KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Paulius Repšys

Kešavimo priemonių tyrimas ir taikymas įmonių katalogo informacinės sistemos greitaveikai spartinti

Magistro darbas

Darbo vadovas
lekt. dr. Gytenis Mikulėnas

Kaunas, 2013

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Paulius Repšys

Kešavimo priemonių tyrimas ir taikymas įmonių katalogo informacinės sistemos greitaveikai spartinti

Magistro darbas

Recenzentas
Lekt. dr. Rokas Zakarevičius
2013-05-1

Vadovas
Lekt. dr. Gytenis Mikulėnas
2013-05-10

Atliko
IFN-1/1 gr. stud. Paulius Repšys
2013-05-10

Kaunas, 2013

Kešavimo priemonių tyrimas ir taikymas įmonių katalogo informacinės sistemos greitaveikai spartinti

Santrauka

Šiame darbe aprašyta atlikta esamų panašių informacinių sistemų lyginamoji analizė, atlikta įmonių katalogo interneto informacinės sistemos reikalavimų analizė bei specifikavimas, projektavimas ir realizavimas.

Darbe aprašytas atliktas tyrimas. Aprašyti išanalizuoti bei palyginti esami kešavimo sprendimai, suformuotos jų naudojimo prielaidos. Remiantis šiomis prielaidomis, atliktas eksperimentas testuojant sukurtos informacinės sistemos greitaveiką su kiekvienu į sistemą įdiegtu kešavimo sprendimu atskirai. Apibendrinti bei pateikti eksperimento rezultatai, pateiktos kešavimo priemonių naudojimo rekomendacijos. Šių rekomendacijų pagrindu buvo pasirinkti ir įmonių katalogo informacinėje sistemoje realizuoti didžiausią greitaveiką suteikiantys kešavimo sprendimai.

Research and application of caching solutions for increasing performance of companies catalog information system

Summary

In this work it is described a comparative analysis of similar information systems, made requirements analysis and specification, projection and realization of companies catalog information system.

A made research is described in the work. Present caching solutions were analysed and compared, assumptions of their usage formed. According to these assumptions, an experiment was made by testing performance of created information system with every caching mean, installed into the system, separately. Results of the experiment and recomendations how to use them where summarized and presented. On the basis of these results, solutions that give biggest performance increase were realized in the companies catalog information system.

Turinys

1. ĮVADAS

Vis didėjantis informacijos kiekis internete vartotojams kelia problemą: kaip atskirti, kuri informacija vertinga, o kuri pasenusi ir bevertė? Informacijos sisteminimas prieš pateikiant ją vartotojams įgaus vis didesnę reikšmę, o šie bus linkę naudotis tomis paslaugomis, kurios leis greičiau surasti vertingą, apdorotą informaciją.

Viena iš sričių, ko žmonės ieško internete, yra įmonės ir jų teikiamos paslaugos/produktai. Šiame darbe bus kuriama įmonių katalogo interneto IS, kurią būtų galima lanksčiai pritaikyti įvairioms paslaugų sritims, ir ne tik įmonėms.

Tokio tipo sistemos dažnai turi didelės apimties duomenų bazę ir susiduria su problema - kaip užtikrinti aukštą sistemos veikimo greitį. Nesiėmus veiksmų, mažinančių aparatinės įrangos naudojimo resursus, ši gali būti labai apkraunama bei sukeliami nepatogumai vartotojui. Todėl sistemos greitaveikos spartinimas turi būti neišvengiamas etapas kuriant interneto IS.

Vienas iš efektyviausių būdų padidinti sistemos greitaveiką - kešavimas. Tai metodika, kuomet sukompiliuoti algoritmų duomenys yra išsaugojami laikinojoje atmintyje ir vėliau pateikiami vartotojui nebeapdorojant tų pačių algoritmų dar kartą. Egzistuoja įvairūs kešavimo tipai bei priemonės, tačiau visų jų tyrimas yra per platus. Todėl šiame darbe buvo pasirinktos bei analizuotos kešavimo priemonės, susijusios su aplikacijos kodu.

Darbo tikslas - atlikti kešavimo priemonių tyrimą siekiant suformuoti jų taikymo rekomendacijas interneto sistemų greitaveikos spartinimo procese bei remiantis šiomis rekomendacijomis realizuoti didžiausią įtaką greitaveikai turinčias kešavimo priemones sukurtoje įmonių katalogo informacinėje sistemoje.

