Web sustav za prodaju proizvoda uz nadzor njihove prehrambene i ekološke prihvatljivosti

Klešić, Ivan

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

Sveučilišni diplomski studij računarstva

Web sustav za prodaju proizvoda uz nadzor njihove prehrambene i ekološke prihvatljivosti

Diplomski rad

Ivan Klešić

Osijek, 2020.

SADRŽAJ

1. UVOD 1

1.1. Zadatak diplomskog rada 1

2. PREHRAMBENA I EKOLOŠKA PRIHVATLJIVOST PROIZVODA I PREGLED POSTOJEĆIH RJEŠENJA 2

2.1. Parametri prehrambene prihvatljivosti proizvoda 2

2.2. Parametri ekološke prihvatljivosti proizvoda 5

2.3. Sustavi stvaranja preporuka 6

2.4. Pregled postojećih rješenja 8

3. IDEJNO RJEŠENJE I MODEL SUSTAVA 10

3.1. Funkcionalni zahtjevi na aplikaciju 10

3.1.1. Profil korisnika i međusobna komunikacija 10

3.1.2. Profil proizvoda 11

3.1.3. Veza između korisnika i proizvoda 11

3.1.4. Sustav preporuka temeljen na analizi grafova 13

3.2. Nefunkcionalni zahtjevi na aplikaciju 15

4. PROGRAMSKO RJEŠENJE WEB SUSTAVA ZA PRODAJU PROIZVODA SA SUSTAVOM DAVANJA PREPORUKA I IZDAVANJA UPOZORENJA 17

4.1. Programske tehnologije, okviri i arhitekture korištene za izradu web aplikacije 17

4.1.1. HTML 17

4.1.2. CSS i Bootstrap 17

4.1.3. JavaScript i jQuery 17

4.1.4. PHP, Symfony i MVC 18

4.1.5. Baza podataka Neo4j 18

4.1.6. Sustav za kreiranje preporuka Reco4PHP 19

4.2. Programsko rješenje sustava na strani poslužitelja 20

4.2.1. Registriranje i prijava korisnika 20

4.2.2. Dodavanje proizvoda u košaricu 21

4.2.3. Korištenje PayPal API-ja za ostvarenje transakcija 22

4.2.4. Postavljanje baze Neo4j 23

4.2.5. Logika stvaranja preporuka 24

4.3. Programsko rješenje sustava na strani klijenta 29

4.3.1. Dodavanje i validacija proizvoda 29
4.3.2. Ocjenjivanje proizvoda 31
4.3.3. Prikazivanje upozorenja i preporuka 32
5. PRIKAZ NAČINA RADA I ISPITIVANJE WEB SUSTAVA 35
5.1. Definiranje testnih slučajeva 35
5.2. Izvršavanje testnih slučajeva i pregled aplikacije 36
5.2.1. Registriranje, prijava i odjava korisnika 36
5.2.2. Profiliranje novog proizvoda 38
5.2.3. Komunikacija između korisnika 39
5.2.4. Kupnja proizvoda 39
5.2.5. Pregled preporuka za korisnika 41
5.2.6. Analiza i opis preporuka 42
6. ZAKLJUČAK 45
LITERATURA 46
SAŽETAK 48
ABSTRACT 49
ŽIVOTOPIS 50
PRILOZI 51

1. UVOD

U današnje vrijeme kupnja proizvoda putem Interneta je vrlo razvijena te postoji velik broj web trgovina koje nude razne usluge i proizvode. Prodaja hrane putem Interneta zahtijeva dodatne mjere opreza radi korisnika koji iz bilo kojeg razloga moraju ograničiti prehranu. Također, modernije web trgovine sadržavaju i mehanizme preporučivanja proizvoda korisnicima.

Cilj ovog rada je ostvariti sustav koji omogućuje višekriterijsku analizu nutritivnih i ekoloških svojstava hrane. Korisnici će imati mogućnosti prijave korisničkim računom uz definiranje plana prehrane u obliku dijete, te definiranja zdravstvenih stanja koja mogu predstavljati problem kod prehrane. Nadalje, korisnici će moći kupovati proizvode putem Interneta te dobivati preporuke i upozorenja kod kupnje proizvoda na temelju korisničkog profila. Dodatno, korisnici će imati mogućnost komuniciranja putem sustava poruka te ocjenjivanja kupljenih proizvoda.

U drugom poglavlju će se predstaviti teorijske osnove na kojima se temelji cijeli sustav. Također će biti navedena i postojeća rješenja koja imaju nešto zajedničko s ovim. Treće poglavlje će prikazati idejno rješenje i model sustava u obliku funkcionalnih i nefunkcionalnih zahtjeva. U četvrtom poglavlju će se opisati korištene tehnologije i kompletno rješenje sustava na strani poslužitelja i na strani klijenta. Konačno, peto poglavlje će sadržavati prikaz rada sustava i testiranje glavnih mogućnosti.

1.1. Zadatak diplomskog rada

U teorijskom dijelu rada potrebno je analizirati parametre koji opisuju i omogućuju procjenu prehrambene i ekološke prihvatljivosti proizvoda u web prodaji, osmisliti model profiliranja korisnika i proizvoda, način prikupljanja povratnih informacija od korisnika i komuniciranja među korisnicima, potrebne postupke analize podataka, postupak on-line plaćanja, te mehanizam davanja preporuka i izdavanja upozorenja korisnicima kod kupnje proizvoda. Koristeći prikladni programski predložak arhitekture rješenja, programska sučelja, tehnologije, okvire i jezike, web sustav je u praktičnom dijelu rada potrebno programski ostvariti i ispitati na odgovarajućem skupu podataka. Također, potrebno je provesti nefunkcionalno testiranje sustava.

2. PREHRAMBENA I EKOLOŠKA PRIHVATLJIVOST PROIZVODA I PREGLED POSTOJEĆIH RJEŠENJA

Potreba za ovakvim programskim rješenjem dolazi od činjenice da je u moderno doba svijest o zdravoj prehrani te očuvanju okoliša sve više prisutna kod ljudi (barem u razvijenijim zemljama). Sve više smo svjesni zdravstvenih problema koji mogu biti posljedica nezdrave i neadekvatne prehrane. Također, utjecaj emisije ugljikovog dioksida na okoliš je ozbiljna tema koja za posljedicu ima razvoj novih, modernih tehnologija koje su (većim dijelom ili potpuno) orijentirane prema iskorištavanju obnovljivih izvora energije.

2.1. Parametri prehrambene prihvatljivosti proizvoda

Prema [1], pretilost je svjetski poznata i rasprostranjena bolest na koju utječe više faktora, a definira se kao pretjerana nakupina masti koja ugrožava zdravlje čovjeka. Povezana je s kardiovaskularnim bolestima, dijabetesom i rakom. Zbog značajnih efekata na zdravlje i stopu smrtnosti, pretilost je jedan od većih problema javnog zdravstva. Glavni uzrok pretilosti je energetska neravnoteža između unesenih i potrošenih kalorija; međutim, iza ovoga stoji puno kompleksnije međudjelovanje bioloških, genetskih i psihosocijalnih čimbenika.

Da bi se postigao uspješan gubitak kilograma, u [2] su predložene promjene u životnom stilu, dijetu koja smanjuje unošenje viška energije, te povećava cjelokupnu kvalitetu prehrane. Dijete su većinom bazirane na uključivanju i/ili isključivanju pojedinih namirnica ili grupa namirnica (prikazano u tablici 2.1), a mogu se podijeliti u 3 glavne skupine:

1. Dijete bazirane na kontroli unosa mikronutrijenata i makronutrijenata (npr. dijeta sa smanjenim unosom masti, dijeta sa smanjenim unosom ugljikohidrata itd.)
2. Dijete bazirane na zabrani unosa određenih namirnica ili grupa namirnica (npr. bezglutenska dijeta, Paleo dijeta, vegetarijanska/veganska dijeta itd.)
3. Dijete bazirane na vremenskom odmaku (npr. *fasting* dijeta)

Tab. 2.1. *Prikaz najpoznatijih dijeta i odgovarajućih namirnica (nedozvoljene namirnice su označene znakom X)*

Dijeta / Namirnice	**Atkins**	**Keto**	**Zone**	**Ornish**	**Paleo**	**Mediteranska dijeta**
Povrće koje ne sadrži škrob						
Povrće koje sadrži škrob	X	X			X	
Voće koje ne sadrži škrob						
Voće koje sadrži škrob	X	X			X	
Crveno meso				X		
Perad				X		
Morski plodovi				X		
Mliječni proizvodi s niskim udjelom masti					X	
Mliječni proizvodi s visokim udjelom masti				X	X	
Orašasti plodovi i sjemenke						
Biljna ulja						
Jaja				X		
Mahunarke	X	X			X	
Cijelovite žitarice	X	X			X	
Rafinirane žitarice	X	X	X		X	
Šećer	X	X	X		X	X

Također, osim kontrole unosa određenih namirnica ili skupina namirnica, u [1,2] je preporučeno i pridržavanje omjera makronutrijenata u prehrani. Pod makronutrijente pripadaju ugljikohidrati, lipidi i bjelančevine. Neke podjele umjesto lipida prikazuju masti, međutim, masti su podskup lipida tako da podjele koje prikazuju lipide obuhvaćaju veći dio nutrijenata.

Nadalje, dijeta nije jedini mogući razlog za izbacivanje određenih namirnica ili grupa namirnica iz prehrane. Naime, postoje brojna zdravstvena stanja koja imaju veze s prehranom a mogu na manjoj ili većoj razini ograničavati unos hrane u organizam. Prema [3], postoji točno određen popis takvih stanja te odgovarajućih mikronutrijenata i makronutrijenata čija konzumacija je ograničena ili zabranjena ako se žele izbjeći negativne posljedice. Najznačajnija zdravstvena stanja te vrste su alergije na hranu. Za alergije ne postoji znanstveno dokazani lijek, tako da je jedini način za sprječavanje alergijskih reakcija kontrola prehrane.

Najpoznatije i najznačajnije skupine alergija na hranu, navedene u [4], su:

- Alergija na mliječne proizvode
- Alergija na ribu
- Alergija na jaja
- Alergija na rakove i školjke
- Alergija na orašaste plodove
- Alergija na pšenicu
- Alergija na soju

Osim alergija, postoje i neke druge bolesti koje utječu na prehranu. Prema [3], te bolesti su:

- Celijakija
- Crohnova bolest
- Netolerancija na laktozu
- Gastroezofagealni refluks

Celijakija je bolest koja šteti tankom crijevu, a simptomi nastupaju nakon konzumacije glutena. Gluten je bjelančevina koja se prirodno nalazi u žitaricama, te je uobičajen u svim namirnicama koje sadrže žitarice. Celijakija može bitno utjecati na probavni sustav, a ako se čovjek ne pridržava pravila, može imati dugotrajne posljedice.

Crohnova bolest je probavna bolest koja također šteti tankom crijevu te uzrokuje upale i iritaciju. Za izbjegavanje simptoma, potrebno je izbaciti iz prehrane namirnice kao što su gazirana pića i hrana bogata vlaknima.

Netolerancija na laktozu nije samo drugi naziv za alergiju na mlijeko i mliječne proizvode, jer iz medicinske točke gledišta to nisu iste bolesti. Međutim, za potrebe ovog rada, ta dva zdravstvena stanja će se poistovjetiti jer, barem što se tiče prehrane, imaju ista ograničenja a to su izbjegavanje namirnica koje sadrže laktozu.

Gastroezofagealni refluks je bolest koju karakterizira vraćanje sadržaja iz želuca u jednjak, što nadalje može izazvati žgaravicu. Što se tiče prehrane, preporučuje se izbjegavati alkohol, kavu, čokoladu, masnu i začinjenu hranu te rajčicu i proizvode od rajčice.

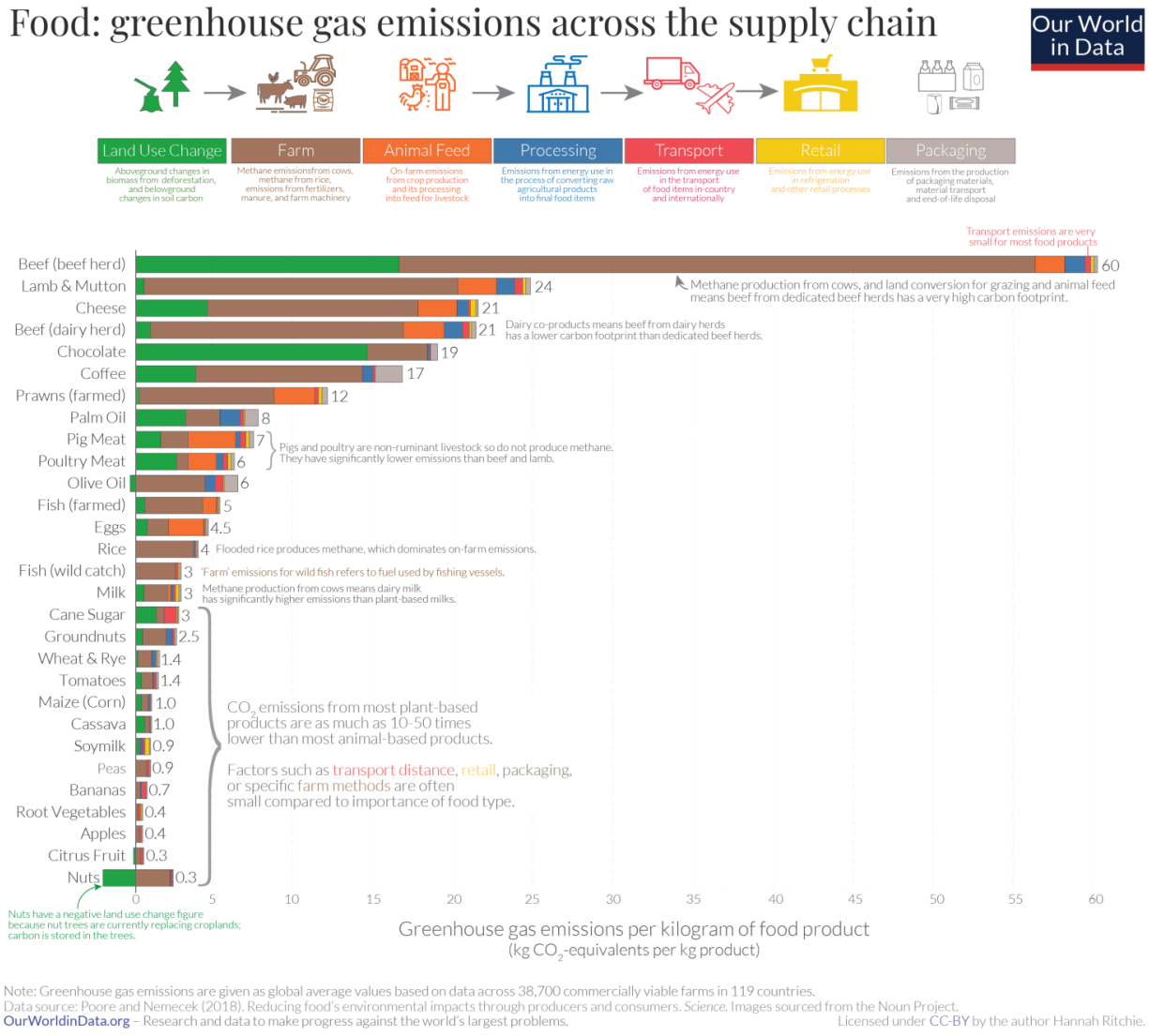
2.2. Parametri ekološke prihvatljivosti proizvoda

Ljudi širom svijeta postaju sve više zabrinuti u vezi promjena klime. Prema [5], proces proizvodnje hrane je odgovoran za četvrtinu svjetske emisije štetnih plinova. Često se može čuti preporuka da se prednost treba davati domaće proizvedenoj hrani, zato što transport također igra veliku ulogu u emisiji ugljikovog dioksida. To bi imalo smisla da transport stvarno značajno utječe na emisiju ugljikovog dioksida, međutim, to jednostavno nije istina za veliku većinu transporta hrane.

