KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

Ričardas Šmaižys

ELEKTRONINĖS KOMERCIJOS SISTEMŲ ARCHITEKTŪRINIŲ SPRENDIMŲ TYRIMAS

Baigiamasis magistro projektas

Vadovas

Doc. dr. Tomas Blažauskas

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

ELEKTRONINĖS KOMERCIJOS SISTEMŲ ARCHITEKTŪRINIŲ SPRENDIMŲ TYRIMAS

Baigiamasis magistro projektas

Programų sistemų inžinerija (kodas 621E16001)

Vadovas

(parašas) Doc. dr. Tomas Blažauskas

(data)

Recenzentas

(parašas) Doc. dr. Liudas Motiejūnas

(data)

Projektą atliko

(parašas) Ričardas Šmaižys

(data)

KAUNAS, 2016

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS

(Fakultetas)

RIČARDAS ŠMAIŽYS

(Studento vardas, pavardė)

„Programų sistemų inžinerija“ valstybinis kodas 621E16001

(Studijų programos pavadinimas, kodas)

„ELEKTRONINĖS KOMERCIJOS SISTEMŲ ARCHITEKTŪRINIŲ SPRENDIMŲ TYRIMAS“

AKADEMINIO SĄŽININGUMO DEKLARACIJA

20 ___ . _____ ____ .

Kaunas

Patvirtinu, kad mano, **Ričardo Šmaižio**, baigiamasis projektas tema „ELEKTRONINĖS KOMERCIJOS SISTEMŲ ARCHITEKTŪRINIŲ SPRENDIMŲ TYRIMAS“ yra parašytas visiškai savarankiškai ir visi pateikti duomenys ar tyrimų rezultatai yra teisingi ir gauti sąžiningai. Šiame darbe nei viena dalis nėra plagijuota nuo jokių spausdintinių ar internetinių šaltinių, visos kitų šaltinių tiesioginės ir netiesioginės citatos nurodytos literatūros nuorodose. Įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs.

Aš suprantu, kad išaiškėjus nesąžiningumo faktui, man bus taikomos nuobaudos, remiantis Kauno technologijos universitete galiojančia tvarka.

_____ _____

(vardą ir pavardę įrašyti ranka) *(parašas)*

SANTRAUKA

El. komercija ir el. parduotuvės yra neatsiejama interneto ir programuotojų kasdienybė, tačiau besiplečiant elektroniniam verslui jų savininkai ir kūrėjai susiduria su naujais iššūkiais, pavyzdžiui, dideli duomenų kiekiai.

Magistro darbe pateikiama aiški ir detali populiariausių atvirojo kodų el. komercijos sistemų analizė, aptariami jų taikomi sprendimai dviem didžiausioms el. komercijos problemoms spręsti: paieškai ir indeksuoto filtro naudojimui, jų greitaveikai. Pateikta architektūrinė analizė, aptarti egzistuojantys problemos sprendimo įrankiai rinkoje bei jų privalumai ir trūkumai.

Magistrinio darbo metu iš atliktų išvadų ir sistemų analizės pasirinkta el. komercijos platformai *PrestaShop* sukurti filtro bei paieškos modulį sprendžiantį didelių duomenų kiekio problemą paieškos ir filtravimo metu.

Darbo metu sukurtas modulis ir jo rezultatai eksperimentiniu būdu palyginti su kitais rinkoje esančiais standartiniais ir nestandartiniais įrankiais ar moduliais, gauti rezultatai aprašyti, įvertinti, pateikiamos išvados ir rezultatai.

SUMMARY

E-commerce and e-shops are integral part of the Internet and everyday work of programming experts. However, as the electronic businesses expand their owner and developers are faced with new challenges such as large amounts of product information, latency, search and filtering usage on their ecommerce projects. The paper provides clear and detailed analysis of most popular open source ecommerce platforms on The Internet, evaluates solutions applied to largest ecommerce issues such as search and filter performance with big data.

In addition, a module for Prestashop ecommerce platform using third-party ElasticSearch service is introduced as a solution based on the evaluations and analysis with a detailed architecture, functional and non-functional requirements outlined with experimental comparison of other reviewed systems and their components.

Finally, both system and experimental results and findings are outlined.

TURINYS

PAVEIKSLĖLIŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

TERMINŲ SANTRUPOS IR ŽODYNAS

1. *DB* –duomenų bazė, organizuotas (susistemintas) duomenų rinkinys, kuriuo galima naudotis elektroniniu būdu.
2. *PHP* – programavimo kalba (angl. *PHP* programing language) – plačiai paplitusi dinaminė interpretuojama atvirojo kodo programavimo kalba, pritaikyta interneto svetainių ir aplikacijų kūrimui.
3. *MySQL* – populiariausia atvirojo kodo reliacinių duomenų bazių valdymo sistema, dirbanti *SQL* kalbos pagrindu.
4. *SQL* – duomenų bazių programavimo kalba, skirta manipuliuoti reliacinių duomenų bazių valdymo sistemose.
5. *RAM* – operatyvioji atmintis
6. Įterptasis langas (angl. iframe) – HTML programavimo kalbos sintaksės žymė (angl. tag) leidžianti įterpti interneto puslapį kito puslapio viduje.
7. *HTML* - (Hyper text Markup Language „Hiperteksto žymėjimo kalba“) – tai standartizuota „W3 konsorciumo“ kompiuterinė žymėjimo kalba, naudojama pateikti turinį internetiniuose puslapiuose.
8. Modelio-Kontrolerio-Vaizdinio (angl. MVC) modelis – tai programinės įrangos projektavimo modelis, kuriuo metu aiškiai ir tiksliai atskiriama biznio logika nuo vaizdinio ir duomenų bazės esybių.
9. Turinio Valdymo Sistema (TVS) – programinės įrangos paketas, supaprastinantis informacinio turinio kūrimą, redagavimą ir valdymą taip, kad keičiant turinį nereiktų jokių programavimo žinių.
10. *IP* – kompiuterio identifikatorius IP protokolu paremtuose tinkluose
11. *Apache* – tinklo serveris
12. *ElasticSearch* – realaus lako paieškos ir analitikos sistemos infrastruktūra
13. *CRON task* – periodinė užduotis *Unix* tipo sistemose, kurią galima vykdyti, kas tam tikrą specialiai aprašius ir nurodžius vykdymų pasikartojimą
14. *EAV*- Entity–attribute–value model – esybių organizavimo modelis, kur esybė išskaidoma į esybės-atributo-atributo_reikšmės dalis

1. ĮVADAS

Atvirojo kodo el. komercijos parduotuvių sprendimai šiuo metu ypač konkurencingi rinkoje ir kiekvienas jų bando pateikti savo privalumus, kuo jis vienu ar kitu aspektu geresnis už konkurentus. Tačiau dažniausiai šie privalumai yra siejami su marketingo klausimais arba funkcionalumu, kuris matomas ir naudotinas labiausiai el. parduotuvių administratoriams, o ne programuotojams ar jų kūrėjams.

Išsamaus architektūrinio palyginimo atvirojo kodo el. parduotuvių platformų ir jų taikomų sprendimų, problemų aptarimo ar privalumų ir trūkumų identifikavimo nėra, o jei ir yra, jie nedetalūs, orientuojasi į neesmines charakteristikas, kurios negeba tinkamai apibrėžti sistemos galimybių, problemų esant nestandartinėms situacijoms.

Magistro darbe pateikiama aiški ir detali sistemų analizė, aptariami jų taikomi sprendimai dviem didžiausios el. komercijos problemoms spręsti: paieškai ir indeksuoto filtro naudojimui, jų greitaveikai. Pateikta analizė, kuria remiantis bus galima įvertinti kuriamų projektų sėkmę ar numatomas bei galimas problemas nestandartiniais atvejais: itin didelės apkrovos serverių, didžiuliai kiekiai prekių, filtro, paieškos ir kt. problemos. Projektas leidžia įvertinti konkrečių sistemų esminius privalumus ir trūkumus, tinkamai vertinti projekto riziką.

Gauti patarimai ir sprendimai programuotojų komandai susipažinusiai su atliktu magistro darbu ir jo tyrimu leis šablonų principu priimti sprendimus, spręsti susidariusias nestandartines situacijas su įmonės dideliais projektais.

Magistro darbe pateikiamas realizuotas sprendimas pagal išanalizuotą visų aptariamų el. komercijos sprendimų atrastus privalumus, kaip sprendimas pagrindinėms problemos: paieškos ir filtro problemoms su didžiuliu duomenų kiekiu el. parduotuvėje ir tai tik vienas iš būdų šabloniškai spręsti susidariusią nestandartinę situaciją išnagrinėjus el. komercijos platformas ir jų taikomus sprendimus.

1.1. Dokumento paskirtis

Šiame dokumente pateikiama ir aptariama populiariausių el. komercijos sprendimų naudojami sprendimai darbui su dideliais duomenimis, apžvelgiami egzistuojantys standartiniai ir nestandartiniai „premium“ lygio sprendimai darbui su dideliais duomenimis. Dokumente pateikiamas sukurtas sprendimas paieškos ir filtro problemai su dideliais duomenimis spręsti pasirinktai el. komercijos sistema *PrestaShop*, kuri šiuo metu yra viena populiariausių Europoje.

1.2. Darbo tikslas

Ištirti naudojamus sprendimus el. komercijos darbui su dideliais duomenimis bei įvertinti sukurto sprendimo naudingumą darbui su dideliais duomenimis pasirinktoje el. komercijos platformoje lyginant su konkurentų esamais sprendimais rinkoje.

1.3. Mokslinis naujumas

1. Pasiūlytas trečiųjų šalių jau sukurtos sistemos *ElasticSearch* panaudojamumas konkrečiai paieškos ir filtro darbui su dideliais duomenų kiekiais problemai spręsti.
2. Suprojektuota ir sukurta posistemė el. komercijos *PrestaShop* platformai
3. Atlikti greitaveikos ir palyginimo tyrimai nustatyti ar pasiūlytas sprendimas veiksmingas ir siekiant palyginti su esamai rinkos žaidėjais.

1.4. Uždaviniai

1. Išanalizuoti populiariausias el. komercijos platformas
2. Nustatyti el. komercijos platformose naudojamus sprendimus darbui su duomenimis (prekėmis filtro ir paieškos metu) esant įvairiems duomenų kiekiams.
3. Suprojektuoti ir sukurti posistemę pasirinktai el. komercijos platformai *PrestaShop* problemai darbui su dideliais duomenimis spręsti
4. Ištyrinėti ir aprašyti realizuoto sprendimo naudą ir palyginti su esamais rinkoje konkurentais
5. Atlikti paieškos ir filtro greitaveikos eksperimentus analizuojant skirtingų duomenų kiekių įtaką greitaveikai.

2. EL. KOMERCIJOS PLATFORMŲ DARBUI SU DIDELIAIS DUOMENIMIS ANALIZĖ

2.1. Temos aktualumas ir tikslingumas

Elektroninė komercija – prekybinės veiklos būdas, kai sutartys sudaromos, o prireikus – ir vykdomos, naudojant informacines technologijas bei priemones, kompiuterių tinklais keičiantis elektroniniais duomenų pranešimais [1]. El. komercijos augimas išlieka stabilus kiekvienais metais ir didėja apytiksliai po 20 proc. per metus [2, 3]. Reikalingi techniniai sprendimai palaikyti tokį el. komercijos augimą taip pat sunkėja ir standartiniai problemų sprendimai dažnai nebetinkami susiduriant su esminėmis el. parduotuvių problemomis: greitaveika, dideliais prekių kiekiais, greitos paieškos ir rezultatų grąžinimu, prekių filtrais. Anot *Shopify* atliktos analizės [4] net 21 proc. paliktų (angl.

abandonded) krepšelių yra dėl per ilgai veikusių sistemos procesų bei net 24 proc. jų dėl sutrikusių sistemų darbo.

Šios darbo ir analizės metu siekiama išsiaiškinti *PrestaShop* el. komercijos platformos (viena populiariausių šiuo metu Europoje ir pasaulyje el. komercijos platforma [5]) trūkumus lyginant su kitais konkurentų ir trečiųjų šalių taikomais sprendimais minėtose problemose bei kaip tuos trūkumus panaudojant geriausias praktikas (angl. best-practices) ir šabloninius sprendimus (angl. design patterns) būtų galima geriausiai išspręsti bei pritaikyti, t.y. patobulinti *PrestaShop* sistemą.

Darbo metu orientuojamasi į pagrindinius aspektus:

1. Paieškos indeksavimo ir rezultatų atvaizdavimo problema esant dideliems duomenų kiekiams
2. Prekių filtro indeksavimo, rezultatų atvaizdavimo ir veikimo problema esant dideliems duomenų kiekiams
3. Standartinio prekių importo optimizavimo išnaudojant objektus greitaveika
4. Architektūrinius sistemos sprendimus (įskaitant duomenų bazės struktūrą) atvaizduojant kategorijų puslapius esant dideliems duomenų kiekiams

2.2. Egzistuojantys sprendimai

PrestaShop el. komercijos sistemoje susiduriama su problema, kad esant bent 20 tūkst. ir daugiau prekių standartinis paieškos indeksavimas ir veikimas tampa nefunkcionalus dėl per ilgai generuojamo prekių indekso, kuris trunka nuo 2-3 val. priklausomai nuo serverio pajėgumų vertinant tai, kad serveris atitinka vidutinius *PrestaShop* sistemai reikalingus reikalavimus ir resursus.

Standartinė paieška taip pat neefektyviai išnaudoja serverio operatyviąją atmintį kaupdama didžiulius kiekius duomenų kintamuosiuose.

Analogiška problema pastebima ir su filtrais el. komercijos platformose, kurio iš esmės veikimo principas toks pats – pirmiausia sukuriamas duomenų indeksas, kuris vėliau panaudojamas rezultatų atvaizdavimui.

Egzistuojantys sprendimai, jų veikimo būdai išskirti pagal tiriamas problemas.

Analizuojami esami algoritmai, taikomi sprendimai ir paruošti produktai bei jų veikimas.

Atlikus rinkos analizę nustatyta, kad standartinis modulis paieškos modulis turi esminių trūkumų pačiame funkcionalume ir apskritai negeba palaikyti specifikuojamo prekių kiekio.

Nustatytas *EIP Concepts Inc*. JAV įsikūrusios įmonės specialus modulis *AJAXFilter*, kuris neišsprendžia didelio kiekių problemos (veikia greitai tik su prekių kiekiu iki 5000-10000 prekių kategorijoje). Taip pat nesuteikia paieškos galimybių ir neoptimizuoja sistemos veikimo atliekant paprastą tekstinę paiešką.

2.2.1. Paieškos indeksavimas ir rezultatų atvaizdavimas

Rinkoje galima rasti tokiu pačiu principu veikiančių sprendimų, kurie iš esmės nesprendžia esminės problemos – duomenų indekso generavimo greitaveikos ir algoritmo. Šiuo atveju indeksavimas įmanomas dviem variantais:

1. Indeksuojama po kiekvieno prekės atnaujinimo ar išsaugojimo
2. Indeksuojama periodinės užduoties (angl. cron task) metu visas indeksas

Pirmasis metodas tinkamas tuomet, kai sistemoje nėra sinchronizuojamos ar atnaujinami prekių duomenys periodiškai, tarkime, kas valandą. Esant dideliems prekių kiekiams net ir vieno indekso atnaujinimas po kiekvieno prekės saugojimo sistemoje tampa neefektyvus, nes prekių sinchronizavimas nespėja atlikti iki galo savo sinchronizacijos dėl lėtumo.

Antrasis metodas, išnaudojantis periodines užduotis, susiduria su kita problema – indeksas kuriamas taip ilgai, kad dalis duomenų būna jau pasenę, apkraunami neproporcingai daug serverio resursai ir tai lėtina sistemos visą darbą. Lankytojams jaučiamas sistemos sulėtėjimas, lankytojai gali nesuprasti dėl kokių priežasčių el. parduotuvė veikia lėtai, o šiuo atveju lėtumas ir pačios el. parduotuvės greitaveika ne tik yra *SEO* paieškos variklių optimizavimo dalis, bet ir turi neigiamos įtakos lankytojų elgsenai: jie gali išeiti iš parduotuvės nieko nenupirkę ir daugiau nebegrįžti.

Rinkoje visiškai pakeičiančių paieškos varikliuko (angl. engine) aprašytą algoritmą ar veikimą, jo būdą ar metodiką paieškos indeksavimo, atvaizdavimo klausimais *PrestaShop* sistemai nėra.

2.2.2. *ElasticSearch* panaudojimas

Esant dideliems duomenų kiekiams paieškos problema gali būti sprendžiama išnaudojant trečiosios šalies sukurtą nemokamą parskirstytą infrastruktūrą ir sistemą *ElasticSearch* [6, 7]. *ElasticSearch* siūlo efektyvų šios problemos sprendimą pasiūlydami realaus laiko grąžinamus rezultatus iš savo sistemos. Tačiau konkrečiai *PrestaShop* el. komercijos sistemai integracija su šia trečiosios šalies sistema nėra atlikta ar viešai prieinama, todėl siekiant šį plačiau žinomą būdą integruoti į sistemą reiktų sukurti specialų tik *PrestaShop* pritaikytą modulį, kuris išnaudotų esamą infrastruktūrą. Modulis būti vienas iš šabloniškų sprendimo būdų programuotojams ateityje taikyti ir spręsti problemą iš esmės, greitai ir taupant kaštus.

Analogiškų ir panašių trečiųjų šalių sistemų didelių duomenų paieškai optimizuoti galima rasti ir daugiau [8]. Kiekvienam jų reiktų kurti atskirą integraciją. Konkurentų rinkoje *Magento* el. komercijos platformai galima rasti jau atliktą ir paruoštą *ElasticSearch* integraciją [9].

2.2.3. Didelio proceso skaidymas į mažesnius

Išnagrinėjus konkurentų sistemą ir paieškos modulio veikimą, pastebėta, kad naudojama praktika skaidyti didelį darbą į mažesnius procesus registruojant juos kaip įvykius (angl. event).

Visas algoritmas ir procesas išskaidomas į atskirus elementus – iš pradžių registruojami kategorijų ryšių įvykiai, vėliau prekių informacijos ir ryšių įvykiai, kurie laukia vienoje eilėje (angl. queue) ir apdorojami po truputį be didelių ir ilgai besitęsiančių *SQL* užklausų.