Darbo uždaviniai:

1. atlikti rinkoje esančių analogiškų įmonių katalogo informacinių sistemų apžvalgą;
2. išsiaiškinti užsakovo įmonės poreikius bei specifikuoti reikalavimus;
3. suprojektuoti sistemos architektūrą, duomenų bazę ir komponentus;
4. realizuoti įmonių katalogo informacinę sistemą;
5. atlikti interneto informacinių sistemų kešavimo priemonių analizę;
6. atlikti sistemos greitaveikos eksperimentą su pasirinktomis kešavimo priemonėmis;
7. suformuoti sistemos greitaveikos spartinimo rekomendacijas;

8. realizuoti didžiausią teigiamą įtaką greitaveikai turinčias kešavimo priemones įmonių katalogo informacinėje sistemoje.

Tyrimo sritis - kešavimo priemonės ir jų taikymas interneto informacinėse sistemose.

Probleminė sritis - lėtas interneto informacinių sistemų veikimas esant dideliam sistemos apkrovimui (lygiagrečių vartotojų skaičiui).

2. ĮMONIŲ KATALOGO IS PROBLEMATIKA

2.1 Sistemos paskirtis

Atsiradus internetui, per keletą pastarųjų dešimtmečių paprastam žmogui pasiekiamas informacijos kiekis pasikeitė nuo nepakankamo iki ypač gausaus. Tai visuomenei teikia didžiulias galimybes, tačiau kartu kelia ir nemažų problemų. Kaip atskirti, kuri informacija vertinga, o kuri pasenusi ir bevertė?

Didėjant informacijos kiekiui internete, tampa svarbi informacijos vadyba. Galima daryti prielaidą, kad informacijos sisteminimas internete įgaus vis didesnę reikšmę ir interneto vartotojai rinksis tuos informacijos teikėjus, kurie ją pateiks susistemintą ir apdorotą.

Viena iš sričių, ko žmonės ieško internete, yra įmonės ir jų teikiamos paslaugos/produktai. Dėl nesistemingo informacijos pateikimo internete kartais yra sunku rasti būtent tokią įmonę, kokia atitiktų lūkesčius. Pavyzdžiui, Google paieškos sistemos rezultatų pirmajame puslapyje atsiduriančios įmonės galbūt turi gerus SEO specialistus, bet ne paslaugas, kurių ieško žmogus.

Lietuvoje yra ne vienas portalas, vienijantis tam tikros srities paslaugas ar produktus teikiančias įmones. Vartotojas, ieškantis specifinės paslaugos, gali greitai peržiūrėti įmones, susidaryti įspūdį ir, pasirinkęs keletą jam labiausiai patikusių, kreiptis į jas dėl išsamesnės informacijos.

Bene didžiausia problema kuriant tokias sistemas yra turimos informacijos atnaujinimas, pasenusios informacijos kontrolė. Esant dideliam įmonių skaičiui, sistemos administratoriams nerealu patiems sukontroliuoti pateikiamos informacijos korektiškumą. Taip pat svarbus klausimas kaip vartotojui bus pateikta galimai didelė įmonių duomenų bazė, kad tarp šimtų ar tūkstančiųįmonių jis galėtų rasti labiausiai jo poreikius atitinkančią. Kitas svarbus dalykas - pagal ką įmonės bus išrikiuotos. Praktika rodo, kad vartotojai neturi kantrybės labai ilgai ieškoti ar naudotis filtravimo galimybėmis. Dažnai jie apsiriboja peržiūrėdami pirmus 10-20 įrašų.

Vienas iš svarbiausių užsakovo reikalavimų kuriamai IS yra tas, kad sukurta IS galėtų būti lengvai pritaikoma įvairias paslaugas ar produktus teikiančioms įmonėms, ir ne tik joms: tai galėtų būti katalogas, vienijantis menininkus, fotografus, turizmo objektus ir daugelį kitų gyvenimo sričių. Be to, siekiant pranašumo prieš konkurentus, būtina sukurti modernaus dizaino vartotojo sąsają,

išlaikant paprastumą, kad IS galėtų naudotis ir mažai patyrę interneto vartotojai. Taigi IS turi būti orientuota į pagrindinių funkcijų teikiamą naudą ir jų pritaikomumą įvairioms sritims.