Kao što je navedeno u [6], ugljikov dioksid je najznačajniji plin koji negativno utječe na atmosferu, ali nije jedini. Da bi se u obzir uzeli svi stakleнički plinovi, u sljedećem grafičkom prikazu (Sl. 2.1), emisija stakleničkih plinova je izražena u kilogramima ekvivalentne količine ugljikovog dioksida po kilogramu proizvoda, ili skraćeno „CO_2-e“. Kao što je vidljivo na slici 2.1, hrana životinjskog porijekla ima mnogo veći stupanja emisije stakleničkih plinova npr. proces proizvodnje jednog kilograma govedine ima emisiju od 60 CO_2-e, dok proces proizvodnje jednog kilograma graška ima emisiju od 0.9 do 1 CO_2-e.

Za većinu hrane, osobito one koja proizvodi velike količine CO_2-e, vrijedi činjenica da se većina CO_2-e emitira u početnim fazama proizvodnje, a to su faza iskorištavanja zemlje (prikazano zelenom bojom) te faza uzgoja (prikazano smeđom bojom). Te dvije faze zajedno čine više od 80% emisije CO_2-e kod većine proizvedene hrane. Osim transporta, na emisiju stakleničkih plinova

utječu i procesiranje, maloprodaja, te pakiranje proizvoda. Svi ti faktori su uzeti u obzir pri stvaranju grafičkog prikaza na slici 2.1 preuzetog iz [6].



Sl. 2.1. *Grafički prikaz emisije stakleničkih plinova različitih skupina hrane [6]*

2.3. Sustavi stvaranja preporuka

Sustavi preporuka, kako je navedeno u [7], su jedni od najzastupljenijih sustava na web stranicama koje imaju veze s prodajom. Smanjuju opterećivanje korisnika nepotrebnim informacijama te poboljšavaju prikaz potrebnih informacija ciljnim skupinama. Implementacija sustava za preporuke

pomaže kod zadržavanja korisnika i povećanja prihoda. Algoritmi koji se koriste u takvim sustavima su mnogobrojni, a mogu biti vrlo jednostavni, kao npr. spajanje istih ključnih riječi u profilu korisnika, i puno kompleksniji kao npr. rudarenje podataka i grupiranje poslužiteljskih zapisnika (eng. *server log clustering*). Sustav preporuka se često predstavlja kao sustav s dva tipa parametara, a to su najčešće ljudi (primarni parametar) i objekti (sekundarni parametar).

Glavni problem sustava preporuka je sljedeći: kako iz velikog skupa podataka (u nekim slučajevima se radi o stotinama tisuća objekata, ili čak milijunima) izvući one koji su određenom korisniku bitni, a da ga pri tome ne opterećuju tj. da je krajnji skup dovoljno mali da odgovara osobnim sklonostima korisnika.

U [8] su predstavljene dvije široke kategorije sustava za preporuke. Prva kategorija je sustav koji preporučuje na temelju sadržaja npr. korisniku se preporučuju objekti koji imaju neke karakteristike slične kao objekti na koje je isti korisnik reagirao u prošlosti (korisnička reakcija u ovom kontekstu može značiti kupnja u slučaju web trgovine, pregled u slučaju web sustava za gledanje filmova itd.). Ovakav postupak se može svesti na četiri koraka:

1. Za svaki objekt treba postojati vektor vrijednosti ili „profil“ objekta
2. Potrebno je pronaći druge objekte koji su prepoznati kao „slični“ u odnosu na referentni objekt
3. Za svakog korisnika treba postojati vektor vrijednosti ili „profil“ korisnika
4. Potrebno je preporučiti objekte koji imaju korelaciju s korisničkim profilom

Prednost ovakvog pristupa je činjenica da za uspješno pretraživanje nisu potrebni podaci drugih korisnika, dok je nedostatak nemogućnost uspješnog preporučivanja za korisnike koji su tek nedavno kreirali svoje profile, pa sustav nema dovoljno podataka da bi izvršio pretraživanje.

Druga kategorija sustava za preporuke su sustavi bazirani na kolaborativnom filtriranju. Ovaj postupak korisnicima preporučuje objekte koje preferiraju slični korisnici. Taj proces se naziva kolaborativnim jer više sličnih korisnika međusobno utječu na preporuke koje im se prikazuju (iako korisnici toga ne moraju biti svjesni). Ovakav pristup rješava problem ranije navedenih sustava jer korisnik ne mora reagirati na objekte da bi mu se isti mogli preporučiti, nego samo mora kreirati profil s određenim parametrima. S druge strane, ovakav pristup ima problema s nedostatnim skupovima podataka, koji se javljaju kada korisnici reagiraju na mali podskup objekata koji se nalaze u sustavu.

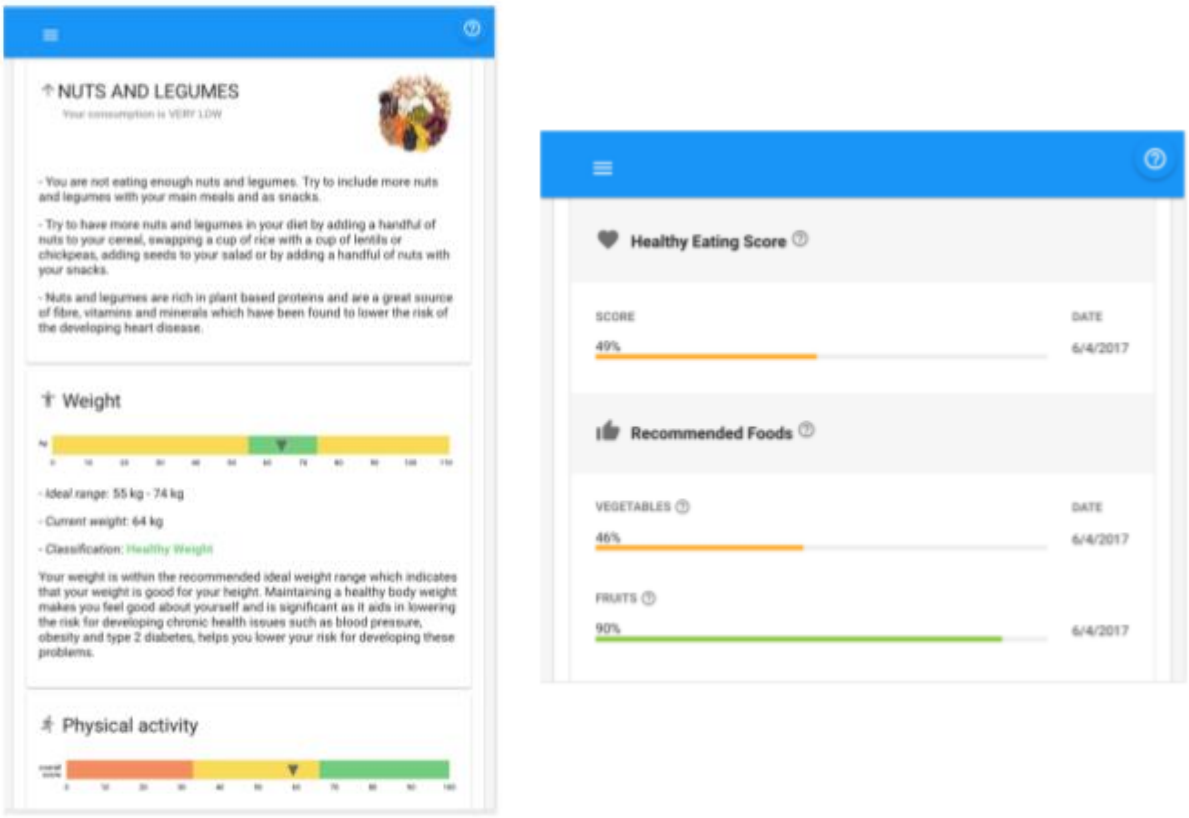
2.4. Pregled postojećih rješenja

Sustavi za preporuke su zastupljeni u većini poznatih web stranica koje nude multimedijski sadržaj (videozapisi, glazba itd.). Dobar primjer toga je sustav za preporuke koji koristi YouTube prikazan u [9]. Taj sustav konkretno radi na temelju neuronskih mreža, gdje se sadržaj koji korisnik gleda koristi kao početni skup podataka. Sustav se sastoji od dvije neuronske mreže, od kojih prva služi za generiranje kandidata za preporuke, a druga služi za rangiranje tj. davanje ocjena pojedinim preporukama. Programski kod generiranja kandidata za preporuke se koristi ranije spomenut algoritam kolaborativnog filtriranja.

Sustav predstavljen u [10] je hibridni sustav baziran na zdravstvenim parametrima i parametrima ukusa pojedinog korisnika. Zdravstveni aspekt algoritma za preporuke procjenjuje koliko je pojedini proizvod adekvatan za korisnika na temelju njegovih sklonosti i sastava samog proizvoda. Algoritam također uzima u obzir ukus tako što analizira korisnikove prehrambene navike i veću važnost daje proizvodima koji su slični onima koje je korisnik već uvrstio u plan prehrane.

Amazonov sustav preporuka prikazan u [11] je vrlo složen i sastoji se od više slojeva. Sustav prati kategorije koji korisnik pretražuje te preporučuje proizvode koji pripadaju istima. Ako je korisnik kliknuo na proizvod, u preporuke se automatski uvrštavaju i ostali proizvodi istog tipa, ali drugih dimenzija, proizvođača i slično. Amazon također preporučuje novije inačice uređaja, u slučaju da je korisnik kupio neki od uređaja tog tipa npr. ako je korisnik kupio Samsung Galaxy S9, u skup preporučenih proizvoda automatski ulazi Samsung Galaxy S10, S20 i novije inačice, ako postoje. Preporuke u web sustavima za prodaju imaju cilj povećati prosječan broj narudžbi koje korisnik izvršava u nekom vremenskom periodu. Prema tome, Amazon je uveo i preporuke koje prikazuju proizvode koji se najčešće kupuju zajedno. Naime, korisnici koju su npr. kupili mobilni telefon mogu očekivati da će im se u preporukama prikazati maske, futrole, slušalice itd. Prema analizama koje provodi Amazon, preporuke koje se šalju korisnicima na e-mail imaju veći stupanj isplativosti od preporuka koje se prikazuju direktno na stranici. Konkretan algoritam koji koristi Amazonov sustav za preporuke je „*item-to-item collaborative filtering*“, koji je nastao kao kombinacija ranije navedena dva algoritma, te kombinira prednosti oba pristupa kako bi poboljšao kvalitetu preporuka.

U [12] je predstavljen web sustav za preporučivanje proizvoda preko Interneta „e-Nutri“. Sustav najprije od korisnika zahtjeva da ispune kratki upitnik vezan za količinu i frekvenciju unosa hrane (eng. *Food frequency questionnaire*). Zatim se podaci koriste za izračunavanje AHEI indeksa (eng. *Alternative Healthy Eating Index*) pa se AHEI dalje koristi za kreiranje preporuka i izvještaja o napretku u odnosu na prošle rezultate (Sl. 2.2 preuzeta iz [12]). Također računa idealni raspon tjelesne mase koji je baziran na BMI-u (eng. *Body Mass Index*) i daje povratne informacije o preporučenoj razini tjelesne aktivnosti.



Sl. 2.2. *Grafički prikaz preporuka i izvještaja o zdravoj prehrani [12]*

3. IDEJNO RJEŠENJE I MODEL SUSTAVA

U ovom poglavlju će se, na temelju teorijskih osnova iz prošlog poglavlja, predstaviti konačni model aplikacije. Prvo će biti predstavljeni funkcionalni i nefunkcionalni zahtjevi te UML dijagram preko kojeg se dolazi do modela baze podataka, a zatim i glavni dio logike aplikacije za sustav preporuka.

3.1. Funkcionalni zahtjevi na aplikaciju

Aplikacija razvijena u okviru ovog rada je web sustav koji se bazira na korisnicima i sadržajem koji isti kreiraju. Prema tome, treba imati podršku za korisničke račune. Korisnički računi će imati omogućene osnovne *CRUD* (eng. *Create-Read-Update-Delete*) operacije, tako da korisnici mogu kreirati svoje profile, mijenjati ih, i brisati. Kada se korisnik prijavi u aplikaciju s vlastitim korisničkim računom, prikazat će mu se drugačiji sadržaj ovisno o tome koja je uloga tog korisnika. Korisnici također moraju imati omogućene CRUD operacije nad proizvodima. U nastavku su detaljnije objašnjeni funkcionalni zahtjevi.

3.1.1. Profil korisnika i međusobna komunikacija

Za funkcionalnost potrebnu u ovom sustavu dovoljna su dva tipa korisnika i jedan administrator. Prvi tip korisnika će biti prodavači, točnije korisnici koji imaju omogućene *CRUD* operacije nad proizvodima. Naime, prodavači će imati potpunu kontrolu nad postavljanjem, održavanjem, mogućim izmjenama ili brisanjem proizvoda (samo onih proizvoda koje su sami postavili na prodaju). Drugi tip korisnika su kupci, točnije korisnici koji imaju uvid u proizvode postavljene na prodaju, te ih mogu pretraživati po određenim kriterijima, dodavati u košaricu za kupnju, te na kraju i kupiti. Nakon kupnje, kupci mogu ocijeniti proizvod ocjenom od 1 do 10. Ostali korisnici koji pretražuju proizvode će imati uvid u prosjek ocjena pojedinog proizvoda. Ocjene će se također uzimati u obzir pri kreiranju preporuka.

Kupci će pri kreiranju vlastitog korisničkog računa moći odabrati i sva zdravstvena stanja s kojima imaju problema, a koja također imaju utjecaja na prehranu. Također, kupci će imati mogućnost izbora jedne od poznatih i dokazanih dijeta. Te informacije će biti spremljene u njihov profil, a bit će korištene pri kreiranju potrebnih preporuka i upozorenja prije ostvarenja kupnje.

Administrator će imati uvid u sve dijelove aplikacije, te će moći brisati korisničke račune ako to bude potrebno. Također, imat će mogućnost dodavanja novih kategorija proizvoda i zdravstvenih stanja u aplikaciju.

