sylius se pojavio 2010. kao online trgovina za prodaju kozmetike koja je 2011. implementirana na Symfony PHP razvojnom okruženju u obliku kolekcije komponenti e-trgovine. U 2017. Sylius je pustio prvu stabilnu verziju svoje platforme za razvoj e-trgovina koju je razvio Sylius tim na čelu s Pawełom Jędrzejewskim. Ova platforma otvorenog koda sastoji se od osnovnih komponenti i Symfony paketa *(*engl. *bundle)*. Svaki paket nudi određene funkcionalnosti koje je moguće izmijeniti, ali isto je tako moguće i dodati novi paket kao ekstenziju sustava koji će pokriti potrebne funkcionalnosti. Sylius je modularan i lako proširiv sustav koji se temelji na popularnim konceptima dizajna, odnosno teži kvalitetnom kodu i korištenju najboljih praksi tijekom razvoja softvera. Glavne funkcionalnosti koje Sylius pokriva su prodaja na više kanala s različitim valutama i jezicima, katalog proizvoda s opcijama, atributima i kategorijama, upravljanje porezima, upravljanje korisničkim računima, logika promocija, kupona, košarice, narudžbi, isplate i dostave. Sylius koristi nekoliko popularnih tehnologija za razvoj softvera (Sl.2.2.), a neki od njih su:

- Symfony
- Doctrine

- TWIG
- Behat
- phpspec

Sl. 2.2. Tehnologije koje koristi Sylius

Sylius je izrađen u Symfony PHP razvojnom okruženju. Doctrine se koristi kao ORM (*Object relational mapping*) apstrakcija baze, a TWIG se koristi kao mehanizam za upravljanje *template*-ima. Behat i phpspec su tehnologije korištene u svrhu testiranja i BDD-a (*behaviour driven development*) [6].

2.6. Izazovi razvoja e-trgovina pomoću Sylius-a

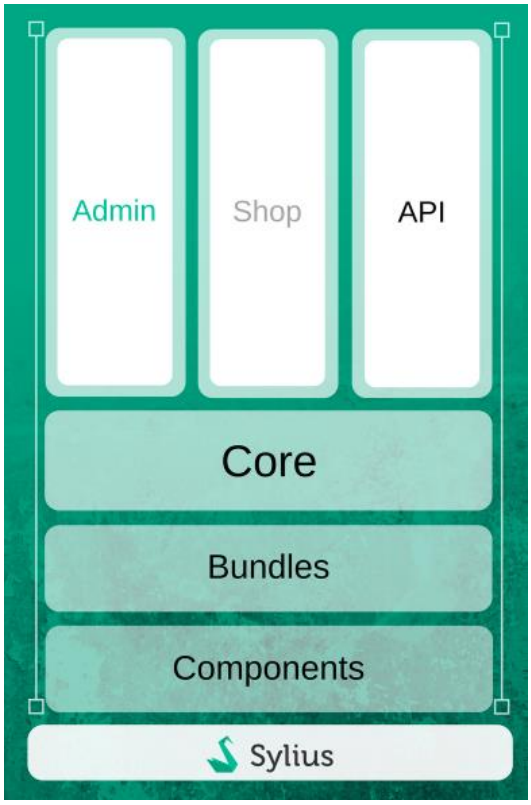
Sylius kao relativno mlada platforma nudi modularno i lako proširivo rješenje visokih performansi koje je primjereno za implementaciju manjih e-trgovina. Modularnost, proširivost i visoke performanse karakteristike su sustava koje su proizišle iz korištenja najnovijih obrazaca dizajna i najboljih praksi kodiranja tijekom implementacije komponenti Sylius-a. Budući da je Sylius mlađa platforma, tijekom razvoja e-trgovine uočljiv je nedostatak tema, ekstenzija i priručnika za razvoj s obzirom na njene zrelije konkurente. Upravo ti nedostatci predstavljaju najveće izazove tijekom razvoja jer je potrebno razvojem vlastitih ekstenzija ostvariti funkcionalnosti koje nedostaju. Funkcionalnosti koje će biti ostvarene u obliku ekstenzija i čiji će razvoj biti opisan u ovom radu su:

- lista želja
- kreiranje i primjena promocija izvan košarice

3. PLANIRANJE DIZAJNA SUSTAVA UNUTAR POSTOJEĆE ARHITEKTURE SYLIUS PLATFORME

Prilikom planiranja dizajna sustava potrebno je detaljno definirati zahtjeve na sustav te proučiti mogućnosti Sylius-a kako bi se utvrdila ostvarivost traženih funkcionalnosti. Potrebno je utvrditi za koje od tih funkcionalnosti postoje rješenja unutar Sylius-a te ukoliko ne postoje potrebno je osmisliti dizajn funkcionalnosti unutar novih ekstenzija. Unutar idućih poglavlja opisana je arhitektura Sylius-a te procesi definiranja zahtjeva i dizajna funkcionalnosti.

3.1. Arhitektura Sylius-a



Sl. 3.1. Arhitektura Sylius sustava

Sastavne dijelove Sylius arhitekture predstavljaju komponente, paketi i jezgra sustava na kojima se temelji viši sloj kojeg sačinjavaju admin sučelje, e-trgovina i API (Sl.3.1.). Komponente su samostalni dijelovi Sylius sustava kao što je to komponenta za oporezivanje. Ona ne mora biti implementirana, ali može na način da objekti na kojima se želi primijeniti oporezivanje implementiraju sučelje za oporezivanje TaxationInterface. Pakete koriste Symfony programeri koji žele implementirati određene komponente unutar paketa svog sustava kako bi s minimalnom konfiguracijom mogli koristiti gotove funkcionalnosti Sylius-a. Jezgra sustava povezuje sve komponente u funkcionalnu web trgovinu. Admin sučelje predstavlja sam AdminBundle i u njega su uključene funkcionalnosti administracije sustava. Sučelje web trgovine predstavlja

ShopBundle s funkcionalnošću standardnog B2C sustava. Sylius API koristi REST pristup i može koristiti kontrolere aplikacije budući da su neovisni o formatu [7].

3.2. Definiranje zahtjeva na sustav

Izgled postavljenih zahtjeva na sustav:

1. Web trgovina mora biti na hrvatskom i engleskom jeziku
2. Potrebno je unijeti cjelokupnu strukturu kategorija s trenutne stranice
3. Nakon što su prenesene kategorije, potrebno je prenijeti i proizvode.
4. Podešavanje plaćanja pouzećem i fiksna naplata dostave
5. Postavljanje liste želja. U Syliusu lista želja nije dio standardne funkcionalnosti. Svaki registrirani korisnik treba imati mogućnost dodavanja proizvoda u svoju listu želja koja će biti spremljena. Sljedeći put kad se prijavi, korisnik može vidjeti svoju listu želja s proizvodima koji su u tom trenutku dostupni. Ukoliko korisnik prvi dan doda jedan proizvod na listu želja. Drugi dan se taj proizvod povuče iz ponude. Treći dan korisnik ne bi trebao vidjeti taj proizvod na listi želja.
6. Implementacija promocija izvan košarice po uzoru na Magento. Promocije izvan košarice daju mogućnost kreiranja pravila koja će aktivirati određeni popust, ako se ta pravila ispune. Promocije izvan košarice ne upotrebljavaju kupon kodove jer se okidaju prije nego je proizvod stavljen u košaricu. Bitno je za ovu funkcionalnost da ima vremenski interval kako bi se mogle isplanirati prodajne akcije. Primjeri promocija izvan košarice:

- Od 23. do 31. listopada 2017, sve kape imaju 15% popusta.
- Na Halloween 31. listopada 2017, svi narančasti i crveni proizvodi imaju 20 kuna popusta.

Prvo je potrebno utvrditi koje od traženih funkcionalnosti pokriva Sylius, ukoliko Sylius ima vlastita rješenja moguće je ostvariti zahtjeve jednostavnom konfiguracijom trgovine iz admin sučelja. Ukoliko neke funkcionalnosti nisu uključene u Sylius, potrebno je osmisliti dizajn tih funkcionalnosti unutar ekstenzija. Podjela traženih funkcionalnosti nalazi se u tablici 3.1.

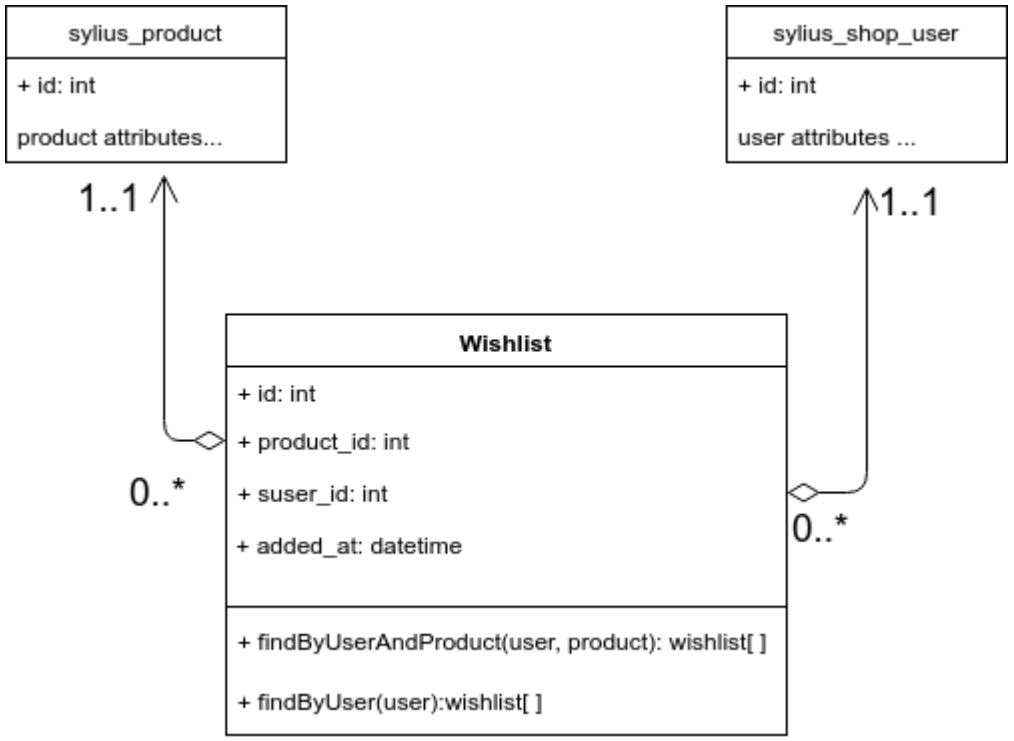
Tab. 3.1. Podjela traženih funkcionalnosti s obzirom na način realizacije

Funkcionalnosti ostvarene unutar Sylius-a	Funkcionalnosti koje je potrebno ostvariti unutar ekstenzija
- višejezičnost - dodavanje kategorija - dodavanje proizvoda i varijanti proizvoda	- lista želja - dodavanje i primjena promocija izvan košarice

- postavljanje načina naplate i dostave	

3.3. Dizajn željenih funkcionalnosti unutar ekstenzija

Realizacija ekstenzija uključuje kreiranje optimalnog dizajna entiteta i relacija među entitetima s obzirom na postavljene zahtjeve. Ekstenzija dodavanja u listu želja zahtjeva entitet koji će predstavljati stavku iz liste želja te koji će povezati proizvod stavljen u listu i korisnika koji ga je stavio u listu. Dizajn ovakvog Wishlist entiteta prikazan je na slici 3.2. pomoću UML dijagrama. Svaka stavka liste želja ima isključivo jednog korisnika te sadrži samo jedan proizvod. Atributi *suser_id* i *product_id* predstavljaju relaciju između jedne stavke liste želja, jednog korisnika i proizvoda. Osim relacijskih atributa stavka sadrži vlastiti identifikacijski atribut (id) i vrijeme nastanka stavke (added_at). Budući da korisnik nema više lista želja, nije potrebno predstavljati listu želja s posebnim entitetom. Sve je ostvarivo s jednim entitetom koji sadrži stavke liste želja. Jedna stavka ima jednog korisnika, jedan proizvod i vrijeme nastanka te ukoliko je potrebno prikazati korisniku njegovu listu, moguće je dohvatiti sve stavke s istim korisničkim brojem (id).