Todėl išanalizavus užsakovo poreikius, suformuluotos šios svarbiausios kuriamos įmonių katalogo IS savybės, kuriomis remiantis bus atlikta detali sistemos funkcinių ir nefunkcinių reikalavimų specifikacija:

- lengvas sistemos palaikymas, dinamiškumas - vartotojai patys gali registruotis sistemoje, užtikrinamas jų duomenų validumas ir išsamumas, pasenusios informacijos kontrolė be sistemos administratorių įsikišimo;
- aiškūs, lengvai keičiami kriterijai pagal ką įmonės yra išrikiuojamos kataloge;
- patogus ir lankstus įmonių filtravimas, paieška;
- IS turi būti lengvai pritaikoma įvairių veiklų sritims;
- IS privalo gebėti nesunkiai naudotis net ir mažai patyrę IT naudotojai.

2.2 Esamų sprendimų lyginamoji analizė

Atlikus analogiškos programinės įrangos paiešką internete paaiškėjo, kad didžioji dalis katalogų programinės įrangos yra skirti vienos konkrečios įmonės produktams saugoti elektroniniu pavidalu. Taip pat daugelyje jų yra integruotas el. parduotuvės modulis. Programinės įrangos, kuri būtų skirta vienos konkrečios paslaugų srities įmonėms registruoti, rasti nepavyko. Plačiau atliktos analizės rezultatus detalizuoja 1 lentelė, kurioje pateikti trys skirtingi elektroninio katalogo programinės įrangos variantai.

Lentelė 1. Esamų elektroninių katalogų programinės įrangos sprendimų palyginimas

Pavadinimas	**E-Katalogas**	**Interneto katalogas**	**Joomla DJ Catalog 2**
Kūrėjas	„Configure One“ (Amerika)	„9TH SPHERE“ (Kanada)	„DJ Extensions“ (Amerika)
Nuoroda internete	http://www.configureone.com	http://www.9thsphere.com	http://extensions.joomla.org/extensions/directory-a-documentation/directory/18857
Paskirtis	Įmonės produktų elektroniniam katalogui sudaryti verslas-vartotojui arba verslas-verslui pagrindu	Įmonės produktų elektroniniam katalogui sudaryti	Joomla TVS plėtinys produktų publikavimui internete
Ar atviro kodo	Ne, komercinis	Ne, komercinis	Ne, komercinis
Programiniai sprendimai	• .NET programavimo kalba; • Gausus Ajax technologijos naudojimas.	• PHP programavimo kalba; • Naudojamas9th sphere's CORE karkasas.	• PHP programimo kalba; • Veikia tik su Joomla TVS.
Pagrindinės funkcijos ir privalumai	• Integruota el. parduotuvė; • vartotojų registracija; • Duomenų importavimas Excel, Xml ir kt. formatais; • greita ir universali įrašų paieška; • galimybė katalogą optimizuoti SEO.	• Labai lengvas įrašų pridėjimas, redagavimas, pašalinimas; • pilnai keičiamas dizainas.	• Lengvas produktų pridėjimas, kategorizavimas; • galimybė pritaikyti skirtingas temas.
Trūkumai	• Didelė kaina.	• Prisirišimas prie mažai žinomo 9th sphere's CORE karkaso.	• Prisirišimas prie Joomla TVS.

„Configure One“ įmonės sukurtas E-Katalogas skirtas itin didelio dydžio sistemoms kurti, jis be papildomų investicijų gali talpinti iki 5 mln. katalogo įrašų. Tai didelio masto, kompleksinis sprendimas, tinkantis ypač didelį produktų asortimentą turinčioms įmonėms.

„9TH SPHERE“ įmonės interneto katalogas paprastesnis, naudojantis šios įmonės sukurtą karkasą 9th sphere's CORE. Tai paprastesnis sprendimas, galintis patenkinti daugelio vidutinio dydžio įmonių poreikius, turintis visas pagrindines lengvam įrašų redagavimui reikalingas funkcijas. Tačiau vėlgi, jis yra daugiau skirtas produktams publikuoti.