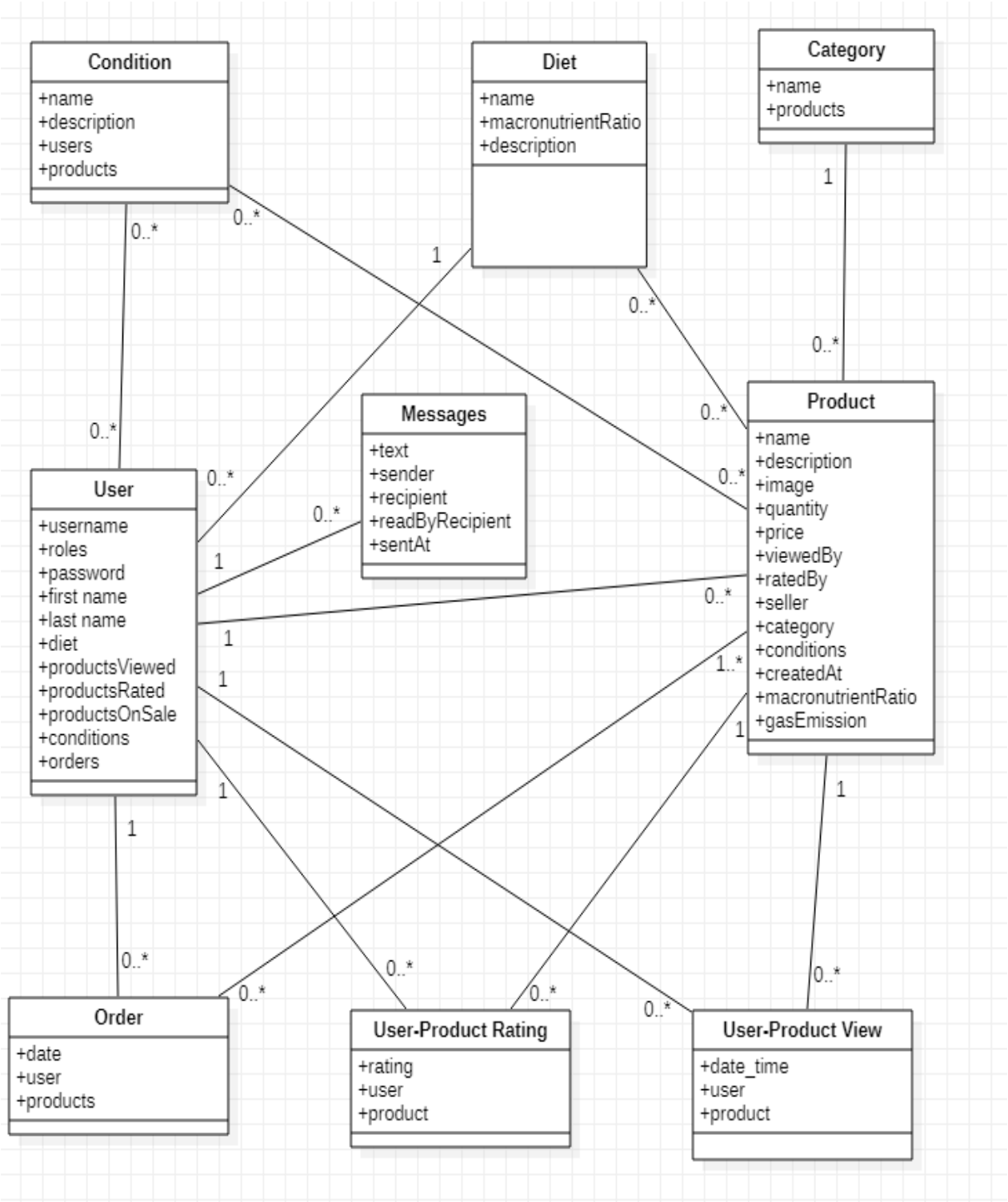
Iako se ovdje ne radi o društvenoj mreži, korisno je implementirati i *chat* funkcionalnost tako da korisnici mogu kontaktirati prodavače ako imaju dodatnih pitanja u vezi određenih proizvoda, a i ostale kupce u slučaju da im treba savjet ili preporuka drugog korisnika koji npr. ima više iskustva s nekim prodavačem. Neovisno o tome je li korisnik trenutno spojen na web aplikaciju ili ne, poruke su spremljene u sustav.

3.1.2. Profil proizvoda

Proizvode u ovom sustavu mogu kreirati, mijenjati i brisati samo prodavači, a pretraživati ih mogu kupci. Pri kreiranju proizvoda, prodavač može definirati naziv, opis, fotografiju te količinu koja je dostupna. Prodavači također imaju obvezu profiliranja proizvoda u vezi prehrambene i ekološke prihvatljivosti koje su opisane u prošlom poglavlju. Svaki proizvod mora imati definiran omjer makronutrijenata koje sadrži. Ako proizvod pripada nekoj od kategorija za koje je poznata količina CO_2-e koji proizvodi, to je potrebno upisati pri postavljanju proizvoda u sustav. Isto tako, ako proizvod sadrži bilo kave nutrijente koji mogu imati posljedice ako konzument boluje od određene bolesti, to je potrebno naznačiti.

3.1.3. Veza između korisnika i proizvoda

Kao što je navedeno ranije, aplikacija izrađena u okviru ovog rada će imati sustav preporuka koji će korisnicima davati preporuke na osnovi više različitih kriterija. Parametri koje će se spremati u bazu podataka i analizirati prilikom kreiranja preporuka su sljedeći: „pregled“ proizvoda od strane korisnika, kupnja proizvoda od strane korisnika, ocjena proizvoda od strane korisnika, te parametri koji se tiču samog korisnika odnosno samog proizvoda, kao npr. izbor određene dijete kada je u pitanju korisnik ili emisija CO_2 kada je u pitanju proizvod. Na slici 3.1 se može vidjeti UML dijagram koji prikazuje model strukture podataka koji će se spremati u bazu, te veze između istih.

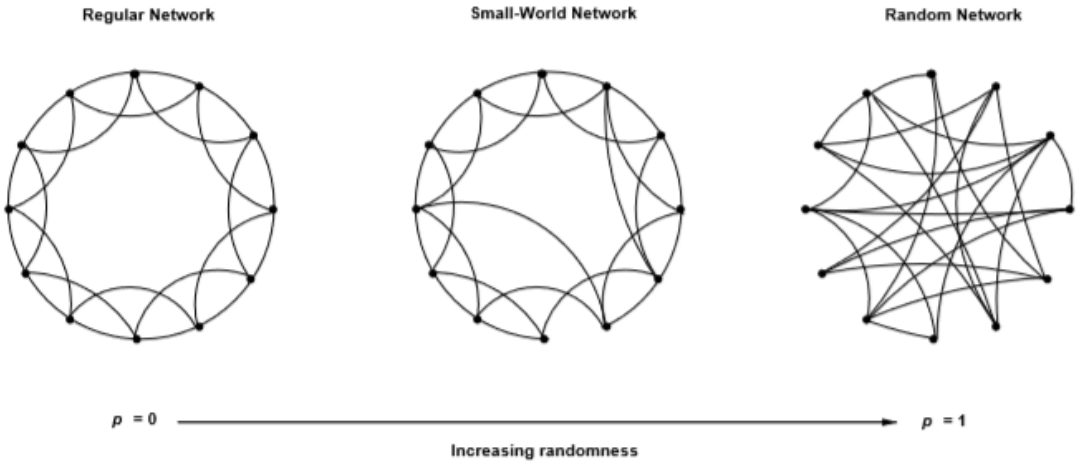


Sl. 3.1. *UML dijagram baze podataka*

3.1.4. Sustav preporuka temeljen na analizi grafova

U kontekstu ovog rada koristit će se više kriterija po kojima će se kreirati preporuke, a u ovom potpoglavlju će biti objašnjen sam način korištenja danih parametara za kreiranje smislene preporuke.

Koncept analiziranja grafova za kreiranje preporuka je već dugo zastupljen u različitim sustavima. Jedan od najranijih istraživačkih radova na tu temu je [13]. Cilj rada je bio modeliranje veza između ljudi u mreži te predviđanje zajedničkih interesa, koristeći elektroničke poruke između korisnika kao veze. Takav način korištenja čvorova i veza među njima je primijenjen u velikom broju područja kao npr. programi za pretraživanje Internet sadržaja, traženje veza među poznatim kriminalcima, medicinska primjena itd. Na slici 3.2 preuzetoj iz [13] mogu se vidjeti primjeri povezanih grafova s povećavanjem stupnja nasumičnosti (eng. *randomness*).



Sl. 3.2. *Prikaz nastanka nasumične mreže [13]*

Sa sigurnošću se može reći da su mreže u stvarnim sustavima puno više nalik na desnu mrežu na slici 3.2. Većina mreža koje predstavljaju realan model podataka imaju vrlo visok stupanj nasumičnosti, što je za očekivati jer su glavni parametri u takvim mrežama ljudi, pa je realno za pretpostaviti da neće imati simetrične interese i sklonosti. Preporuke za ovakav sustav je puno lakše predočiti i ostvariti koristeći bazu podataka baziranu na grafovima. Glavni razlog tome je činjenica da su u bazama podataka baziranim na grafovima veze između čvorova definirane kao entiteti koji imaju istu važnost kao i same strukture podataka u bazi (što nije slučaj kod npr. baza podataka baziranih na SQL-u). S druge strane, prema istraživanjima znanstvenika na Sveučilištu u

Mississippi-ju [14], relacijske baze podataka imaju bolje performanse kod upita podatkovnog tipa (upiti gdje se npr. traži točno određen skup objekata koji odgovaraju specificiranom parametru). Iz tog razloga, u okviru ovog rada će se koristiti primarno MySQL baza podataka, a samo za potrebnu funkcionalnost stvaranja preporuka će se koristiti Neo4j, baza podataka koja će se detaljnije objasniti u četvrtom poglavlju.

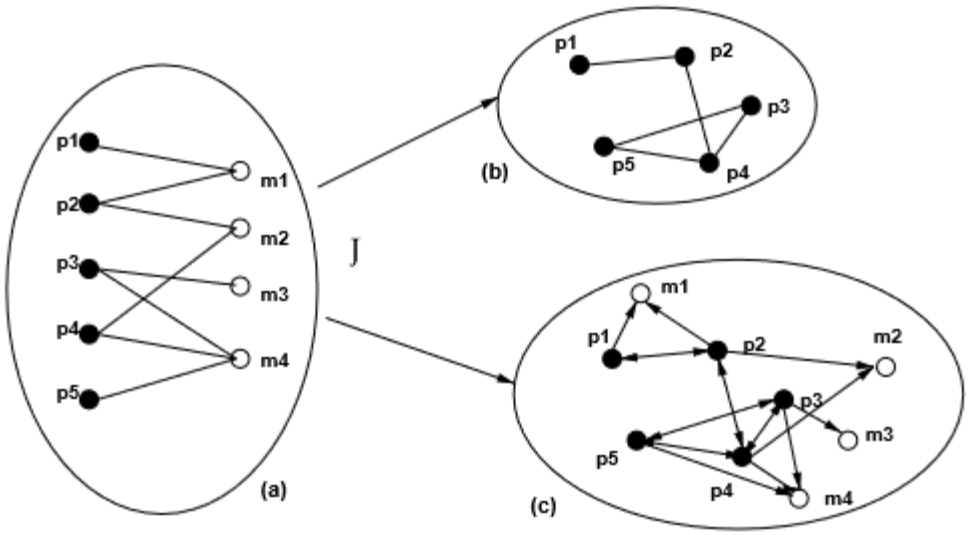
U [15] je predstavljen koncept skokova kroz veze između čvorova (eng. *jumping connections*) kako bi se prikazao sustav parametara koji se koristi, te preporuke koje se dobivaju analizom tih parametara.

Skup podataka koji se koristi za preporuke može biti bilo kakva vrsta interakcije između objekata u sustavu. U slučaju ovog rada, skup podataka će prikazivati veze između korisnika i proizvoda te će biti podijeljen na tri glavna tipa:

- korisnik je „pogledao“ određeni proizvod
- korisnik je kupio određeni proizvod
- korisnik je ocijenio određeni proizvod ocjenom od jedan do deset

Takav skup podataka se može prikazati bipartitnim grafom. Bipartitni graf je graf čiji se čvorovi mogu jasno podijeliti u dva tipa. Može se reći da se takav graf sastoji od tri skupa podataka. Dva skupa (proizvodi i korisnici) su modelirani kao čvorovi, a veze između njih su modelirane kao poveznice između tih čvorova. Veze obavezno imaju definiran tip (ime veze) a također mogu imati i pridružene parametre (npr. vremenska oznaka, ako je bitno u kojem trenutku je veza između dva čvora stvorena).

Skok (eng. *jump*) je funkcija koja kao parametar uzima strukturu grafa te kao rezultat vraća parove objekata (može se odnositi na parove korisnik-korisnik ili parove korisnik-objekt). Skok je osnovna funkcija kojom se u ovakvim strukturama podataka dolazi do preporuka. Bitno je naglasiti da definicija skoka ne specificira kriterije prema kojima se mapiranje izvršava niti specificira da li je potrebno koristiti cijelu strukturu grafa ili samo određeni podskup. Ti parametri trebaju biti definirani u logici same aplikacije. Na slici 3.3 preuzetoj iz [15] se može vidjeti primjer bipartitnog grafa, grafa socijalne mreže te grafa preporuka.



Sl. 3.3. *Ilustracija skokova između čvorova grafa; a) bipartitni graf korisnika i objekata, b) graf društvene mreže, c) graf preporuka [15]*

Opisani postupak skokova kroz čvorove grafa će se primijeniti u četvrtom poglavlju, pri ostvarenju aplikacije. To je osnovni postupak koji će se koristiti pri definiranju pravila koja će se nadalje koristiti za uključivanje i isključivanje pojedinih čvorova u postupku stvaranja preporuka.

3.2. Nefunkcionalni zahtjevi na aplikaciju

S obzirom na to da se radi o sustavu za prodaju proizvoda i kreiranje preporuka, postoje vrlo važni nefunkcionalni zahtjevi koji se moraju ispuniti.

Sigurnost je bitna stavka jer se radi o kupnji i prodaji proizvoda. Prema tome, korisničke lozinke moraju biti adekvatno zaštićene. Također, potrebno je uzeti u obzir i mogućnost napada na sustav korištenjem metoda kao što je npr. CSRF (eng. *Cross-Site Request Forgery*) gdje se u aplikaciju putem formi ubacuje programski kod koji se kasnije može izvršiti radi krađe ili neovlaštene izmjene podataka na poslužitelju. Nadalje, sustav mora biti pouzdan, tako da se korisnici mogu sa sigurnošću osloniti na preporuke i informacije koje im sustav daje. Ovo podrazumijeva točnost podataka koji opisuju određeni proizvod, ali i pravilno odlučivanje o izdavanju upozorenja i preporuka. Logiku koja se ovdje koristi je potrebno ispitati na dovoljno velikom skupu podataka da bi se bilo kakve moguće greške mogle otkloniti.

Sljedeći zahtjev je brzina izvođenja aplikacije, što je usko povezano s korisničkim iskustvom. Radnje kao što su npr. dodavanje proizvoda u košaricu, kupnja proizvoda i razgovor između korisnika moraju biti implementirani na takav način da ne zahtijevaju puno vremena za izvođenje, te da je cjelokupno iskustvo fluidno i intuitivno. To se postiže učinkovitim programskim kodom, te prikazivanjem adekvatnih notifikacija tako da korisnik zasigurno razumije što se događa u aplikaciji. Prema tome i korisničko sučelje mora pratiti ista načela. Treba biti transparentno i dovoljno razumljivo za velik broj korisnika.

4. PROGRAMSKO RJEŠENJE WEB SUSTAVA ZA PRODAJU PROIZVODA SA SUSTAVOM DAVANJA PREPORUKA I IZDAVANJA UPOZORENJA

Budući da je aplikacija razvijena u okviru ovog rada napravljena za prodavače i korisnike bez obzira na njihovo područje djelovanja te radi dostupnosti široj publici, ista je napisana na engleskom jeziku, ali će se u ovom radu prevesti odnosno objasniti na hrvatskom jeziku. Za mogućnosti testiranja i prikaza aplikacije u svrhu ovog rada, aplikacija će biti postavljena na lokalni poslužitelj XAMPP te pokrenuta na Internet pregledniku Google Chrome.