Tokiu principu taip pat išvengiama duomenų bazės užrakinimo problemos. Kadangi apžvelgiamose sprendimuose naudojama *MySQL* duomenų bazių valdymo programavimo kalba, orientuojamasi ir į šiai duomenų bazei būdingą užraktų veikimą. Eilučių (angl. row) užraktas naudojamas *MySQL InnoDB* metode, kai užrakinamos yra tik pačios eilutės jų modifikavimo metu, visos lentelės užraktas naudojamas *MyISAM* metode, kai rakinama visa duomenų bazės lentelė. Problema praktikoje sutinkama ta, kad didelių duomenų kiekių problema atsiranda vėliau nei yra projektuojama pati DB architektūra, o vėliau pakeisti *MySQL* veikimo būdą ir lentelių rakinimą gali būti labai problematiška ar iš vis neįmanoma, todėl skaidymas į procesus išlieka problematiškas tam tikrais atvejais.

2.2.4. Prekių filtro indeksavimas, rezultatų atvaizdavimas ir veikimas

Filtro duomenų indeksui standartiškai naudojamas tas pats iš dviejų būdų susidedantis algoritmas (indeksavimas po prekės atnaujinimo, pilnas indeksavimas periodinės užduoties metu). Visos problemos aptartos prie pirmos problemos, todėl papildomai nebekartojamos.

Šiuo atveju dar viena specifinė filtro indeksavimo problema ta, kad net ir sąlyginai nedideliems duomenų rinkiniams (40-50 tūkst. prekių sistemoje) sukuriami neefektyviai dideli indeksai su 4-5 mln. įrašų duomenų bazėje. Atsižvelgiant į tai, kad *PrestaShop* naudoja vieną populiariausių *PHP* aplinkos duomenų bazių *MySQL* ir jos techninį veikimą, kai arba lentelės duomenų apdorojimo metu yra užrakinamos (generuojant didelius įrašus blokuojama visa didelė lentelė, o kol ji neatblokuota sulėtėja visos sistemos darbas, nes lankytojų užklausos laukia eilėje, kol pasibaigs didelė užklausa) arba duomenų lentelių eilutės yra užrakinamos (greitaveikos problema jaučiama įterpiant duomenis į duomenų bazę, kas lėtina sinchronizacijas ir duomenų importus) [10].

Rinkoje esantys nemokamas standartinis *blocklayerd* [11] ir komercinis *Presto-Changeo* prižiūrimas modulis *AJAX Filter* [12] neišsprendžia aprašytų problemų filtro indeksavimo ir atvaizdavimo

klausimais. Standartinis nemokamas modulis, nors ir sukuria indeksą pakankama greitai nesprendžia duomenų bazės rakinimo problemos, taip pat veikia lėtai atvaizduojant duomenis ir netiksliai juos atvaizduodamas. Komercinis modulis sukuria perteklinį kiekį duomenų bazės įrašų, kuris sulėtina darbą visos sistemos, indeksavimas vyksta neprotingai ilgą laiką lyginant su turimais duomenų kiekiais.

2.2.5. *Magento* sistemos palyginimas ir sprendimo būdai

Magento sistemoje matomas aiškus stebėtojo (angl. observer) programinio šablono panaudojimas. Sistemoje stebėtojas atlieka indeksavimą tuomet, kai jo prireikia – perindeksuoja viską, indeksuoja kiekvieną dieną (speciali funkcija), indeksuoja po duomenų atnaujinimo, jei to reikia. Indeksavimas taip pat standartiškai skaidomas į dalis: kainos filtrą, požymį ir savybių filtrą, kategorijų filtrą.

Indeksavimas vyksta iteruojant per kategorijas, o kategorijų viduje per prekes atnaujinant išskaidytas indekso dalis mažomis užklausomis.

Iteravimas vykdomas tam tikru, nuostatose nustatytu, duomenų dydžiu (angl. batch), sistema turi savikontrolės funkcijas, kurios tikrina ar tas pats procesas atlieka iteravimą ir ar jo nereikia nutraukti pritrūkus resursų ar artėjant prie serverio veikimo laiko limito.

2.2.6. *OpenCart* sistemos palyginimas ir sprendimo būdai

OpenCart sistemos sprendimas apskritai paprastas ir nesudėtingas, neturi indekso ir veikia papildomų lentelių principu. Rinkoje nėra jokių nemokamų, atviro kodo filtro ar paieškos modulių bent prilygstančių standartiniams *PrestaShop*.

2.3. Įgyvendinimo problemos

2.3.1. Versijavimo problema

Siekiant išlaikyti sistemą ir visus programavimo elementus universalius sinchronizacijos, filtrai, indeksavimai turi būti kuo artimesni *PrestaShop* struktūrai ir išnaudoti, kiek įmanoma daugiau objektų, standartinių DB lentelių, kad būtų lengvesnis atnaujinimas pasikeitus sistemos struktūrai.

Funkcionalumas turi būti universalus ir pritaikomas nesudėtingai naujoms versijoms.

2.3.2. Duomenų sinchronizavimo greitaveikos apribojimas

Daroma prielaida, kad sistema nuolat atnaujina duomenų kiekius ir informaciją per standartinius objektus, todėl indeksavimui tiek filtro, tiek paieškos yra taikomi laiko ribojimai

2.4. Analizės išvados

Atlikta populiariausių el. komercijos projektų analizė ir jų naudojamų metodų taikymas paieškos ir filtro problemai spręsti su įvairiais duomenų kiekiais. Atliktus analizę yra aišku, kad reikia vengti pačios duomenų bazės naudojimo, nes ji apkrauna lankytojams srautą, sukuria perteklinius įrašus, veikimas gali būti nenuspėjamai užrakintas (angl. deadlock), o vienas iš sprendimo būdų, kuris pripažintas ir rinkoje, yra *ElasticSearch* artimai realiojo laiko paieškai failinėje sistemoje (angl. file system). Realizuojant savo sistemą patartina naudoti būtent trečiųjų šalių įrankį *ElasticSearch* kaip pagrindą funkcionalumui dėl jo teikiamų pranašumų.

Siekiant sutaupyti laiko ir kaštų kuriant filtro modulį bus per panaudojami standartiniai *PrestaShop* filtro modulio *blocklayered* administracinė ir vartotojo terpė ir vaizdas bei standartinio paieškos modulio *blocksearch* vartotojams matomo terpė ir vaizdas.

3. PROJEKTINĖ DALIS

3.1. Sistemos paskirtis

PrestaShop el. komercijos sistema pagal savo architektūrą ir standartinius modulius geba tinkamai veikti tik iki 10 tūkst. prekių. Atlikus rinkos analizę ir pagal užsakovo poreikius pastebėta, kad rinkoje nėra specialaus paieškos ir filtro modulio, kuris gebėtų pritaikyti sistemą tinkamam veikimui su daugiau nei 10 tūkst. prekių.

Duomenų bazės nėra pajėgios apdoroti ir suindeksuoti savyje didelių kiekių duomenų, t.y. jei indeksuojame 40 tūkst. prekių, indeksas tampa milijoninį įrašų kiekį turinti duomenų bazės lentelė, kurioje vykdoma standartinė *SQL* užklausų paieška tampa neefektyvi.

Pagrindiniai sistemos tikslai ir paskirtis:

- Užpildyti trūkstamą nišą *PrestaShop* el. komercijos modulių srityje, kur filtro ir paieškos moduliai gebėtų apdoroti didelius duomenų (prekių) kiekius.
- Sukurti modulį užsakovo esamiems ir naujiems klientams
- Parduoti modulį, kaip paslaugą, už fiksuotą kainą užsakovo klientams

Sistema skirta išspręsti vieną pagrindinių el. komercijos problemų darbui su dideliais duomenų kiekiais, kas yra vienas iš šio magistrinio darbo uždavinių.

3.2. Potencialūs sistemos vartotojai

3.1 lentelė. Potencialūs sistemos vartotojai

Vartotojų grupė	Rolė	Patirtis	Svarbios sistemos charakteristikos
Programuotojai	Integruoja sprendimą	Profesionalūs naudotojai	Sprendimo kokybė, lengvas per panaudojamumas, diegimo lengvumas
Parduotuvės administratorius	Naudoja sprendimą be modifikacijų	Vidutinė arba profesionalūs	Intuityvus administravimo valdymas
Naudotojas modulio	El. parduotuvės pirkėjas	Nepatyręs	Intuityvumas naudotis sistema, paprastumas.

3.3. Projekto apribojimai

Tai apribojimai, kurie turi įtakos reikalavimų specifikacijai, sistemos kūrimo eigai bei charakteristikoms

3.3.1. Įpareigojantys apribojimai

Aprašymas: Suderinamas modulis su *PrestaShop1.6* versija

Pagrindimas: Senesnės versijos reikalauja per daug resursų suderinamumui sukurti, pasikeitusi visapusiškai jų architektūra, tai pagrindinė naudojama versija šiuo metu.

Aprašymas: Vieno paspaudimo diegimas administravime

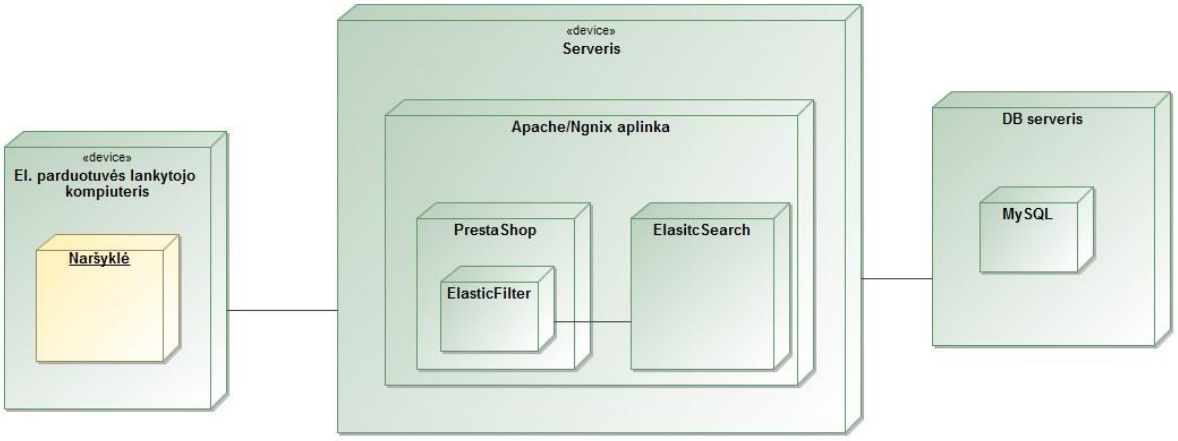
Pagrindimas: Moduliai yra per sudėtingi ir dažnai nepateisina perkančiųjų lūkesčių. Paprastumas diegiant esminis bruožas ir išskirtinumas.

Aprašymas: Palaikyti mobiliąją sąsają

Pagrindimas: Retas el. komercijos sprendimas nepritaikytas mobiliesiems įrenginiams, tai interneto tendencija.

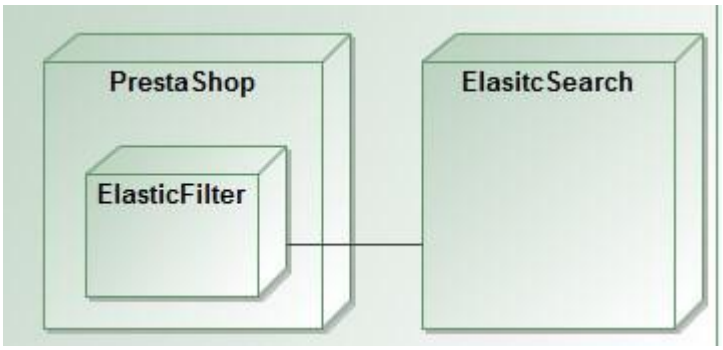
3.3.2. Diegimo aplinka

Realizacijos schema puikiai atspindi ir atvaizduoja diegimo aplinką, jos ryšius.



3.1 pav. diegimo aplinkos diagrama

3.3.3. Bendradarbiaujančios sistemos



3.2 pav. bendradarbiaujančių sistemų diagrama

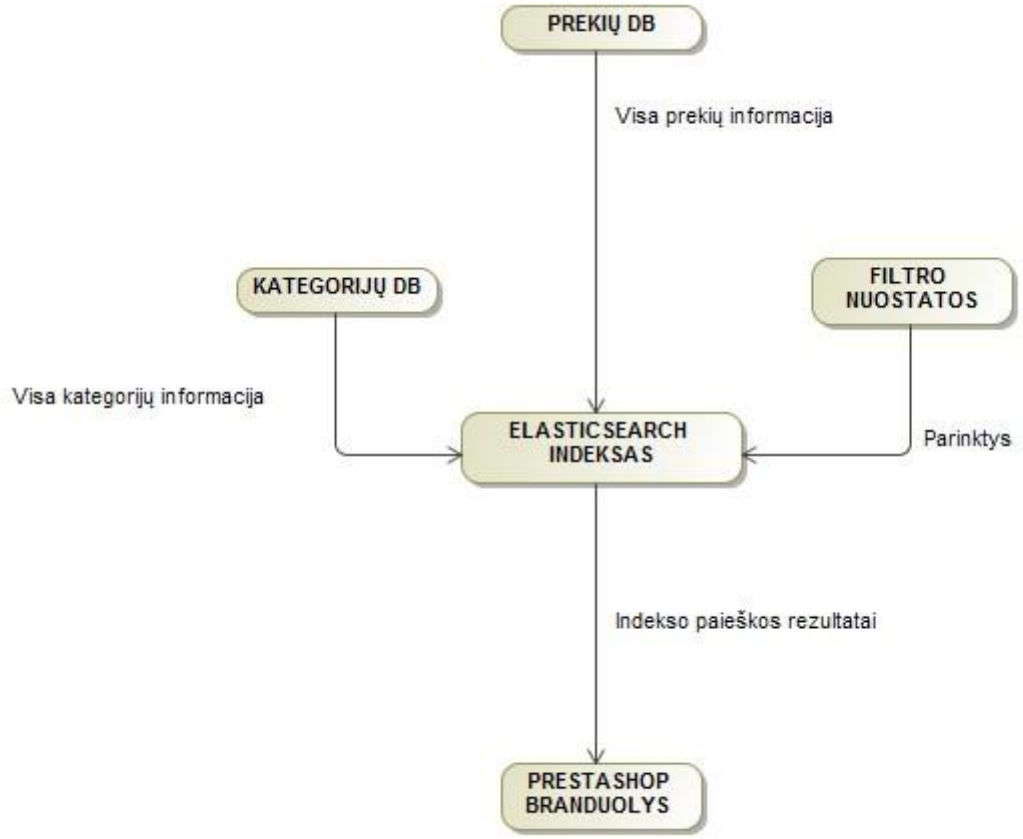
3.2 pav. pavaizduota komponentų diagrama, kuri susideda iš *PrestaShop* branduolio ir kuriamo modulio komponentų. Kuriamas modulis nenaudoja jokių kitų komponentų savo viduje.

3.3.4. Komerciniai specializuoti programų paketai

Naudojama *ElasticSearch* biblioteka filtro ir paieškos indekso kūrimui, apdorojimui ir pačiai paieškai.

3.4. Funkciniai reikalavimai

3.4.1. Veiklos sfera

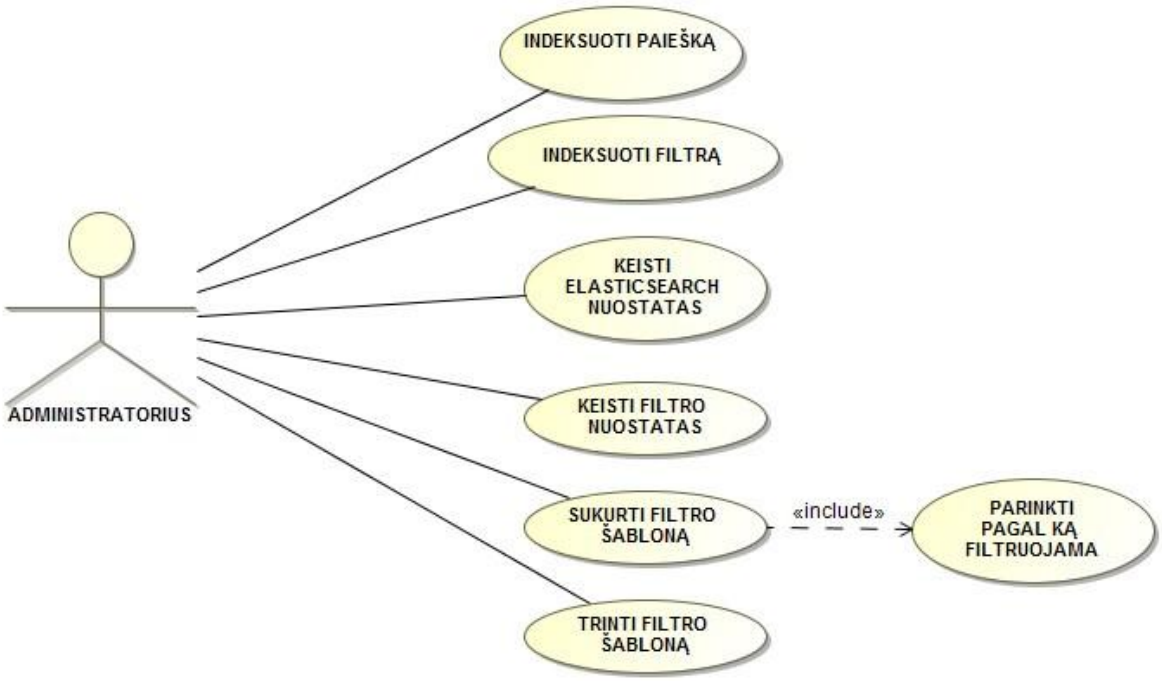


3.3 pav. veiklos konteksto diagrama

ElasticSearch iš standartinių *PrestaShop* el. komercijos platformos naudojamų duomenų bazių lentelių sugeneruoja statinį failų principu sudarytą indeksą, kuris vėliau naudojamas modulio (įterpto į *PrestaShop*) branduolį funkcionalumui tenkinti.

Tarpinis variantas *ElasticSearch* indekso naudojamas, nes indeksavimas į šio tipo indeksą bei paieška jame pagal atitinkamus kriterijus vykdoma ypač greitai lyginant su standartinėmis priemonėmis – duomenų baze. Duomenų bazės nėra pajėgios apdoroti ir suindeksuoti savyje didelių kiekių duomenų, t.y. jei indeksuojame 40 tūkst. prekių, indeksas tampa milijoninį įrašų kiekį turinti duomenų bazės lentelė, kurioje vykdoma standartinė *SQL* užklausų paieška tampa neefektyvi.