Labiau šiame darbe kuriamos sistemos poreikius atitinka Joomla DJ Catalog 2 plėtinys, skirtas Joomla turinio valdymo sistemai. Jame galima publikuoti įvairių tipų esybes, ne tik produktus. Tačiau pagrindinė priežastis, kodėl nesirenkama atviro kodo TVS ir jai sukurti moduliai yra ta, kad prisirišus prie atviro kodo TVS, vėliau būna labai sunku modifikuoti ją pagal savo poreikius. Lietuvoje jau esama keletas interneto katalogų, sukurtų remiantis šiuo metodu. Nė vienas iš jų netenkina siekiamo kokybės lygio.

Apsvarsčius jau esamus katalogų sprendimus nuspręsta kurti savo įmonių katalogo sistemą dėl šių priežasčių:

- nerasta sukurtų sistemų, kurios tiksliai atitiktų kuriamos sistemos pobūdį;
- reikalinga kompatiška, neperkrauta nereikalingomis funkcijomis, moderniausias interneto technologijas naudojanti IS;
- patiems sukūrus IS, vėliau ją bus daug lengviau tobulinti pagal savo poreikius.

2.3 Platformos pasirinkimas

Kuriant bet kokią IS, reikalaujančią programavimo, susiduriama su problema, kokį jau egzistuojantį struktūrinį sprendimą pasirinkti, kad nereikėtų išradinėti daugelio dalykų, kurie jau sukurti. Galima išskirti tris sistemos kūrimo variantai: naudoti turinio valdymo sistemą (TVS), naudoti karkasą arba tiesiog viską sukurti nuo nulio, pritaikant konkrečiai pagal savo poreikius. 2 lentelė charakterizuoja kiekvieną iš šių variantų.

Lentelė 2. Tvs, karkasų ir savarankiškos platformos palyginimo matrica

Platforma	**Tvs**	**Karkasas**	**Kūrimas nuo nulio**
Kūrimo greitis	didelis	vidutinis	mažas

Panaudotinų plėtinių skaičius, sukurtų bendruomenės narių	didelis	vidutinis	mažas
Kūrimo kaštai	maži	vidutiniai	dideli
Sistemos palaikymo kaštai	nedideli	vidutiniai	dideli
Galimybės optimizuoti sistemos greitaveiką	mažos - vidutinės	didelės	didelės
Sistemos išplėtimo galimybės	mažos	didelės	didelės
Bendruomenės palaikymas	labai geras	geras	-
Sistemos dydis	mažas	vidutinis - didelis	didelis

TVS leidžia labai greitai ir mažais kaštais sukurti mažus ar net vidutinio dydžio interneto projektus. Tačiau vėliau, norint įdiegti specifinio funkcionalumo, galima susidurti su problemomis, bandant apeiti TVS esančius standartus. Vis tik tokios sistemos yra kuriamos siekiant universalumo, ir dažnai tai labai apriboja jų galimybes. Be to, daugelis TVS yra sukurti nesilaikant MVC principo.

Pradėti kurti sistemą nuo nulio yra beveik tas pats kas pačiam sukurti TVS. Kiekviename projekte neišvengiamai reikės bazinių CRUD funkcijų (angl. create, read, update, delete), vartotojų autorizacijos, validavimo ir t.t. Todėl šis būdas pasirenkamas tik tuomet, jei norima susikurti savo TVS ir pritaikyti ją grynai pagal savo poreikius, atsisakant visų nereikalingų funkcijų ir taip minimizuojant kodą. Šis būdas reikalaujalabai didelių laiko sąnaudų, o sistemos palaikymas vėliau taip pat gali būti komplikuotas, jei neatsiras programuotojų, žinančių visas sistemos subtilybes.

Šiuo metu optimaliausias sprendimas vidutinio dydžio ir didelėms sistemoms, naudojamas visame pasaulyje, yra karkasai. Savyje jie jau turi klases, leidžiančias atlikti daugelį pasikartojančių kasdienių operacijų, užtikrinti pirminį sistemos funkcionalumą. Programuotojui lieka pagrindinė užduotis - suprogramuoti pačią sistemos logiką. Tai daryti reikia laikantis karkaso standartų. Juos perpratus, galima pasiekti aukštos kokybės rezultatų per pakankamai trumpą laiką. Todėl įmonių katalogo IS bus kuriama naudojant karkasą.