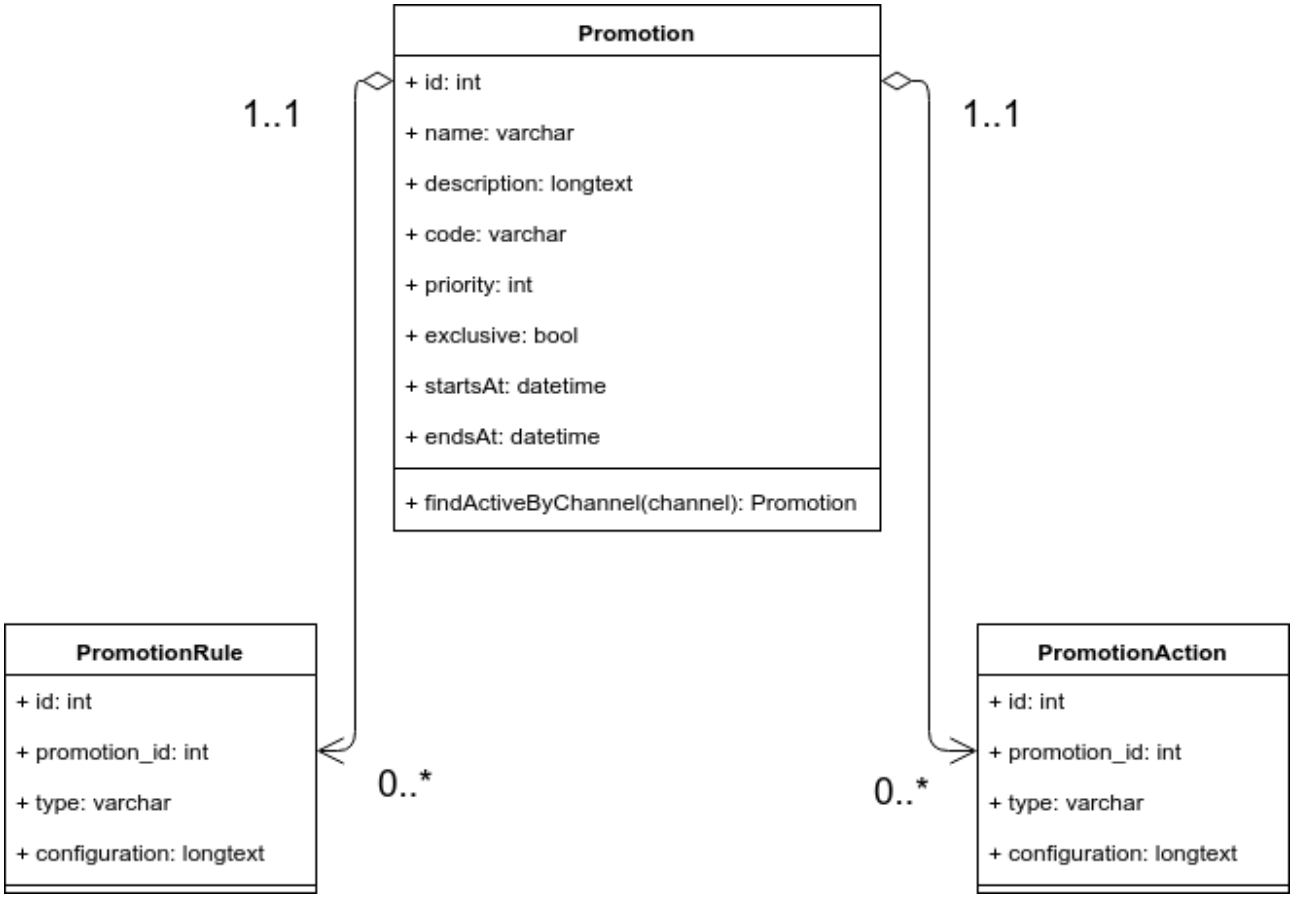


Sl. 3.2. UML dijagram Wishlist entiteta

Promocije izvan košarice realiziraju se pomoću tri entiteta:

- Promocija (tablica *Promotion*)
- Promocijsko pravilo (tablica *PromotionRule*)
- Promocijska akcija (tablica *PromotionAction*)

Entitet promocije sastoji se od osnovnih informacijama kao što su naziv, opis, trajanje promocije i prioriteti. Promocijsko pravilo i akcija nalaze se u relaciji s promocijom i sadrže informaciju o tipu i konfiguraciji pravila i akcije. Iz konfiguracije je vidljivo koja akcija se provodi na cijeni, ukoliko je pravilo zadano u konfiguraciji zadovoljeno u vrijeme trajanja promocije u skladu s prioritetom i ukoliko u isto vrijeme ne postoji promocija koja ima *exclusive* prioritet. Na slici 3.3. prikazan je UML dijagram s atributima i relacijama spomenutih entiteta. Promocije su vezane za promocijske akcije i pravila pomoću atributa *promotion_id* koji se nalazi unutar obje tablice. Jedna promocija može imati više pravila i više akcija.



Sl. 3.3. UML dijagram entiteta *Promotion, PromotionRule i PromotionAction*

4. IMPLEMENTACIJA E-TRGOVINE

Implementacija e-trgovine izvršena je korištenjem tehnologija i alata opisanih u idućim poglavljima. Implementacija je opisana kroz proces kreiranja novih ekstenzija te opis strukture i ključnih dijelova koda ekstenzija.

4.1. Korištene tehnologije i alati

Prilikom izrade e-trgovine, korištena je već opisana Sylius platforma koja se temelji na Symfony razvojnom okruženju pisanom u PHP programskom jeziku.

PHP je skriptni jezik otvorenog koda koji djeluje s poslužiteljske strane, a dizajniran je za web. Većina web-a danas pogonjena je PHP-om [8], a trenutna verzija i verzija korištena za izradu aplikacije je PHP 7.

Symfony je vodeće PHP razvojno okruženje za izgradnju web stranica i aplikacija, koje se sastoji od Symfony komponenti i paketa. Symfony se temelji na MVC (Model-View-Controller) dizajnu što znači da se sastoji od modela koji predstavljaju određene entitete aplikacije, pogleda koji predstavljaju način na koji je korisniku prikazana aplikacija i kontrolera koji sadrže većinu programske logike te izvršavaju određenu akciju s obzirom na odabranu rutu. Prikaz aplikacije sastoji se od *template*-a koji su realizirani pomoću Twig *template* jezika. Symfony koristi Doctrine za rad s entitetima i bazom [9].

Doctrine je ORM *(Object Relational Mapper*) i DBAL (*Database Abstraction Layer*) sustav koji nam omogućuje dodatni sloj apstrakcije prilikom komunikacije s bazom. Doctrine mapira podatke unutar entiteta koji odvajaju logiku aplikacije od podataka spremljenih u bazi, odnosno podatcima u bazi se pristupa preko PHP objekata što doprinosi objektno orijentiranom pristupu izrade aplikacije [10].

4.2. Kreiranje novih ekstenzija unutar Sylius-a i njihova struktura

Sylius ekstenzije razvijaju se unutar *plugin*-a koji su poput uobičajenih paketa s vlastitim testnim okruženjem. Najjednostavniji način za kreiranje novog *plugina*-a je pomoću Composera s naredbom:

```
1. composer create-project sylius/plugin-skeleton SyliusMyFirstPlugin
```

Pomoću ove naredbe kreira se kostur *plugin*-a.

Glavni elementi plugin-a su direktoriji tests, features i src. Direktorij tests sadrži Symfony aplikaciju za testiranje ekstenzije, features direktorij sadrži Sylius scenarije, a src direktorij

sadrži kontrolere, repozitorije, entitete, resurse, template i konfiguracijske datoteke, odnosno glavninu programskog koda.

U slučaju razvoja *plugin*-a unutar aplikacije potrebno je učiniti sljedeće:

- registrirati *plugin* unutar app/AppKernel.php datoteke [11]

```
pr. app/AppKernel.php
public function registerBundles()
{
  $bundles = [
    // ...
    new \Acme\SyliusExamplePlugin\AcmeSyliusExamplePlugin(),
    // ...
  ];
}
```

- registrirati rute unutar app/config/routing.yml datoteke:

```
pr. app/config/routing.yml
acme_sylius_example_shop:
    resource: "@AcmeSyliusExamplePlugin/Resources/config/app/shop_routing.yml"
```

- urediti *autoload* komponentu composer.json datoteke unutar korijenskog direktorija aplikacije, odnosno dodati *namespace* *plugin-a* i njegovu putanju. AcmeSyliusExamplePlugin je naziv plugin-a koji je moguće izmijeniti u željeni naziv pri čemu treba pripaziti na konvenciju pisanja imena, ali je također potrebno promijeniti i nazive na već spomenutim mjestima na kojima se pojavljuje naziv *plugin*-a.

4.3. Konfiguracija postavki sustava

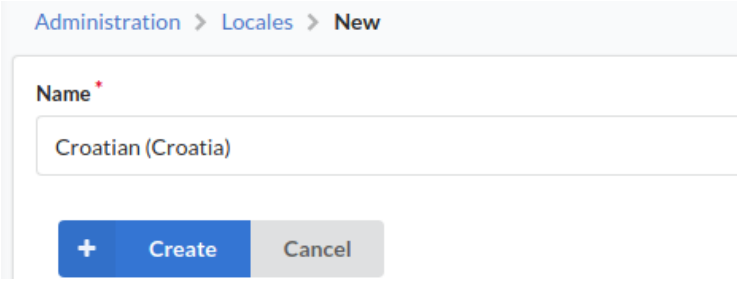
Određeni zahtjevi su ostvarivi unutar Sylius funkcionalnosti te ih je potrebno postaviti konfiguracijom trgovine unutar admin sučelja. Iduće funkcionalnosti moguće je postaviti unutar admin sučelja:

- višejezičnost
- unos kategorija
- unos proizvoda i varijanti proizvoda
- postavljanje načina plaćanja i dostave

Višejezičnost se postiže kreiranjem nove *Locale* opcije unutar admin izbornika: *Locale* -> *Create* i odabirom *Locale* opcije iz izbornika na slici 4.1. Dodavanje novog ili izmjena postojećeg prijevoda moguća je unutar direktorija app\Resources\translations. Izgled prijevoda unutar yml datoteke messages.hr.yml kojemu se pristupa s pomoću sylius.form.address.street ključa:

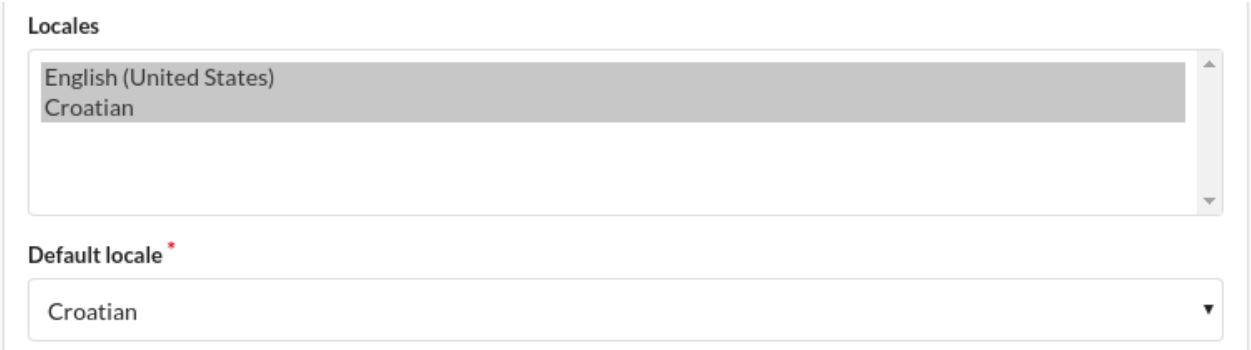
```
1. sylius:
2.     form:
```

```
3.          address:
4.              street: 'Ulica i kućni broj'
```



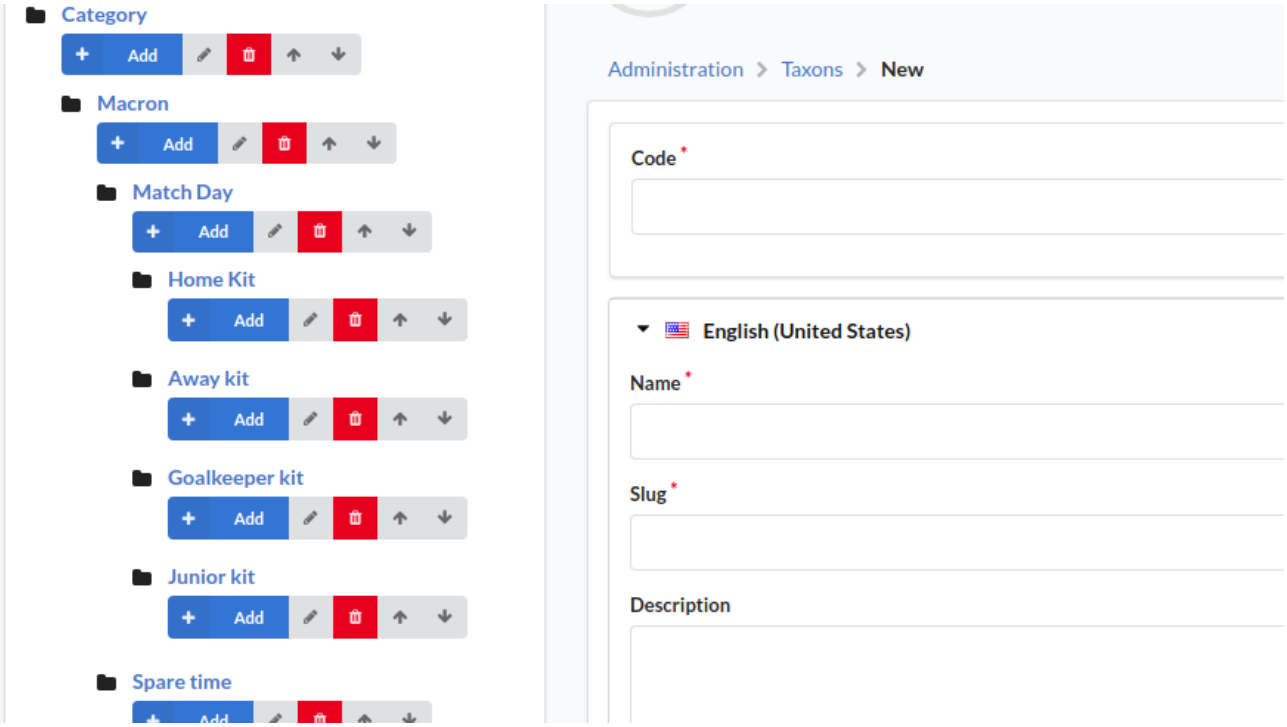
Sl. 4.1. Postavljanje nove *Locale* opcije unutar admin sučelja

Nakon toga potrebno je unutar izbornika odabrati opciju *Channels* te urediti kanal koji se koristi i unutar njega odabrati jezike koje se želi koristiti unutar trgovine. Prikaz izbora nalazi se na slici 4.2.



Sl. 4.2. Prikaz izbora jezika unutar sučelja za uređivanje kanala trgovine

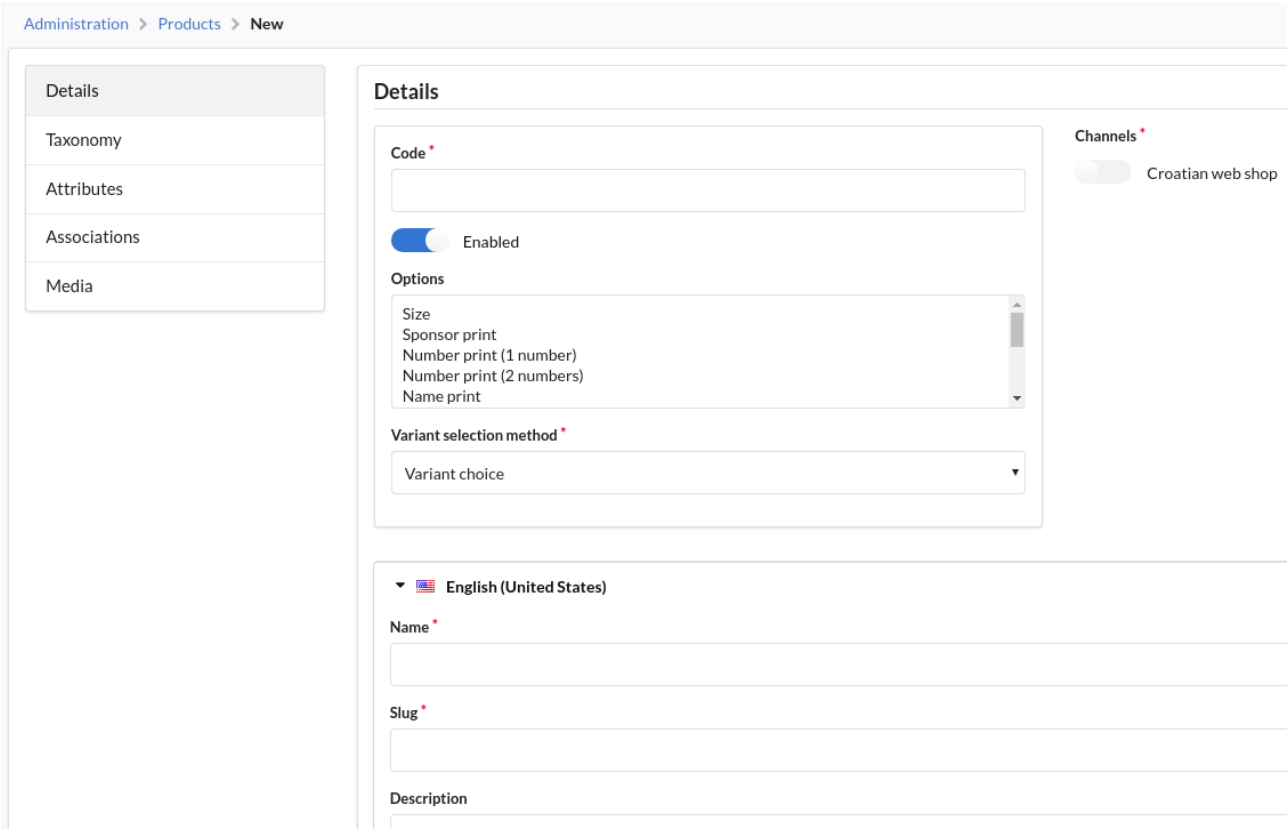
Kategorije se dodaju unutar *Taxons* opcije admin izbornika kao što je prikazano na slici 4.3. Dovoljno je unijeti potrebne podatke o kategoriji kao što su kod, naziv i opis te nakon toga odabrati opciju *Create*. Na lijevoj strani vidljiva je struktura kategorija i mogućnost dodavanja, brisanja i premještanja kategorije unutar određenog dijela strukture.



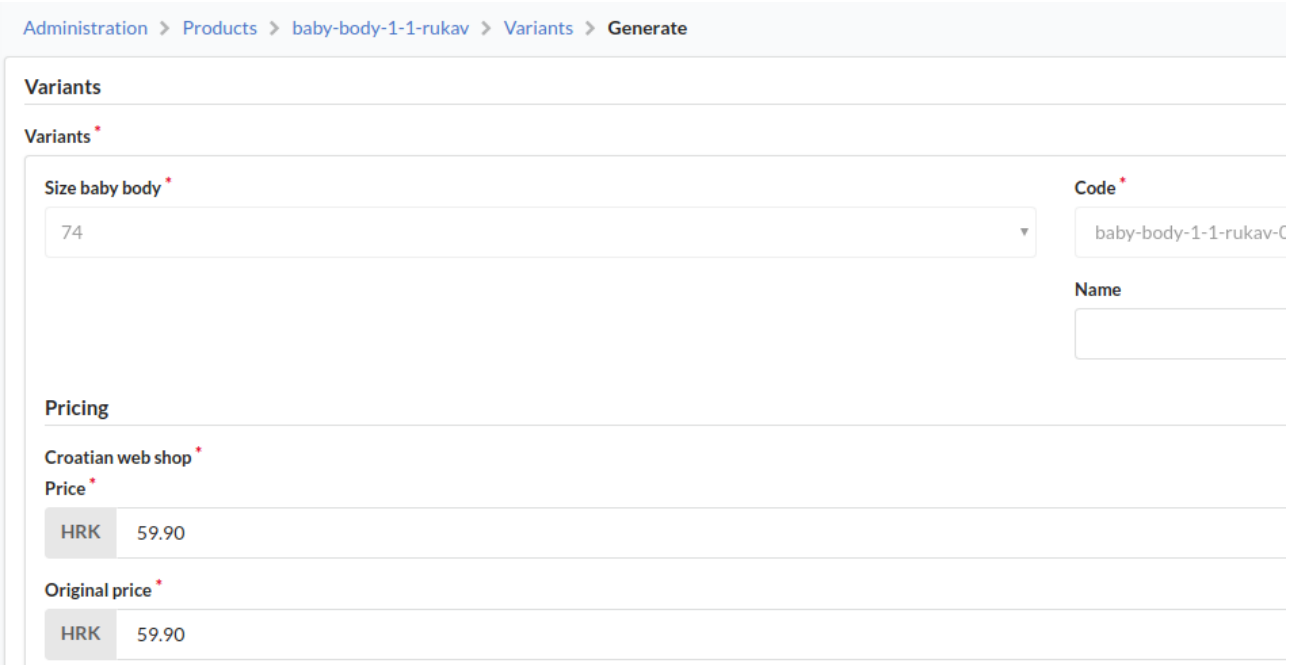
Sl. 4.3. Prikaz sučelja za dodavanje nove kategorije

Unos proizvoda odvija se unutar *Products* dijela admin izbornika gdje su prikazani trenutni proizvodi i ponuđena je opcija kreiranja novog proizvoda. Odabirom *Create* opcije i vrste proizvoda (jednostavni ili konfigurabilni) pojavljuje se forma za unos podataka o proizvodu. Unosom podataka u formu sa slike 4.4. i odabirom *Create* opcije, kreira se novi proizvod.

Unos varijanti proizvoda odvija se odabirom *Manage variant* opcije unutar liste proizvoda. Varijante je moguće urediti unosom cijena za različite kombinacije opcija kao što je to prikazano na slici 4.5.