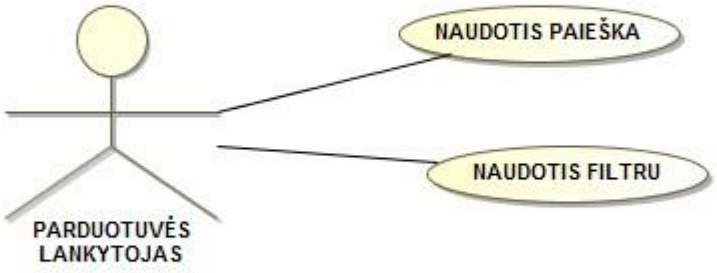
3.4.2. Produkto veiklos sfera



3.4 pav. administratoriaus panaudos diagrama

Pagrindinės administratoriaus galimybės ir funkcionalumas:

- Indeksuoti paiešką mygtuko paspaudimu rankomis
- Indeksuoti filtrą mygtuko paspaudimu rankomis
- Keisti *ElasticSearch* nuostatas: nustatyti *ElasticSearch* serverį
- Keisti filtro nuostatas: nustatyti kategorijų gylį filtre, ar rodyti galimų rezultatų skaičių šalia parinkties
- Sukurti filtro šabloną: išnaudojamas standartinio modulio *blocklayered* funkcionalumas: kuriamas filtro šablonas, kuris matomas tam tikrose kategorijose. Taip pat nustatyti, kuriuos atributus (angl. attributes and groups) bei ypatybes (angl. features) galima filtruoti nurodytame filtro šablone.
- Trinti filtro šabloną: galimybė ištrinti susikurto kategorijos filtro šabloną



3.5 pav. vartotojo (pirkėjo) panaudos diagrama

El. parduotuvės lankytojas, kuriame įdiegtas kuriamas produktas, gali:

- Naudotis paieška
- Naudotis filtru

3.4.3. Panaudojimo atvejų sąrašas

3.2 lentelė. PANAUDOJIMO ATVEJIS: Indeksuoti paiešką

1. PANAUDOJIMO ATVEJIS: Indeksuoti paiešką
Vartotojas/Aktorius: Administratorius
Aprašas: Administratorius rankomis suindeksuoja paieškos indeksą paspausdamas mygtuką.
Prieš sąlyga: Vartotojas prisijungęs prie administracinės aplinkos Vartotojas modulio konfigūravimo puslapyje *ElasticSearch* nuostatos nustatytos ir teisingos
Sužadinimo sąlyga: Indeksuojama paieška ir prekės
Po-sąlyga: Suindeksuota paieška su prekių informacija

3.3 lentelė. PANAUDOJIMO ATVEJIS: Indeksuoti filtrą

2. PANAUDOJIMO ATVEJIS: Indeksuoti filtrą **Vartotojas/Aktorius:** Administratorius **Aprašas:** Administratorius rankomis suindeksuoja filtro indeksą paspausdamas mygtuką. **Prieš sąlyga:** Vartotojas prisijungęs prie administracinės aplinkos Vartotojas modulio konfigūravimo puslapyje *ElasticSearch* nuostatos nustatytos ir teisingos **Sužadinimo sąlyga:** Filtras turi būti suindeksuotas, kad jį būtų galima naudoti lankytojams. **Po-sąlyga:** Suindeksuota filtro prekių informacija.

3.4 lentelė. PANAUDOJIMO ATVEJIS: Keisti *ElasticSearch* nuostatas

3. PANAUDOJIMO ATVEJIS: Keisti *ElasticSearch* nuostatas **Vartotojas/Aktorius:** Administratorius **Aprašas:** *ElasticSearch* nuostatos suderinamos – nurodoma į kur bus indeksuojama filtro ir paieškos informacija. **Prieš sąlyga:** Vartotojas prisijungęs prie administracinės aplinkos Vartotojas modulio konfigūravimo puslapyje *ElasticSearch* sudiegta ir sukonfigūruota serveryje **Sužadinimo sąlyga:** Filtras ir paieška negali būti suindeksuoti, kol neparuošta sąsaja su *ElasticSearch* **Po-sąlyga:** Galimas indeksavimas filtro ir paieškos

3.5 lentelė. PANAUDOJIMO ATVEJIS: Sukurti filtro šabloną

4. PANAUDOJIMO ATVEJIS: Sukurti filtro šabloną **Vartotojas/Aktorius:** Administratorius **Aprašas:** Visoms arba konkrečiai prekių kategorijai filtras gali būti atskiras ir kiekvienai kategorijai ar visoms kuriamas atskiras filtro šablonas **Prieš sąlyga:** Vartotojas prisijungęs prie administracinės aplinkos Vartotojas modulio konfigūravimo puslapyje **Po-sąlyga:** Nustatytas filtro šablonas konkrečiam kategorijų sąrašui

3.6 lentelė. PANAUDOJIMO ATVEJIS: Keisti filtro nuostatas

5. PANAUDOJIMO ATVEJIS: Keisti filtro nuostatas
Vartotojas/Aktorius: Administratorius
Aprašas: Nustatomos pagrindinės filtro vaizdavimo ir veikimo nuostatos
Prieš sąlyga: Vartotojas prisijungęs prie administracinės aplinkos Vartotojas modulio konfigūravimo puslapyje
Po-sąlyga: Filtro nuostatos pakeistos ir išsaugotos

3.7 lentelė. PANAUDOJIMO ATVEJIS: Trinti filtro šabloną

6. PANAUDOJIMO ATVEJIS: Trinti filtro šabloną
Vartotojas/Aktorius: Administratorius
Aprašas: Filtro šablonas kategorijų sąrašui gali būti panaikintas
Prieš sąlyga: Vartotojas prisijungęs prie administracinės aplinkos Vartotojas modulio konfigūravimo puslapyje
Po-sąlyga: Filtro šablonas nustatytoms kategorijoms pašalintas

3.8 lentelė. PANAUDOJIMO ATVEJIS: Naudotis paieška

7. PANAUDOJIMO ATVEJIS: Naudotis paieška
Vartotojas/Aktorius: El. parduotuvės lankytojas
Aprašas: El. parduotuvės lankytojas naudodamasis paieškos laukeliu suveda raktinius žodžius - informaciją ir gauna paieškos rezultatus.
Prieš sąlyga: Vartotojas lankosi el. parduotuvėje
Po-sąlyga: Gaunami efektyviai pateikiami paieškos rezultatai pagal raktažodžio paiešką

3.9 lentelė. PANAUDOJIMO ATVEJIS: Naudotis filtru

8. PANAUDOJIMO ATVEJIS: Naudotis filtru
Vartotojas/Aktorius: El. parduotuvės lankytojas
Aprašas: El. parduotuvės lankytojas naudodamasis šoninės skilties filtro parinktimis (pasirinkimais) nustato, kurias prekes nori matyti (filtruoja)
Prieš sąlyga: Vartotojas lankosi el. parduotuvėje
Po-sąlyga: Gaunami efektyviai pateikiami prekių rezultatai pagal vartotojo parinktys

3.4.4. Funkcinių reikalavimų sąrašas

3.10 lentelė. Reikalavimas #9.1

Reikalavimas #:	9.1	Reikalavimo tipas:	1	PA #:	1
Aprašymas:	Suindeksuoti paieškos indeksą				
Pagrindimas:	Administratorius turi turėti galimybę suindeksuoti paiešką rankomis, jeigu abejoja ar tai pavyko automatiškai				
Šaltinis	Administratoriai				
Tinkamumo kriterijus:	Administratorius gali specialiu mygtuku suindeksuoti paieškos indeksą				
Užsakovo patenkinimas:	5	Užsakovo nepatenkinimas:	5		
Priklausomybės:	-	Konfliktai:	-		
Papildoma medžiaga:	-				
Istorija:	Užregistruotas 2015 gegužės 21 d.				

3.11 lentelė. Reikalavimas #9.2

Reikalavimas #:	9.2	Reikalavimo tipas:	1	PA #:	2
Aprašymas:	Suindeksuoti filtro indeksą				
Pagrindimas:	Administratorius turi turėti galimybę suindeksuoti filtrą rankomis, jeigu abejoja ar tai pavyko automatiškai				
Šaltinis	Administratoriai				
Tinkamumo kriterijus:	Administratorius gali specialiu mygtuku suindeksuoti filtro indeksą				
Užsakovo patenkinimas:	5	Užsakovo nepatenkinimas:	5		
Priklausomybės:	-	Konfliktai:	-		
Papildoma medžiaga:	-				
Istorija:	Užregistruotas 2015 gegužės 21 d.				

3.12 lentelė. Reikalavimas #9.3

Reikalavimas #:	9.3	Reikalavimo tipas:	1	PA #:	3
Aprašymas:	Nustatyti *ElasticSearch* serviso serverį				
Pagrindimas:	Nustatomas į kur indeksuojama: gali būti nutolęs arba vietinis *ElasticSearch* serveris				
Šaltinis	Administratoriai				
Tinkamumo kriterijus:	Laukelis serverio nurodymui				
Užsakovo patenkinimas:	5	Užsakovo nepatenkinimas:	5		
Priklausomybės:	-	Konfliktai:	-		
Papildoma medžiaga:	-				
Istorija:	Užregistruotas 2015 gegužės 21 d.				

3.13 lentelė. Reikalavimas #9.4

Reikalavimas #:	9.4	Reikalavimo tipas:	1	PA #:	4
Aprašymas:	Priskirti kategorijų sąrašą filtrui				
Pagrindimas:	Filtras (filtro šablonas) gali būti pritaikytas tik konkrečiam kategorijų sąrašui				
Šaltinis	Administratoriai				
Tinkamumo kriterijus:	Kategorijų medžio/sąrašo pasirinkimas				
Užsakovo patenkinimas:	5	Užsakovo nepatenkinimas:	5		
Priklausomybės:	-	Konfliktai:	-		
Papildoma medžiaga:	-				
Istorija:	Užregistruotas 2015 gegužės 21 d.				

3.14 lentelė. Reikalavimas #9.5

Reikalavimas #:	9.5	Reikalavimo tipas:	1	PA #:	4
Aprašymas:	Priskirti filtruojamus atributus				
Pagrindimas:	Ne visi atributai gali būti reikalingi konkrečiam kategorijų sąrašui				
Šaltinis	Administratoriai				
Tinkamumo kriterijus:	Aktyvių atributų sąrašas/medis				
Užsakovo patenkinimas:	5	Užsakovo nepatenkinimas:	5		
Priklausomybės:	-	Konfliktai:	-		
Papildoma medžiaga:	-				
Istorija:	Užregistruotas 2015 gegužės 21 d.				

3.15 lentelė. Reikalavimas #9.6

Reikalavimas #:	9.6	Reikalavimo tipas:	1	PA #:	4
Aprašymas:	Priskirti filtruojamas savybes				
Pagrindimas:	Ne visos savybės gali būti reikalingos konkrečiam kategorijų sąrašui				
Šaltinis	Administratoriai				
Tinkamumo kriterijus:	Aktyvių savybių sąrašas/medis				
Užsakovo patenkinimas:	5	Užsakovo nepatenkinimas:	5		
Priklausomybės:	-	Konfliktai:	-		
Papildoma medžiaga:	-				
Istorija:	Užregistruotas 2015 gegužės 21 d.				

3.16 lentelė. Reikalavimas #9.7

Reikalavimas #:	9.7	Reikalavimo tipas:	1	PA #:	6
Aprašymas:	Trinti filtro šabloną				
Pagrindimas:	Tam tikri sukurti filtro šablonai konkrečiam kategorijų sąrašui gali būti nebereikalingi				
Šaltinis	Administratoriai				
Tinkamumo kriterijus:	Pašalintas filtro šablonas				
Užsakovo patenkinimas:	5	Užsakovo nepatenkinimas:	5		
Priklausomybės:	-	Konfliktai:	-		
Papildoma medžiaga:	-				
Istorija:	Užregistruotas 2015 gegužės 21 d.				

3.17 lentelė. Reikalavimas #9.8

Reikalavimas #:	9.8	Reikalavimo tipas:	1	PA #:	8
Aprašymas:	Filtro aktyvių savybių išvedimas vartotojui				
Pagrindimas:	Vartotojas turi gebėti pasirinkti pagal ką filtruoti				
Šaltinis	El. Parduotuvės lankytojai				
Tinkamumo kriterijus:	Matomos aktyvių savybių parinktys filtro skiltyje				
Užsakovo patenkinimas:	5	Užsakovo nepatenkinimas:	5		
Priklausomybės:	-	Konfliktai:	-		
Papildoma medžiaga:	-				
Istorija:	Užregistruotas 2015 gegužės 21 d.				

3.18 lentelė. Reikalavimas #9.9

Reikalavimas #:	9.9	Reikalavimo tipas:	1	PA #:	8
Aprašymas:	Filtro aktyvių atributų išvedimas vartotojui				
Pagrindimas:	Vartotojas turi gebėti pasirinkti pagal ką filtruoti				
Šaltinis	El. Parduotuvės lankytojai				
Tinkamumo kriterijus:	Matomi aktyvių atributų pasirinkimai filtro skiltyje				
Užsakovo patenkinimas:	5	Užsakovo nepatenkinimas:	5		
Priklausomybės:	-	Konfliktai:	-		
Papildoma medžiaga:	-				
Istorija:	Užregistruotas 2015 gegužės 21 d.				

3.19 lentelė. Reikalavimas #9.10

Reikalavimas #:	9.10	Reikalavimo tipas:	1	PA #:	7
Aprašymas:	Paieškos laukelio išvedimas				
Pagrindimas:	Vartotojas turi galėti suvesti raktinius žodžius į paieškos laukelį				
Šaltinis	El. Parduotuvės lankytojai				
Tinkamumo kriterijus:	Matomas paieškos laukelis sistemoje				
Užsakovo patenkinimas:	5	Užsakovo nepatenkinimas:	5		
Priklausomybės:	-	Konfliktai:	-		
Papildoma medžiaga:	-				
Istorija:	Užregistruotas 2015 gegužės 21 d.				

3.5. Nefunkciniai reikalavimai

3.5.1. Reikalavimai sistemos išvaizdai

Reikalavimo #:	10.1
Reikalavimo aprašymas:	Vartotojo sąsaja turi būti intuityvi vartotojui

Reikalavimo #:	10.2
Reikalavimo aprašymas:	Vartotojo sąsaja turi būti nesudėtingai perprantama ir suvokiama

Reikalavimo #:	10.3
Reikalavimo aprašymas:	Vartotojo sąsaja turi būti nesiskirti ir išnaudoti *PrestaShop* sistemos bendrą vartotojo sąsają

Reikalavimo #:	10.4
Reikalavimo aprašymas:	Vartotojo sąsaja turi būti pritaikyta mobiliesiems įrenginiams

Reikalavimo #:	10.5
Reikalavimo aprašymas:	Sistema turi atrodyti profesionaliai

Reikalavimo #:	10.6
Reikalavimo aprašymas:	Sistemos modulio vartotojo sąsaja turi būti švari ir sudaryti įvaizdį gero kokybiško kodo.

3.5.2. Reikalavimai panaudojamumui

Reikalavimo #:	11.1
Reikalavimo aprašymas:	Vartotojo sąsaja turi būti pritaikyta mobiliesiems įrenginiams

Reikalavimo #:	11.2
Reikalavimo aprašymas:	Sistema turi būti valdoma vidutines el. komercijos sistemos valdymo žinias turinčio asmens.

Reikalavimo #:	11.3
Reikalavimo aprašymas:	Sistema neturi būti sudėtingesnė už standartinius *PrestaShop* modulius ir vartotojo sąsają.

Reikalavimo #:	11.4
Reikalavimo aprašymas:	Sistema turi mokėti naudotis žmogus prieš tai moduliu nesinaudojęs.

Reikalavimo #:	11.5
Reikalavimo aprašymas:	Vartotojo sąsają turi būti galima išversti naudojant standartines *PrestaShop* vertimo galimybes.
Tinkamumo kriterijus:	Išnaudojama failų sistema *PrestaShop* vertimams

Reikalavimo #:	11.6
Reikalavimo aprašymas:	Vartotojo sąsaja turi būti draugiška diakritinėms ženklams.

3.5.3. Reikalavimai vykdymo charakteristikoms

Reikalavimo #:	12.1
Reikalavimo aprašymas:	Bet kokia sąsaja su vartotoju turi netrukti ilgiau 10 sekundžių.

Reikalavimo #:	12.2
Reikalavimo aprašymas:	Indeksuojamas prekių kiekis turi sėkmingai veikti su 40 tūkst. prekių

Reikalavimo #:	12.3
Reikalavimo aprašymas:	Prekių indeksavimas turi trukti protingą laiką priklausomai nuo prekių kiekio. 1000 prekių suindeksuojama per ne daugiau 120 sekundžių.

3.5.4. Reikalavimai veikimo sąlygoms

Reikalavimo #:	13. 1
Reikalavimo aprašymas:	Sistema turi būti galima naudotis administratoriaus/biuro tipo aplinkoje.

Reikalavimo #:	13.2
Reikalavimo aprašymas:	Sistema turi pilnai funkcionuoti naujausiose naršyklėse.
Tinkamumo kriterijus:	Sistemos visas funkcionalumas veikia naudojant naršyklių *Mozilla Firefox* 30+ ir *Google Chrome* 36+ versijas, *IE10*

Reikalavimo #:	13.3
Reikalavimo aprašymas:	Sistema turi būti suderinama su naujausia *PrestaShop* versija
Tinkamumo kriterijus:	Sistema turi būti suderinama su *PrestaShop* 1.6

Reikalavimo #:	13.4
Reikalavimo aprašymas:	Sistema turi būti perduodama klientams suspausta *.zip* formatu.

Reikalavimo #:	13.5
Reikalavimo aprašymas:	Sistema turi būti galima įdiegti naudojant standartinę *PrestaShop* modulių diegimo aplinką.

.

3.5.5. Reikalavimai sistemos priežiūrai

Reikalavimo #:	14.1
Reikalavimo aprašymas:	Sistema turi būti galima išplėsti ar papildyti papildomomis funkcijomis.

Reikalavimo #:	14.2
Reikalavimo aprašymas:	Sistemos modulis turi veikti iki kol neišleidžiama nesuderinta su moduliu *PrestaShop* versija.

3.5.6. Reikalavimai saugumui

Reikalavimo #:	15.1
Reikalavimo aprašymas:	Sistema turi būti prieinama visam aptarnaujančiam personalui.
Tinkamumo kriterijus:	Bet kuris aptarnaujančio personalo darbuotojas gali naudotis sistema.