Pagal užsakovo reikalavimą, sistema bus kuriama naudojant PHP programavimo kalbą. Nors tokios kalbos kaip .NET ar JAVA laikomos pranašesnėmis už PHP, pastaroji irgi turi savų privalumų. Pagal Suzumura T., Trent S. ir kt. atliktą PHP, JAVA ir C kalbų greičio palyginimo tyrimą [16], PHP veikimo greičio ir programinės įrangos kūrimo kaštų santykis dažnai lenkia kitas kalbas.

Renkantis kokį PHP karkasą naudoti, iš kelių galimų variantų pasirinktas CakePHP atviro kodo karkasas. Tai vienas iš plačiausiai naudojamų PHP karkasų, turintis tokias funkcijas kaip kodo generavimas per komandinę eilutę, objektinis duomenų bazės modeliavimas, kešavimas, validavimas, autentifikavimas, daugiakalbiškumas ir kitas.

2.4 Apibendrinimas

Kuriama įmonių katalogo IS turi būti lanksčiai pritaikoma įvairių sričių įmonių ar kitokių ūkio subjektų publikavimui ir paieškai internete, orientuota į lengvą palaikymą administratoriams ir naudojimą išorės vartotojams. Analogiškų sistemų, kurias būtų galima gauti nemokamai ar nusipirkti ir kurios tiksliai atititiktų lūkesčius, rasti nepavyko. Sistemai kurti kaip pagrindas pasirinktas CakePHP karkasas.

3. ĮMONIŲ KATALOGO IS PROJEKTO MODELIS

3.1 Reikalavimų specifikacija

„Pagrindinis reikalavimų darbų tikslas - taip nukreipti sistemos kūrimo procesą, kad būtų sukurta tinkama sistema [...] Svarbu, kad dėl reikalavimų sutartų tiek sistemos užsakovai, tiek būsimieji vartotojai, tiek kūrėjai“ (Butkienė, 2013: 97). Todėl šiame skyriuje bus detalizuojami ir konkretizuojami antrame skyriuje išaiškinti reikalavimai sistemai. Šalia aprašymų, įvairūs sistemos procesai bus vaizduojami grafiniu pavidalu naudojant standartinę modeliavimo kalbą UML. Ši kalba yra plačiai vartojama tiek modeliuojant programinę įrangą, tiek veiklos procesus, tiek kitokias sistemas [14].

3.1.1 Užsakovai, pirkėjai ir kiti sistema suinteresuoti asmenys

Sistemos užsakovas ir pirkėjas: UAB „Synergy technologies“.

Lentelė 3. Sistemos vartotojų kategorijos

Vartotojo kategorija	**Paprasti vartotojai**	**Registruoti vartotojai**	**Katalogo redaktoriai** (įmonių, norinčių reklamuoti savo paslaugas, atstovai)	**Administratoriai**
Vartotojo sprendžiami uždaviniai:	Susirasti ir išsirinkti įmonę, kuri teiktų jam reikalingas paslaugas / produktus	Susirasti ir išsirinkti įmonę, kuri teiktų jam reikalingas paslaugas / produktus. Įtakoti katalogo įrašų reitingą paliekant atsiliepimus ir vertinimus pagal savo turimą patirtį	Padidinti savo teikiamų paslaugų žinomumą; pagerinti įmonės įvaizdį vartotojų akyse; padidinti pardavimus	Valdyti sistemą
Patirtis dalykinėje srityje:	Naujokas	Naujokas	Srities specialistas	Srities specialistas
Patirtis informacinėse technologijose:	Naujokas	Naujokas	Naujokas	Patyręs
Apsimokymo poreikis:	Nereikia	Nereikia	Nereikia	Reikia

Prioritetas	2 lygmuo (ši vartotojų grupė labai svarbi. Kuo daugiau šių vartotojų naudosis sistema, tuo didesni šansai sistemos komercinei sėkmei)	2 lygmuo (ši vartotojų grupė labai svarbi. Kuo daugiau šių vartotojų naudosis sistema, tuo didesni šansai sistemos komercinei sėkmei)	1 lygmuo (ši vartotojų grupė svarbiausia, nes nuo jų priklauso informacijos kiekis sistemoje, ir ar ji bus aktuali paprastiems vartotojams)	4 lygmuo (ši vartotojų grupė prižiūri sistemą. Kuo jos darbas nepastebimesnis, tuo jis yra atliktas geriau. Nuo šių vartotojų daug priklauso, ar sistema veiks sklandžiai ir komerciškai sėkmingai)

3.1.2 Apribojimai reikalavimams

Apribojimai sprendimui

Sistema veiks tik internete, vienu konkrečiu adresu. Sistema turi gerai veikti visose populiariausiose interneto naršyklėse, nepriklausomai nuo operacinės sistemos tipo. Sistema turi būti pritaikyta išmaniesiems mobiliesiems įrenginiams.