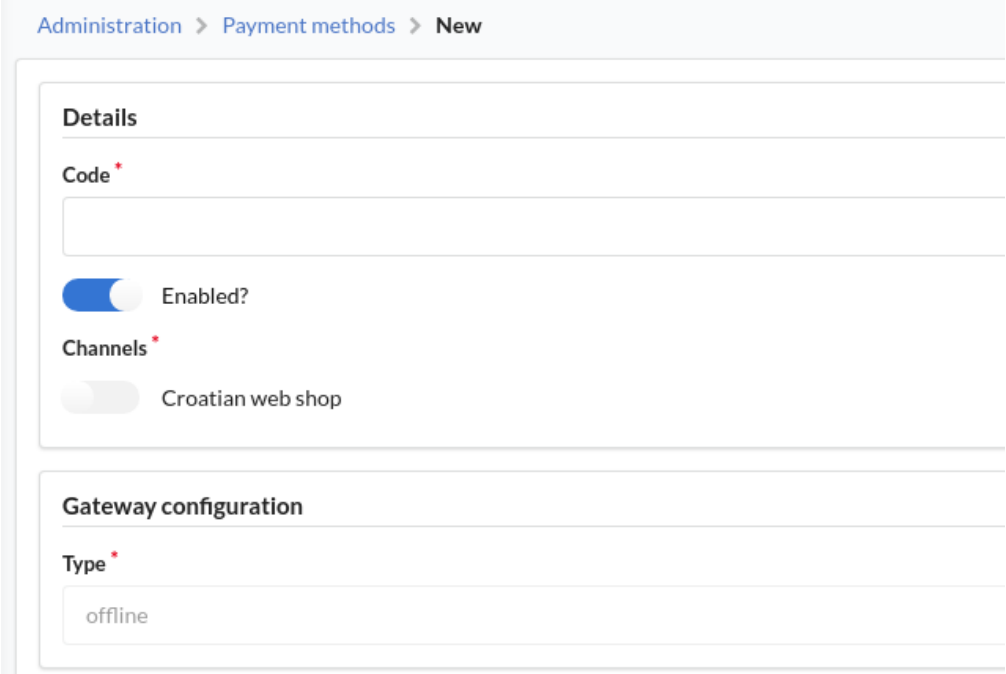


Sl. 4.4. Forma za dodavanje novog proizvoda



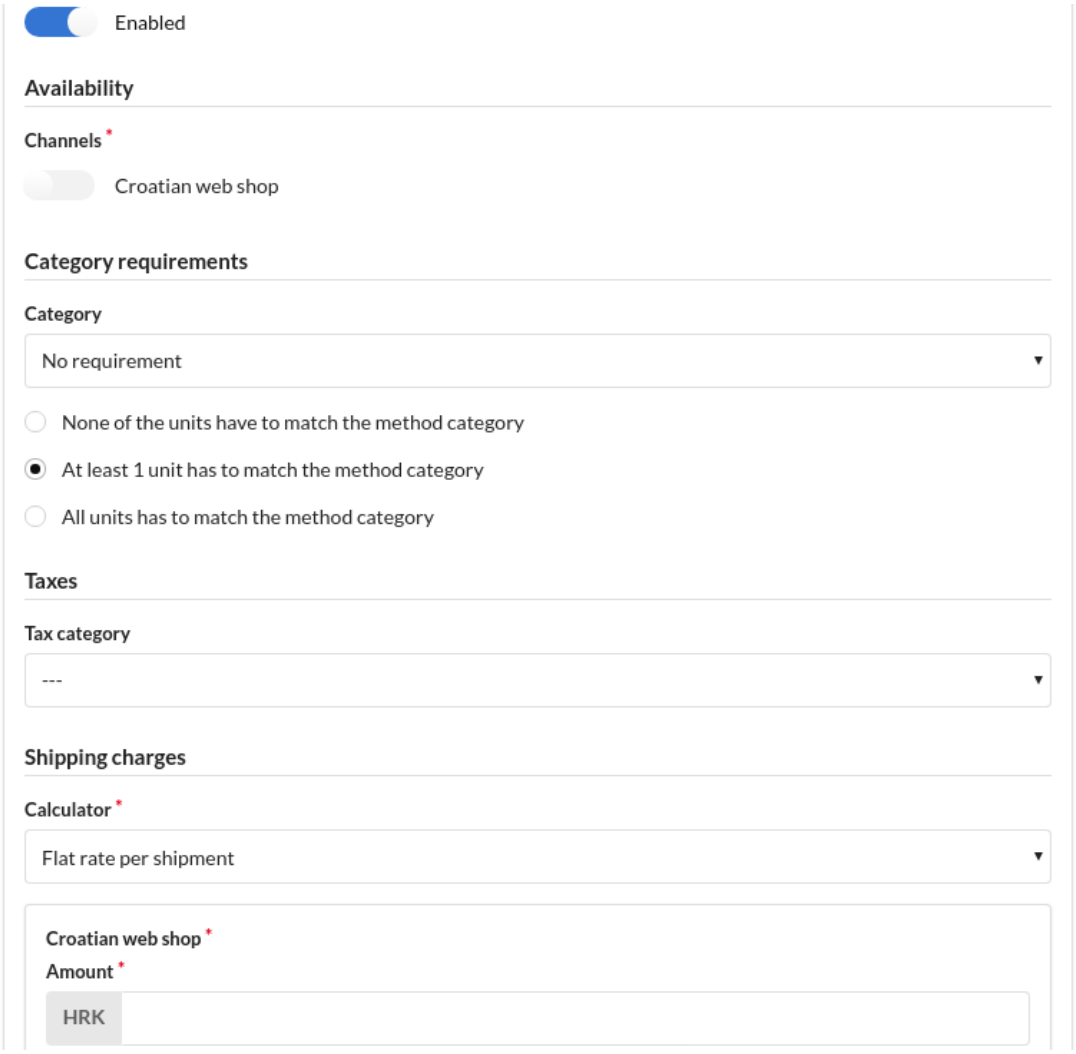
Sl. 4.5. Forma za uređivanje varijanti proizvoda

Načini plaćanja uređuju se unutar *Payment methods* opcije admin izbornika gdje je moguće kreirati novu metodu određenog tipa (*Offline, Paypal express checkout, Stripe checkout*). Prikaz kreiranja nove metode tipa *Offline* nalazi se na slici 4.6.



Sl. 4.6. Dodavanje nove metode plaćanja

Metoda dostave dodaje se unutar *Shipping Methods* opcije admin izbornika. Odabirom *Create* opcije prikazuje se forma za unos informacija o metodi poput zone na koju se metoda primjenjuje, vrste naplate dostave (npr. plaćanje fiksne cijene), kanala, cijene i tako dalje (Sl.4.7.).



Sl. 4.7. Prikaz opcija podešavanja metode dostave

4.4. Opis ključnih dijelova koda ekstenzije za dodavanje u listu želja

Sljedeća poglavlja prikazuju ključne dijelove koda za realizaciju ekstenzije za dodavanje u listu želja. Osim koda i opisa definirani su i obrasci dizajna koji se koriste.

4.4.1. Kreiranje modela

Kako bi se kreirao model potrebno je proći iduće korake [12]:

1. Registrirati *SensioGeneratorBundle* unutar app/AppKerenel.php datoteke kako bi se mogao izgenerirati entitet, odnosno da bi se mogla koristiti naredba *generate:doctrine:entity*.

2. Upisati u terminal iz korijenskog direktorija projekta:

```
1.  $ php bin/console generate:doctrine:entity
```

3. Upisati ime entiteta npr. Wishlist, odabrati format za mapiranje podataka (yml,xml...) i upisati atribute novog entiteta.

4. Provjeriti je li baza podataka ažurirana prije dodavanja novog entiteta:

```
$ php bin/console doctrine:migrations:diff
```

Unosom naredbe stvara se nova *migration* datoteka.

5. Da bi se baza ažurirala pomoću nove *migration* datoteke kreirane u prethodnom koraku, potrebno je upisati naredbu:

```
$ php bin/console doctrine:migrations:migrate
```

6. Otvoriti generiranu klasu modela i dodati da klasa implementira ResourceInterface.

```php
<?php

namespace PraktinchooneriSyliusWishlistPlugin\Entity;

use Sylius\Component\Resource\Model\ResourceInterface;

class Wishlist implements ResourceInterface
{
    // ...
}
```

7. Registrirati entitet, odnosno kreirati resources.yml datoteku, ako već ne postoji i dodati u nju sljedeće linije:

```yaml
# app/config/resources.yml
sylius_resource:
    resources:
        app.wishlist:
            driver: doctrine/orm # You can use also different driver here
            classes:
                model: PraktinchooneriSyliusWishlistPlugin\Entity\Wishlist
```

Nakon toga je potrebno registrirati app/config/resources.yml datoteku unutar app/config/config.yml datoteke kako bi bila učitana prilikom učitavanja postavki sustava.

```yaml
# app/config/config.yml
imports:
    - { resource: "resources.yml" }
```

8. Provjeriti je li sve dobro postavljeno:

```
$ php bin/console debug:container | grep wishlist
```

Za potrebe projekta kreira se entitet *Wishlist* koji sadrži *id* entiteta, vrijeme kreiranja, *id* proizvoda koji predstavlja strani ključ te relaciju na entitet *Product* i korisnički *id* koji predstavlja strani ključ te relaciju s entitetom *ShopUser*.

Sustav se mogao implementirati unutar 2 entiteta. Jednog koji predstavlja listu želja vezanu za korisnika i drugi koji predstavlja stavku liste želja koja je povezana s proizvodom, no budući da

se radi o jednostavnoj listi želja te jedan korisnik ima samo jednu listu, dovoljan je jedan entitet. Na ovaj je način također olakšano dohvaćanje korisnika i proizvoda određene liste želja.

4.4.2. Povezivanje entiteta pomoću stranih ključeva

Klasa entiteta mora sadržavati povezane entitete. U ovom slučaju svaka lista želja mora biti povezana s jednim korisnikom i proizvodom.

Relacije su definirane unutar: src/Resources/config/doctrine/Wishlist.orm.yml datoteke. Format yml je jedan od formata koji se može koristiti za mapiranje podataka.

Primjer povezivanja *Wishlist* entiteta s *Product* entitetom na relaciji više na jedan [13]:

```
manyToOne:
    product:
      targetEntity: Sylius\Component\Core\Model\Product
      mappedBy: Wishlist
      joinColumn:
        name: product_id
        referencedColumnName: id
        onDelete: CASCADE
```

Nakon izmjene potrebno je upisati naredbe kako bi se ažurirala baza podataka:

```
$ php bin/console doctrine:migrations:diff
$ php bin/console doctrine:migrations:migrate
```

Osim relacije s *Product* entitetom definirana je i više na jedan relacija s *ShopUser* entitetom koji predstavlja korisnika.

Unutar relacija definirano je *onDelete: CASCADE* ponašanje kako bi se pri brisanju proizvoda obrisala i stavka iz liste želja koja je vezana uz taj proizvod.