Reikalavimo #:	15.2
Reikalavimo aprašymas:	Tik administratoriai konfigūruoja filtro ir paieškos parametrus
Tinkamumo kriterijus:	Administratoriai, kuriems suteiktos teisės konfigūruoti modulį

Reikalavimo #:	15.3
Reikalavimo aprašymas:	Sistemai galioja *PrestaShop* privatumo politika ir reikalavimai.

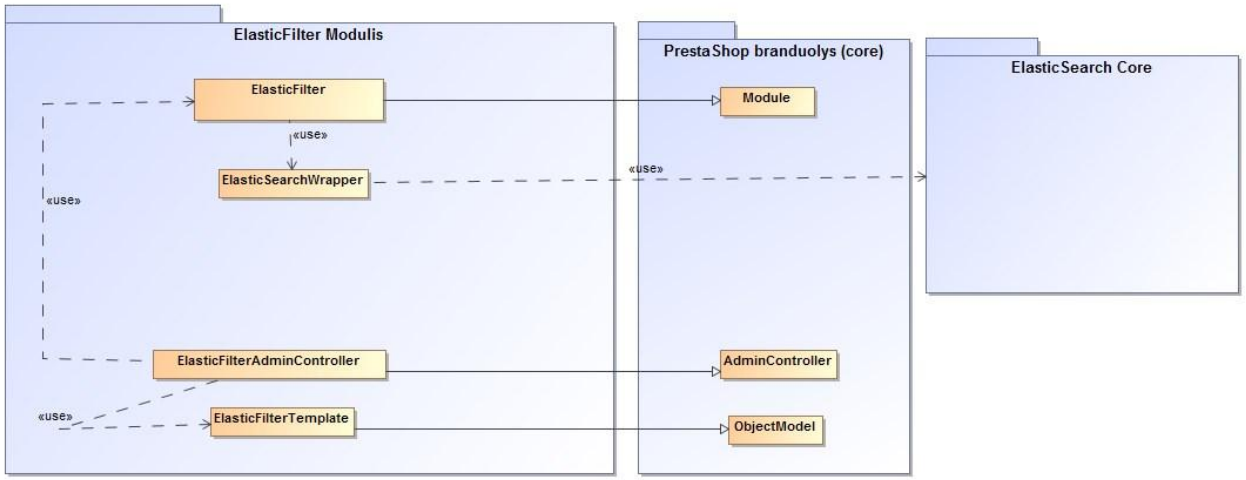
3.6. Sistemos architektūros modelis

Architektūros modelis skirtas sistemos architektūros aprašymui. Šiame darbe architektūra apibrėžiama įvairiais požiūriais, ir kiekvienam požiūriui pavaizduoti naudojamas atskiras modelis.

Architektūriniams sprendimams įtakos turinys reikalavimai:

- Modulis turi veikti *PrestaShop* el. komercijos sistemoje
- Visas valdymas modulio paremtas bendrais *PrestaShop* el. komercijos naudojamais principais
- Išnaudojama *PrestaShop* el. komercijos branduolio architektūra ir funkcionalumas
- Derinamasi ir laikomasi *PrestaShop* el. komercijos sprendimo naudojamų architektūros taisyklių atliekant branduolio perrašymą (angl. override)
- Naudojamas objektinis programavimas
- Sudarant sistemos architektūrą turi būti atsižvelgta į specifikavimo reikalavimus ir charakteristikas, funkcinius ir nefunkcinius reikalavimus

3.6.1. Sistemos statinis vaizdas



3.6 pav. klasių diagrama

Modulis yra neatsiejamas nuo *PrestaShop* branduolio, todėl pav. pavaizduotame statiniame sistemos vaizde papildomai atvaizduojamos ir pagrindinės modulio naudojamos *PrestaShop* el. komercijos platformos klasės ir kontroleriai.

Modulis išnaudoja trečiųjų šalių klasių biblioteką, kuri taip pat pažymėta kaip atskiras paketas.

Klasė *ElasticFilter*, kuri yra pagrindinis stuburas ir pagrindinė klasė be kurios modulis apskritai negalėtų veikti, būti įdiegtas, valdyti kitų kontrolerių ir klasių siejama su *PrestaShop* branduolio *Module* klase. Modulio klasė atsakinga už tinkamą modulio veikimą, *ElasticSearch* nuostatų apie serverį vietinį arba nutolusį saugojimą, pasijungimą prie įskiepių (angl. hook) sistemoje, jų apdorojimą ir atvaizdavimą.

Modulio klasė taip pat apdoroja ir vykdo planuotų užduočių (angl. cron job) eigą.

ElasticFilterAdminController atsakingas už nuostatų, susijusių su filtru ir paieška rodymu ir apdorojimu. Šis kontroleris yra susietas ir naudoja dalį *AdminController* funkcionalumo, kuris yra bendras visiems *PrestaShop* administravimo kontroleriams.

ElasticFilterTemplate yra esybė (angl. entity) sistemoje – atitinka kuriamo ir saugojamo filtro šabloną susietą su konkrečiomis kategorijomis. Esybės *PrestaShop* sistemoje turi bendrą *ObjectModel* funkcionalumą. Esybė saugo informaciją apie sukurtą šabloną susietą su kategorijomis.

ElasticSearchWrapper tai *ElasticSearch* branduolio ir paketo abstrakcija – tarpinis ir pagrindinis komunikatorius tarp *PrestaShop* modulio ir trečiosios šalies paketo – serviso. Šis kontroleris atsakingas už visą indeksavimo darbą, teisingą informacijos paruošimą indeksui, paties indekso valdymą ir apdorojimą.

3.6.2. Paketų detalizavimas

3.6.2.1. Duomenų nuskaitymo modulio klasių diagrama.

3.7 pav. duomenų nuskaitymo modulio klasių diagrama

Pagrindinių metodų paaiškinimas:

- install() – diegia modulį ir atlieka būtinas funkcijas diegimo metu
- uninstall() – atlieka veiksmus būtinus pašalinant modulį iš sistemos
- installDB() – sudiegia būtina SQL duomenų bazės struktūrą
- uninstallDB() – pašalina reikiamas lenteles iš SQL duomenų bazės
- doCronJob() – funkcija atsakinga už planuotos užduoties vykdymą
- checkSecureKey – sutikrinami planuotos užduoties paleidimo saugumo raktas (angl. tokian)
- hook[funkcijos pavadinimas] – įskiepiai į sistemos vidų – atsakingi už atitinkamų funkcijų papildomus pakeitimus
- doProductSearch() – vykdo prekių paiešką – komunikuoja su *Elastic*
- parseFilterQuery() – išanalizuoja pateiktus filtro pasirinkimus ir suformuoja užklausą paieškai indekse
- parseSearchQuery() – išanalizuoja paieškos pateiktus raktažodžius ir suformuoja užklausą paieškai indekse
- installTab() – diegia specialų administravimo kontrolerį į sistemą.

3.6.2.2. Detali *ElasticFilterAdminController* klasės diagrama

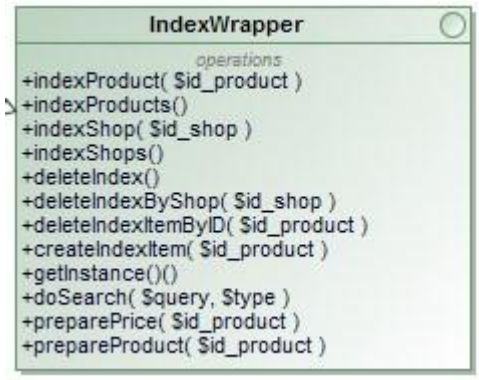
3.8 pav. detali ElasticFilterAdminController klasės diagrama

Pagrindinių metodų paaiškinimas:

- init() – paveldimas metodas iš pagrindinio administravimo kontrolerio, atsakingas už veiksmus inicijuojant kontrolerį
- postProcess() – atsakinga už formos apdorojimą
- getContent() – formos kūrimas ir išvedimas

- getSearchSettingsFields() - paieškos nuostatų laukų formavimas
- getFilterSettingsFields() - filtro nuostatų laukų formavimas
- getSearchSettingsFieldsValues() - paieškos nuostatų laukų reikšmių gavimas ir formavimas
- getFilterSettingsFieldsValues() - filtro nuostatų laukų reikšmių gavimas ir formavimas
- renderFilterSettingsForm() – filtro nuostatų forma sugeneruojame specialių pagalbinių klasių aprašymu
- deleteFilterTemplate () – ištrinamas filtro šablonas
- createFilterTemplate() – sukuriamas filtro šablonas ir saugomas DB

3.6.2.3. Detali *IndexWrapper* klasės diagrama



3.9 pav. detali *IndexWrapper* klasės diagrama

Abstrakti klasė siekianti integruoti kitus panašaus pobūdžio varikliukus (angl. engine) indekso kūrimui ir apdorojimui.

Pagrindinių metodų paaiškinimas:

- indexProduct() – prekės suindeksavimas
- indexproducts() – visų prekių indeksavimas
- indexShop() – indeksavimas vienos subparduotuvės
- indexShops() – indeksavimas visų subparduotuvių
- deleteIndex() – viso indekso šalinimas
- deleteIndexByShop() – indekso šalinimas pagal subparduotuvę
- deleteIndexItemById() – indekso įrašo šalinimas pagal prekės ID

- createIndexItem() – indekso įrašo kūrimas
- getInstance() – grąžinama klasės inicijuotas objektas
- doSearch() – vykdoma paieška indekse
- preparePrice() – paruošiama indeksui kaina pagal prekės rėžius
- prepareProduct() – paruošiama ir gaunama visa reikalinga informacija apie prekę indekso įrašo kūrimui.

3.6.2.4. Detali *ElasticSearchWrapper* klasės diagrama

3.10 pav. detali *ElasticSearchWrapper* klasės diagrama

Pagrindinių metodų paaiškinimas:

- checkStatus() – tikrinamas vietinio ar nutolusio serviso pasiekiamumas

3.6.2.5. Detali *ElasticFilterTemplate* klasės diagrama

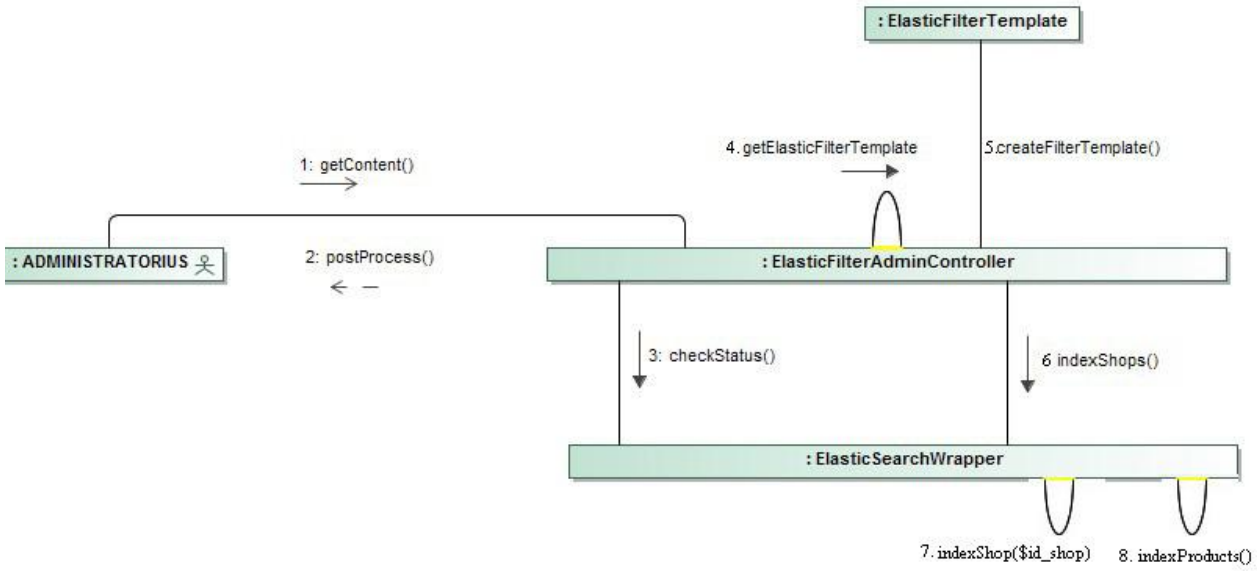
3.11 pav. detali *ElasticFilterTemplate* klasės diagrama

Pagrindinių metodų paaiškinimas:

- getElasticFilterTemplate() – gaunamas filtro šablonas ir jo informacija

Visi kiti metodai paveldami iš *PrestaShop* branduolio klasės, pavyzdžiui, *save()* ir kt.

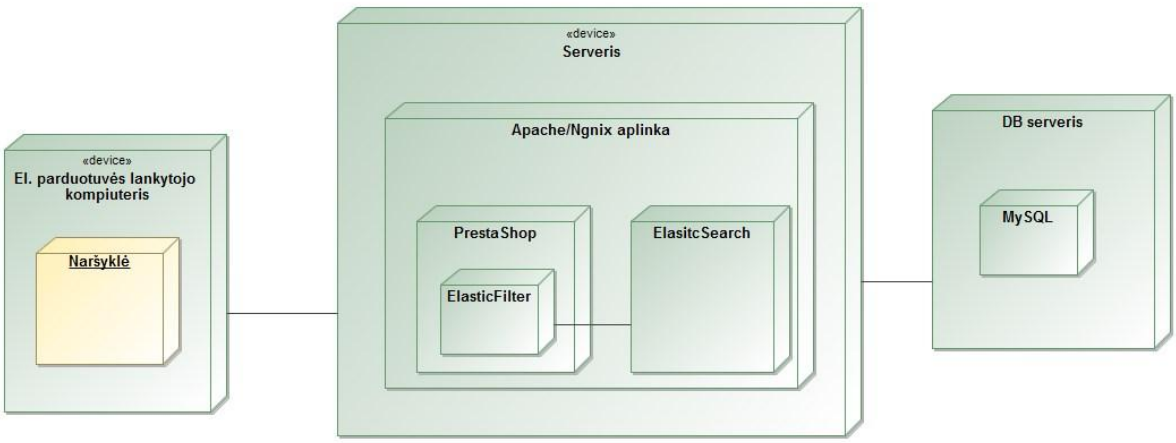
3.6.3. Sistemos dinaminis vaizdas



3.12 pav. bendradarbiavimo diagrama

3.12 pav. pavaizduota bendradarbiavimo diagrama sudaro pagrindinį įspūdį ir veiksmų seką iš administratoriaus pozicijos, t.y. kad sukuriamas filtro šablonas, suindeksuojamas indeksas pagal šį šabloną.

3.6.4. Išdėstymo vaizdas



3.13 pav. išdėstymo diagrama

Realizacijos schema puikiai atspindi ir atvaizduoja diegimo aplinką, jos ryšius.

3.6.5. Duomenų vaizdas

Duomenų esminį vaizdą aprašo nuostatų ir filtro šablonų saugojimo lentelė duomenų bazėje veikiančioje SQL principu. Šių lentelių struktūra ir bendras principas panaudojami iš standartinio *PrestaShop* modulio *blocklayered*.

3.20 lentelė. layered_filter DB lentelė

	Laukas	Tipas	Apibūdinimas
raktas	id_layered_filter	int(11)	Unikalus filtro šablono identifikatorius
	name	varchar(128)	Filtro šablono pavadinimas
	filters	TEXT	Saugomas sutrauktas (angl. serialize) *PHP* kalbos filtrų parinkčių rezultatas
	n_categories	int(11)	
	date_add	datetime	Filtro šablono sukūrimo data

3.21 lentelė. layered_filter_shop DB lentelė

	Laukas	Tipas	Apibūdinimas
	id_layered_filter	int(11)	Unikalus filtro šablono identifikatorius
	id_shop	int(11)	Subparduotuvės ID

ElasticSearch indekso vienas įrašas (angl. item):

```
{
    "_index" : "INDEKSO UNIKALUS KODAS",
    "_type" : "INDEKSO TIPAS, pavyzdžiui, products",
    "_id" : "PREKĖS ID",
    "_score" : 1.0,
    "_source":
                    {
                    "reference":"UNIKALUS KODAS",
                    "name_1":"PAVADINIMAS SU KALBOS ID",
                    "link_rewrite_1":"SEO URL SU KALBOS ID",
                    "description_short_1":"APRAŠYMAS SU KALBOS ID",
                    "search_keywords_1":
```

[

"20938",

"Obsessive Police dress",

"Blazingly yellow body on a racing car jumper style. Front and back deep cleavage. Lower part in a hot pant style nicely underlines the bottocks.",

"quantity":0,

"price":23.65,

"categories":["130","87"],

"condition":"new",

"id_manufacturer":"59",

"manufacturer_name":"Obsessive",

"weight":"0.000000",

"out_of_stock":2,

"id_category_default":"87",

"ean13":"",

"available_for_order":"1",

"customizable":"0",

"minimal_quantity":"1",

"show_price":"1",

"id_image":"111899",

"attribute_group_4":["1536"],

"attribute_group_5":["1792"],

"feature_10":"61",

"feature_9":"36",

"price_min_2":23,

"price_max_2":29,

"price_min_1":81,

"price_max_1":99}

}

Skaičiai prie pabaigos pavadinimų atitinkamai reiškia tų esybių (angl. entity) unikalų sistemoje esanti identifikatorių arba tiesiog ID. Laukams, kurie yra tekstinės informacijos tipo, pavyzdžiui, aprašymas ar trumpas aprašymas, pabaigoje esantys skaičiai atitinka sistemoje esančios kalbos ID.

4. TYRIMAI

Tyrimams ir eksperimentams atlikti buvo pasirinktos trys populiariausios el. komercijos sistemos: *Magento, PrestaShop, OpenCart*. Visos sistemos yra atvirojo kodo (*Magento* turi atvirojo kodo bendruomenės versiją (angl. community edition)), nemokamos ir pasiekiamos parsisiuntimui be jokių papildomų kaštų ar žinių.

Atvirojo kodo sistemos pasirinktos dėl patogumo, licencijos kaštų taupymo ir siekiant padėti programuotojams ar el. parduotuvių kūrėjams atsakyti į kylantį klausimą, kokią iš šių tarpusavyje konkuruojančių sistemų pasirinkti savo kitam projektui, jeigu žinoma apie didelių duomenų panaudojamumą ateityje. Komercinės sistemos dažniausiai turi savo palaikymo kaštus ir komandas, todėl jos neįtrauktos į tyrimą, kaip visiškai atskira ir nekonkuruojanti dalis tarp atvirojo kodo el. parduotuvių platformų.