Diegimo aplinka

Sistema bus kuriama PHP programavimo kalba. Bus naudojama tokia aparatinė įranga:

- PHP 5.3.13 Apache 2.4 serveris;
- Mysql 5.5.24 duomenų bazė;

Bendradarbiaujančios sistemos.

Bendradarbiaujančios sistemos - tai nekomerciniai specializuoti programų paketai, kurie bus naudojami sistemoje. Komercinių paketų naudojama nebus.

Bus naudojami šie atviro kodo programų paketai:

- CakePHP karkasas - tai vienas iš plačiausiai naudojamų PHP karkasų, turintis tokias funkcijas kaip kodo generavimas per komandinę eilutę, objektinis duomenų bazės modeliavimas, kešavimas, validavimas, autentifikavimas, daugiakalbiškumas ir kitas. Nuoroda: http://cakephp.org/
- "Uploader" - failų įkėlimo ir nuotraukų redagavimo paketas, skirtas CakePHP karkasui. Nuoroda: http://milesj.me/code/cakephp/uploader

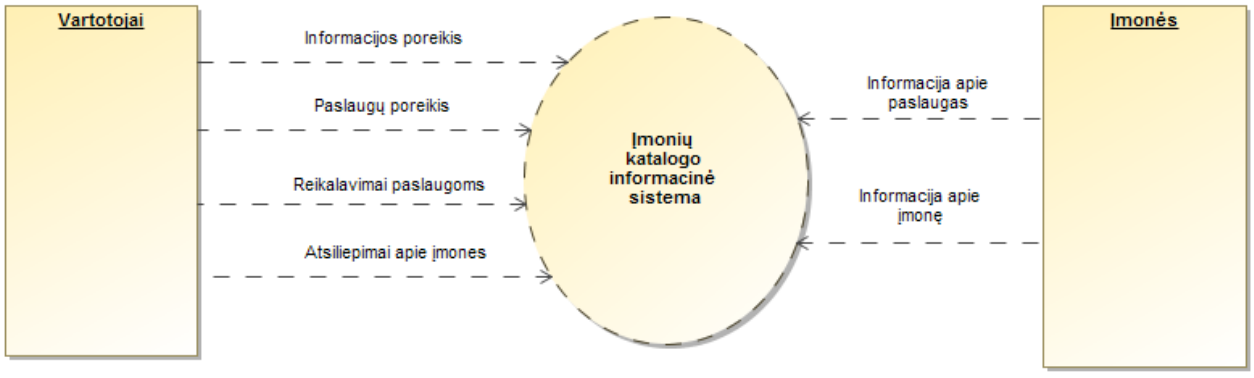
- "Alaxos" - vartotojų rolių ir teisių valdymo vartotojo sąsajos paketas, skirtas CakePHP karkasui.Nuoroda: http://www.alaxos.net/blaxos/pages/view/cakephp_2
- JQuery karkasas - tai Javascript programavimo kalbos biblioteka, gerokai palengvinanti kliento pusės programavimo sintaksę ir ištaisanti Javascript programavimo kalbos neatitikimus tarp įvairių naršyklių. Nuoroda: http://jquery.com
- JQuery UI biblioteka - interaktyvios vartotojo sąsajos elementų rinkinys, sukurtas jQuery karkaso pagrindu. Nuoroda: http://jqueryui.com
- Twitter Bootstrap - vaizdo dalies karkasas, skirtas efektyviam šiuolaikiškų vartotojo sąsajų kūrimui. Nuoroda: http://twitter.github.com/bootstrap/

Svarbūs faktai ir prielaidos

- Kuriama įmonių katalogo IS turi būti lengvai panaudojama įvairių veiklų įmonėms;
- Sistema privalo gebėti naudotis net ir mažai patyrę internetinių technologijų naudotojai.