4.4.3. Klasa entiteta *Wishlist*

Klasa *Wishlist* je osnovna klasa ekstenzije koja predstavlja entitet *Wishlist*, odnosno stavku liste želja. Lokacija klase unutar *plugin*-a je src/Entity/ direktorij. Klasa implementira sučelje *WishlistInterface* koje proširuje *ResourceInterface* sučelje. *Wishlist* klasa sadržava metode za postavljanje i dohvaćanje varijabli te same varijable:

- int **id** - *Wishlist* id
- Datetime **added_at** - vrijeme kreiranja
- ProductInterface **product** - povezani proizvod
- ShopUserInterface **suser** - povezani korisnik

4.4.4. Akcija dodavanja u listu želja

Akcija je implementirana unutar *WishlistController* klase, a sastoji se od provjere korisnika, dohvaćanja proizvoda čiji *id* je proslijeđen akciji kontrolera, kreiranja nove stavke liste želja,

provjere sadržaja liste želja i dodavanja nove stavke. Ukoliko je prijavljen, korisnik će biti u mogućnosti izvršiti navedenu akciju, a u suprotnom će biti preusmjeren na stranicu za prijavu. Akcija kontrolera ima jedan parametar, a to je *product_id* pomoću kojeg je moguće dohvatiti iz repozitorija proizvoda proizvod koji se trenutno dodaje u listu želja. Akcija kontrolera izvršava se odabirom opcije *Dodaj u uži izbor* te se tada prosljeđuje i *product_id*. Ukoliko je proizvod uspješno dohvaćen, pretražuje se lista želja kako bi se utvrdilo postoji li već korisnik s tim proizvodom**.** Pretraga se izvršava izvođenjem funkcije novog repozitorija liste želja koja dohvaća stavku liste želja za zadanog korisnika i proizvod. Ukoliko korisnik nema taj proizvod u listi želja, kreira se nova stavka liste želja pomoću *factory*-ja liste želja te se nova stavka dodaje u repozitorij liste želja. Ukoliko stavka već postoji unutar korisnikove liste želja, samo će se prikazati korisnikova lista želja s već od prije dodanim proizvodima.

Glavne točke izrade su bile kreiranje:

- kontrolera
- *factory*-ja
- repozitorija

4.4.5. Kreiranje kontrolera

Kontroler se koristi za definiranje akcija za određene rute, a nalazi se unutar src/Controller/ direktorija te proširuje klasu Controller [14]. Rute kontrolera definirane su unutar yml datoteke src/Resources/config/app/shop_routing.yml.

Primjer registriranja rute:

```
1.  praktinchooneri_wishlist_add_item:
2.      path: /add-wishlist-item/{product_id}
3.      defaults:
4.          _controller: PraktinchooneriSyliusWishlistPlugin:Wishlist:addToWishlist
5.          name: ~
```

Ruta pod nazivom *praktinchooneri_wishlist_add_item* definirana je pomoću putanje /add-wishlist-item/{product_id}, kontrolera *WishlistController* i metode *addToWishlistAction*(*product_id*).

4.4.6. Kreiranje *factory*-ja

Factory je klasa koja implementira *factory* obrazac dizajna koji se koristi za kreiranje objekata klasa, nalazi se unutar *plugin* direktorija src/Factory/. Klasa *WishlistFactory* implementira sučelje *WishlistFactoryInterface* koje proširuje sučelje *FactoryInterface*. Klasa mora definirati funkciju *CreateNew*() jer je ona dio *FactoryInterface* sučelja. Funkcija *CreateNew*() vraća objekt klase *Wishlist* te se koristi unutar funkcije *createForShopUserandProduct($shopUser,$product)*

kojoj se kao parametri predaju trenutni korisnik i proizvod. Parametri se postavljaju korištenjem metoda za postavljanje koje su definirane unutar *Wishlist* klase**.** Funkcija vraća novi objekt klase *Wishlist* s postavljenim korisnikom i proizvodom. Na taj način je kreiran novi objekt klase *Wishlist* s određenim korisnikom i proizvodom.

Factory metoda:

```
1.  public function createForShopUserandProduct(ShopUserInterface $shopUser, ProductInterface $product): WishlistInt
    erface
2.  {
3.      $wishlist = $this->decoratedFactory->createNew();
4.      $wishlist->setShopUser($shopUser);
5.      $wishlist->setProduct($product);
6.      $wishlist->setAddedAt(new \DateTime('now'));
7.      return $wishlist;
8.  }
```

Factory klasu je potrebno registrirati unutar src/Resources/config/app/resources.yml datoteke i unutar /src/Resources/config/services.yml datoteke [15].

Primjeri registracije:

```
1.  #custom service.yml
2.  services:
3.    praktinchooneri.basefactory.wishlist:
4.          class: '%praktinchooneri.basefactory.wishlist.class%'
5.          arguments:
6.              - '%praktinchooneri_wishlist.model.wishlist.class%'
7.    praktinchooneri_wishlist.factory.wishlist:
8.        class: Praktinchooneri\SyliusWishlistPlugin\Factory\WishlistFactory
9.        arguments:
10.           - '@praktinchooneri.basefactory.wishlist'
```

Varijable kao *%praktinchooneri.basefactory.wishlist.class%* definirane su unutar src/Resources/config/app/parameters.yml datoteke.

Primjer definicije parametara:

```
1.  # custom parameters.yml parameters:
2.  praktinchooneri_wishlist.model.wishlist.class: Praktinchooneri\SyliusWishlistPlugin\Entity\Wishlist
3.  praktinchooneri_wishlist.factory.wishlist.class: Praktinchooneri\SyliusWishlistPlugin\Factory\WishlistFactory
4.  praktinchooneri.basefactory.wishlist.class: Sylius\Component\Resource\Factory\Factory
```

Također je potrebno je učitati konfiguracijske datoteke ekstenzije unutar src/DependencyInjection/PraktinchooneriSyliusWishlistExtension.php datoteke poput service.yml datoteke unutar koje se definiraju servisi poput *factory* servisa.

Primjer registracije:

```
1.  $loader = new Loader\YamlFileLoader($container, new FileLocator(__DIR__.'/../Resources/config'));
2.  $loader->load('services.yml');
```

4.4.7. Kreiranje repozitorija

Repozitorij je implementacija *Repository* obrasca dizajna koji se koristi za definiranje učestalih funkcija za komuniciranje s bazom poput dohvaćanja stavki iz liste želja. Klasa *WishlistRepository* nalazi se unutar *plugin* direktorija src/Repository te se registrira unutar resources.yml i services.yml datoteka [16].

Nakon registracije repozitorij je moguće dohvatiti pomoću *get* metode:

```
1.  $repository = $this->get('praktinchooneri.repository.wishlist');
```

Novi repozitorij mora proširiti klasu ORM\EntityRepository kao bi mogao koristiti metode poput *add* i *remove* metoda za manipulaciju sadržaja repozitorija. Unutar repozitorija se kreiraju potrebne metode za dohvaćanje *Wishlist* objekata. U ovom slučaju kreiran je *WishlistRepository* s metodama *findByUserAndProduct(ShopUser $user, Product $product)* i *findbyUser(ShopUser $user)* koje se koriste za pronalaženje stavke liste želja određenog korisnika s određenim proizvodom i pronalaženje više stavki liste želja jednog korisnika.

Izgled metode repozitorija:

```
1.  public function findByUserAndProduct(ShopUser $user, Product $product)
2.  {
3.      return $this->createQueryBuilder('w')
4.          ->andWhere('w.suser = :user')
5.          ->andWhere('w.product = :product')
6.          ->setParameter('user', $user)
7.          ->setParameter('product', $product)
8.          ->getQuery()
9.          ->getOneOrNullResult()
10.         ;
11. }
```

4.4.8. *Add to Wishlist* opcija

Opcija *Add to Wishlist* dodana je unutar prikaza proizvoda. Kreirana je nova *Twig* datoteka koja se dodaje u osnovnu *Twig* datoteku za prikaz proizvoda unutar *ShopBundle*-a. Izgled novog *Twig*-a s poveznicom za dodavanje u uži izbor koja se prikazuje samo ukoliko je korisnik prijavljen što se provjerava pomoću *is_granted('ROLE_USER')* metode:

```
1.  {% if is_granted('ROLE_USER') %}
2.  <a href="{{ path('praktinchooneri_wishlist_add_item', {'product_id': product.id}) }}" class="ui huge primary fluid lab
    eled icon button"> {{ 'sylius.ui.add_to_wishlist'|trans }}</a href>
3.  {% endif %}
```

Akcija kontrolera koja je opisana u ranijim poglavljima te se odvija odabirom opcije:

```
1.  public function addToWishlistAction(?int $product_id)
2.  {
3.      // if user is logged in:
4.      //get current user
```

```
5.      if ($user = $this->getUser()) {
6.          //fetch product repository
7.          $productrepository = $this->get('sylius.repository.product');
8.          //get specific product by id
9.          $product = $productrepository->findOneBy(['id' => $product_id]);
10.         //check existence
11.         if ($product != null) {
12.             //fetch wishlist repository
13.             $repository = $this->get('praktinchooneri.repository.wishlist');
14.             //check existance in wishlist
15.             $wishlistItems = $repository->findByUserAndProduct($user, $product);
16.             //if item isn't in wishlist
17.             if ($wishlistItems == null) {
18.                 //fetch wishlist factory
19.                 $factory = $this->get('praktinchooneri_wishlist.factory.wishlist');
20.                 /** @var WishlistInterface $wishlist
21.                  * create user with specific product and user
22.                  * **/
23.                 $wishlist = $factory->createForShopUserandProduct($user, $product);
24.                 //add to database
25.                 $repository->add($wishlist);
26.             }
27.             //redirect -> show wishlist
28.             return $this->redirectToRoute('praktinchooneri_wishlist_show');
29.         } else {
30.             //product missing trow exception
31.             throw new NotFoundHttpException("Page not found");
32.         }
33.     } else {
34.         //user is not logged in
35.         return $this->redirectToRoute('sylius_shop_login');
36.     }
37. }
```

4.4.9 Prikaz liste želja

Prvo se radi provjera se je li korisnik prijavljen unutar kontrolera. Ukoliko je korisnik prijavljen, dohvaća se repozitorij liste želja, pomoću metode repozitorija findByUser(ShopUser $user) dohvaćaju se sve stavke iz liste želja trenutno prijavljenog korisnika. Stavke iz liste želja prosljeđuju se iz kontrolera u *Twig* datoteku za prikaz liste želja. Unutar *Twig* datoteke se s pomoću metode za pristup *Wishlist* klase dohvaćaju proizvodi svake stavke liste želja te se prosljeđuju dalje na klasičan *Twig* za prikaz liste proizvoda. Unutar *Twig* datoteke se također provjerava je li proizvod omogućen kako se ne bi prikazali onemogućeni proizvodi.

Prikaz metode kontrolera:

```
1.  public function showWishlistAction()
2.  {
3.      //check user
4.      if ($user = $this->getUser()) {
5.          //fetch wishlist repository
6.          $repository = $this->get('praktinchooneri.repository.wishlist');
7.          //get user's wishlistItems
8.          $wishlistItems = $repository->findByUser($user);
9.          //show wishlist
10.         return $this-
    >render('@PraktinchooneriSyliusWishlistPlugin/wishlist_show.html.twig', array('wishlist' => $wishlistItems));
11.     } else {
12.         return $this->redirectToRoute('sylius_shop_login');
```

```
    }
}
```

Prikaz *Twig* datoteke:

```
{% for wish in wishlist %}
    {% set product = wish.getProduct %}
        {% if product.enabled == 1 %}
```

4.4.10 Opcija brisanja iz užeg izbora

Unutar *Wishlist* kontrolera definirana je akcija brisanja koja kao parametar uzima product_id. Brisanje započinje provjerom korisnika. Ukoliko je korisnik prijavljen, dohvaća se repozitorij proizvoda i traži se proizvod po *id*-u. Vrši se provjera je li *id* valjan, odnosno postoji li takav proizvod u bazi. Ukoliko proizvod postoji, dohvaća se repozitorij liste želja i pretražuje se pomoću metode repozitorija, postoji li stavka sa zadanim korisnikom i proizvodom, ukoliko postoji vrši se brisanje pomoću *remove*() metode repozitorija.