4.1. Programske tehnologije, okviri i arhitekture korištene za izradu web aplikacije

U nastavku će biti objašnjene one tehnologije koje su korištene za izradu aplikacije u okviru ovog rada. Stare i poznate tehnologije ne zahtijevaju opširno opisivanje, tako da će biti opisane ukratko.

4.1.1. HTML

HTML (eng. *HyperTextMarkupLanguage*) je opisni jezik za izradu web stranica. Pomoću HTML-a stvaraju se hipertekstualni dokumenti, oblikuje se njihov sadržaj te se stvaraju hiperveze. Svoju raširenost zahvaljuje jednostavnosti i činjenici da je od početka bio zamišljen kao besplatan i tako dostupan svima. Prikaz hipertekst dokumenta omogućuje Internet preglednik.

4.1.2. CSS i Bootstrap

CSS (eng. *Cascading Style Sheets*) je stilski opisni jezik, koji se koristi za opis prikaza dokumenta napisanog u jednom od opisnih (npr. HTML ili XML) jezika. CSS-om se uređuje sam izgled i raspored stranice. Bootstrap je razvojna cjelina otvorenog programskog koda koja služi za dizajn web stranica i web aplikacija. Sastoji se od predložaka za dizajn koji su bazirani na HTML-u i CSS-u te se odnose na tipografiju, forme, tipke, navigaciju te druge komponente sučelja, kao i JavaScript ekstenzije.

4.1.3. JavaScript i jQuery

JavaScript je skriptni programski jezik koji se izvršava u Internet pregledniku na strani korisnika. JavaScript s AJAX (Asynchronous JavaScript And XML) tehnologijom omogućuje web aplikacijama komunikaciju s programom na poslužiteljskoj strani, što čini web aplikacije interaktivnijim i lakšim za korištenje. JQuery je JavaScript biblioteka napravljena s ciljem pojednostavljenja skriptiranja na strani klijenta. Sintaksa je dizajnirana tako da olakšava

pretraživanje dokumenta, odabir određenih elemenata u dokumentu, kreiranje animacija, rukovanje događajima (eng. *events*), te korištenje AJAX tehnologije.

4.1.4. PHP, Symfony i MVC

PHP (eng. *PHP: Hypertext Preprocessor*) [16] je programski jezik baziran na C i Perl sintaksi, namijenjen prvenstveno programiranju dinamičnih web stranica. Ističe se širokom podrškom raznih baza podataka i Internet protokola kao i raspoloživosti brojnih programskih biblioteka. PHP programski kod se može ugraditi u HTML strukturu i koristiti u kombinaciji s raznim modulima za obrasce (eng. *templating engines*), sustavima za upravljanje te razvojnim cjelinama. PHP programski kod prevodi PHP prevoditelj (eng. *interpreter*). Symfony [17] je razvojna cjelina otvorenog programskog koda (eng. *open source*) za web aplikacije bazirana na programskom jeziku PHP. Razvio ga je Fabien Potencier s ciljem razvoja web aplikacija prateći MVC (eng. *Model-View-Controller*) arhitekturu. Koncept MVC arhitekture podrazumijeva korištenje modela koji definiraju informacije, upravitelja koji definiraju logiku aplikacije te pogleda koji prikazuju informacije definirane u modelima na način koji određuje upravitelj.

4.1.5. Baza podataka Neo4j

Prema [18], Neo4j je NoSQL baza podataka otvorenog programskog koda bazirana na grafovima. Glavna prednost ove baze je mogućnost spremanja velike količine povezanih podataka. Kako količina podataka raste, tako raste i broj veza među njima. Neo4j podržava upravljanje velikim količinama podataka bez ugrožavanja broja tipova i veza među istima. Svaki čvor (eng. *node*) i veza (eng. *relationship*) može imati proizvoljan broj parametara. Neo4j koristi posebne strukture podataka kako bi se podaci spremali i dohvaćali na učinkovit način. Za razliku od relacijskih baza podataka, kompleksni upiti nad podacima se ne računaju u trenutku izvršavanja upita. Umjesto takvog konvencionalnog pristupa, veze između entiteta su spremljene izravno. Kao što se u relacijskim bazama podataka najčešće koristi SQL, tako se u Neo4j koristi CQL (eng. *Cypher query language*). Sintaksa CQL-a je vrlo razumljiva i intuitivna te se ne koristi samo za modeliranje i spremanje podataka, nego i za izvršavanje upita nad podacima. Na slici 4.1 vidljiv je primjer jednog CQL upita.

```
MATCH (customer:User)-[:HAS_BOUGHT]->(:Product)<-[:HAS_BOUGHT]-
(otherCustomer:User)-[:HAS_BOUGHT]->(recommendation:Product)
WHERE c.id = 5 AND NOT (customer)-[:BOUGHT]->(recommendation)
RETURN recommendation.name, count(*) as reco_frequency
ORDER BY reco_frequency DESC LIMIT 100;
```

Sl. 4.1. *Primjer CQL upita*

4.1.6. Sustav za kreiranje preporuka Reco4PHP

Za stvaranje preporuka će biti korišten sustav *GraphAware Reco4PHP* [19], koji je napisan u PHP-u tako da je kompatibilan s razvojnim okruženjem Symfony te radi na temelju grafova u bazi podataka Neo4j. U nastavku će biti objašnjen sam način rada sustava, a u sljedećem potpoglavlju i konkretan primjer primjene u ovom radu. *Reco4PHP* radi na sljedeći način: proces određivanja preporuka počinje ulaznim čvorom (predstavlja korisnika za kojeg se preporuke kreiraju). Zatim se iz klase *DiscoveryEngine* poziva metoda *discoveryQuery*() koja kao parametar prima ranije spomenuti ulazni čvor. U samoj metodi se nalazi CQL upit koji definira način na koji se mapiranje treba izvršiti. Ovaj proces kao rezultat vraća skup mogućih preporuka koje još nisu filtrirane ni detaljnije analizirane. Nadalje, na skup potencijalnih preporuka se primjenjuju filteri u obliku klasa *Filter* i *Blacklist*. *Filteri* uspoređuju ulazni čvor s preporučenim čvorom i odlučuju je li pogodan za preporuku. *Blacklist* je predefiniran skup čvorova koji ne bi smjeli ući u preporuke za određenog korisnika npr. ako se korisniku preporučuje proizvod, treba osigurati da ga korisnik nije kupio ranije, jer onda preporuka ne bi imala smisla. Filtrirani skup preporuka se nadalje prosljeđuje klasi *PostProcess*, koja prolazi kroz cijeli skup preporuka te daje mogućnost pridruživanja bodova preporukama koje su po definiranom kriteriju bolje od drugih. Ovakav način rada omogućava vrlo precizno upravljanje preporukama definiranjem kriterija koji preporuke poredaju po važnosti te ih na kraju vraćaju samom korisniku.

4.2. Programsko rješenje sustava na strani poslužitelja

U ovom potpoglavlju će biti prikazana implementacija sustava na strani poslužitelja.

4.2.1. Registriranje i prijava korisnika

Registriranje i prijava korisnika su jedni od osnovnih dijelova svake web trgovine. Kako bi se definirao korisnik, potrebno je kreirati model koji će sadržavati parametre te upravitelj koji će izvršavati logiku registracije i prijave te potrebnih postupaka spremanja podataka. Bitno je naglasiti da se korisnička lozinka ne sprema u izvornom obliku, nego se prvo provlači kroz *hashing* algoritam. Taj algoritam kao rezultat daje niz znakova koji se spremaju u bazu (u ovoj aplikaciji korišten je algoritam *bcrypt*). Time se korisnički računi osiguravaju od neovlaštenog pristupa. Čak i ako netko pristupi bazi podataka, ne može znati korisničke lozinke jer su spremljene kao *hash*. Iz tog razloga se pri prijavi korisnika uzima šifra koju je upisao, provlači se kroz isti algoritam te ako rezultat odgovara zapisu u bazi, korisniku se omogućuje prijava u aplikaciju. Na slici 4.2 je prikazan dio programskog koda koji se izvršava pri registraciji korisnika.

```
$user = new User();

$form = $this->createForm(UserType::class, $user, array(
    'requiredPassword' => true,
    'user' => $user
));

$form->handleRequest($request);

if ($form->isSubmitted() && $form->isValid()) {

    $plainPassword = $user->getPassword();
    $encodedPassword = $encoder->encodePassword($user, $plainPassword);
    $user->setPassword($encodedPassword);

    $entityManager->persist($user);
    $entityManager->flush();
    $this->addFlash(
        'success',
        'Profile created successfully!'
    );
    return $this->redirectToRoute('app_login');
}

return $this->render('user/create.html.twig', array(
    'form' => $form->createView(),
    'user' => $user,
    'edit' => false
));
```

Sl. 4.2. *Programski kod za kreiranje korisničkog računa*

Iz programskog koda na slici 4.2 se može zaključiti sljedeće: ako forma nije poslana na poslužitelj, kreira se nova prazna forma te se ista prikazuje korisniku. Nakon što je korisnik upisao potrebne podatke i poslao formu, korisnička lozinka se enkriptira te se u enkriptiranom obliku sprema u bazu s ostalim podacima. U programskom kodu je vidljiv samo rad s lozinkom zato što se jedino ona ne sprema u izvornom obliku. Symfony automatski pridružuje podatke odgovarajućem objektu uz pomoć sučelja FormBuilder. Nakon uspješnog spremanja u bazu, korisniku se ispisuje odgovarajuća poruka.

4.2.2. Dodavanje proizvoda u košaricu

Za implementaciju košarice pogodno je koristiti *session* varijable. *Session* je poseban podatkovni prostor na poslužitelju koji se koristi za spremanje podataka, ali alociran je samo dok je korisnik prijavljen u aplikaciju. Kada se korisnik odjavi, brišu se sve *session* varijable koje mu pripadaju. Svaki korisnik ima svoj *session* u koji će se spremati proizvodi koje je dodao u košaricu. Kada korisnik klikne na tipku „Dodaj u košaricu“, poziva se AJAX zahtjev koji na asinkron način poziva pripadajuću funkciju na poslužitelju te joj predaje ID proizvoda. Takvim pristupom korisniku se omogućuje brzo dodavanje proizvoda u košaricu, bez da se svaki puta mora vraćati na popis proizvoda. Programski kod koji se izvršava pri dodavanju proizvoda u košaricu je vidljiv na slici 4.3.

```
$productID = $request->request->get('productID');
$quantity = (int)$request->request->get('quantity');
$product = $entityManager->getRepository('App:Product')-
>find($productID);
/** @var User $currentUser */
$currentUser = $this->getUser();
if($product && $quantity){
    $itemArray = array('id' => $productID, 'name'=>$product->getName(),
'description'=>$product->getDescription(), 'quantity'=>$quantity,
'price'=>$product->getPrice(), 'image'=>$product->getImage(),
'conflict'=>$conflict);
    $cartArray = $session->get('cart', null);
    $found = false;
    if($cartArray){
foreach($cartArray as $k => $v){
                if($productID === $v['id']){
                    $found = true;
                    $cartArray[$k]['quantity'] += $quantity;
                    if($product->getQuantity() <
$cartArray[$k]['quantity']){
                        $cartArray[$k]['quantity'] = $product-
>getQuantity(); }   }   }
        if (!$found){
            $cartArray[] = $itemArray; }
    } else{
        $cartArray = array();
        $cartArray[] = $itemArray;    }
    $session->set('cart', $cartArray);
    return new JsonResponse(['msg' => "Added product to cart", 'size'
=> count($cartArray) ], 200);}
return new JsonResponse(['msg' => 'There was an error.' ], 400);
```

Sl. 4.3. *Logika dodavanja proizvoda u košaricu*

Iz programskog koda na slici 4.3 je vidljivo da se prvo dohvaća proizvod i količina preko parametara poslanih u AJAX zahtjevu te ako oba parametra postoje, izvršavaju se uvjetne provjere koje određuju je li proizvod prethodno dodan u košaricu. Ako je, potrebna količina se nadodaje na postojeću. Ako nije, dodaje se samo količina poslana u zahtjevu. Na kraju se korisniku ispisuje poruka o uspješnom dodavanju proizvoda te se ažurira podatak o ukupnom broju proizvoda u košarici.

4.2.3. Korištenje PayPal API-ja za ostvarenje transakcija

PayPal je jedna od najkorištenijih usluga za plaćanje preko Interneta. Većina korisnika web trgovina preferira plaćati putem PayPala ili direktno putem kartice. Iz tog razloga, u ovoj aplikaciji su implementirana ta dva rješenja. Za tu svrhu pogodno je koristiti *PayPal Sandbox* [20] . To je virtualno testno okruženje koje simulira realne transakcije, ali bez upotrebe pravog novca. Nudi mogućnosti kreiranja lažnih korisničkih računa koji se mogu koristiti za simulaciju i testiranje

transakcija dok aplikacije nije postavljena na poslužitelj. Kada se kreira korisnički račun putem PayPala, otvara se mogućnost kreiranja testnih računa kako bi se izvršile virtualne transakcije. Pri tome se koristi PayPal Sandbox API kojemu se prosljeđuju osnovni parametri proizvoda u košarici (npr. količina, cijena itd.). PayPal dohvaća poslane parametre, računa ukupnu cijenu, a zatim prebacuje taj iznos s korisničkog računa kupca na korisnički račun prodavača, te je time završena jedna transakcija. Forma koja se šalje PayPal API-ju vidljiva je na slici 4.4.

```
<input type="hidden" name="cmd" value="_ext-enter">
<form class="PayPalform" action="https://www.sandbox.Paypal.com/us/cgi-
bin/webscr" method="post">
    <input type="hidden" name="cmd" value="_cart">
    <input type="hidden" name="business" value="sb-
3ogde1914139@business.example.com">
    <input type="hidden" name="upload" value="1">
    {% for item in cart %}
        <input type="hidden" name="item_name_{{ loop.index }}"
value="{{item["name"]}}">
        <input type="hidden" name="quantity_{{ loop.index }}"
value="{{item["quantity"]}}">
        <input type="hidden" name="amount_{{ loop.index }}"
value="{{item["price"]}}">
    {% endfor %}
    <input type="hidden" name="currency_code" value="EUR">
    <input type="image" src="http://www.PayPal.com/en_US/i/btn/x-click-
but01.gif" name="submit" alt="Make payments with PayPal - it's fast, free
and secure!">
</form>
```

Sl. 4.4. *HTML forma za ostvarenje transakcija putem PayPal Sandbox usluge*

4.2.4. Postavljanje baze Neo4j

Da bi se uopće moglo doći do preporuka, prvo se trebaju modelirati podaci u bazi Neo4j. To je ostvareno korištenjem Cyphera, točnije CQL naredbi koje će biti detaljnije objašnjene u nastavku. Na slici 4.5 je vidljiv primjer CQL koda koji modelira novi proizvod kao novi čvor te kreira sve potrebne veze između istog i ostalih čvorova.

```
$parameters = [ 'productID' => $product->getId(),
                'categoryID' => $product->getCategory()->getId(),
                'name' => $product->getName(),
                'proteinPercent' => $product->getProteinPercent(),
                'lipidPercent' => $product->getLipidPercent(),
                'carbohydratePercent' => $product->getCarbohydratePercent(),
                'emission' => $product->getGasEmission()
];
$query = 'MATCH (category:Category {categoryID: {categoryID}})
          CREATE (product:Product {productID: {productID}, name: {name},
proteinPercent: {proteinPercent}, lipidPercent: {lipidPercent},
carbohydratePercent: {carbohydratePercent}, emission: {emission}})
CREATE (product)-[rel:BELONGS_TO]->(category)
          WITH product ';

foreach ($product->getConditions() as $index => $condition){
    $query .= 'MATCH (condition'.$index.':Condition {conditionID:
{condition'.$index.'ID}})
              CREATE (product)-[rel:IS_DANGEROUS_TO]->(condition'.$index.')
              WITH product
              ';
    $parameters['condition'.$index.'ID'] = $condition->getId();
}

foreach ($product->getDiets() as $index => $diet){
    $query .= 'MATCH (diet'.$index.':Diet {dietID: {diet'.$index.'ID}})
              CREATE (product)-[rel:IS_EXCLUDED_FROM]->(diet'.$index.')
              WITH product ';
    $parameters['diet'.$index.'ID'] = $diet->getId();
}

$query .= 'RETURN product';
$client->run($query, $parameters);
```

Sl. 4.5. *Logika dodavanja novog proizvoda u bazu*

Ključna riječ CREATE kreira novi čvor ili vezu s parametrima predanim kao argument. Sintaksa je sljedeća: prvo se navodi identifikator čvora ili veze (identifikator je moguće proslijediti ostatku CQL koda koristeći ključnu riječ WITH), zatim se, poslije dvotočke, navodi tip čvora ili veze te parametri. Koristeći ključnu riječ MATCH moguće je dohvatiti postojeći čvor ili vezu.