3.1.3 Veiklos kontekstas

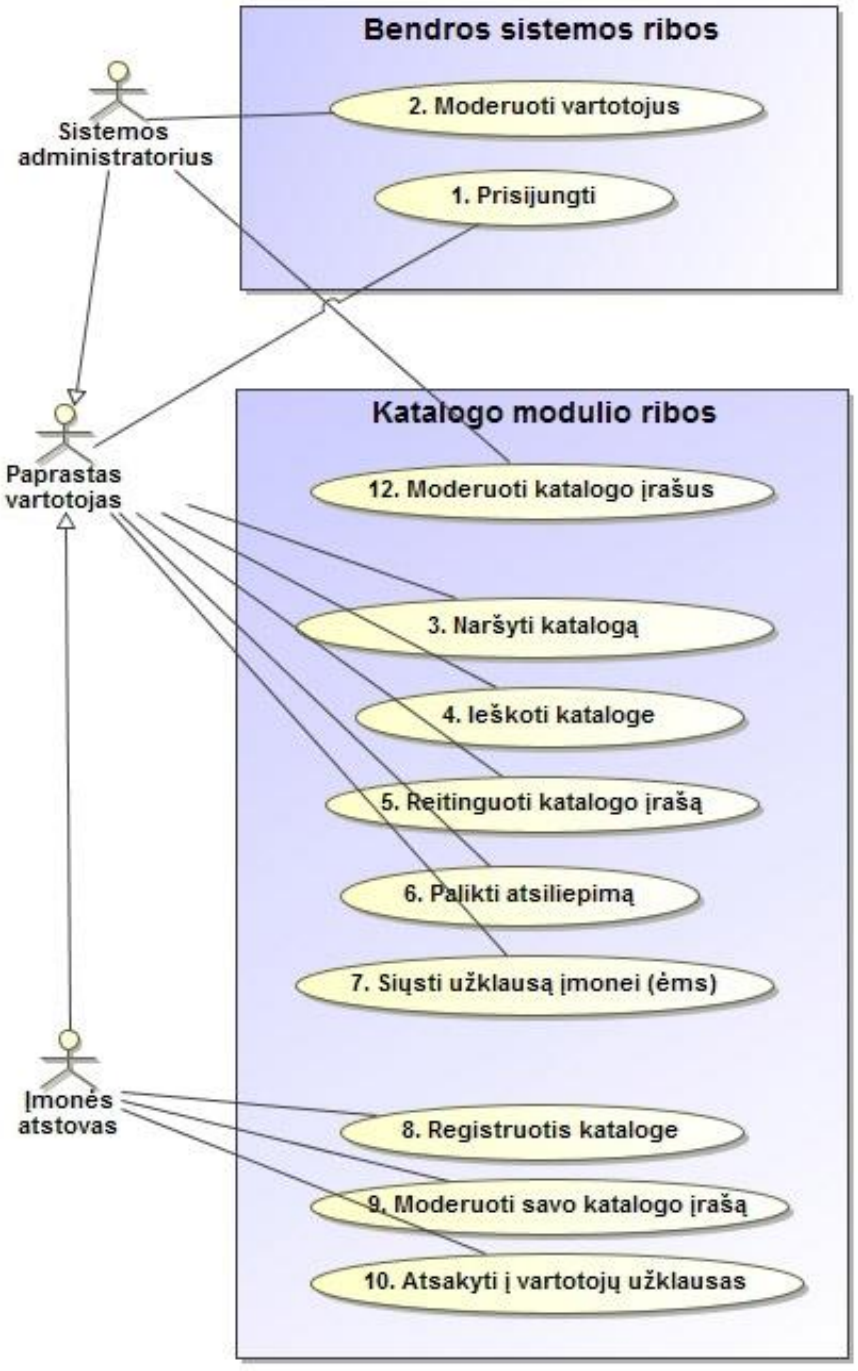
Kuriamos sistemos veiklos kontekstas parodytas 1 paveikslėlyje. Matome, kad yra tarsi dvi viena kitos ieškančios pusės: vartotojai ir įmonės, o sistema leidžia abiejų pusių reikalavimus ir teikiamą informaciją apjungti ir padaryti prieinamą susistemintos informacijos pavidalu.