Prikaz metode za brisanje:

```
public function deleteFromWishlistAction(?int $product_id)
{
    //check user
    if ($user = $this->getUser()) {
        //fetch product repository
        $productrepository = $this->get('sylius.repository.product');
        //get specific product by id
        $product = $productrepository->findOneBy(['id' => $product_id]);
        //check existence
        if ($product != null) {
            //fetch wishlist repository
            $repository = $this->get('praktinchooneri.repository.wishlist');
            //check existence in wishlist
            //@var WishlistInterface $wishlistItems
            $wishlistItem = $repository->findByUserAndProduct($user, $product);
            if ($wishlistItem != null) {
                //remove from database
                $repository->remove($wishlistItem);
            } else {
                //wishlist item missing, trow exception
                throw new NotFoundHttpException("Page not found");
            }
        } else {
            //product missing, trow exception
            throw new NotFoundHttpException("Page not found");
        }
        //redirect -> show wishlist
        return $this->redirectToRoute('praktinchooneri_wishlist_show');
    } else {
        return $this->redirectToRoute('sylius_shop_login');
    }
}
```

4.5 . Ekstenzija promocija izvan košarice

Sustav promocija izvan košarice temelji se na entitetima promocije, promocijskog pravila i promocijske akcije. Sustav tri entiteta je realiziran unutar tri tablice u bazi podataka, tri klase entiteta i sučelja entiteta, *Factory* klasa i *Repository* klasa. *Repository* klasa promocije sadrži metode pomoću kojih se dohvaćaju trenutno aktivne promocije trenutnog kanala poredane po prioritetu.

```
1.  //find promotions in certain channel and filter by active
2.  public function findActiveByChannel(ChannelInterface $channel): array
3.  {
4.      return $this->filterByActive($this->createQueryBuilder('o'))
5.          ->andWhere(':channel MEMBER OF o.channels')
6.          ->setParameter('channel', $channel)
7.          ->addOrderBy('o.priority', 'desc')
8.          ->getQuery()
9.          ->getResult();
10. }
11. // filter by active, check date intervals
12. protected function filterByActive(QueryBuilder $queryBuilder, ?\DateTimeInterface $date = null): QueryBuilder
13. {
14.     return $queryBuilder
15.         ->andWhere('o.startsAt IS NULL OR o.startsAt < :date')
16.         ->andWhere('o.endsAt IS NULL OR o.endsAt > :date')
17.         ->setParameter('date', $date ?: new \DateTime());
18. }
```

Kreiranje i funkcionalnost *Entity, Factory, Repository i Controller* klasa već su opisani unutar opisa ekstenzije za dodavanje u listu želja, stoga su unutar ovog poglavlja opisani dijelova koda specifični za izradu logike promocija izvan košarice.

4.5.1. Dodavanje opcije promocije unutar admin sučelja

Sustav promocija izvan košarice podrazumijeva opciju dodavanja promocija unutar admin sučelja. Kako bi se dodala opcija unutar admin izbornika, potrebno je unutar *AdminMenuListener* klase i funkcije *addAdminMenuItems* proslijediti objekt klase *MenuBuilderEvent* pomoću kojeg će se dohvatiti sam element izbornika i njegov *marketing* potomak. *Marketing* potomku dodan je novi potomak *catalog_promotions* te su mu pridruženi ruta i naziv.

Primjer dodavanja opcije u izbornik:

```
1.  final class AdminMenuListener
2.  {
3.      /**
4.       * @param MenuBuilderEvent $event
5.       */
6.      // ad Catalog promotions to admin menu
7.      public function addAdminMenuItems(MenuBuilderEvent $event): void
8.      {
```

```
        $marketing = $event->getMenu()->getChild('marketing');
        $marketing
            ->addChild('catalog_promotions', ['route' => 'app_admin_catalog_promotion_index'])
            ->setLabel('Catalog Promotions');
    }
}
```

Nakon toga *Listener* klasa se unutar service.yml datoteke registrira za slušanje događaja *sylius.menu.admin.main*.

Nakon dodavanja opcije u izbornik, potrebno je izraditi početni prikaz entiteta promocije korištenjem *grid* komponente. *Grid* komponenta je yml datoteka koja omogućuje prikaz osnovnih informacija o promociji kao što su naziv klase entiteta, kod, naziv, prioriteti, filteri te opcije brisanja, uređivanja i dodavanja novih promocija.

Prikaz dijela *grid* komponente:

```
sylius_grid:
    grids:
        app_admin_promotion:
            driver:
                name: doctrine/orm
                options:
                  class: Praktinchooneri\SyliusCatalogPriceRulesPlugin\Entity\Promotion
            sorting:
                priority: desc
            fields:
                priority:
                    type: twig
                    label: sylius.ui.priority
                    sortable: ~
                    options:
                        template: "@SyliusUi/Grid/Field/position.html.twig"
```

Opcije uređivanja i dodavanja novih promocija ostvarene su pomoću *form* komponenti. Forme su klase koje nasljeđuju klasu *AbstractResourceType* te se definiraju unutar Form/Type direktorija. Funkcija *buildForm* sadrži *builder* objekt pomoću kojeg se dodaju polja forme poput naziva, opisa i isključivosti. Za svako polje, moguće je definirati opcije poput naziva, klase koja određuje tip polja i naziva koji će se prikazati adminu. Prikaz metode *buildForm*:

```
public function buildForm(FormBuilderInterface $builder, array $options): void
{
    $builder
        ->add('name', TextType::class, [
            'label' => 'sylius.form.promotion.name',
        ])
        ->add('description', TextareaType::class, [
            'label' => 'sylius.form.promotion.description',
            'required' => false,
        ])
        ->add('exclusive', CheckboxType::class, [
            'label' => 'sylius.form.promotion.exclusive',
        ])
```

Najvažnija polja koja se dodaju su polja za prikaz odabira promocijskog pravila i akcije koja su definirana pomoću *CollectionType* klasa koje predstavljaju kolekciju pravila i akcija te nasljeđuju klasu *AbstractConfigurationCollectionType*. Prikaz dodavanja *PromotionRuleCollectionType* klase unutar forme:

```
->add('rules', PromotionRuleCollectionType::class, [
    'label' => 'sylius.form.promotion.rules',
    'button_add_label' => 'sylius.form.promotion.add_rule',
])
->add('actions', PromotionActionCollectionType::class, [
    'label' => 'sylius.form.promotion.actions',
    'button_add_label' => 'sylius.form.promotion.add_action',
])
```

Unutar navedenih kolekcijskih klasa definirani su tipovi koje kolekcija sadrži, odnosno tip pravila (*PromotionRuleType*) i tip akcije (*PromotionActionType*). Oba tipa nasljeđuju klasu *AbstractConfigurablePromotionElementType* i sadrže svoju *buildForm* funkciju unutar koje se dodaju polja definirana *ChoiceType* klasama. *ChoiceType* klase definiraju opcije za izbor tipa pravila ili akcije te nasljeđuju *AbstractType* klasu. Prikaz PromotionRuleType klase:

```
final class PromotionRuleType extends AbstractConfigurablePromotionElementType
{
    /**
     * {@inheritdoc}
     */
    public function buildForm(FormBuilderInterface $builder, array $options = []): void
    {
        parent::buildForm($builder, $options);
        //every PromotionRuleType have PromotionRuleChoiceType which enables rule type choice
        $builder
            ->add('type', PromotionRuleChoiceType::class, [
                'label' => 'sylius.form.promotion_rule.type',
                'attr' => [
                    'data-form-collection' => 'update',
                ],
            ])
        ;
    }
```

Prikaz postavljanja opcija u registrirana pravila:

```
public function configureOptions(OptionsResolver $resolver): void
{
    // rule choice type has choices->rules registered in rulechecker
    $resolver->setDefaults([
        'choices' => array_flip($this->rules),
    ]);
}
```

Pomoću *resolver* objekta postavljaju se opcije u obliku niza registriranih pravila ili akcija. Akcije koje su se koristile unutar promocija su klasične akcije primjene fiksnog popusta i popusta u postotku. Korištene akcije su već definirane unutar Sylius promocijske komponente. Pravila su

definirana unutar konfiguracijskih tipova koji su također klase za izradu formi s buildForm funkcijom. Primjeri takvih klasa su *ProductAttributeConfigurationType* i *TaxonConfigurationType* klase koje predstavljaju pravilo primjene promocije po vrijednosti atributa i po kategoriji.

Pravila i akcije registriraju se unutar klasa koje implementiraju *CompilerPassInterface* sučelje i registriraju pravila ili akcije unutar process funkcija. Navedene klase se nalaze unutar DependencyInjection/Compiler direktorija ekstenzije. Sve navedene klase koje se koriste za implementaciju promocijske forme moraju se registrirati unutar service.yml datoteke. Servisi koji se registriraju mogu imati atribute koji se prosljeđuju klasi koja je registrirana kao servis. Klasa servisa kako bi ispravno funkcionirala treba atribute, odnosno ovisi o njima. Atributi su najčešće objekti drugih klasa, a koncept koji omogućuje ovakvu međusobnu injekciju ovisnosti naziva se Dependency Injection. Najčešći tip injekcije je injekcija u konstruktor klase, gdje se definiraju svi objekti potrebni za normalan rad klase. Zahvaljujući ovom mehanizmu nema potrebe za instanciranjem objekata unutar samog konstruktora, nego se objekt instancira izvan klase i proslijedi kao parametar konstruktoru klase. Na ovaj način značajno je olakšano testiranje i održavanje koda.

Posljednji korak implementacije forme je prikazivanje forme unutar *Twig*-a:

```
1.  <div class="ui segment">
2.      <div class="two fields">
3.          {{ form_row(form.code) }}
4.          {{ form_row(form.name) }}
5.      </div>
6.      {{ form_row(form.description) }}
7.  </div>
```

4.5.2. Primjena promocija na cijene proizvoda

Kako bi se primijenile promocije, potrebno je izmijeniti glavnu klasu u kojoj se računaju cijene proizvoda, a to je *Component/Core/Calculator/ProductVariantPriceCalculator* klasa. Navedena klasa unutar metode *calculate* računa cijenu varijante proizvoda za trenutni kanal. Za izračun se koristi metoda *process PromotionProcessor* klase. Prikaz metode *calculate*:

```
1.  public function calculate(ProductVariantInterface $productVariant, array $context): int
2.  {
3.      Assert::keyExists($context, 'channel');
4.      // calculate channel pricing with PromotionProcessor
5.      $channelPricing = $this->processor->process($productVariant, $context['channel']);
```

Metoda process prima argumente varijantu proizvoda i kanal te na temelju njih dohvaća cijenu varijante proizvoda na trenutnom kanalu. Nakon dohvaćanja cijene, dohvaćaju se aktivne

promocije pomoću metode repozitorija *findActiveByChannel* i provjerava se jesu li uvjeti pravila promocije zadovoljeni pomoću *checkRule* metode i je li promocija ekskluzivna. Ukoliko je ekskluzivna, primjenjuje se akcija pomoću applyAction metode i vraća se *channelPricing* objekt u klasu kalkulatora. U suprotnom slučaju primjenjuju se i sve ostale promocije za koje su zadovoljena pravila te se tek onda vraća *channelPricing* objekt kalkulatoru. Prikaz izračuna cijene unutar process metode:

```
1.  // get product variant channel pricing
2.  if ($subject->getChannelPricingForChannel($channel)) {
3.      // create new channel pricing and set code and price from current product variant channel pricing
4.      $channelPricing = new ChannelPricing();
5.      $channelPricing->setChannelCode($channel->getCode());
6.      $channelPricing->setPrice($subject->getChannelPricingForChannel($channel)->getPrice());
7.      //foreach active promotion
8.      foreach ($this->promotionRepository->findActiveByChannel($channel) as $promotion) {
9.          //check for exclusive promotion and check rules for current product variant
10.         if ($promotion->isExclusive() && $this->checkRule($subject, $promotion)) {
11.             //if exclusive promotion exists apply action and return channel pricing (applied on just one promotion with high
    est priority)
12.             $this->applyAction($promotion, $channelPricing);
13.             return $channelPricing;
14.         }
15.     }
16.     foreach ($this->promotionRepository->findActiveByChannel($channel) as $promotion) {
17.         //if exclusive promotion doesnt exist check rules for current product variant
18.         if ($this->checkRule($subject, $promotion)) {
19.             //apply all eligible promotions
20.             $this->applyAction($promotion, $channelPricing);
21.         }
22.     }
23.     return $channelPricing;
```