4.2.5. Logika stvaranja preporuka

Kao što je spomenuto u prošlom potpoglavlju, do preporuka se dolazi korištenjem klasa *DiscoveryEngine*, *Filter*, *Blacklist* te *PostProcess*. U nastavku će biti prikazani konkretni primjeri tih klasa korišteni u ovom radu. Postupak skakanja kroz čvorove opisan u trećem poglavlju će se ovdje primijeniti tako da će biti definirana konkretna pravila za skakanje kroz čvorove te pronalaženje čvorova koje treba uključiti ili isključiti iz skupa preporuka. Prvo je potrebno

implementirati metode za otkrivanje preporuka discoveryQuery(). Metode su vidljive na slikama 4.6 i 4.7.

```
public function discoveryQuery(Node $input, Context $context) :
StatementInterface
{
    $query = 'MATCH (input:User) WHERE id(input) = {id}
    MATCH (input)-[:BOUGHT]->(product)<-[:BOUGHT]-(o)
    WITH distinct o
    MATCH (o)-[:BOUGHT]->(reco)
    RETURN distinct reco LIMIT 100';

    return Statement::create($query, ['id' => $input->identity()]);
}
```

Sl. 4.6. *Logika otkrivanja preporuka bazirana na povijesti kupnje korisnika*

```
public function discoveryQuery(Node $input, Context $context) :
StatementInterface
{
    $query = 'MATCH (input:User) WHERE id(input) = {id}
    MATCH (input)-[:IS_USING]->(diet)
    WITH diet
    MATCH (diet)<-[:IS_USING]-(user)-[:BOUGHT]->(reco)
    RETURN distinct reco LIMIT 100';

    return Statement::create($query, ['id' => $input->identity()]);
}
```

Sl. 4.7. *Logika otkrivanja preporuka bazirana na dijeti koju korisnik primjenjuje*

Iz slike 4.6 se može zaključiti sljedeće: pretpostavka je da je korisnik kojem se generiraju preporuke već kupio neke proizvode. Prema tome, dohvaćaju se korisnici koji su kupili iste proizvode kao i početni korisnik, a zatim se dohvaćaju proizvodi koje su također kupili ti korisnici. Ti proizvodi čine početni skup preporuka za prvi *DiscoveryEngine*. Ovdje se javlja potencijalni problem, a to je slučaj u kojem je korisnik već kupio neke od preporučenih proizvoda. Taj problem će se riješiti u nastavku, gdje će se pomoću *Blacklista* upravo takvi proizvodi izbaciti iz skupa preporuka. Na slici 4.7 se vidi drugi dio logike otkrivanja preporuka, gdje se dohvaća dijeta koju korisnik primjenjuje, pa se zatim dohvaćaju proizvodi koje su kupili korisnici koji također primjenjuju tu istu dijetu.Nakon procesa otkrivanja, na skup proizvoda se primjenjuju filteri tipa *Blacklist.* Metode blacklistQuery() koje sadrže logiku je moguće vidjeti u nastavku.

```
public function blacklistQuery(Node $input)
{
    $query = 'MATCH (input) WHERE id(input) = {inputId}
    MATCH (input)-[:BOUGHT]->(product)
    RETURN distinct product as item';

    return Statement::create($query, ['inputId' => $input->identity()]);
}
```

Sl. 4.8. *Logika izbacivanja prethodno kupljenih proizvoda iz skupa preporuka*

```
public function blacklistQuery(Node $input)
{
    $query = 'MATCH (input) WHERE id(input) = {inputId}
    MATCH (input)-[:HAS_PROBLEMS_WITH]->(condition)<-[:IS_DANGEROUS_TO]-
(product)
    RETURN distinct product as item';

    return Statement::create($query, ['inputId' => $input->identity()]);
}
```

Sl. 4.9. *Logika izbacivanja proizvoda koji su neadekvatni za korisnika s obzirom na zdravstveno stanje iz skupa preporuka*

```
public function blacklistQuery(Node $input)
{
    $query = 'MATCH (input) WHERE id(input) = {inputId}
    MATCH (input)-[:IS_USING]->(diet)<-[:IS_EXCLUDED_FROM]-(product)
    RETURN distinct product as item';

    return Statement::create($query, ['inputId' => $input->identity()]);
}
```

Sl. 4.10. *Logika izbacivanja proizvoda koji su neadekvatni za korisnika s obzirom na dijetu iz skupa preporuka*

CQL upit na slici 4.8 prvo dohvaća trenutnog korisnika a zatim vraća sve proizvode koje je isti već kupio. CQL upit na slici 4.9 prvo dohvaća trenutnog korisnika i njegova zdravstvena stanja te vraća sve proizvode koji su za ta zdravstvena stanja nepogodni. CQL upit na slici 4.10 prvo dohvaća trenutnog korisnika i moguću dijetu koju koristi te vraća sve proizvode koji su za zu dijetu nepogodni. Konačno, nakon filtera, na skup preporuka primjenjuju se klase *PostProcess* čija svrha je dodjeljivanje dodatnih bodova proizvodima koji se po određenom kriteriju ističu. U nastavku su prikazani konkretni primjeri korištenja *PostProcessa*. Svaki *PostProcess* sadrži dvije metode. Metoda buildQuery() izvršava CQL upit i vraća podatke potrebne za odlučivanje o dodjeljivanju

bodova, a metoda postProcess() dohvaća podatke koje je metoda buildQuery() vratila te ovisno o njihovim vrijednostima dodaje bodove preporukama.

```
public function buildQuery(Node $input, Recommendations $recommendations)
{
    $query = 'UNWIND {ids} as id
    MATCH (n) WHERE id(n) = id
    RETURN id(n) as id, n.emission as emission';

    $ids = [];
    foreach ($recommendations->getItems() as $item) {
        $ids[] = $item->item()->identity();
    }

    return Statement::create($query, ['ids' => $ids]);
}

public function postProcess(Node $input, Recommendation $recommendation,
Record $record)
{
    $emission = $record->get('emission');

    if($emission <= 10 && $emission >= 0){
        $score = 10;
    }
    else if ($emission <= 25 && $emission > 10 ){
        $score = 5;
    }
    else{
        $score = 0;
    }
    $recommendation->addScore($this->name(), new SingleScore($score,
'emission'));
}
```

Sl. 4.11. *Logika dodavanja bodova proizvodima koji imaju manju emisiju stakleničkih plinova*

```
public function buildQuery(Node $input, Recommendations $recommendations)
{
    $query = 'UNWIND {ids} as id
    MATCH (n) WHERE id(n) = id
    MATCH (input:User) WHERE id(input) = {id}
    RETURN id(n) as id, EXISTS ((input)-[:VIEWED]->(n)) as has_viewed';

    $ids = [];
    foreach ($recommendations->getItems() as $item) {
        $ids[] = $item->item()->identity();
    }

    return Statement::create($query, ['ids' => $ids, 'id' => $input-
>identity()]);
}

public function postProcess(Node $input, Recommendation $recommendation,
Record $record)
{
    $score = 0;
    if ($record->get('has_viewed')){
        $score = 5;
    }
    $recommendation->addScore($this->name(), new SingleScore($score));
}
```

Sl. 4.12. *Logika dodavanja bodova proizvodima koji se nalaze u korisnikovoj povijesti pregledanih proizvoda*

```
public function buildQuery(Node $input, Recommendations $recommendations)
{
    $query = 'UNWIND {ids} as id
    MATCH (n) WHERE id(n) = id
    MATCH (n)<-[r:RATED]-(u)
    WITH n, count(r) as num_ratings, reduce(total=0, number in r.rating |
total + number) as full_rating
    RETURN id(n) as id, toFloat(full_rating) / num_ratings as score';

    $ids = [];
    foreach ($recommendations->getItems() as $item) {
        $ids[] = $item->item()->identity();
    }

    return Statement::create($query, ['ids' => $ids]);
}

public function postProcess(Node $input, Recommendation $recommendation,
Record $record)
{
    $recommendation->addScore($this->name(), new SingleScore($record-
>get('score')));
}
```

Sl. 4.13. *Logika dodavanja bodova proizvodima koji su dobro ocijenjeni od strane drugih korisnika*

Na slici 4.11 je prikazan *PostProcess* koji dodjeljuje bodove s obzirom na emisiju stakleničkih plinova. Metoda buildQuery() najprije dohvaća sve preporučene proizvode te njihove parametre emisije stakleničkih plinova izražene u CO_2-e. Metoda postProcess() dodaje bodove preporučenim proizvodima, i to kako slijedi: ako je vrijednost emisije između 0 i 10 CO_2-e dodaje se deset bodova, ako je vrijednost emisije između 10 i 25 CO_2-e dodaje se 5 bodova i konačno, ako je vrijednost emisije više od 25 CO_2-e ne dodaju se bodovi. Na slici 4.12 je prikazan *PostProcess* koji dodaje bodove ovisno o tome je li korisnik pogledao određeni proizvod. Metoda buildQuery() na prethodno opisan način dohvaća preporučene proizvode, ali u ovom slučaju vraća informaciju o tome postoji li veza između čvorova korisnika i proizvoda koja govori da je korisnik pogledao proizvod. Ako takva veza postoji, proizvodu se dodaje 5 bodova. Na slici 4.13 je prikazan zadnji *PostProcess* koji dodjeljuje bodove ovisno o ukupnoj ocjeni proizvoda. Metoda buildQuery() zbraja ukupne ocjene proizvoda te ih dijeli s brojem ocjena. Taj podatak se prosljeđuje metodi postProcess() koja dodaje bodove u iznosu koji je jednak prosječnoj ocjeni za taj proizvod.

4.3. Programsko rješenje sustava na strani klijenta

U ovom potpoglavlju će biti prikazana implementacija sustava na strani klijenta.

4.3.1. Dodavanje i validacija proizvoda

Za dodavanje novih entiteta u bazu pogodno je koristiti koncept *Form Buildera* u Symfony razvojnom okruženju. Rad s običnim HTML formama je vrlo repetitivan i dugotrajan jer programer mora prikazati sve potrebne forme, napraviti validaciju forme i mapirati podatke iz poslanih formi u objekte. *Form Builder* automatizira velik dio posla i nudi dodatne mogućnosti. Na slici 4.14 je prikazan *Form Builder* za prikazivanje forme koja se koristi pri dodavanju novog proizvoda u bazu.

```
$builder
    ->add('name', TextType::class, [
        'attr' => ['autofocus' => true],'label' => 'Name'])
    ->add('description', TextareaType::class, [
        'label' => 'Description'    ])
    ->add('quantity', IntegerType::class, ['label' => 'Quantity (stock)'])
    ->add('price', NumberType::class, ['scale' => 2,'label' => 'Price (EUR)'])
    ->add('category', EntityType::class, ['class' => Category::class,
        'choice_label' => function ($category) {
            return $category->getName();        }    ])
    ->add('image', FileType::class, ['label' => 'Image (jpg/jpeg or png file,
leaving this empty will not interfere with existing image)',
        'mapped' => false, 'required' => false, 'constraints' => [
new File([     'maxSize' => '10M','mimeTypes' => [
               'image/jpeg','image/png',],
```

```
'mimeTypesMessage' => 'Please upload a valid .jpeg or .png image',     ])
],  ])
    ->add('proteinPercent', IntegerType::class, [
        'label' => 'Protein percent in macronutrient ratio'])
    ->add('carbohydratePercent', IntegerType::class, [
        'label' => 'Carbohydrate percent in macronutrient ratio'])
    ->add('lipidPercent', IntegerType::class, [
        'label' => 'Lipid percent in macronutrient ratio'
    ])
->add('conditions', EntityType::class, array(
        'class' => Condition::class,
        'multiple' => true, 'required' => false,
        'label' => 'If your product contains any of the ingredients below,
please check them', 'expanded' => true,
        'choice_label' => function ($condition) {
            return ($condition->getDescription());    }    ))
    ->add('diets', EntityType::class, array(
        'class' => Diet::class,
        'multiple' => true,'required' => false,
        'label' => 'If your product contains any of the ingredients below,
please check them',
        'expanded' => true,
        'choice_label' => function ($diet) {
            return ($diet->getDescription());    }    ))
    ->add('gasEmission', IntegerType::class, [
        'label' => 'Gas emission in CO2-e'    ])
    ->add('submit', SubmitType::class, [
        'label' => 'Submit',
        'attr' => array('class' => 'btn btn-default')
    ]);
```

Sl. 4.14. *Generiranje forme za dodavanje proizvoda u bazu*

Prikazana forma sadrži sve potrebne tipove podataka za profiliranje proizvoda, kao npr. emisiju stakleničkih plinova i omjer makronutrijenata koje proizvod sadržava. Objektu *$builder* se dodaju elementi forme koristeći metodu add() koja se može ulančavati. Svakom elementu forme se mogu dodavati određeni parametri kao npr. naziv, tip podatka koji se u tom elementu koristi, je li nužno ispuniti element forme itd.

Nakon što je korisnik ispunio formu, svi podaci prolaze kroz validaciju u obliku pravila koja se definiraju u samom modelu. To su obično pravila kao npr. minimalna i maksimalna duljina stringa, dozvoljeni tipovi datoteka kod postavljanja na poslužitelj itd. U ovom slučaju je bilo potrebno implementirati vlastito pravilo za validaciju omjera makronutrijenata, korištenjem klase *ConstraintValidator* čija je metoda validate() vidljiva na slici 4.15.