Tyrimas išskaidytas į dvi atskiras logines dalis:

1. filtro indeksavimas ir veikimas;
2. paieškos indeksavimas ir veikimas

Kiekviena iš dalių turi savo specifikų, kuriuos aptartos tolimesniuose poskyriuose.

Tyrimo metu atliktas palyginimas tarp pačių *PrestaShop* esančių rinkoje modulių ir atvirojo kodo sistemų.

4.1. Indeksuojamo filtro greitaveikos tyrimas

El. parduotuvėje ir analizuojamose el. parduotuvėse filtras yra kelių lygių navigacijos blokas, kuris leidžia klientams pasirinkti ir atsifiltruoti prekių katalogą prekių kategorijos puslapyje pagal tam tikrus požymius. Pavyzdžiui, prekes galima filtruoti pagal jų tam tikrus požymius tarp kurių dažniausiai išskiriami prekių variantai (angl. attributes and groups) bei ypatybės (angl. features).

Prekių variantai (angl. attributes and groups) – tai prekių pasirinkimai turintys įtakos prekės kainai, sudėčiai ir/ar galimai svoriui. Prekių variantai, kitaip tariant, yra viena prekė su keletu pasirinkimu,

pavyzdžiui, suknelė S, M, L, XL dydžio ir kartu turinti pasirinkimus: raudona, žalia, geltona. Iš viso įmanoma turėti variantų, tarkime, S-raudona; S-žalia; S-geltona; M-žalia; ir t.t.

Prekių ypatybės (angl. features) – yra tekstinio pobūdžio charakterizuojančios prekę ypatybės, pavyzdžiui, prekė stalas gali būti metalinis arba medinis, tačiau klientas neturi galimybės pasirinkti kažkurios vienos iš abiejų ypatybių, arba stalas yra medinis, arba stalas yra metalinis. Šiuo atveju taip pat nėra variantų ir nesusidaro pasirinkimų matrica.

Kategorijų filtravimas (angl. categories) – moduliuose yra galimybės filtruoti pagal prekėms priskirtas kategorijas. Analogiška struktūra ir principu yra filtruojami ir gamintojai (angl. manufacturers) bei prekių tiekėjai (angl. suppliers).

Prekės kainos filtravimas (angl. price filter) – moduliuose yra galimybė filtruoti prekes pagal jų kainą. Šioje situacijoje gali skirtis rezultatai priklausomai nuo taikomos skaičiavimo logikos, viena iš analizuojamų sistemų *OpenCart* tokio filtravimo neturi.

Į filtro galimybes taip pat visose sistemose įtrauktas ir filtravimas pagal prekės būseną (yra/nėra sandėlyje).

4.1.1. *PrestaShop* standartinio modulio filtro indekso veikimas – modulis *blocklayered*

Kiekviena sistema atskirai ir individualiai skaičiuoja kainas, tarkime, *PrestaShop* el. komercijos sistema savo DB saugo tik prekės įvedamą kainą, o mokesčius, nuolaidas ir kt. skaičiuoja atvaizdavimo metu, todėl prieš kuriant filtro indeksą (angl. index) atliekami sudėtingi skaičiavimai, kurie užima laiką, kuris gali iš esmės skirtis nuo kitų sistemų. *PrestaShop* prekių filtro kainų indekse saugomos reikšmės nuo mažiausios įmanomos iki didžiausios įmanomos kainos, tačiau vaizduojant su nuolaidomis ir be nuolaidų neretai gaunamos paklaidos.

Visose sistemose išskyrus *OpenCart* filtravimas atliekamas indekso principu, t.y. prekių filtrui paruošiamas indeksas trečiųjų šalių sistemose, pavyzdžiui, *ElasticSearch* arba el. platformos naudojamoje duomenų bazėje.

4.1.2. *PrestaShop* komercinio *AjaxFilter* filtro indekso veikimas

Komercinis *Presto-Changeo* kūrėjų modulis filtrą kuria panašiu principu, kaip ir standartinis veikimas. Duomenų bazėje atskirose lentelėse saugojamas duomenų indeksas:

4.1 pav. *ajaxfilter* modulio filtro indekso duomenų bazės struktūra

Kiekviena duomenų bazės lentelės struktūra atvaizduota žemiau esančiuose paveikslėliuose:

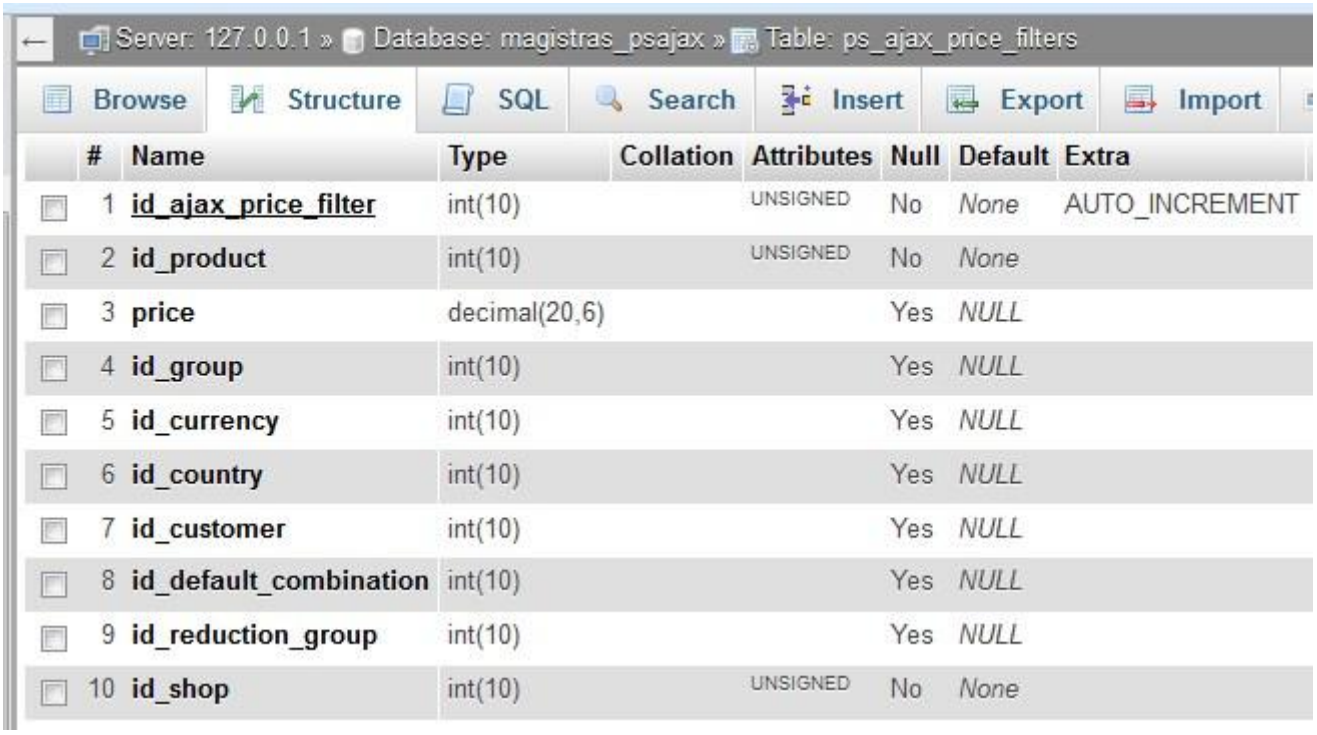
Server: 127.0.0.1 » Database: magistras_psajax » Table: ps_ajax_filters

Browse | Structure | SQL | Search | Insert | Export | Imp

#	Name	Type	Collation	Attributes	Null	Default	Extra
1	id_ajax_filter	int(10)		UNSIGNED	No	None	AUTO_INCREMENT
2	id_filter	int(10)		UNSIGNED	No	None	
3	filter_type	tinyint(3)		UNSIGNED	No	None	
4	id_filter_value	int(10)		UNSIGNED	No	None	
5	group_enabled	tinyint(1)		UNSIGNED	No	None	
6	enabled	tinyint(1)		UNSIGNED	No	None	
7	element	tinyint(1)		UNSIGNED	No	None	
8	increment	int(10)		UNSIGNED	No	None	
9	prefix	varchar(5)	utf8_general_ci		Yes	NULL	
10	postfix	varchar(5)	utf8_general_ci		Yes	NULL	
11	extra	text	utf8_general_ci		No	None	
12	location	tinyint(1)		UNSIGNED	No	None	
13	group_position	tinyint(3)		UNSIGNED	No	None	
14	filter_position	tinyint(3)		UNSIGNED	No	None	
15	id_shop	int(10)		UNSIGNED	No	None	

4.2 pav. *ps_ajax_filters* DB lentelės struktūra

4.2 pav. aprašytoje duomenų bazės lentelėje saugoma modulio nuostatų (angl. settings) informacija, t.y. filtrų pozicijos (kategorijų pasirinkimai gali būti aukščiau atributų ar savybių ir pan.), įjungtų/išjungtų filtrų ir pasirinkimų būsenos bei jų reikšmių būsena (atvaizduojama filtre ar ne) bei paties filtro pasirinkimo nuostata, pavyzdžiui, tai pasirinkimas ar slankiklis (angl. slider), ar pasirinkimas (angl. selectbox) ir t.t.



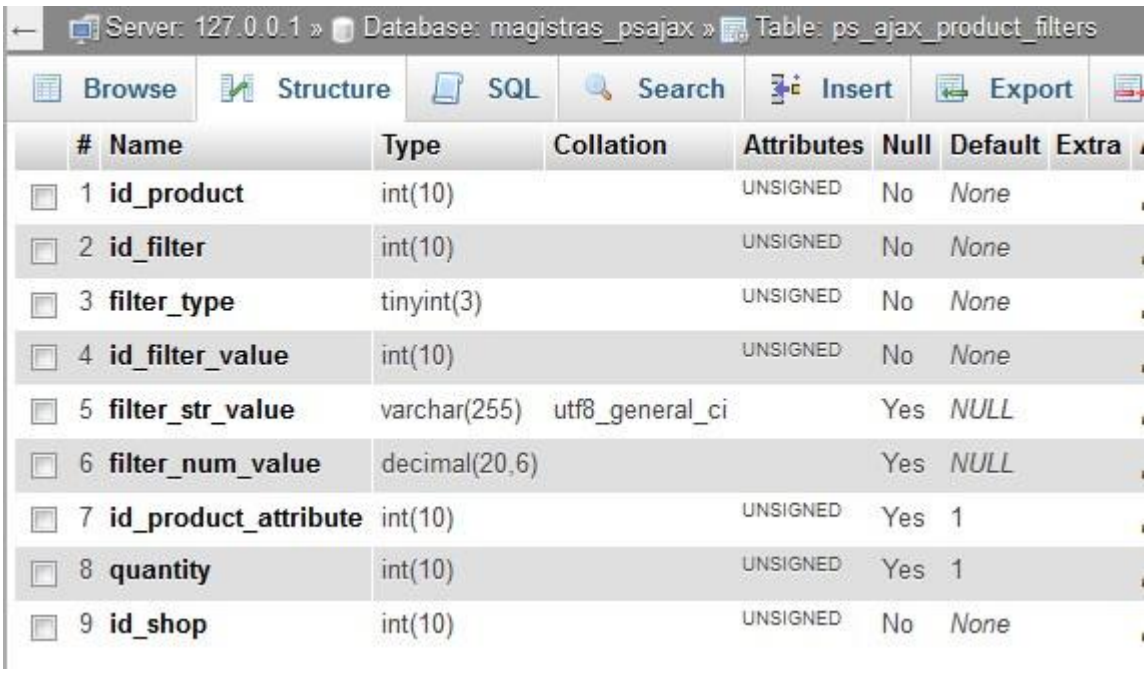
#	Name	Type	Collation	Attributes	Null	Default	Extra
1	id_ajax_price_filter	int(10)		UNSIGNED	No	None	AUTO_INCREMENT
2	id_product	int(10)		UNSIGNED	No	None	
3	price	decimal(20,6)			Yes	NULL	
4	id_group	int(10)			Yes	NULL	
5	id_currency	int(10)			Yes	NULL	
6	id_country	int(10)			Yes	NULL	
7	id_customer	int(10)			Yes	NULL	
8	id_default_combination	int(10)			Yes	NULL	
9	id_reduction_group	int(10)			Yes	NULL	
10	id_shop	int(10)		UNSIGNED	No	None	

4.3 pav. *ps_ajax_price_filters* DB lentelės struktūra

ps_ajax_price_filters DB lentelėje saugojama informacija susijusi su kainomis ir jų pasiskirstymu.

Kiekvienam prekės atributui ar, kitaip tariant, variantui indeksavimo metu skaičiuojama galutinė kaina (skirtumas nuo standartinio, kai indeksuojama jau yra galutinė kaina, o ne rėžis nuo-iki). Indeksavimo metu išskaičiuojama visa įmanoma logika ir surašomi visi įmanomi variantai į DB lentelę taip sukeliant daugybę įrašų DB kainų filtro lentelėje.

Indeksuojant kainą atsižvelgiama į tokius parametrus, kaip vartotojas, vartotojo grupė, valiuta, šalis, nuolaidos pritaikymas, pagrindinė prekė. Įmanomų variantų matrica ypač didelė.



#	Name	Type	Collation	Attributes	Null	Default	Extra
1	id_product	int(10)		UNSIGNED	No	None	
2	id_filter	int(10)		UNSIGNED	No	None	
3	filter_type	tinyint(3)		UNSIGNED	No	None	
4	id_filter_value	int(10)		UNSIGNED	No	None	
5	filter_str_value	varchar(255)	utf8_general_ci		Yes	NULL	
6	filter_num_value	decimal(20,6)			Yes	NULL	
7	id_product_attribute	int(10)		UNSIGNED	Yes	1	
8	quantity	int(10)		UNSIGNED	Yes	1	
9	id_shop	int(10)		UNSIGNED	No	None	

4.4 pav. *ps_ajax_product_filters* DB lentelės struktūra

Nagrinėjant šį modulį verta atkreipti dėmesį, kad modulis naudoja savo susikurtus *MySQL* DB indeksus [13], kurie DB padeda paieškos metu neskaityti nuo pirmos eilutės vis iš eilės kitą, o pirmiausia nusistato galimą poziciją pagal indeksus ir tik tuomet ieško toliau DB įrašų.

Komercinis filtras taip pat geresnis tuo, kad nuskaito iš duomenų bazės duomenų kiekį sluoksniais (angl. batches). Modulio architektūra yra tokia, kad kiekvieną galimą reikšmę įrašo kaip atskirą įrašą DB lentelėje. Tarkime, jei įrašoma bent viena informacija apie prekės savybę, atributą ar kategoriją, sandėlio būseną, kiekvienas informacijos blokas tampa atskiru DB lentelės įrašu, ši informacija saugoma *ps_ajax_product_filters* lentelėje.

Modulis atlieka ypač daug įrašymo operacijų indeksavimo metu.

4.1.3. *PrestaShop* realizuoto modulio filtro indekso veikimas

Filtro ir indekso paieškos kūrimas yra vienodas ir lygiai toks pats. Abiem atvejais sukuriamas tas pats anksčiau darbe aprašytas duomenų vienetas, neskaidant papildomai ar atskirai žodžių – viso teksto paieška (angl. fulltext search).

Kainos filtravimas atliekamas tokiu pačiu principu kaip ir standartiniame modulyje siekiant išlaikyti tą pačią logiką naudojamą *PrestaShop* sistemoje. Kainos skaičiuojamos „nuo-iki“ įmanomos mažiausios ir didžiausios kainos (dėl to įmanomos paklaidos filtro rezultatų rodyme esant nuolaidoms).

Visa kita informacija gaunama paprastų duomenų užklausų metu iš visų prekių sąrašo naudojant iteratorių tam, kad nebūtų vienu metu iš DB užkraunamas didelis kiekis duomenų ir neišeikvotų papildomos atminties.

Indeksas sukuriamas trečiųjų šalių *ElasticSearch* saugykloje anksčiau darbe minėta saugojimo forma kaip tekstinis įrašas.

Išsamų veikimo procesą ir kt. galima susidaryti iš ankstesnių magistrinio darbo dalių apibrėžiančių sistemos architektūrą.

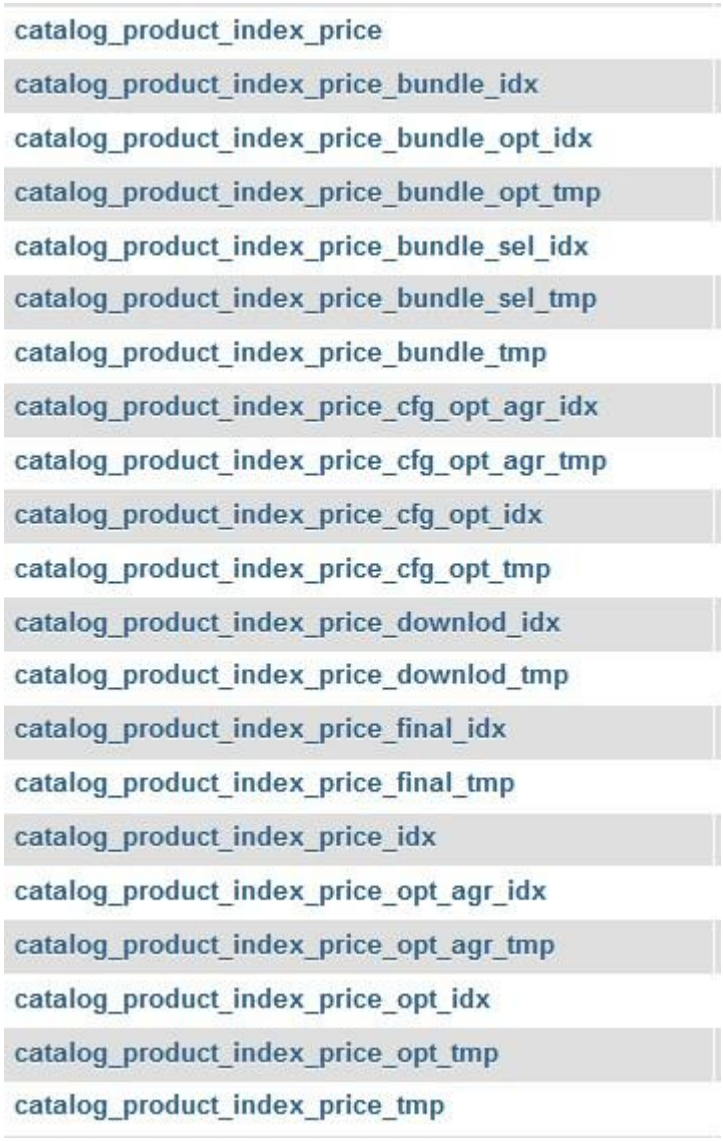
4.1.4. *Magento* standartinio filtro indekso veikimas

Magento naudoja savotišką denormalizavimą (angl. denormalizaton), kad paieškos sistema veiktų greičiau bei išnaudoja *EAV(*angl. Entity-Attrbute-Value) modelį, kuriuo kaip ir *PrestaShop* el. komercijoje prekių informacija išskaidoma į mažesnes lenteles, tarkime, lenteles: *product, product_attribute_values, product_attributes* ir t.t.