Metoda *CheckRule* provjerava za svako pravilo promocije je li zadovoljen uvjet s obzirom na danu varijantu proizvoda i promociju. Provjera zadovoljenosti pravila izvršava se pomoću metode *isEligible* koja pripada klasi za provjeru pravila kao što je to TaxonRuleChecker klasa koja implementira RuleCheckerInterface sučelje. Primjer metode za provjeru pravila:

```
1.  //check rules and count eligible, check if number of eligible rules is same as number of rules -
    > all promotion rules must be eligible to execute promotion
2.  private function checkRule(ProductVariantInterface $subject, Promotion $promotion)
3.  {
4.      $counter = 0;
5.      $numberOfRules = $promotion->getRules()->count();
6.      foreach ($promotion->getRules() as $rule) {
7.          $ruleChecker = $this->getRuleCheckerByType($rule->getType());
8.          if ($ruleChecker->isEligible($subject, $rule->getConfiguration())) {
9.              $counter++;
10.             if ($counter == $numberOfRules)
11.                 return true;
12.         }
13.     }
14.     return false;
15. }
```

Metoda *applyAction* primjenjuje promociju pomoću *execute* metode klase za primjenu promocijske akcije na cijenu poput *FixedDiscountPromotionActionCommand* klase koja implementira *PromotionActionCommandInterface* sučelje. Primjer metode za primjenu akcije:

```
1.  private function applyAction(Promotion $promotion, ChannelPricing $channelPricing)
2.  {
3.      //get promotion actions and it's commands by type, execute commands
4.      foreach ($promotion->getActions() as $action) {
5.          $actionCommand = $this->getActionCommandByType($action->getType());
6.          $actionCommand->execute($channelPricing, $action);
7.      }
8.  }
```

Nakon što je *channelPricing* objekt vraćen u klasu kalkulatora, pomoću metode *getPromotionPrice* klase *ChannelPricing* dohvaća se trenutno snižena cijena i klasa kalkulatora ju vraća za daljnji prikaz.

```
1.  //get promotion price from custom Promotion ChannelPricing
2.  return $channelPricing->getPromotionPrice();
```

Klasa *ChannelPricing* koja nasljeđuje *BaseChannelPricing* klasu definira metodu *getPromotionPrice* koja vraća razliku originalne cijene i iznosa sniženja koji je postavljen unutar *execute* metoda promocijske akcije.

```
1.  //calculate promotion price -> original price - promotion amount
2.  public function getPromotionPrice()
3.  {
4.      return $this->getPrice() - $this->getPromotionAmount();
5.  }
```

4.5.3. Prikaz promocijske cijene i originalne cijene

Unutar _price.html.twig datoteke koja prikazuje cijenu proizvoda pozivaju se *macro* funkcije za izračun promocijske i originalne cijene. *Macro* funkcije pozivaju *sylius_calculate_price* i *sylius_calculate_original_price* funkcije definirane unutar *PriceExtension* klasa.

```
1.  {% import "@SyliusShop/Common/Macro/money.html.twig" as money %}
2.   {% set priceMoney = money.calculatePrice(product|sylius_resolve_variant) %}
3.      <span class="ui huge header" id="product-price">
4.      {{ priceMoney }}
5.      </span>
6.   {% set priceOriginalMoney = money.calculateOriginalPrice(product|sylius_resolve_variant) %}
7.   {% if priceMoney != priceOriginalMoney  %}
8.      <span class="ui header" >
9.          <s id="product-price-original"> {{ priceOriginalMoney }} </s>
10.     </span>
11.  {% endif %}
```

PriceExtension klasi se prosljeđuje Sylius *Helper* klasa kojoj se unutar service.yml datoteke kao argument prosljeđuje klasa kalkulatora cijena koja u konačnici vraća cijenu za prikaz. Registracija navedenih klasa unutar service.yml datoteke:

```
app.twig.extension.price:
        class: Praktinchooneri\SyliusCatalogPriceRulesPlugin\Twig\PriceExtension
        arguments:
         - '@app.templating.helper.price'
        tags:
        - { name: twig.extension }
app.templating.helper.price:
        class: Sylius\Bundle\CoreBundle\Templating\Helper\PriceHelper
        arguments:
          - '@app.calculator.product_variant_price'
        tags:
          - { name: templating.helper, alias: sylius_calculate_original_price }
```

5. KORIŠTENJE I TESTIRANJE E-TRGOVINE

Unutar sljedećih poglavlja dan je prikaz rada funkcionalnosti dodavanja u listu želja, prikaza liste želja te kreiranja, primjene i prikaza promocija izvan košarice.

5.1. Primjer korištenja liste želja

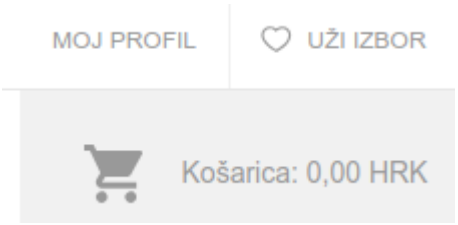
Opcija dodavanja u listu želja dostupna je nakon odabira određenog proizvoda unutar detalja o proizvodu, ali samo ukoliko je korisnik prijavljen (Sl. 5.1. i 5.2).

Sl. 5.1. Izgled detalja o proizvodu kada korisnik nije prijavljen

Sl. 5.2. Prikaz opcije "Dodaj u uži izbor" u slučaju da je korisnik prijavljen

Odabirom opcije dodavanja u uži izbor, odnosno listu želja, proizvod je dodan u uži izbor te je korisnik preusmjeren u svoju listu želja (Sl. 5.3.) gdje također može doći i odabirom opcije "Uži izbor" iz izbornika (Sl. 5.4.).

Sl. 5.3. Prikaz liste želja



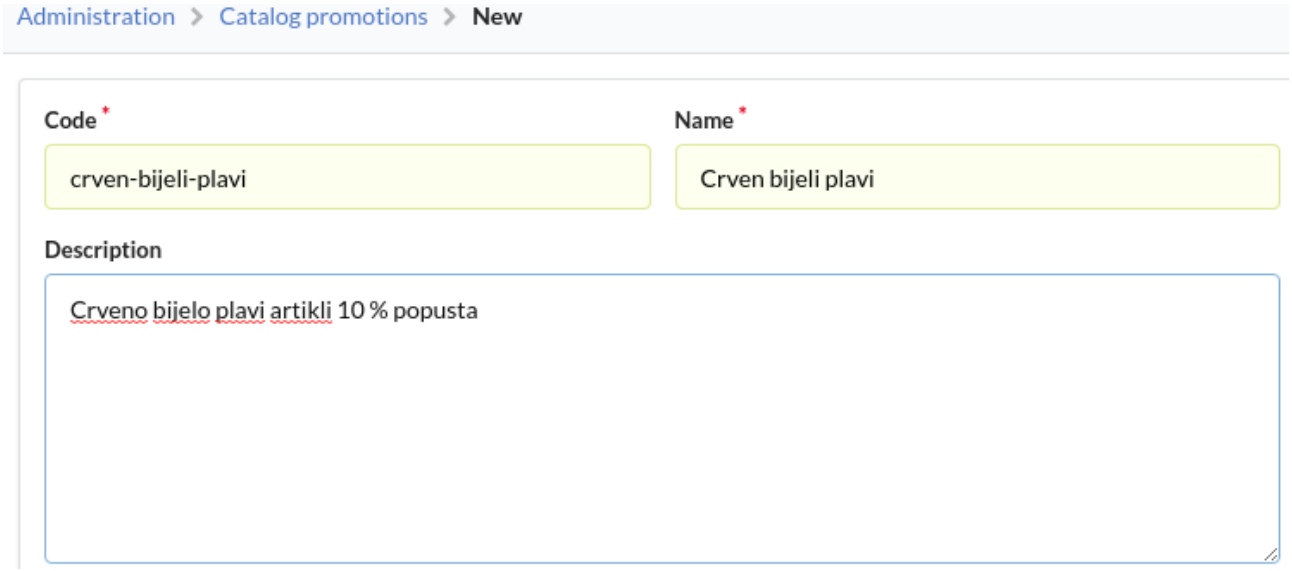
Sl. 5.4. Prikaz opcije "Uži izbor" unutar izbornika

Korisnik može ukloniti proizvod iz liste želja tako da odabere element za brisanje naznačen pomoću znaka x (Sl. 5.5.).

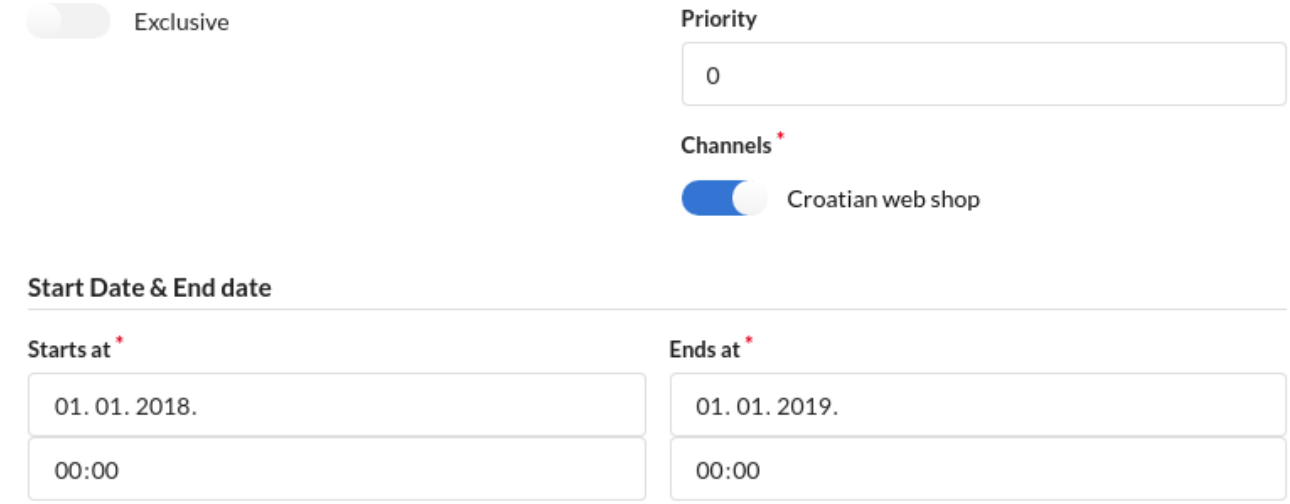
Sl. 5.5. Prikaz prazne liste želja nakon izvršenog brisanja

5.2. Opis kreiranja novih promocija unutar administratorskog sučelja

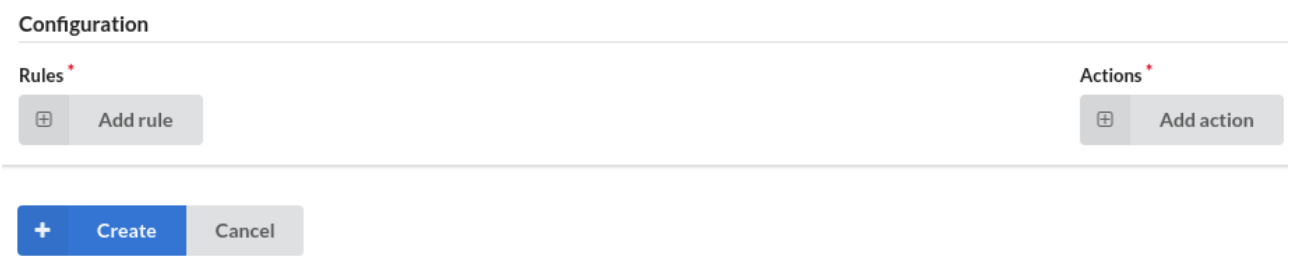
Kreiranje promocija izvan košarice izvodi se unutar *Catalog promotions* opcije admin izbornika. Unos podataka o promociji kao što su kod, naziv, opis, vrijeme trajanja i status (Sl. 5.6. i 5.7.) te odabir promocijskih pravila i akcija (Sl. 5.8. i 5.9.).