Validacija forme se izvršava tako da se, nakon što je korisnik kliknuo na tipku „Submit“, prvo provjerava zadovoljava li forma pravila napisana u metodi validate(). Ako ih ne zadovoljava, korisniku se ispisuje poruka koja objašnjava što treba ispraviti. U ovom slučaju omjer makronutrijenata mora biti izražen u postocima tj. kada se zbroje vrijednosti, zbroj mora biti 100.

```
public function validate($object, Constraint $constraint)
{
    if (!$object instanceof Product && !$object instanceof Diet) {
        throw new UnexpectedTypeException($constraint,
RatioConstraint::class);
    }
    if ($object->getCarbohydratePercent() + $object->getProteinPercent() +
$object->getLipidPercent() != 100 ) {
        $this->context->buildViolation($constraint->message)
            ->atPath('proteinPercent')
            ->addViolation();}}
```

Sl. 4.15. *Validacija omjera makronutrijenata*

4.3.2. Ocjenjivanje proizvoda

Nakon kupnje određenog proizvoda, korisniku se otvara mogućnost ocjenjivanja proizvoda ocjenom od 1 do 10. To je ostvareno korištenjem Javascript *plugina* RateYo. Na slici 4.16 je vidljiva inicijalizacija i konfiguracija *plugina*, te pozivanje AJAX zahtjeva kada korisnik ocijeni proizvod.

```
let rateElement = $("#rateYo");
rateElement.rateYo({
    onSet: function (rating, rateYoInstance) {
        $.ajax({
            url: "{{ path('product_rate') }}",
            method: 'POST',
            data: {
                productID: $('#addToCart').attr('data-id'),
                rating: rating }, }); }});
rateElement.rateYo({
    starWidth: "40px"});
rateElement.rateYo("option", "maxValue", 10);
rateElement.rateYo("option", "fullStar", true);
rateElement.rateYo("option", "numStars", 10);
{% if userRating %}
    rateElement.rateYo("option", "rating", {{ userRating }});
{% endif %}
```

Sl. 4.16. *Korištenje plugina RateYo*

4.3.3. Prikazivanje upozorenja i preporuka

Kao što je spomenuto ranije, proizvodi imaju profile koji sadrže podatke o prehrambenim i ekološkim parametrima. Konkretnije, svaki proizvod u bazi sadrži veze prema dijetama i zdravstvenim stanjima za koje nije pogodan. Svaki korisnik u svom profilu ima spremljene podatke o dijeti i zdravstvenom stanju. Prema tome, prije kupnje proizvoda potrebno je provjeriti podudaranje između profila proizvoda i korisnika i ako ono postoji, potrebno je ispisati upozorenje. Programski kod za provjeru podudaranja i ispisivanje upozorenja se može vidjeti na slici 4.17.

```
foreach($product->getConditions() as $condition){
    if($currentUser->getConditions()->contains($condition)){
        $conflict = true;
        break;
    }
}
foreach($product->getDiets() as $diet){
    if($currentUser->getActiveDiet() === $diet){
        $conflict = true;
        break;
    }
}
if($cartArray){
    foreach($cartArray as $item){
        if($item['conflict']){
            $this->addFlash(
                'warning','Your cart contains product(s) that conflict with
Your diet and/or medical condition profile. Please review products in the
cart before proceeding to checkout.');
 break;        }    }
    $cartSize = count($cartArray);
}
```

Sl. 4.17. *Provjera i ispisivanje upozorenja korisniku*

Ispisivanje preporuka korisniku je riješeno na sljedeći način: potrebno je registrirati servis u Symfonyu koji će predati identifikator korisnika i generirati preporuke koristeći sve komponente opisane u prošlom potpoglavlju. Zatim, skup objekata se šalje na stranu klijenta i prikazuje se u Twig pogledu. Programski kod koji prikazuje preporuke je vidljiv na slici 4.18.

```
<h1 align="center">Based on Your activity and profile, we recommend these
products:</h1>
<div class="container">
    <br>
    <div class="row">
        <div class="col-lg-1"></div>
        <div class="col-lg-10">
            <div class="row"><br>
                {% for product in recommendations %}
                    <div class="col-lg-4 col-md-6 mb-4">
                        <div class="card h-100">
<a href="{{ path('product_details', {id:product.id}) }}"><img class="card-
img-top" src="{{ asset('/uploads/images/' ~
product.image)|imagine_filter('fixed_filter_small') }}" alt=""></a>
                            <div class="card-body">
                                <h4 class="card-title">
                                    <a href="{{ path('product_details',
{id:product.id}) }}">{{ product.name }}</a>
                                </h4>
                                <h5>€{{ product.price }}</h5>Stock: {{
product.quantity }}
                                <p class="card-text">{{
product.description|length > 50 ? product.description|slice(0, 50) ~ '...' :
product.description  }}</p>
                            </div>
                            <div class="card-footer">
                                <small class="text-muted"></small>
                            </div>
                        </div>
                    </div>
                {% endfor %}
            </div>
        </div>
        <div class="col-lg-1"></div>
    </div>
</div>
```

Sl. 4.18. *Prikazivanje preporuka korisniku*

U ovom poglavlju prikazani su glavni dijelovi aplikacije na strani klijenta i na strani poslužitelja. U nastavku je prikazan sažetak u obliku tablice koja sadrži sve tehnologije korištene u ovom radu te njihovu primjenu (Tab. 4.1).

Tab. 4.1. *Lista tehnologija i odgovarajućih opisa primjene*

Ime tehnologije	**Opis primjene tehnologije**
HTML	Definiranje osnovnih elemenata u pogledu
CSS i Bootstrap	Definiranje izgleda HTML elemenata
JavaScript, JQuery i AJAX	Dinamičko dohvaćanje i manipulacija s HTML elementima, asinkrona komunikacija s poslužiteljem
Symfony (PHP, Doctrine)	Definiranje glavne logike aplikacije na poslužitelju (upravitelji, PHP), definiranje modela i spajanje istih s bazom (Doctrine)
MySQL	Spremanje podataka u relacijsku bazu
Neo4J	Spremanje podataka potrebnih za ostvarenje sustava preporuka, NoSQL
Reco4PHP	Definiranje pravila za stvaranje preporuka

5. PRIKAZ NAČINA RADA I ISPITIVANJE WEB SUSTAVA

U ovom poglavlju će se testirati web sustav ostvaren u okviru ovog rada. Izvršit će se nekoliko testnih slučajeva koji će zajedno prikazati sve bitne dijelove sustava te njihov način rada.

5.1. Definiranje testnih slučajeva

U ovom potpoglavlju kreirano je pet testnih slučajeva koji zajednički obuhvaćaju sve dijelove aplikacije te dobro prikazuju pojedine funkcionalnosti. U nastavku je prikazana tablica 5.1 koja sadrži opise, korake i odgovarajuće funkcionalnosti koje testni slučajevi pokazuju.

Tab. 5.1. *Testni slučajevi*

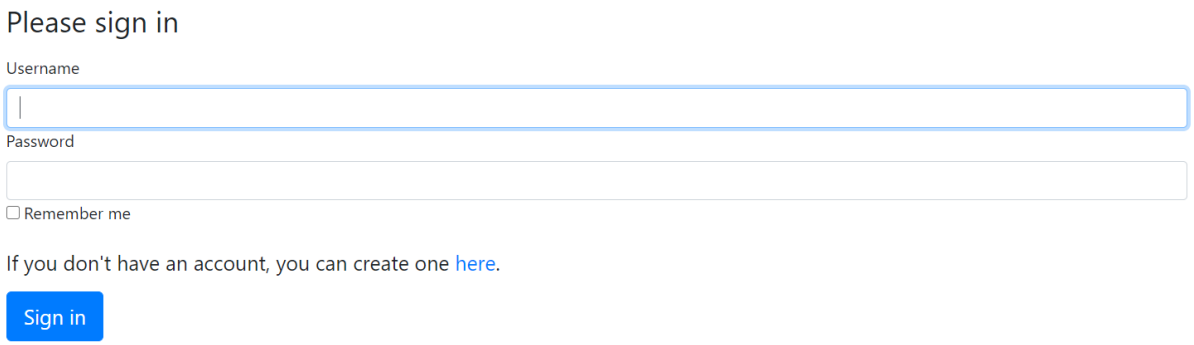
Funkcionalnost	Koraci	Opis
Registriranje, prijava i odjava korisnika	1. Otvaranje forme za registraciju novog korisnika 2. Ispunjavanje korisničkih podataka 3. Klik na tipku za registraciju korisnika 4. Prijava korisnika 5. Klik na tipku za odjavu korisnika	Provjera autentifikacije korisnika
Profiliranje novog proizvoda	1. Otvaranje forme za kreiranje novog proizvoda 2. Ispunjavanje profila novog proizvoda 3. Klik na tipku za kreiranje proizvoda	Provjera ispravnosti postavljanja novog proizvoda na prodaju
Komunikacija između korisnika	1. Otvaranje pogleda za komunikaciju 2. Slanje poruka drugom korisniku	Provjera ispravnog rada komunikacije među korisnicima

Kupnja proizvoda	1. Dodavanje željenih proizvoda u košaricu 2. Pregled košarice 3. Otklanjanje proizvoda koji predstavljaju problem s obzirom na dijetu ili zdravstveno stanje 4. Ostvarenje kupnje putem PayPal usluge	Provjera procesa dodavanja proizvoda u košaricu i kupnje proizvoda putem PayPal usluge
Pregled preporuka za korisnika	1. Otvaranje pogleda za prikaz preporuka 2. Uspoređivanje podataka u bazi s prikazanim preporukama	Provjera generiranja preporuka

5.2. Izvršavanje testnih slučajeva i pregled aplikacije

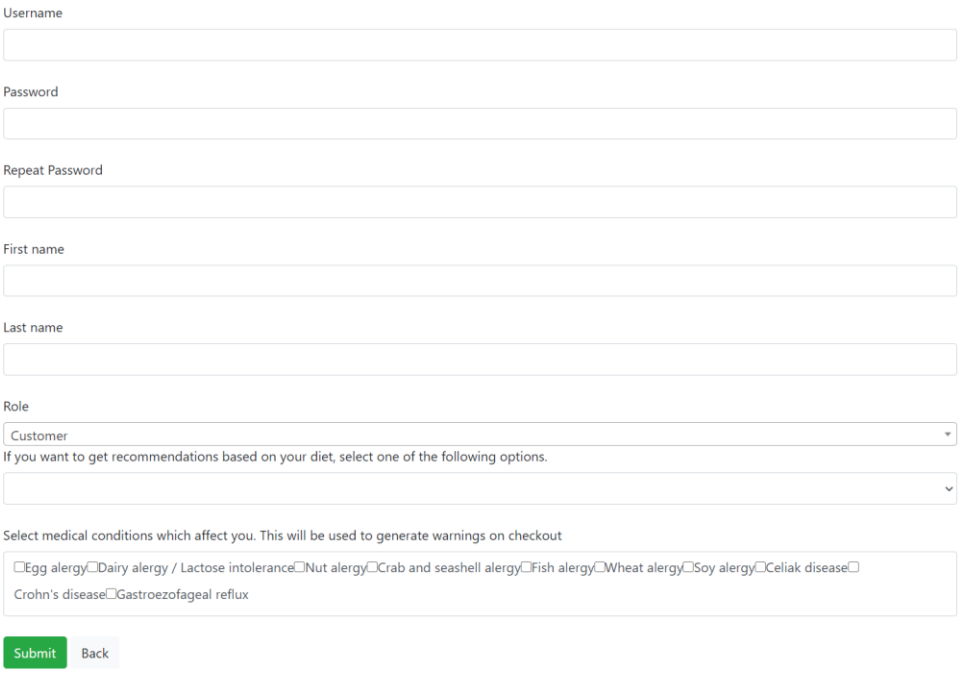
5.2.1. Registriranje, prijava i odjava korisnika

Pri otvaranju aplikacije prikazuje se forma za unos korisničkog imena i lozinke. Formu je moguće vidjeti na slici 5.1.



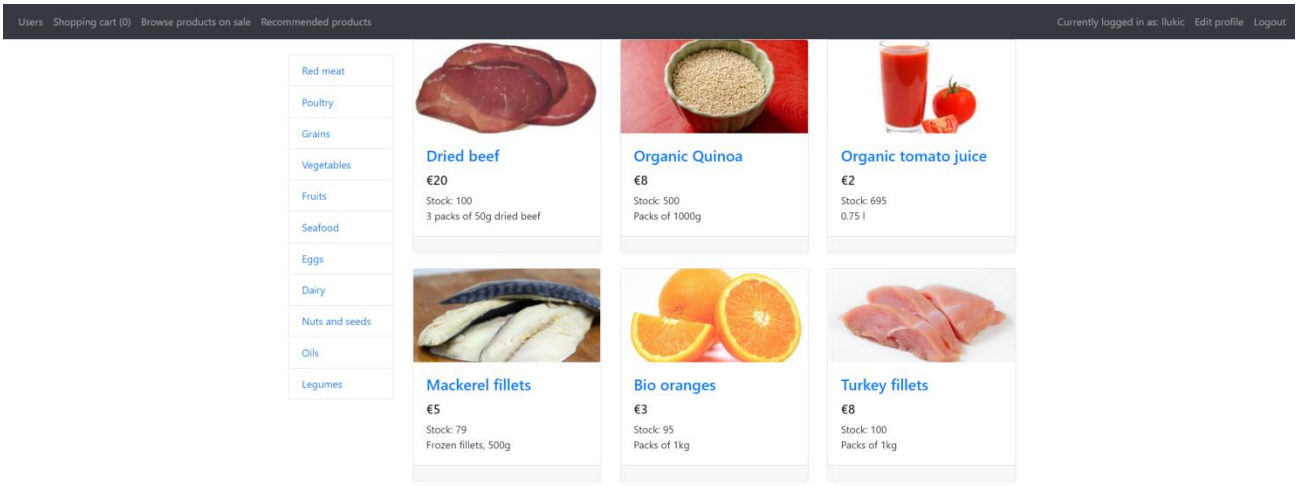
Sl. 5.1. *Forma za prijavu korisnika*

Nakon klika na poveznicu, prikazuje se forma za registraciju korisnika. Forma je vidljiva na slici 5.2.



Sl. 5.2. *Forma za registraciju korisnika*

Nakon što je ispunio sve potrebne podatke, korisnik klikom na tipku „Submit“ kreira vlastiti korisnički račun te se može prijaviti u aplikaciju. Izgled početnog ekrana aplikacije za kupce je prikazan na slici 5.3.