Magento sistemoje paieškos indeksavimas yra gudresnis lyginant su *PrestaShop* moduliais ir standartiniu veikimu dėl jau naudojamų Fabriko (angl. factory design pattern) ir Iteratoriaus (angl. iterator) šablonų. Indeksams valdyti ir kurti yra sukurta *IndexIteratorFactory* klasė, kuri naudojama įvairiems indeksams kurti ir apgalvota iš anksto galimų indeksų pridėjimui ateityje.

Magento filtras išskaidytas į 9 atskirus filtrus apimančius *EAV* modulio indeksaciją, kainų indeksaciją, paieškos indeksaciją.

Filtrui naudojamas prekių indeksas išskaidytas per lenteles DB:



4.5 pav. *Magento* kainų indekso lentelės

EAV DB lentelės yra bendros paieškos ir filtro indeksacijai, todėl plačiau aptartos prie paieškos.

4.6 pav. *catalog_product_index_price* DB lentelė *Magento* sistemoje

4.1.5. *OpenCart* standartinio filtro indekso veikimas

OpenCart iš esmės daug primityvesnė sistema nei kitos aptartos magistriniame darbe. Čia filtravimui, kaip ir paieškai aptariamai toliau šiame darbe, nėra naudojamas joks filtrų indeksas. Patys filtrai kuriami rankomis ir susiejami su prekės atributais, kategorijomis ir kt. vėliau.

Tarkime, mes susikuriame filtrą ir jo reikšmes (angl. values), tas reikšmes priskiriame rankomis atitinkamai norimiems atributams, kategorijoms, pasirinkimams ir kt., todėl prie kiekvienos esybės (angl. entity) atsiranda tarpinė filtro pasirinkimų ir sąsajos DB lentelė, pavyzdžiui, *oc_product_filter*, kurios struktūra pateikta žemiau.

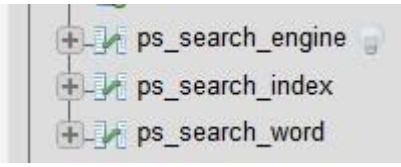
4.7 pav. *OpenCart* prekės filtrų sąsajos lentelė DB

4.2. Paieškos indekso ir naudojimo greitaveikos tyrimas

Paieška el. komercijos sprendimuose sudaro esminę veikimo dalį, todėl turi veikti tinkamai ir greitai. Visose analizuojamose sistemose paieška sukuriama taip pat prieš tai pasiruošiant indeksą iš prekių informacijos išskyrus *OpenCart* el. komercijos platformą.

4.2.1. *PrestaShop* standartinė paieška ir veikimas

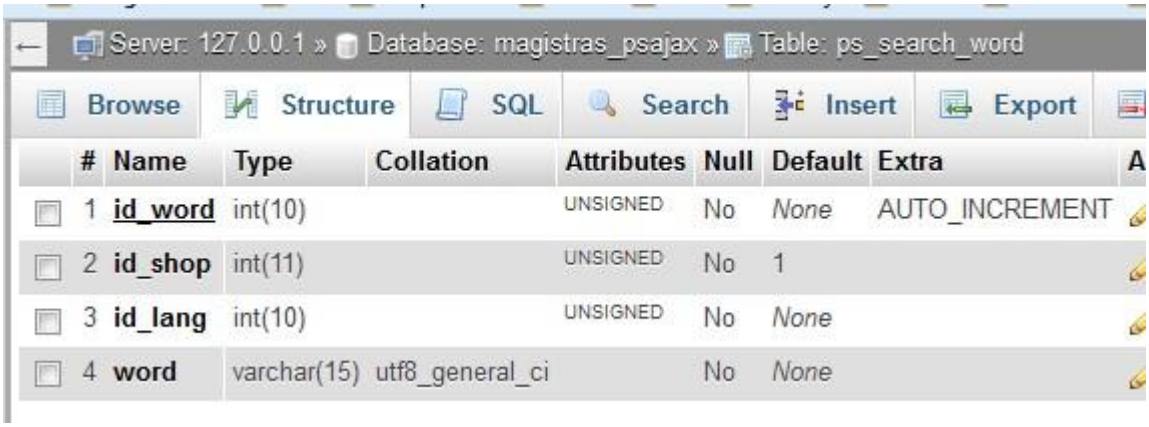
PrestaShop el. komercijos sistemoje standartiškai skiriamos trys duomenų bazės lentelės.



4.8 pav. *PrestaShop* paieškai naudojamos DB lentelės

4.9 pav. *PrestaShop paieškai naudojama lentelės ps_search_engine struktūra*

4.10 pav. *PrestaShop* paieškai naudojama lentelės *ps_search_index* struktūra



4.11 pav. *PrestaShop* paieškai naudojama lentelės ps_search_word struktūra

ps_search_index DB lentelėje saugomi įrašai apie žodžius esančius *ps_search_word* lentelėje ir jų poveikį paieškos rezultatams, t.y. tarkime, žodžiai iš kategorijų, gamintojų, prekės aprašymo, pavadinimo ir kt. gali turėti skirtingą „svorį“, kuris nustatomas administravime. Šie „svoriai“ surašomi lentelėje *ps_search_index*, o patys žodžiai lentelėje *ps_search_word.*

Duomenų pavyzdžiai pateikti žemiau esančiame paveikslėlyje (kairėje *ps_search_word,* dešinėje *search*):

	id_word	id_shop	id_lang	word
e	283	1	1	100
e	24	1	1	2010
e	64	1	1	accessories
e	13	1	1	accessorize
e	566	1	1	adjustable
e	60	1	1	attention
e	54	1	1	beautiful

id_product	id_word	weight
1	1	7
1	2	9
2	2	3
4	2	2
6	2	2

4.12 pav. *PrestaShop* paieškos lentelių duomenų pavyzdžiai

Indeksavimas tokiu pačiu algoritmu atliekamas visoms kalboms.

4.2.2. *PrestaShop* realizuoto modulio paieškos veikimas

Filtro ir indekso paieškos kūrimas yra vienodas ir lygiai toks pats. Abiem atvejais sukuriamas tas pats anksčiau darbe aprašytas duomenų vienetas, neskaidant papildomai ar atskirai žodžių.

Išsamų veikimo procesą ir kt. galima susidaryti iš ankstesnių magistrinio darbo dalių apibrėžiančių sistemos architektūrą.

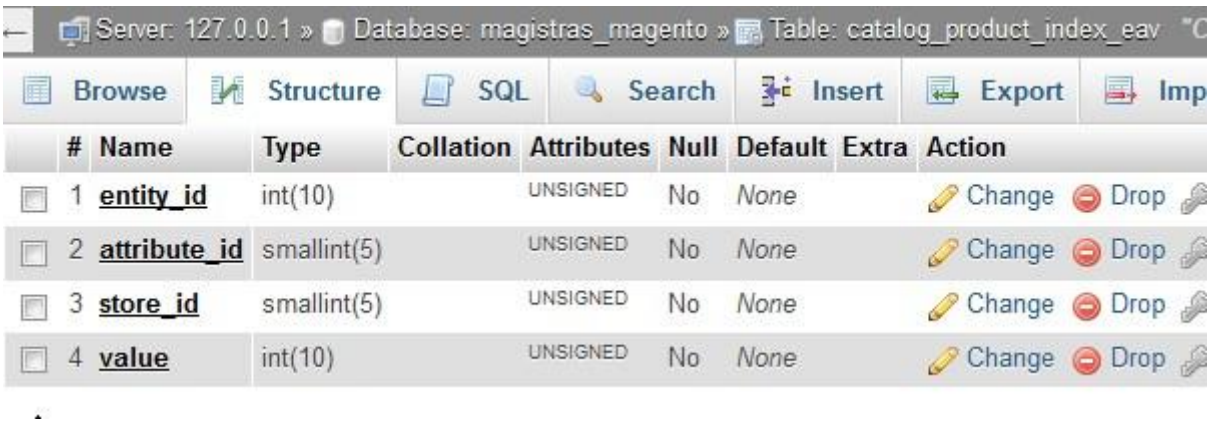
4.2.3. *Magento* standartinio modulio paieškos veikimas

Išlaikoma denormalizacijos ir *EAV* architektūra.

Procesas atsiremia į *MySQL* duomenų indeksą ir DB esančias lenteles bei jų struktūras:

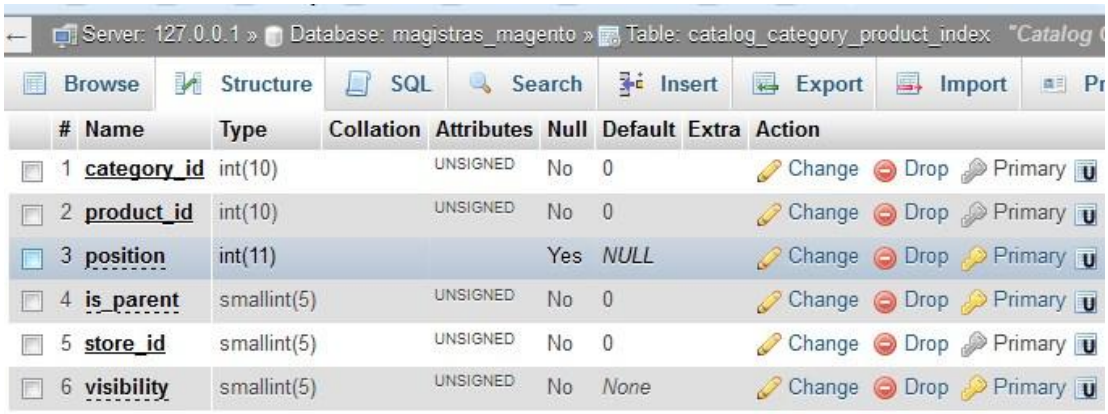
catalog_product_index_eav
catalog_product_index_eav_decimal
catalog_product_index_eav_decimal_idx
catalog_product_index_eav_decimal_tmp
catalog_product_index_eav_idx
catalog_product_index_eav_tmp

4.13 pav. *Magento* DB lentelės paieškos indeksui



4.14 pav. *catalog_product_index_eav* lentelė *Magento* sistemoje

Nurodytos duomenų bazės struktūra iš esmės yra vienoda su kitomis pavaizduotomis lentelėmis, todėl jų struktūra papildomai nekartojama. Tačiau ne tik prekės, bet ir kategorijos yra analogiškai indeksuojamos, kurių indeksas vėliau panaudojamas paieškoje. Už tai atsakingos kategorijos indekso lentelės DB*: catalog_category_product_index* ir *catalog_category_product_index_tmp*, kurios struktūra yra vienoda ir pateikta žemiau.



4.15 pav. *catalog_category_product_index* ir *catalog_category_product_index_tmp* struktūra

Denormalizacijos ir *EAV* architektūra turėtų būti greitesnė už tipinį duomenų bazės indeksą bei jo kūrimą

4.2.4. *OpenCart* standartinio modulio paieškos veikimas

OpenCart primityvesnė sistema, todėl jokio paieškos indekso nenaudoja. Prekių informacija gauna sugeneruodama vieną prekių gavimui per panaudojamą per visą sistemą funkciją *getProducts($filter_data)*, kurios parametras yra surūšiuoti duomenys pagal ką ieškoti prekių DB lentelėse.

```
if (isset($this->request->get['search']) || isset($this->request->get['tag'])) {
    $filter_data = array(
        'filter_name'         => $search,
        'filter_tag'          => $tag,
        'filter_description'  => $description,
        'filter_category_id'  => $category_id,
        'filter_sub_category' => $sub_category,
        'sort'                => $sort,
        'order'               => $order,
        'start'               => ($page - 1) * $limit,
        'limit'               => $limit
    );

    $product_total = $this->model_catalog_product->getTotalProducts($filter_data);

    $results = $this->model_catalog_product->getProducts($filter_data);
```

4.16 pav. *OpenCart* paieškos kodo logikos išeities kodo fragmentas

Sistemoje filtrų parametro kintamasis užpildomas ir paruošiamas pagal įvestą paieškos žodį (angl. search keyword) paprastomis užklausomis ir paprastu masyvas-masyve kodu, pavyzdžiui, atrenkant kategorijas.

5. EKSPERIMENTINĖ DALIS

Eksperimentai atlikti su viena ir ta pačia technine įranga:

- 16 GB *RAM* operatyvioji atmintis
- *Intel Core* i7-4770 3,40 Ghz procesorius
- 64-bit *Windows* aplinka
- Naudota *XAMPP PHP* programavimo aplinka su įdiegtu *ElasticSearch 2.3.3* paleistu procesu
- Programinio kodui keisti ir metrikos nustatyti *Jetbrains PHPStorm*

Testavimo įrankiai ir pagalba:

- *XDebug PHP* programavimo kalbai skirtas įrankis greitaveikai tirti
- *WebGrind* –sugeneruotų greitaveikos failų analizės įrankis su vartotojo sąsaja

Tyrimo metu, kur įmanoma, buvo parinkta po vieną filtro nuostatą: kainos, savybės, atributo, sandėlio būsenos ir pan. nuostatos buvo įjungtos filtravimo metu. Paieškos metu buvo įrašomas vienas tas pats žodis randamas prekėse, pavyzdžiui, „dress“ ar „apple“.

Prekėms dubliuoti ir pradiniams duomenis sukurti buvo naudojami automatiniai generavimo įrankiai- pavyzdinis generavimo įrankio išeities kodas (angl. source code) priedų dalyje – Priedas nr. 1.

Susidūrus su didelių duomenų problema informacija buvo apdorojama ir per komandinę eilutę naudojant *Windows* komandinės eilutės komandas:

sed -n '$=' filename

ir

sed –n –e {count_from},{count_to}p –e {count_to}q filename > part_of_results_file.txt

5.1. Filtro indeksavimo ir panaudojimo greitaveikos eksperimentas

5.1.1. Tyrimo eiga

Tyrimo metu buvo sugeneruoti skirtingi prekių imčių kiekiai ir su jais atliktas filtro indeksavimas, o po to filtro panaudojimas el. komercijos sistemos platformos lankytojams matomoje dalyje, fiksuotas užkrautos sistemos greitis ir užklausų detalizavimas pagal ilgiausiai trunkančias taip sprendžiant ir darant išvadas, dėl ko trunka ilgiausias sistemos veikimas. Kiekvienas eksperimentas atliktas po 3 kartus ir vertinamas eksperimentų vidurkis.

Kadangi kai kurie indeksai išskaidyti į kelias dalis, skaičiuojamas kiekvieno atskiro indekso proceso laikas, o vėliau sumuojamas ir naudojamas tolimesniame palyginime. Pavyzdžiui, atskirus indeksus turi *PrestaShop* sistema: kainų, atributų ir SEO nuorodų indeksai bei *Magento* sistema: prekių-kategorijų, prekių-kainų, prekių-atributų prekių-likučių indeksus ir t.t.

5.1.2. Tyrimo rezultatai

5.1 lentelė. filtro indeksavimo eksperimentas nr. 1

Duomenų kiekis	***Blocklayered* suminis**	***AjaxFilter*, ms**	***ElasticFilter*, ms**	***Magento* suminis, ms**	***OpenCart*, ms**
100	795	7465	6152	826	-
1000	795	75784	50007	1672	-
2000	906	137757	82917	3264	-
5000	1146	3892410	372151	7896	-
10000	1791	2746029	1728257	8323	-
20000	4012	5212340	3348102	14212	-
50000	7155	9760978	7102610	27542	-

5.2 lentelė. filtro indeksavimo eksperimentas nr. 2

Duomenų kiekis	***Blocklayered* suminis**	***AjaxFilter*, ms**	***ElasticFilter*, ms**	***Magento* suminis, ms**	***OpenCart*, ms**
100	770	8309	6154	875	-
1000	864	74814	49947	1901	-
2000	912	132981	98851	3916	-
5000	1123	385552	351853	6994	-
10000	1585	2644424	1714876	8893	-
20000	3946	5420640	3469102	13339	-
50000	6890	9615691	7265412	28751	-

5.3 lentelė. filtro indeksavimo eksperimentas nr. 3

Duomenų kiekis	***Blocklayered* suminis**	***AjaxFilter*, ms**	***ElasticFilter*, ms**	***Magento* suminis, ms**	***OpenCart*, ms**
100	770	7734	6154	801	-
1000	864	75222	49873	1545	-
2000	912	131466	99118	2964	-
5000	1123	391043	394445	7199	-
10000	1585	2446029	1750119	7865	-
20000	3946	5211785	3429102	18645	-
50000	6890	9823154	7411058	26142	-

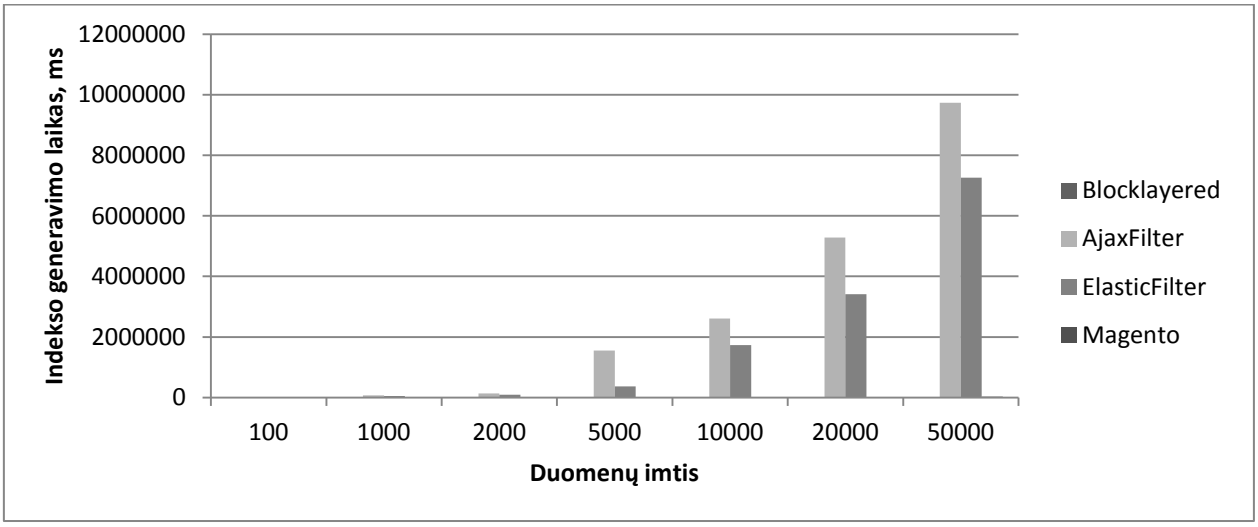
OpenCart šio bandymo metu neturi indeksavimo, todėl papildomai nėra duomenų apie rezultatus, įtraukta į lentelę siekiant parodyti, kad neatliekamas joks indeksavimas apskritai.