Sl. 5.6. Dio forme za unos koda, naziva i opisa



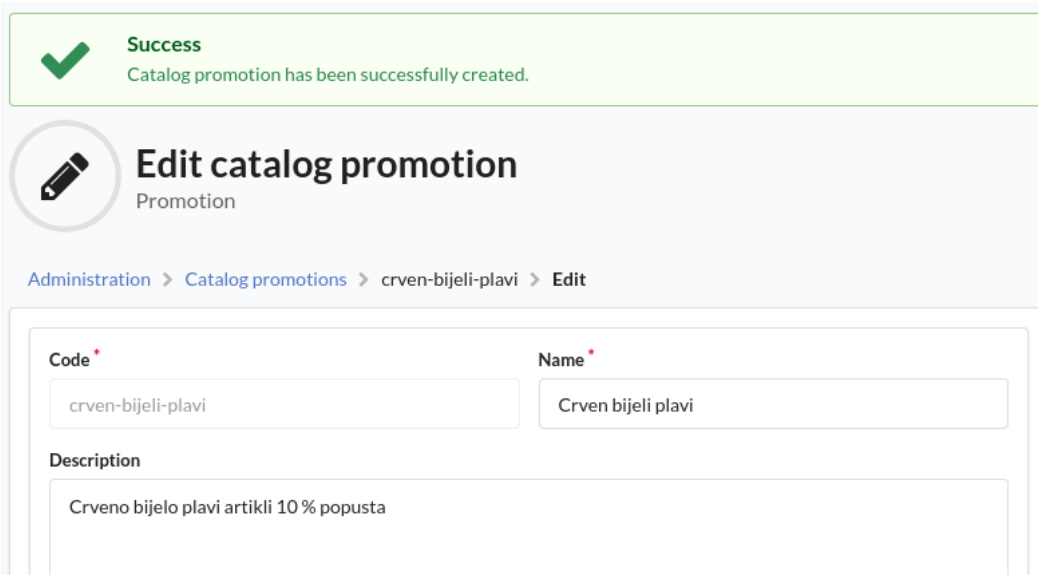
Sl. 5.7. Dio forme za odabir *exclusive* atributa, prioriteta, kanala i vremena trajanja promocije



Sl. 5.8. Dio forme za dodavanje promocijskih pravila i akcija

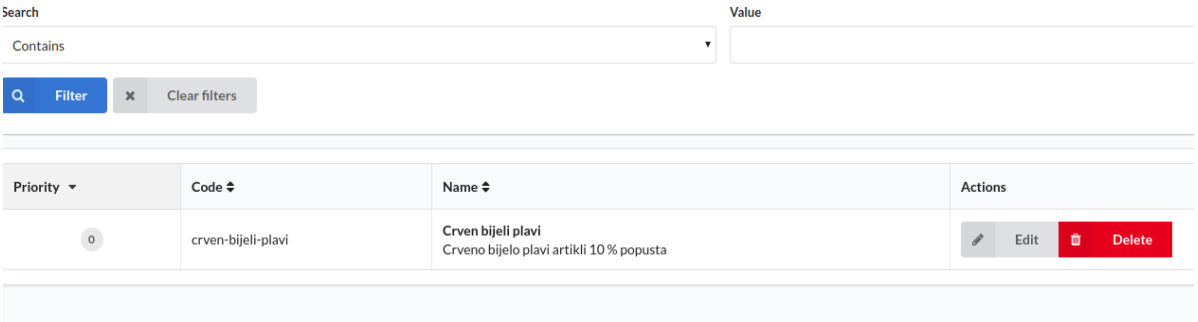
Sl. 5.9. Dio forme za odabir opcija promocijskog pravila i akcije

Nakon unosa i odabira opcije *Create* kreira se nova promocija izvan košarice i prikazuje se forma za uređivanje nove promocije s porukom da je promocija uspješno kreirana (Sl. 5.10.).



Sl. 5.10. Forma za uređivanje promocija s porukom o uspješnosti dodavanja nove promocije

Unutar početne liste promocija može se vidjeti da je uspješno dodana nova promocija (Sl. 5.11.).



Sl. 5.11. Početna lista svih promocija s novom promocijom

5.3. Testiranje postupka primjene promocije

Novokreirana promocija se primjenjuje na artikle koji su crveno bijelo plave boje te smanjuje njihovu cijenu u iznosu od deset posto. Primjena promocije na cijenu artikla može se provjeriti pregledom proizvoda s danom karakteristikom, odnosno s crveno bijelo plavom bojom (Sl. 5.12.).

Sl. 5.12. Prikaz primjene promocije na artikl koji zadovoljava zadano pravilo

6. ZAKLJUČAK

Razvoj moderne web trgovine znatno je olakšan zahvaljujući Sylius platformi. Osim gotovih rješenja za većinu funkcionalnosti koje su potrebne jednoj e-trgovini, Sylius također omogućuje jednostavnu implementaciju dodatnih funkcionalnosti. Zbog tih svojih osobina pokazao se izvrsnim izborom za izradu zadatka rada, odnosno za izradu manje trgovine koja zahtjeva visoku razinu prilagodljivosti sustava. Jednostavnost konfiguracije e-trgovine omogućuje korištenje i korisnicima slabijeg tehničkog znanja, no ukoliko je potrebno izvršiti različite prilagodbe funkcionalnosti unutar ekstenzija, potrebno je dobro poznavanje objektno orijentiranih koncepata.

LITERATURA

[1] Y. A. Nanehkaran, An Introduction To Electronic Commerce, INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 2, ISSUE 4, April 2013, dostupno na: http://www.ijstr.org/final-print/apr2013/An-Introduction-To-Electronic-Commerce.pdf [20.06.2018]

[2] Aimeos, What is an e-commerce framework?, 08.06.2017, dostupno na: https://aimeos.org/tips/ecommerce-framework/ [20.06.2018.]

[3] Bssadmin, What is Magento? The History of Magento, BSSCommerce Magento Blog, 17.06.2015, dostupno na: https://bsscommerce.com/blog/magento-and-history-of-magento/[25.06.2018.]

[4] A. English, Magento vs Shopify – Which is the right platform for you?, Website Builder Expert, 27.06.2018, dostupno na: https://www.websitebuilderexpert.com/magento-vs-shopify/ [28.06.2018]

[5] ORO, Dokumentacija, dostupno na: https://oroinc.com/orocrm/doc/current/architecture [30.06.2018]

[6] Sylius, Sylius eCommerce Framework, dostupno na: https://sylius.com/sylius-ecommerce-framework/ [05.07.2018]

[7] Sylius, Dokumentacija, dostupno na: https://docs.sylius.com/en/latest/index.html [07.07.2018]

[8] L. Welling, L.Thomson, PHP and MySQL Web Development, Fifth Edition, Addison-Wesley, 2017.

[9] Symfony, Dokumentacija, dostupno na: https://symfony.com/what-is-symfony [07.07.2018]

[10] Doctrine, Dokumentacija, dostupno na: https://www.doctrine-project.org/projects/doctrine-orm/en/current/tutorials/getting-started.html [07.07.2018]

[11] Symfony, Kreiranje bundle-a, dostupno na: https://symfony.com/doc/current/bundles.html#creating-a-bundle, [15.07.2018]

[12] Sylius, Kreiranje entiteta, dostupno na: http://docs.sylius.org/en/latest/cookbook/entities/custom-model.html, [21.07.2018]

[13] Symfony, Relacije, dostupno na: https://symfony.com/doc/2.8/doctrine/associations.html, [24.07.2018]

[14]Symfony, Kontroler, dostupno na: https://symfony.com/doc/2.8/controller.html, [30.07.2018]

[15] Symfony, Factory, dostupno na:
https://symfony.com/doc/2.8/service_container/factories.html#symfony4-promo,
[01.08.2018]
[16] Symfony, Repository, dostupno na:
http://symfony.com/doc/3.0/doctrine/repository.html#symfony4-promo, [02.08.2018]

SAŽETAK

Zadatak ovog rada bio je izraditi naprednu e-trgovinu na Sylius platformi te opisati proces razvoja moderne web trgovine. Tijekom izgradnje web trgovine bilo je potrebno upoznati se sa samim konceptima e-trgovine, platforme za razvoj e-trgovina i Sylius platforme. Web trgovinu je trebalo podesiti korištenjem postojećih funkcionalnosti Sylius-a, ali i nadograditi razvojem ekstenzija. Kroz razvoj dodatnih funkcionalnosti poput liste želja i promocija izvan košarice, koriste se napredni koncepti objektno orijentiranog pristupa, odnosno različiti obrasci dizajna. Na samom kraju testira se i opisuje se korištenje ostvarenog sustava kroz par osnovnih slučajeva korištenja.

Ključne riječi: ekstenzija, e-trgovina, obrasci dizajna, Sylius

ABSTRACT

THE DEVELOPMENT OF MODERN WEBSHOP WITH SYLIUS E-COMMERCE SOLUTION

The main purpose of this master's thesis was to develop an advanced webshop using the Sylius platform and outline the process of developing a modern webshop. During the development process, it was necessary to develop a profound understanding of all e-commerce related concepts, platforms and the Sylius platform itself. The webshop had to be configured using built-in Sylius functionalities, but it also had to be upgraded using extensions. During the development of added features and functionalities such as the wish list and catalog promotions, advanced object-oriented programming and advanced design concepts and patterns were used. Ultimately, the developed e-commerce solution was tested and described through a couple of basic use cases.

Keywords: extension, e-commerce, design patterns, Sylius

ŽIVOTOPIS

Kristina Slović rođena je 20.12.1994. u Đakovu. Nakon završene osnovne škole (OŠ Drenje), upisuje 2009. Opću gimnaziju u Đakovu. Nakon završetka srednje škole 2013., upisuje preddiplomski studij računarstva na Elektrotehničkom fakultetu u Osijeku. Nakon završetka preddiplomskog studija 2016., upisuje diplomski studij računarstva, smjer programsko inženjerstvo. Dobitnica rektorove nagrade u akademskoj godini 2017./2018. za izvrstan seminarski rad iz predmeta Računarstvo usluga i analiza podataka pod nazivom: "Klasifikacija slika Pas/Mačka". Trenutno zaposlena u tvrtki Inchoo.

PRILOZI (NA CD-U)

Prilog 1. Dokumentacija diplomskog rada

Prilog 2. Programski kod web aplikacije