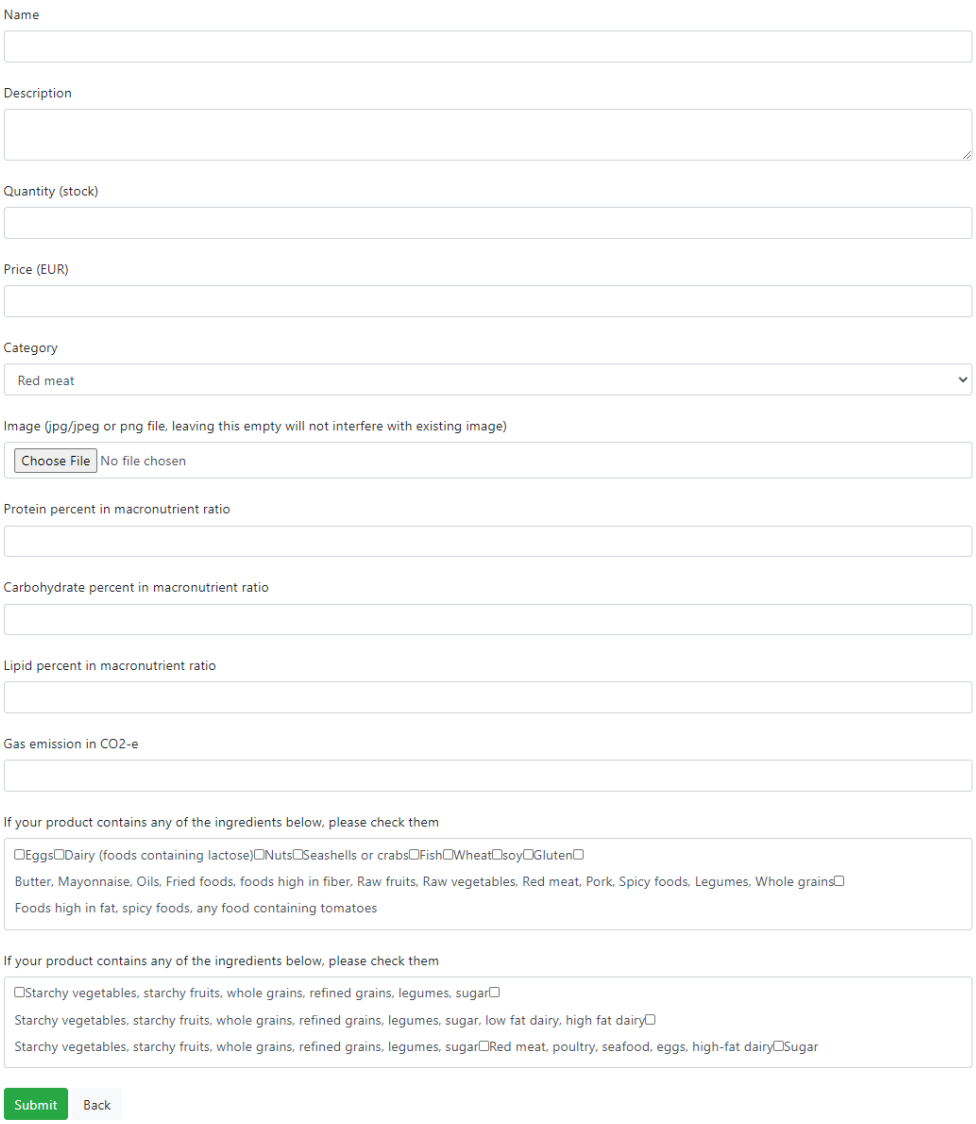


Sl. 5.3. *Početni ekran aplikacije*

Klikom na tipku „Logout“ korisnik se odjavljuje iz aplikacije.

5.2.2. Profiliranje novog proizvoda

Kada prodavač napravi svoj korisnički račun (koristeći prethodno opisan postupak), ima mogućnost postavljanja proizvoda na prodaju. Klikom na tipku „Add product“ otvara se forma za dodavanje proizvoda. Ista sadrži sve potrebne podatke koji se koriste u prodaji, prikazivanju i generiranju upozorenja i preporuka. Kupac može odabrati predefinirane kategorije, omjer makronutrijenata, emisiju stakleničkih plinova te sva zdravstvena stanja i dijete za koje je proizvod štetan. Forma je vidljiva na slici 5.4.

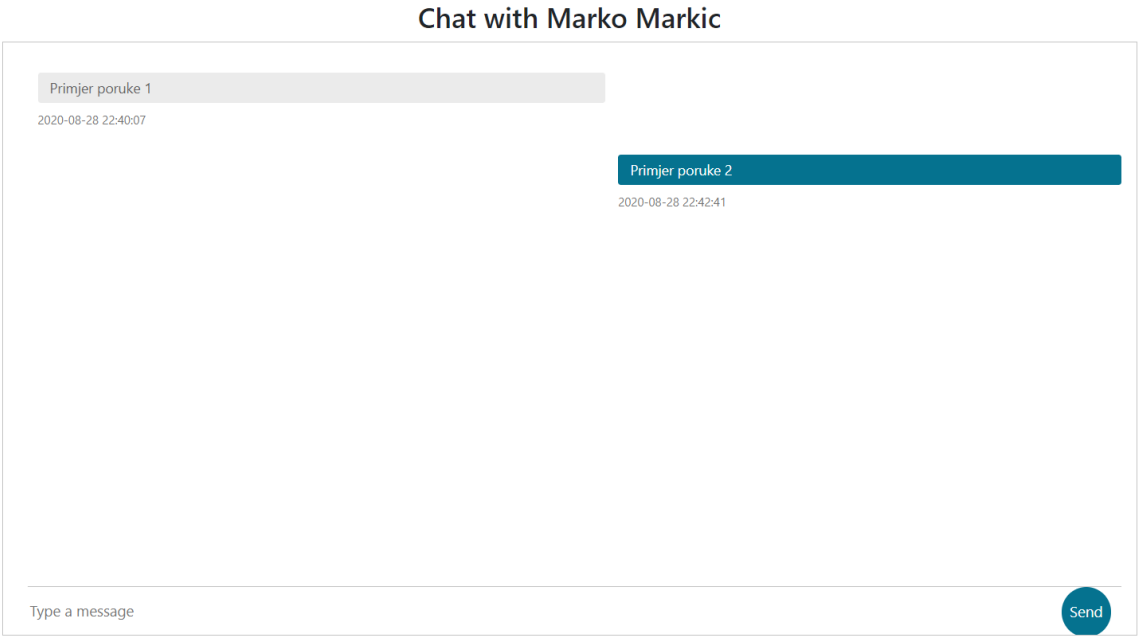


Sl. 5.4. *Forma za kreiranje proizvoda*

Klikom na tipku „Submit“ proizvod je postavljen na prodaju i korisnici ga mogu vidjeti prilikom pretraživanja.

5.2.3. Komunikacija između korisnika

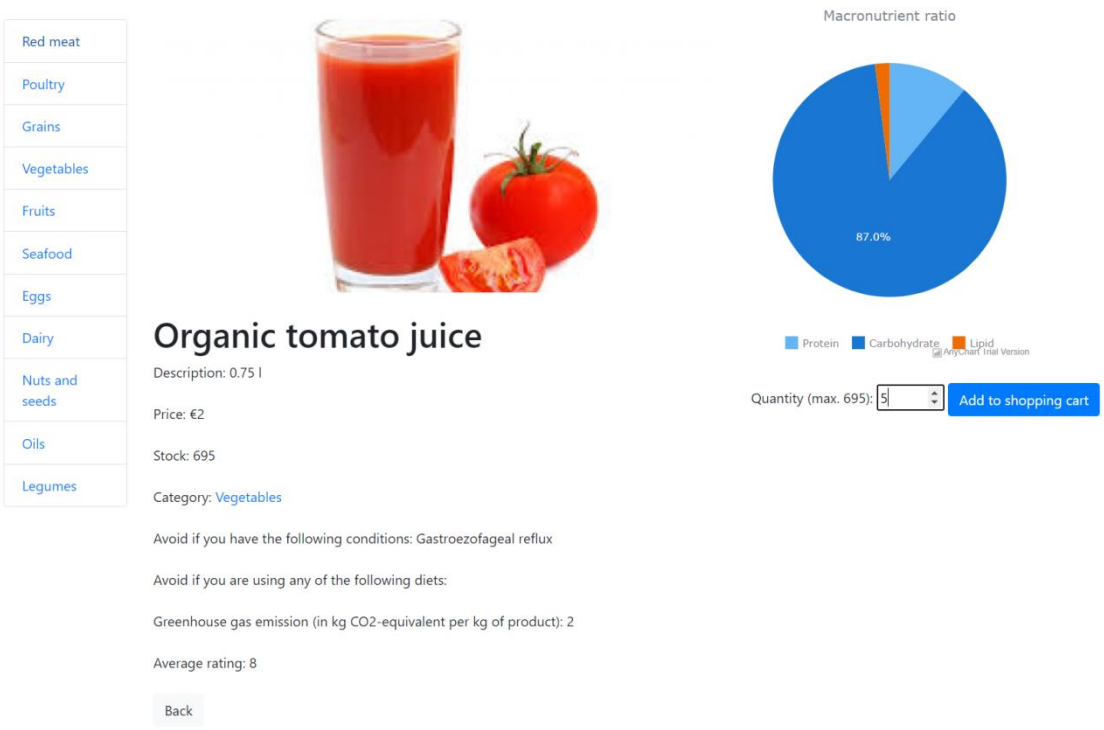
Svaki korisnik ove aplikacije ima mogućnost komunikacije s drugim korisnicima. Potrebno je naći kliknuti na tipku „Users“, naći korisnika s kojim se želi komunicirati te kliknuti na tipku „Chat“. Otvara se prozor za razgovor koji je vidljiv na slici 5.5.



Sl. 5.5. *Prozor za međusobni razgovor korisnika*

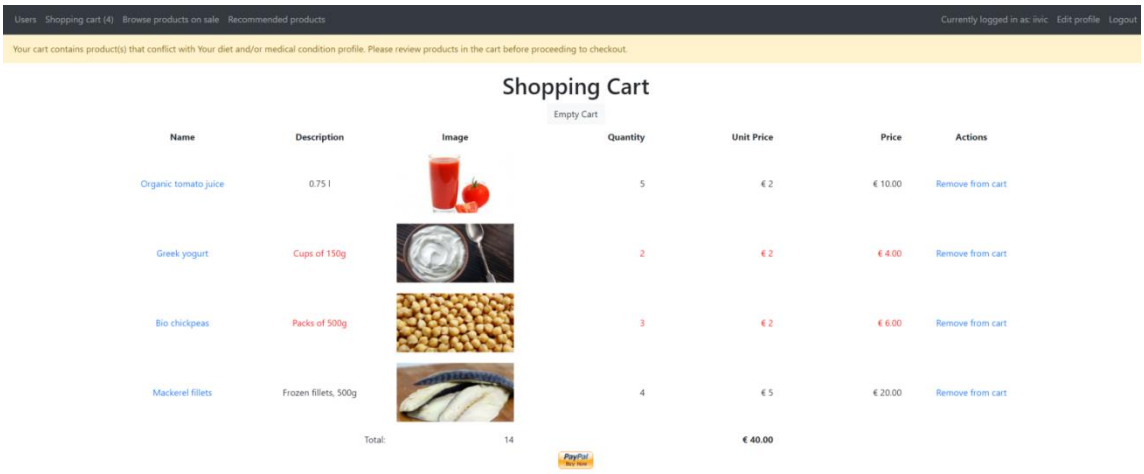
5.2.4. Kupnja proizvoda

Prije kupnje proizvoda, potrebno je dodati nekoliko proizvoda u košaricu. Primjer dodavanja proizvoda u košaricu vidljiv je na slici 5.6.



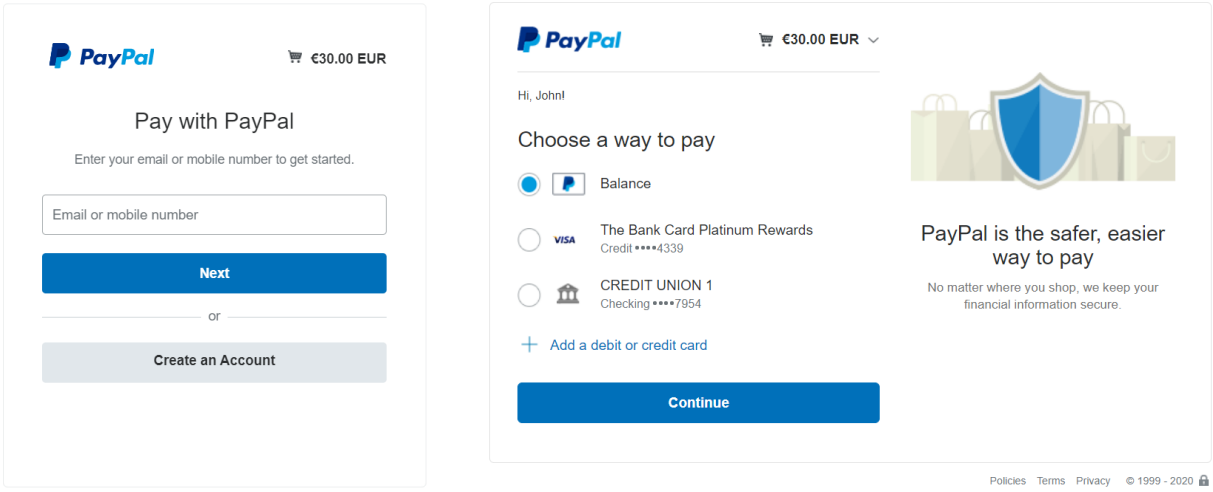
Sl. 5.6. *Pregled proizvoda i dodavanje u košaricu*

Klikom na tipku „Add to shopping cart“ količina upisana u tekstualno polje se dodaje u košaricu. Nakon što je korisnik dodao proizvode koje želi, klikom na tipku „Shopping cart“ prikazuje se pregled svih proizvoda u košarici, a prikazan je na slici 5.7.



Sl. 5.7. *Prikaz košarice i upozorenja*

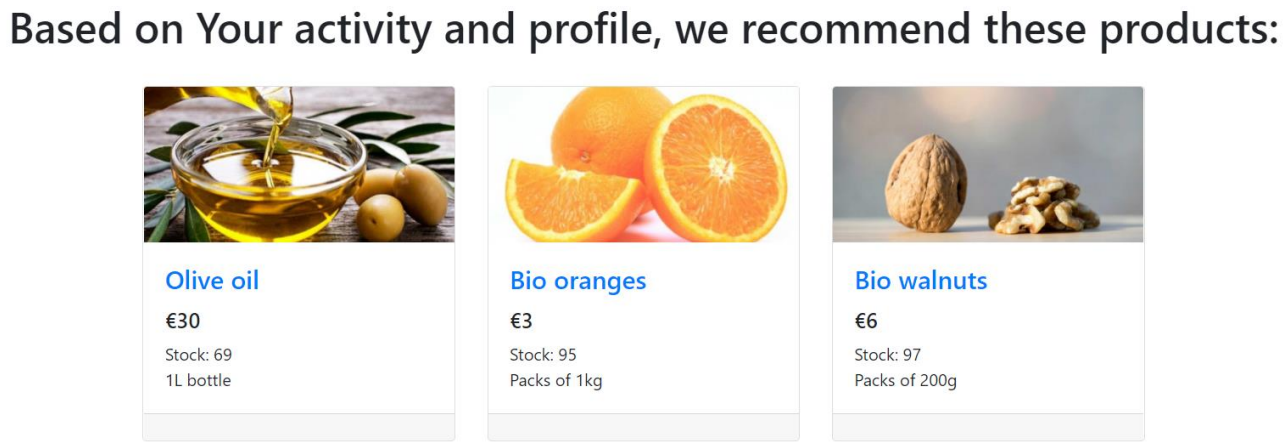
Budući da prijavljeni korisnik u ovom slučaju ima alergiju na mlijeko i mliječne proizvode, a također primjenjuje dijetu „Atkins“, ne bi trebao konzumirati proizvode „Greek yogurt“ i „Bio chickpeas“. Prema tome, ti proizvodi su istaknuti crvenom bojom i na gornjem dijelu pogleda je ispisano upozorenje. Klikom na tipku „Remove from cart“ proizvodi se eliminiraju iz košarice, a zatim se klikom na tipku „PayPal, Buy now“ otvara PayPal Sandbox prozor za ostvarenje transakcije, koji je vidljiv na slici 5.8. Nakon kupnje, korisnik je preusmjeren nazad na aplikaciju.