Iš pateiktų duomenų ir rezultatų lentelių matyti, kad kiekvieno bandymo metu nežymiai skiriasi rezultatai, kas gali būti paaiškinta papildomų funkcijų išnaudojimu, kompiuterio apkrovos skirtumais būtent tomis sekundėmis, kai testuojama ir atliekamas eksperimentas. Visų rezultatų išvestas vidurkis atvaizduotas lentelėje žemiau.

5.4 lentelė. filtro indekso visų eksperimentų vidurkis

Duomenų kiekis	***Blocklayered*, ms**	***AjaxFilter*, ms**	***ElasticFilter*, ms**	***Magento*, ms**	***OpenCart*, ms**
100	782	7836	6153	834	-
1000	987	75273	49942	1706	-
2000	942	134068	93629	3805	-
5000	1122	1556335	372816	7363	-
10000	1684	2612161	1731084	8360	-
20000	3935	5281588	3415435	15399	-
50000	6911	9733274	7259693	27478	-

Gautų rezultatų vidurkis diagrama pavaizduotas vaizdiniam duomenų palyginimui tarp filtrų indeksavimo greičių.



5.1 pav. filtro indekso visų eksperimentų vidurkis

Iš visų rezultatų matyti, kad greičiausias ir labiausiai besiskiriantis indeksavimas yra standartinio *PrestaShop blocklayered* filtro. Nesitikėjus tokių rezultatų ir papildomai išnagrinėjus dar kartą filtro veikimą, matyti, kad filtro efektyvumas yra paremtas tuo, kad jis atlieka darbą tik su DB užklausų lygmenyje: negauna ir neapdoroja jokių duomenų su kintamaisiais ar atmintimi išskyrus kainos generavimo indeksą, kuriame papildomai skaičiuoja informaciją. Dėl tos pačios priežasties ir to paties panaudojimo *Magento* indeksas taip pat žymiai greitesnis už kitų lyginamųjų modulių.

Taip pat verta paminėti, kad greičiau veikiantys moduliai turi išsiskirstę indeksus į atskirus indeksus, kaip buvo minėta anksčiau, *PrestaShop* sistema: kainų, atributų ir SEO nuorodų indeksai bei *Magento* sistema: prekių-kategorijų, prekių-kainų, prekių-atributų prekių-likučių indeksu - tai

akivaizdžiai leidžia sutaupyti laiko: mažiau duomenų apdorojama vienu metu, neapkraunama operatyvioji atmintis, vykdymo laikas trumpėja, visas procesas efektyvesnis.

Visi kiti analizuoti filtro moduliai ir sistemos indeksavimo metu atlieka papildomus duomenų paėmimus iš DB, pavyzdžiui, prekių informacijai surenka savybės, atributus ir tik tuomet apdoroja atgal į duomenų bazę.

Kitų eksperimentų metu buvo nagrinėjama suindeksuotos informacijos filtrams panaudojimas ir greitaveika, kurių rezultatai pateikti lentelėse žemiau.

5.5 lentelė. filtro panaudojimo greitaveikos eksperimentas nr. 1

Duomenų kiekis	***Blocklayered*, ms**	***AjaxFilter*, ms**	***ElasticFilter*, ms**	***Magento*, ms**	***OpenCart*, ms**
100	1487	2834	3244	1864	720
1000	1902	2773	2916	2954	764
2000	2726	2602	3111	4535	897
5000	3620	3328	3431	7349	904
10000	5145	5127	4843	10656	2338
20000	9626	8728	7399	19180	4596
50000	15410	14887	11786	31987	12359

5.6 lentelė. filtro panaudojimo greitaveikos eksperimentas nr. 2

Duomenų kiekis	***Blocklayered*, ms**	***AjaxFilter*, ms**	***ElasticFilter*, ms**	***Magento*, ms**	***OpenCart*, ms**
100	1487	2883	3100	1532	715
1000	1755	2855	3538	2762	729
2000	1932	2531	3660	4570	891
5000	3556	3015	3507	7651	1462
10000	5153	5012	3697	11436	2371
20000	9849	8958	8122	21051	4527
50000	16125	14332	12995	32557	12018

5.7 lentelė. filtro panaudojimo greitaveikos eksperimentas nr. 3

Duomenų kiekis	***Blocklayered*, ms**	***AjaxFilter*, ms**	***ElasticFilter*, ms**	***Magento*, ms**	***OpenCart*, ms**
100	1549	2973	3330	1864	654
1000	2950	2933	3513	2954	710
2000	1962	2507	3684	4535	887
5000	3604	2991	4007	7349	1488
10000	5033	5019	4368	10656	2356
20000	9816	8768	8072	19180	4421
50000	16125	18760	13125	31987	11979

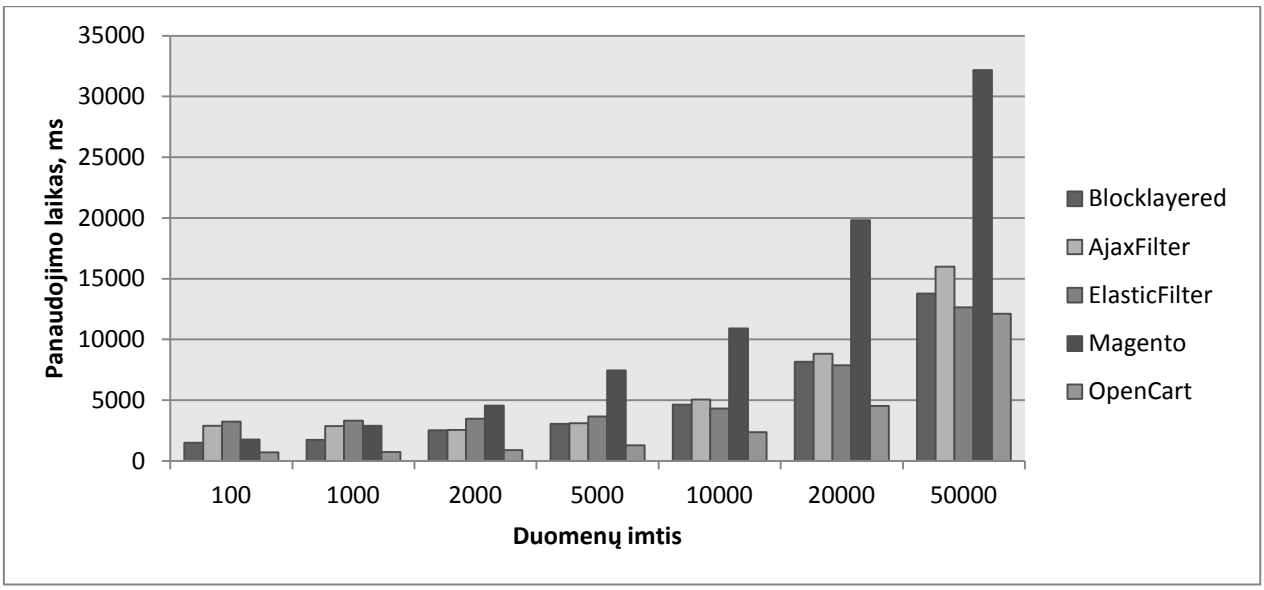
Iš gautų rezultatų matoma, kad didelis duomenų kiekis turi mažai įtakos pačiam filtro veikimui ir lankytojų matomiems rezultatams atvaizduoti, tačiau kai kuriais atvejais 18 ar 30 sekundžių ir daugiau veikimas jau nėra tinkamas ir galimas realiose el. parduotuvėse.

Visų rezultatų išvestas vidurkis atvaizduotas lentelėje žemiau.

5.8 lentelė. suvidurkinti filtro panaudojimo rezultatai

Duomenų kiekis	***Blocklayered*, ms**	***AjaxFilter*, ms**	***ElasticFilter*, ms**	***Magento*, ms**	***OpenCart*, ms**
100	1508	2897	3225	1753	696
1000	1735	2854	3322	2890	734
2000	2536	2547	3485	4547	892
5000	3046	3111	3648	7450	1285
10000	4634	5053	4303	10916	2355
20000	8169	8818	7864	19804	4515
50000	13784	15993	12635	32177	12119

Duomenys geresniam vaizdui ir skirtumams pastebėti atvaizduoti diagrama, kurią matote žemiau.



5.2 pav. suvidurkinti filtro panaudojimo rezultatai

Iš atvaizduotų rezultatų matome, kad nors geriausius rezultatus ties mažais duomenimis rodė *OpenCart*, kuri turi ypač primityvią tiesiog užklausomis veikiančią sistemą netgi be jokio indekso, su dideliais duomenimis tapo praktiškai neefektyvus ir nepanaudojamas bei susilygino su kitomis sistemomis.

Eksperimento metu pastebėta, kad *AjaxFilter* DB sukuria iki 15 kartų daugiau įrašų, nei indeksuojama prekių, kas bendrai lėtina patį indeksavimą. Vieno indekso metu sukuriama iki 6-12 mln. įrašų vienoje indekso DB lentelėje.

Visi kiti filtrai naudojantys indeksus demonstruoja panašius rezultatus, todėl galima daryti pastebėjimą, kad duomenų kiekis pačių duomenų atvaizdavimui lankytojams ir pasiėmimas iš indekso trunka daugmaž vienodai ir sudėtingai nepriklausomai nuo duomenų kiekio, o didžiausia problema yra paruošti tą indeksą, nes indeksavimo laikas, kaip atvaizduota anksčiau buvusiose diagramose, yra itin didelis ir netinkamas šių dienų el. komercijos atvejams.

5.2. Paieškos indeksavimo ir panaudojimo greitaveikos eksperimentas

5.2.1. Tyrimo eiga

Tyrimo metu buvo sugeneruoti skirtingi prekių imčių kiekiai ir su jais atliktas paieškos indeksavimas, o po to suindeksuotos paieškos suaktyvinimas el. komercijos sistemos platformos lankytojams matomoje dalyje Eksperimentų metu fiksuotas užkrautos sistemos greitis ir rezultatų išvedimas į ekraną. Kiekvienas eksperimentas atliktas po 3 kartus ir vertinamas eksperimentų vidurkis.

Šio eksperimentu metu nedalyvauja modulis *AjaxFilter*, nes neturi sąsajų su paieška *PrestaShop* platformoje, pabrėžtina, kad *Magento* sistemos atveju apsiribojama tik vienu bendros tekstinės paieškos indeksu *catalogsearch_fulltext*, o *OpenCart* sistema vėlgi neturi jokio indekso.

Magistrinio darbo metu sukurtas modulis generuoja ir naudoja tą patį indeksą tiek fltravimui, tiek paieška, todėl jo rezultatai sutampa su indekso filtro kūrimo metu toliau pateiktuose tyrimo rezultatuose.

5.2.2. Tyrimo rezultatai

5.9 lentelė. paieškos indekso eksperimentas nr. 1

Duomenų imtis	**Standartinė paieška, ms**	***ElasticFilter*, ms**	***Magento*, ms**	***OpenCart*, ms**
100	10411	6154	1024	-
1000	164201	49873	4398	-
2000	234918	99118	8339	-
5000	642056	394445	14176	-
10000	1106584	1750119	31187	-
20000	5151264	3429102	59256	-
50000	13005972	7411058	148142	-

5.10 lentelė. paieškos indekso eksperimentas nr. 2

Duomenų imtis	**Standartinė paieška, ms**	***ElasticFilter*, ms**	***Magento*, ms**	***OpenCart*, ms**
100	10782	6152	792	-
1000	103521	50007	4172	-
2000	213262	82917	8277	-
5000	653005	372151	15198	-
10000	1098145	1728257	30198	-
20000	5019121	3348102	57129	-
50000	12578945	6802610	150438	-

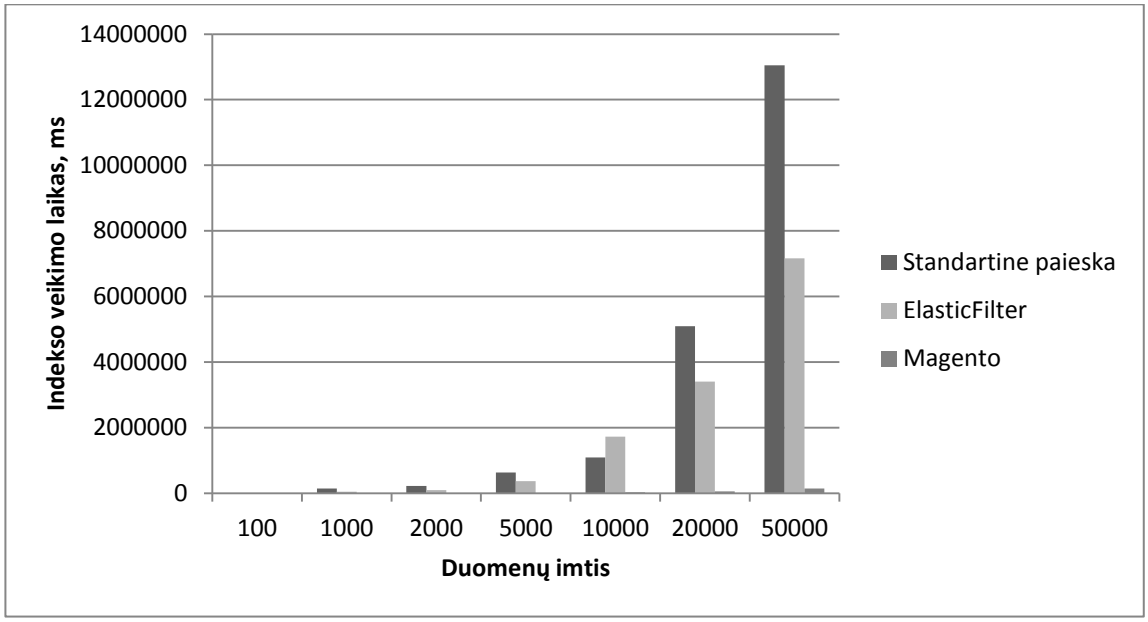
5.11lentelė. paieškos indekso eksperimentas nr. 3

Duomenų imtis	**Standartinė paieška, ms**	***ElasticFilter*, ms**	***Magento*, ms**	***OpenCart*, ms**
100	10341	6154	4165	-
1000	184130	49947	4911	-
2000	228510	98851	6179	-
5000	613308	351853	7834	-
10000	1082180	1714876	7292	-
20000	5118594	3429102	9103	-
50000	13551089	7265412	8913	-

Visų rezultatų išvestas vidurkis atvaizduotas lentelėje žemiau.

5.12 lentelė. vidurkis paieškos indekso eksperimentų

Duomenų imtis	**Standartinė paieška, ms**	***ElasticFilter*, ms**	***Magento*, ms**	***OpenCart*, ms**
100	10511	6153	951	-
1000	150617	49942	4320	-
2000	225563	93629	8336	-
5000	636123	372816	14546	-
10000	1095636	1731084	30446	-
20000	5096326	3402102	58431	-
50000	13045335	7159693	147454	-



5.3 pav. vidurkis paieškos indekso eksperimentų

Iš gautų rezultatų matyti, kad standartinė paieška yra pati neefektyviausia, nors atrodo gana primityvi ir naudoja panašius būdus kaip ir filtravimo metu, tik čia dar įvedama papildoma logika ir apdorojimas, kurie ir eksponentiškai sulėtina darbą, kai yra didelių duomenų kiekis.

Sukurto modulio paieškos indekso laikas yra lygiai toks pats kaip ir filtro indekso laikas, nes atlieka tą patį veiksmą ir naudoja vieną ir tą patį indeksą ir paieškai, ir filtrui.

Eksperimentų metu buvo tikrinamas ir paieškos veikimas iš indekso arba *OpenCart* atveju tiesiog paprasta paieška sistemoje su tomis pačiomis imtimis, rezultatai pateikti žemiau.

5.13 lentelė. paieškos veikimo eksperimentas nr. 1

Duomenų imtis	**Standartinė paieška, ms**	***ElasticFilter*, ms**	***Magento*, ms**	***OpenCart*, ms**
100	1414	2598	5614	623
1000	2764	3470	4729	651
2000	2651	2584	5806	590
5000	3399	2706	6848	904
10000	4571	3151	6932	1150
20000	7077	5587	8963	1736
50000	9532	7521	9272	4262

5.14 lentelė. paieškos veikimo eksperimentas nr. 2

Duomenų imtis	**Standartinė paieška, ms**	***ElasticFilter*, ms**	***Magento*, ms**	***OpenCart*, ms**
100	1494	3286	4165	627
1000	1529	2905	4911	682
2000	1558	2955	6179	713
5000	3332	2761	7834	824
10000	4151	3056	7292	1136
20000	6967	5568	9103	1804
50000	9861	7754	8913	3805

5.15 lentelė. paieškos veikimo eksperimentas nr. 3

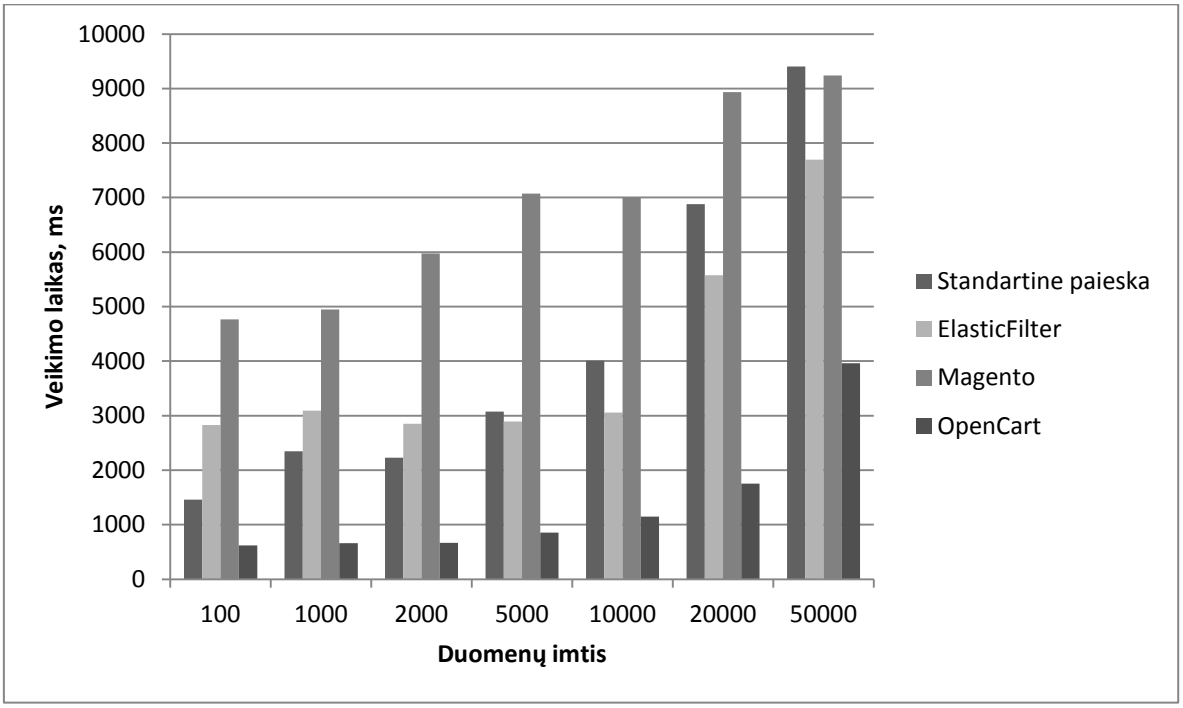
Duomenų imtis	**Standartinė paieška, ms**	***ElasticFilter*, ms**	***Magento*, ms**	***OpenCart*, ms**
100	1472	2596	4523	620
1000	2742	2903	5198	654
2000	2488	3021	5945	705
5000	2488	3209	6534	841
10000	3308	2974	6792	1168
20000	6588	5581	8743	1727
50000	8811	7812	9543	3825

Vidurkis eksperimentų pateikiamas žemiau esančioje lentelėje.

5.16 lentelė. vidurkis paieškos veikimo eksperimentų

Duomenų imtis	**Standartinė paieška, ms**	***ElasticFilter*, ms**	***Magento*, ms**	***OpenCart*, ms**
100	1460	2827	4767	623
1000	2345	3093	4946	662
2000	2232	2853	5977	669
5000	3073	2892	7072	856
10000	4010	3060	7005	1151
20000	6877	5579	8936	1756
50000	9401	7696	9243	3964

Iš paieškos panaudojimo rezultatų matyti, kad kaip ir filtro atveju, indekso dydis neturi reikšmingos įtakos veikimui ir pats veikimas kinta tiesiogine priklausomybe vietoje eksponentinio kai duomenys yra generuojami.



5.4 pav. vidurkis paieškos veikimo eksperimentų

5.3. Eksperimentinių tyrimų apibendrinimas ir bendras palyginimas

Buvo galima tikėtis, kad sistema be visiškai jokio indeksavimo veiks lėčiausiai, tačiau *OpenCart* rezultatai pasirodė esantys vieni geresnių. Reiktų atsižvelgti ir į tai, kad *OpenCart* atveju visiškai nebuvo kainos filtro, o bendra sistemos struktūra yra daug paprastesnė nei kitų dviejų lyginamų: *OpenCart* turi ir

paprastesnę DB struktūrą, ir mažiau saugo esybės duomenų tiek apie prekes, tiek apie kitas esybes. Vis dėlto esant dideliems duomenims el. komercijos platforma tapo mažiau veiksminga už likusias.

PrestaShop standartinis filtras veikia ypač greitai ir iš esmės išsiskiria nuo kitų būdamas greitesnis net iki 1408 kartų už *AjaxFilter* komercinį modulį ar 1050 kartų už sukurtą modulį. Šio greitumo principas yra tas, kad filtras visiškai nenaudoja jokių papildomų funkcijų, o atlieka tik greitas įrašymo į DB operacijas nenaudodamas jokių kitų tarpinių kintamųjų, nesaugodamas informacijos atmintyje, neapdorodamas jos. Vienos tokių užklausų ir vieno iš indeksų generavimo užklausa pateikiama žemiau.

```
/*
 * Generate data product attribute
 */
public function indexAttribute($id_product = null)
{
    if (is_null($id_product))
        Db::getInstance()->execute('TRUNCATE '._DB_PREFIX_.'layered_product_attribute');
    else
        Db::getInstance()->execute('
            DELETE FROM '._DB_PREFIX_.'layered_product_attribute
            WHERE id_product = '.(int)$id_product
        );

    Db::getInstance()->execute('
        INSERT INTO `'._DB_PREFIX_.'layered_product_attribute` (`id_attribute`, `id_product`, `id_attribute_group`, `id_shop`)
        SELECT pac.id_attribute, pa.id_product, ag.id_attribute_group, product_attribute_shop.`id_shop`
        FROM '._DB_PREFIX_.'product_attribute pa'.
        Shop::addSqlAssociation('product_attribute', 'pa').'
        INNER JOIN '._DB_PREFIX_.'product_attribute_combination pac ON pac.id_product_attribute = pa.id_product_attribute
        INNER JOIN '._DB_PREFIX_.'attribute a ON (a.id_attribute = pac.id_attribute)
        INNER JOIN '._DB_PREFIX_.'attribute_group ag ON ag.id_attribute_group = a.id_attribute_group
        '.(is_null($id_product) ? '' : 'AND pa.id_product = '.(int)$id_product).'
        GROUP BY a.id_attribute, pa.id_product , product_attribute_shop.`id_shop`'
    );

    return 1;
}
```

5.5 pav. *blocklayered* atributų indekso SQL užklausa – greitumo paaiškinimas

AjaxFilter indeksuodamas sukuria didžiulius duomenų kiekius ir įrašų kiekį savo indekso lentelėse, atliekami tarpiniai skaičiavimai dėl ko jo greitis indeksuojant yra eksponentiškai priklausomas nuo duomenų imties.

Sukurtas magistrinio darbo metu modulis yra iki 30 proc. efektyvesnis už komercinį *AjaxFilter* modulį, tačiau net iki 1.8 karto greitesnis už standartinę paiešką. Būtina pastebėti, kad sukurtas modulis vieno ir to paties indeksavimo metu suindeksuoja informaciją ir filtrui, ir paieškai dėl ko jis susumavus bendrus paieškos ir standartinio filtro ar paieškos ir komercinio filtro rezultatus yra 0,63-3,01 kartų greitesnis ir efektyvesnis.

5.17 lentelė. standartinės paieškos ir standartinio filtro palyginimas su *ElasticFilter*

Duomenų imtis	**Standartinė paieška, ms**	***Blocklayered*, ms**	**Viso indekso laikas, ms**	***ElasticFilter, ms***	**Skirtumas, kartais**
100	10511	782	11293	6153	1,84
1000	150617	987	151604	49942	3,04
2000	225563	942	226505	93629	2,42
5000	636123	1122	637245	372816	1,71
10000	1095636	1684	1097321	1731084	0,63
20000	5096326	3935	5100262	3415435	1,49
50000	13045335	6911	13052246	7259693	1,80

Palyginus standartinę paiešką ir komercinį *AjaxFilter* matome, kad sukurtas magistro darbo metu filtras yra 2,98-5,88 kartų efektyvesnis ir greitesnis priklausomai nuo duomenų imties.

5.18 lentelė. standartinės paieškos ir *AjaxFilter* palyginimas su *ElasticFlter*

Duomenų imtis	**Standartinė paieška, ms**	***AjaxFilter*, ms**	**Viso indekso laikas, ms**	***ElasticFilter, ms***	**Skirtumas, kartais**
100	10511	7836	18347	6153	2,98
1000	150617	75273,33333	225891	49942	4,52
2000	225563	134068	359631	93629	3,84
5000	636123	1556335	2192458	372816	5,88
10000	1095636	2612160,667	3707797	1731084	2,14
20000	5096326	5281588,333	10377915	3415435	3,04
50000	13045335	9733274,333	22778610	7259693	3,14

Modulio sukūrimas ir jo eksperimento rezultatai įrodo, kad *ElasticSearch* yra veiksminga priemonė daugiau paprastos paieškos be logikos metu, o esant papildomai logikai, pavyzdžiui, kainų skaičiavimui vis vien sugaištama sąlyginai daug laiko atliekant indeksavimo veiksmus. Pagrindinis abiejų modulių *AjaxFilter* ir sukurtojo probleminis aspektas – tarpinių duomenų naudojimas, apdorojimas ir papildomi kreipimaisi į DB jiems gauti.

Magento el. komercijos sistema pasiekė geresnius rezultatus dėka savo išskaidytos ir *EAV* principu paremtos DB. Išskaidyti indeksai ir didžiausias indeksų kiekis tarp lygintų sistemų leido dirbti vienu metu su mažiau duomenų efektyviau, todėl *Magento* rezultatai yra vidutiniški lyginant su kitomis dviem el. komercijos platformomis, tačiau rodė prastesnius rezultatus filtro ir paieškos naudojime. Svarbu atkreipti dėmesį, kad pakartojant tą pačią užklausą antrą, ar trečią kartą, ji suveikia praktiška akimirksniu dėl sistemos keršavimo ir laikinų DB lentelių, tačiau bandymų metu buvo siekiama išgauti visada realų pirmos užklausos laiką.

PrestaShop paieškos indeksas yra pats neefektyviausias iš gautų rezultatų, todėl jį tikslinga pakeisti kitais realizacijos būdais, pavyzdžiui, *ElasticSearch* pilno tipo paieška be kainų skaičiavimo logikos.

6. IŠVADOS

Šiame darbe buvo:

1. Išanalizuotos populiariausios atvirojo kodo el. komercijos sistemos *PrestaShop, OpenCart, Magento* ir jų architektūra bei sprendimo būdai susiduriant su dažniausiomis el. komercijos problemomis: filtro ir paieškos naudojimu su dideliais duomenų kiekiais.
2. Suprojektuotas ir realizuotas specialus modulis pasirinktai el. komercijos platformai *PrestaShop* darbui su filtru ir paieška esant dideliems duomenų kiekiams išnaudojant *ElasticSearch* indeksų galimybes.
3. Atlikti greitaveikos eksperimentai tarp *PrestaShop* sistemos esančių rinkoje modulių, sukurto modulio darbo metu ir dviejų kitų atvirojo kodo el. parduotuvių platformų.

Atlikus eksperimentus galima teigti, kad:

1. Sistemos veikiančios be indeksavimo yra ne tik, kad mažiau tikslios, bet ir tinkamos darbui tik su mažas duomenims, esant dideliam duomenų kiekiui jų greitis susilygina su indeksus naudojančiomis sistemomis.
2. Indekso paieškai ir filtrui priklausomybė nuo duomenų kiekio generuojant juos yra eksponentiškai didėjanti, o naudojant (atvaizduojant iš indekso) priklausomybė – tiesinė.
3. Indekso paruošimo didžiąją dalį laiko sudaro papildomų duomenų paėmimas ir apdorojimas, kainos skaičiavimas ir kt. papildomai atliekami veiksmai. Kai naudojamos tik DB užklausos indekso greičių skirtumai yra iki 1550 kartų greitesni lyginant su sistemomis, kurios atlieka papildomus tarpinius veiksmus.
4. Dideliems duomenų kiekiams ir ypač teksto paieškai ar indekso kūrimui be papildomos logikos rekomenduotina naudoti *ElasticSearch* tipo indeksus, kurie yra efektyvesni iki 5 kartų lyginant su paprastais DB bazės indeksais naudojamais paieškai.
5. Jeigu neįmanoma naudoti *ElasticSearch* tipo indeksų, patartina skaidytų indeksus į atskirus ir į mažesnius duomenų kiekius pagal veikimo logiką.
6. Sukurtas magistrinio darbo metu modulis yra vidutiniškai 1.8 kartų efektyvesnis už standartinę paiešką ir standartinį filtro modulį kartu susumavus, tačiau filtro indekso kūrimo ir panaudojimo metu turi panašius rezultatus, kaip ir kiti moduliai, pavyzdžiui, *AjaxFilter*.
7. Įvertinus aukščiau išvardintus punktus galima teigti, kad didelių duomenų panaudojimas el. komercijos platformose ir ne tik yra vertas dėmesio objektas ir tolimesnių tyrimų bei taikymų žinant ir didėjančius duomenų kiekius (angl. big data) kompiuterijos pasaulyje.

7. LITERATŪRA

[1] WikiPedia, „Wikipedia,“ [Tinkle]. Available: http://lt.wikipedia.org/wiki/Elektronin%C4%97_komercija. [Kreiptasi 10 12 2014].

[2] K. Morrison, „SocialTimes,“ [Tinkle]. Available: http://socialtimes.com/data-growth-e-commerce-infographic_b198687. [Kreiptasi 10 12 2014].

[3] Emarketer, „Emarketer,“ [Tinkle]. Available: http://www.emarketer.com/Article/Global-B2C-Ecommerce-Sales-Hit-15-Trillion-This-Year-Driven-by-Growth-Emerging-Markets/1010575. [Kreiptasi 10 12 2014].

[4] M. Macdonald, „Shopify,“ [Tinkle]. Available: http://www.shopify.com/blog/8484093-why-online-retailers-are-losing-67-45-of-sales-and-what-to-do-about-it. [Kreiptasi 12 10 2014].

[5] Lastdropofink, „Lastdropofink,“ [Tinkle]. Available: http://lastdropofink.co.uk/tools/magento/magento-powers-26-of-the-top-ecommerce-websites/. [Kreiptasi 21 12 2014].

[6] ElasticSearch, „ElasticSearch,“ [Tinkle]. Available: http://www.elasticsearch.org/overview/. [Kreiptasi 12 12 2014].

[7] V. Sprainyte, „Making Sense of Your Logs with Elasticsearch,“ [Tinkle]. Available: https://www.devbridge.com/articles/making-sense-of-your-logs-with-elasticsearch/. [Kreiptasi 13 12 2014].

[8] Predictiveanalyticstoday, „Top 11 Open Source Big data Enterprise Search Software,“ [Tinkle]. Available: http://www.predictiveanalyticstoday.com/top-open-source-big-data-enterprise-search-software/. [Kreiptasi 14 12 2014].

[9] M. Brandon, „Qbox Releases QES Elasticsearch Connector for Magento,“ [Tinkle]. Available: http://blog.qbox.io/qbox-releases-qes-elasticsearch-connector-for-magento. [Kreiptasi 16 12 2014].

[10] Databasejournal, „Databasejournal,“ [Tinkle]. Available: http://www.databasejournal.com/features/mysql/the-pros-and-cons-of-mysql-table-locking.html. [Kreiptasi 12 12 2014].

[11] PrestaShop, „GitHubg repository,“ [Tinkle]. Available: https://github.com/PrestaShop/blocklayered. [Kreiptasi 12 12 2014].

[12] PrestoChangeo, „PrestoChangeo Ajax Filter page,“ [Tinkle]. Available: http://www.presto-changeo.com/en/filter-modules/74-ajax-filter.html. [Kreiptasi 12 12 2014].

[13] Oracle INc, „How MySQL Uses Indexes [Tinkle]. Available:

http://dev.mysql.com/doc/refman/5.7/en/mysql-indexes.html [Kreiptasi 15 05 2016].

[14] Xavier Borderie, PrestaShop's developer tools [Tinkle]. Available:

http://doc.prestashop.com/display/PS16/PrestaShop's+developer+tools [Kreiptasi 19 05 2016].

8. PRIEDAI

8.1. Priedas Nr. 1 – PrestaShop prekių generatoriaus pavyzdys

```
    <?php

    require(dirname(__FILE__).'/config/config.inc.php');

    $iterations = 0;

    while($iterations < 10000)
    {
        $random = rand(1, 7);
        if (Validate::isLoadedObject($product = new Product((int)$random))) {
            $id_product_old = $product->id;
            if (empty($product->price) && Shop::getContext() == Shop::CONTEXT_GROUP)
{
                $shops                                                          =
ShopGroup::getShopsFromGroup(Shop::getContextShopGroupID());
                foreach ($shops as $shop) {
                    if ($product->isAssociatedToShop($shop['id_shop'])) {
                        $product_price = new Product($id_product_old, false, null,
$shop['id_shop']);
                        $product->price = $product_price->price;
                    }
                }
            }
            unset($product->id);
            unset($product->id_product);
            $product->indexed = 0;
            $product->active = 1;
            if ($product->add()
                &&  Category::duplicateProductCategories($id_product_old,  $product-
>id)
                && Product::duplicateSuppliers($id_product_old, $product->id)
                &&                        ($combination_images                      =
Product::duplicateAttributes($id_product_old, $product->id)) !== false
                && GroupReduction::duplicateReduction($id_product_old, $product->id)
                && Product::duplicateAccessories($id_product_old, $product->id)
                && Product::duplicateFeatures($id_product_old, $product->id)
                && Product::duplicateSpecificPrices($id_product_old, $product->id)
                && Pack::duplicate($id_product_old, $product->id)
                &&  Product::duplicateCustomizationFields($id_product_old,  $product-
>id)
                && Product::duplicateTags($id_product_old, $product->id)
                && Product::duplicateDownload($id_product_old, $product->id)) {
                if ($product->hasAttributes()) {
                    Product::updateDefaultAttribute($product->id);
                }

    /*
                if                    (!Tools::getValue('noimage')                  &&
!Image::duplicateProductImages($id_product_old, $product->id, $combination_images)) {
                    $this->errors[] = Tools::displayError('An error occurred while
copying images.');
                } else {
                    Hook::exec('actionProductAdd',       array('id_product'       =>
(int)$product->id, 'product' => $product));
                    if (in_array($product->visibility, array('both', 'search')) &&
Configuration::get('PS_SEARCH_INDEXATION')) {
```

```
                        Search::indexation(false, $product->id);
                    }
                }
    */

                                Hook::exec('actionProductAdd', array('id_product' =>
(int)$product->id, 'product' => $product));

            } else {
                echo "ERROR";
            }
        }
        $iterations++;
    }
```