Sl. 5.8. *Prikaz PayPal transakcije*

5.2.5. Pregled preporuka za korisnika

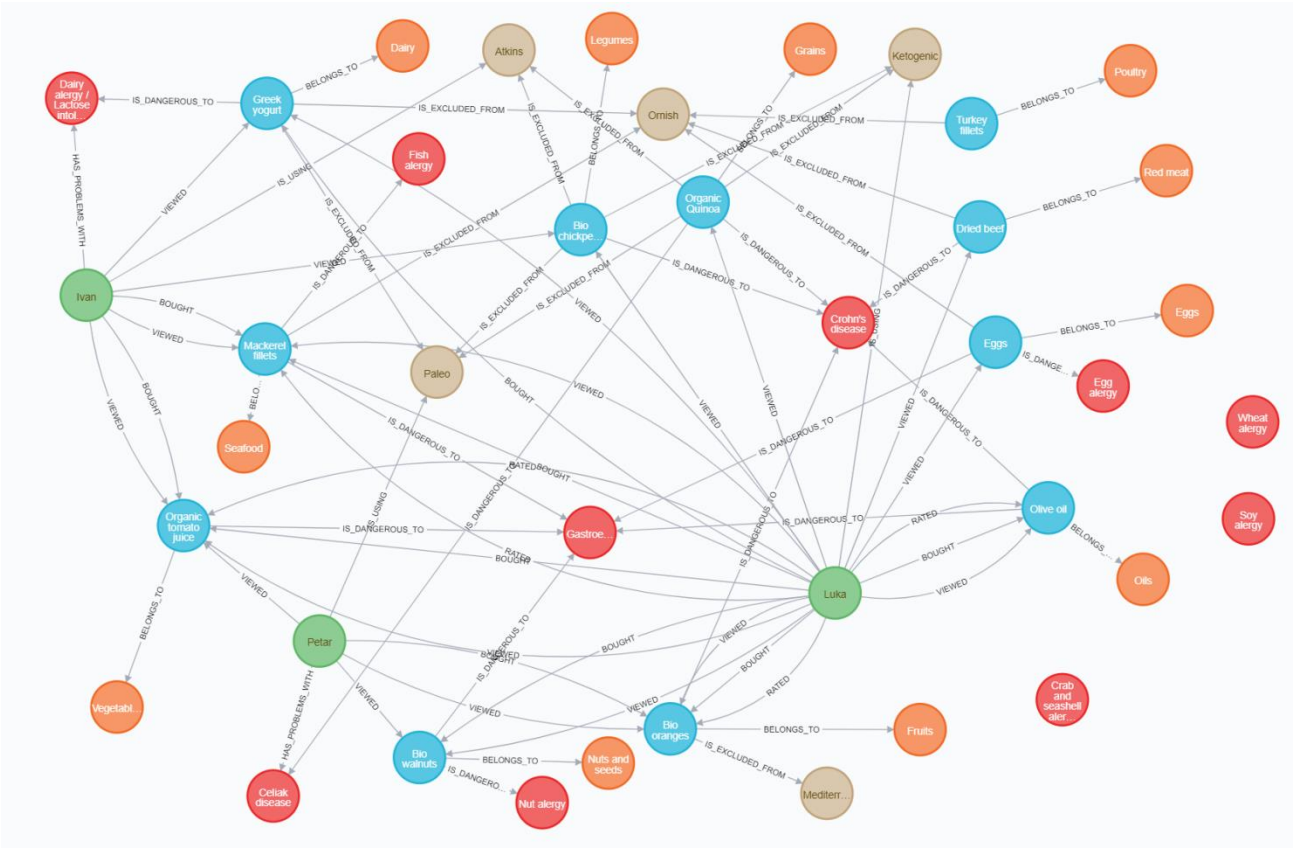
Klikom na tipku „Recommended products“, prikazuju se preporučeni proizvodi. Prikaz je moguće vidjeti na slici 5.9.



Sl. 5.9. *Prikaz preporučenih proizvoda*

5.2.6. Analiza i opis preporuka

Pogledom na bazu podataka (Sl. 5.10) se može zaključiti da su entiteti u bazi prikazani različitim bojama, ovisno o tome kojoj klasi pripadaju. Klase nisu eksplicitno definirane u samoj bazi nego se ime klase navodi pri kreiranju svakog čvora. U bazi su entiteti prikazani na sljedeći način: korisnici su obojeni zelenom bojom, proizvodi plavom bojom, dijete smeđom bojom, kategorije narančastom bojom i zdravstvena stanja crvenom bojom.



Sl. 5.10. *Prikaz grafa baze podataka*

U nastavku će biti objašnjen postupak stvaranja preporuka te će se kao primjer uzeti korisnik Ivan. Korisnik Ivan je, kao što je vidljivo na grafu, prethodno kupio proizvode „Mackerel fillets“ i „Organic tomato juice“. Također je vidljivo da je korisnik Luka kupio iste proizvode s još nekim proizvodima koje korisnik Ivan nije kupio („Greek yogurt“, „Olive oil“, „Bio walnuts“, „Bio oranges“). Sva četiri proizvoda ulaze u skup potencijalnih preporuka za korisnika po imenu Ivan. Iz grafa je vidljivo da korisnik Ivan ima problema s alergijom na mliječne proizvode, stoga se proizvod „Greek yogurt“ izbacuje iz skupa preporuka.

Definiranjem pravila odnosno ograničenja postignut je takav način rada da sustav nikada ne bi trebao preporučiti proizvod koji korisnik ne smije koristiti. Također, ako korisnik pokuša kupiti takav proizvod, ispisuje se upozorenje na koje korisnik može i ne mora obratiti pažnju. Sustav adekvatno upozorava korisnika ali ga ni u čemu eksplicitno ne sprječava. U nastavku je prikazana tablica s još nekoliko korisnika i preporuke koje su im predstavljene. Neki čvorovi su izostavljeni iz baze podataka na slici 5.10 radi bolje vidljivosti.

Tab. 5.2. *Detaljan prikaz preporuka*

Kupac	**Kupljeni proizvodi**	**Preporučeni proizvodi**	**Dijeta**	**Zdravstvena stanja**
Ivan	Organic tomato juice, Mackerel fillets	Bio oranges, Olive oil, Bio walnuts, Eggs	Atkins	Lactose intolerance
Mario	Dried beef, Greek yogurt	Bio oranges	Ketogenic	Gastroezofageal reflux
Luka	Bio walnuts, Olive oil, Eggs, Bio oranges, Greek yogurt, Organic tomato juice, Mackerel fillets	Dried beef	Ketogenic	Soy allergy
Stjepan	-	Bio oranges, Olive oil, Bio walnuts, Eggs, Organic tomato juice, Mackerel fillets, Dried beef, Greek yogurt	Ketogenic	-
Petar	Bio oranges	Olive oil, Bio walnuts, Eggs, Organic tomato juice, Mackerel fillets	Paleo	Celiak disease

Iz tablice 5.2 se može zaključiti da čak i novi korisnici koji do sad nisu kupovali proizvode (korisnik Stjepan) mogu dobiti adekvatne preporuke ako postoje korisnici koji primjenjuju istu dijetu (u ovom slučaju korisnici Mario i Luka). Nadalje, korisnici koji su kupili velik broj proizvoda (korisnik Luka) imaju manji broj proizvoda koji ulaze u potencijalni skup preporuka. Također, korisnici koji pate od zdravstvenih stanja koja isključuju velik broj proizvoda (u ovom slučaju korisnik Mario) imaju značajno smanjen skup potencijalnih preporuka. Konačno, budući da se ovdje radi o kolaborativnom sustavu preporuka, bitan nedostatak je činjenica da preporuke nisu konkretne tj. može se reći da nisu dovoljno prilagođene za ukuse i sklonosti pojedinog korisnika.

6. ZAKLJUČAK

U ovom diplomskom radu osmišljen je, modeliran i programski ostvaren web sustav koji korisnicima pomaže kod odabira proizvoda za kupnju na siguran način, uzimajući u obzir njihove potencijalne zdravstvene probleme i način prehrane. Postavljeni ciljevi ostvareni su pomoću prikladnog programskog okruženja. Predefinirane vrijednosti pojedinih parametara i testiranje su omogućili konačno ostvarenje aplikacije koristeći se postupcima koji su detaljno razjašnjeni prije programskog rješenja. Aplikacija ima dva tipa korisnika, prodavače i kupce, s posebnim naglaskom na kupce i njihove potrebe. Implementirane su funkcionalnosti koje korisnicima omogućuju prijavu u sustav, pregled i kupnju proizvoda. Generiranje preporuka vrši se na temelju profila kupca i proizvoda te aktivnosti drugih kupaca koji imaju slične parametre. Rješenje je također potpomognuto s mogućnosti ocjenjivanja proizvoda ocjenom od 1 do 10 te sustavom poruka za komunikaciju korisnika. Implementiran je i koncept notifikacija koje korisnicima daju informacije o tome što se događa u aplikaciji.

Sustav je dao pouzdane preporuke koristeći jasno definirana pravila koja konkretno ograničavaju stvaranje preporuka te ne preporučuje proizvode koje korisnik ne bi smio koristiti. Nadalje, sučelje je intuitivno i jednostavno za korištenje te je pogodno za velik broj korisnika. Nedostatak sustava preporuka je činjenica da su preporuke kolaborativnog tipa. Naime, nije garantirano da će pojedini korisnik biti zadovoljan sa svim preporukama koje mu sustav kreira. Buduće dorade aplikacije bi trebale dati konkretnije preporuke koje su prilagođene sklonostima i ukusima samog korisnika, bez obzira na ostale korisnike u sustavu.

LITERATURA

[1] R. Freire, Scientific Evidence Of diets for weight loss: Different macronutrient composition, intermittent fasting, and popular diets, Nutrition br. 69, sv. 110549, str. 1-5, 2020.

[2] H.A. Raynor, C.M. Champagne, Position of the Academy of Nutrition and Dietet, J Acad Nutr Diet, br.116, sv. 1, str. 129–147, 2016.

[3] Digestive disorders, https://www.nutrition.gov/topics/diet-and-health-conditions/digestive-disorders (posjećeno 25.4.2020.)

[4] Food alergies, https://www.fda.gov/food/buy-store-serve-safe-food/what-you-need-know-about-food-allergies (posjećeno 26.4.2020.)

[5] Carbon footprint, https://ourworldindata.org/food-choice-vs-eating-local (posjećeno 28.4.2020.)

[6] J. Poore, T. Nemecek, Reducing food's environmental impacts through producers and consumers, Science br. 360, sv. 6392, str. 987-992, lipanj 2018.

[7] P. Resnick, H. Varian, Recommender Systems, Communications of the ACM, br. 3, sv. 40, str. 56-58, ožujak 1997.

[8] C. Tripathy, Y. Pavlidis, Recommendation systems: Principles and practice, XRDS br. 3, sv. 26, str. 54-56, travanj 2020.

[9] P. Covington, J. Adams, E. Sargin, Deep Neural Networks for Youtube Recommendations, RecSys, str. 1-5, rujan 2016.

[10] H. Schäfer, M.C. Willemsen, Rasch-based Tailored Goals for Nutrition Assistance Systems, 24th International Conference on Intelligent User Interfaces, str. 18-22, ožujak 2019

[11] The Amazon Recommendations Secret to Selling More Online, http://rejoiner.com/resources/amazon-recommendations-secret-selling-online/ (posjećeno 2.5.2020.)

[12] R. Zenun Franco, Online Recommender Systems for Personalized Nutrition Advice, RecSys, kolovoz 2017.

[13] M.F. Schwarz, D.C.M. Wood, Discovering Shared Interests using Graph Analysis, Communications of the ACM, br. 8, sv. 36, kolovoz 1993.

[14] C. Vicknair, M. Macias, Z. Zhao, X. Nan, Y. Chen, D. Wilkins, A Comparison of a Graph Database and a Relational Database, A data provenance perspective, ACMSE 2010, travanj 2010.

[15] B. J. Keller, N. Ramakrishnan, B. J. Mirza, Studying Recommendation Algorithms by Graph Analysis, Journal of Intelligent Information Systems, ožujak 2003.

[16] Službena web stranica PHP programskog jezika,
https://www.php.net/
(posjećeno 27.8.2020.)

[17] Službena web stranica Symfony razvojnog okruženja,
https://symfony.com/
(posjećeno 2.6.2020.)

[18] Introducing the Neo4j Symfony bundle,
https://www.sitepoint.com/introducing-the-neo4j-symfony-bundle/
(posjećeno 19.6.2020.)

[19] Github repozitorij Reco4PHP,
https://github.com/graphaware/reco4php
(posjećeno 25.8.2020)

[20] PayPal Developer (Sandbox),
https://developer.PayPal.com/docs/api/overview/
(posjećeno 20.6.2020.)

SAŽETAK

Web aplikacija ostvarena u ovom radu služi kao podrška korisnicima koji žele kupovati hranu putem Interneta, uzimajući u obzir zdravstvene i ekološke parametre proizvoda. Sustav rješava problem nesigurnosti korisnika kojima zdravstveno stanje ograničava prehranu. Aplikacija sprema profile proizvoda i korisnika te ih koristi u generiranju upozorenja i preporuka. Prodavači mogu postavljati proizvode na prodaju i uređivati njihove parametre vezane uz prehrambenu i ekološku prihvatljivost. Stvaranje preporuka se temelji na analizi grafova, tj. prolasku kroz strukturu grafa korisnika i proizvoda te mapiranja određenih proizvoda s korisnicima. Pri tome se uzimaju u obzir sva zdravstvena stanja i dijeta koju je kupac odabrao u svom profilu. Potvrda stabilnosti i pravilnog rada aplikacije osigurana je testiranjem svakog dijela aplikacije. Analizom korištenja može se utvrditi da su preporuke pouzdane, međutim, buduće dorade aplikacije bi trebale implementirati dodatna pravila koja bi rezultirala stvaranjem konkretnijih preporuka koje su prilagođene korisnicima.

Ključne riječi: analiza grafova, prehrambena i ekološka vrijednost proizvoda, sustav preporuka, zdravstveno stanje, web trgovina.

ABSTRACT

Web application created in this graduate thesis serves as support for users who want to buy food on the Internet, taking into account the nutritional and ecological values of products. The system solves the problem of insecurity of users whose health restricts their diet. The application saves the profiles of products and users and uses these parameters to generate warnings and recommendations. Sellers can place products on sale and edit their parameters connected with nutritional and ecological value. Generating recommendations is based on graph analysis or more specifically, traversal over a graph of users and products and mapping specific products to users. When generating recommendations, every medical condition and diet of a user is taken into account. Usage analysis can determine that the recommendations are reliable, however, future refinements of the application should implement additional rules that would result in the creation of more specific recommendations that are tailored to users.

Keywords: graph analysis, nutritional and ecological value of products, recommendation system, medical condition, webshop.

ŽIVOTOPIS

Ivan Klešić rođen je u Osijeku, Republika Hrvatska, 24. studenog 1995. Godine. Pohađao je osnovnu školu Vladimira Becića u Osijeku. U osmom razredu je sudjelovao na državnoj smotri radova iz robotike. Nakon osnovne škole, 2010. godine upisuje Elektrotehničku i prometnu školu Osijek te završava sva četiri razreda s odličnim uspjehom. U drugom razredu srednje škole je sudjelovao na državnom natjecanju iz elektrotehnike. Maturirao je 2014. godine. 2014. godine upisuje Elektrotehnički fakultet u Osijeku, smjer računarstvo. Nakon završetka preddiplomskog studija, upisuje diplomski studij na istom fakultetu (sada Fakultet Elektrotehnike, računarstva i informacijskih tehnologija). Posjeduje osnovna znanja programiranja, posebice programski jezik PHP te razvojna okruženja Laravel i Symfony.

Vlastoručni potpis

_____

Ivan Klešić

PRILOZI

1. Dokument rada u formatu docx
2. Dokument rada u formatu pdf
3. Programski kod web sustava za prodaju proizvoda uz nadzor njihove prehrambene i ekološke prihvatljivosti