Pav. 1. Įmonių katalogo IS veiklos konteksto diagrama

3.1.4 Panaudojimo atvejų modelis

Panaudojimo atvejo modelio sudėtinės dalys yra panaudojimo atvejis, vykdytojas ir juos siejanti asociacija (informacijos srautas) [10]. Sistemos panaudojimo atvejus galima išskirti į dvi grupes: bendras sistemos ribas ir katalogo modulio ribas. 2 paveikslėlyje pateikiami svarbiausi sistemos panaudojimo atvejai:



Pav. 2. Įmonių katalogo IS panaudojimo atvejų diagrama

Išsamiai kiekvienas panaudojimo atvejis aprašytas 8.1 priede „Panaudojimo atvejų detalus aprašymas“.

3.1.5 Funkciniai reikalavimai sistemai

Toliau pateikiami funkciniai reikalavimai sistemai, suformuoti pagal apibrėžtus panaudojimo atvejus.

Reikalavimas #:	1	Reikalavimo tipas:	9a	Įvykis/panaudojimo atvejis #:	1
Aprašymas:	Sistema turi leisti vartotojams užsiregistruoti				
Pagrindimas:	Siekiant sustiprinti vartotojų lojalumą sistemai, turi būti sukurta vartotojų sistema. Tik turėdami savo sąskaitas, paprasti vartotojai galės palikti atsiliepimus, įmonių atstovai galės registruoti savo įmonę kataloge.				
Šaltinis:	Visi sistemos vartotojai				
Tikimo kriterijus:	Bet kuris sistemos lankytojas gali užsiregistruoti sistemoje				
Užsakovo tenkinimas:	1			Užsakovo netenkinimas:	5
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2012.10.14				

Reikalavimas #:	2	Reikalavimo tipas:	9a	Įvykis/panaudojimo atvejis #:	1
Aprašymas:	Sistema turi leisti registruotiems vartotojams prisijungti				
Pagrindimas:	Kad autentifikuotųsi, vartotojas turi galėti prisijungti prie sistemos				
Šaltinis:	Visi sistemos vartotojai				
Tikimo kriterijus:	Bet kuris užsiregistravęs sistemos lankytojas gali prisijungti prie sistemos				
Užsakovo tenkinimas:	1			Užsakovo netenkinimas:	5
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2012.10.14				

Reikalavimas #:	3	Reikalavimo tipas:	9a	Įvykis/panaudojimo atvejis #:	8
Aprašymas:	Sistema turi leisti įmonių atstovams patiems kataloge užregistruoti įmones. Vartotojui užsiregistravus sistemoje, jam turi būti leidžiama užregistruoti įmonę kataloge.				
Pagrindimas:	Būtina suteikti vartotojams kuo daugiau savarankiškų teisių, kitaip jie mažai domėsis sistema. Paspaudus registravimosi kataloge mygtuką, pateikiama registravimosi forma, ją				

	užpildžius ir išsaugojus, įmonė tampa matoma kataloge, o sistemos administratoriui išsiunčiamasel. laiškas apie naują katalogo įrašą.		
Šaltinis:	Įmonių atstovai		
Tikimo kriterijus:	Prisijungęs sistemos lankytojas gali užregistruoti savo įmonę kataloge		
Užsakovo tenkinimas:	1	Užsakovo netenkinimas:	5
Priklausomybės	1	Konfliktai:	Nėra
Papildoma medžiaga:			
Istorija:	Užregistruotas 2012.10.14		

Reikalavimas #:	4	Reikalavimo tipas:	9a	Įvykis/panaudojimo atvejis #:	9
Aprašymas:	Sistema turi leisti įmonių atstovams moderuoti jiems priklausančius katalogo įrašus				
Pagrindimas:	Labai svarbu, kad įmonių atstovai patys kuo dažniau atnaujintų kataloge esančią informaciją, taip sumažėtų darbo krūvis sistemos administratoriams, dažnai atnaujinančioms savo profilį įmonėms būtų galima skirti papildomų reitingo taškų				
Šaltinis:	Administratorius				
Tikimo kriterijus:	Prisijungęs kaip sistemos vartotojas ir pasirinkęs katalogo įrašą, įmonės atstovas gali keisti jo informaciją arba pašalinti jį. Įmonės atstovas negali redaguoti kitų įmonių katalogo įrašų.				
Užsakovo tenkinimas:	3			Užsakovo netenkinimas:	3
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2012.10.14				

Reikalavimas #:	5	Reikalavimo tipas:	9a	Įvykis/panaudojimo atvejis #:	7
Aprašymas:	Sistemoje turi būti įdiegtas užklausų siuntimo modulis				
Pagrindimas:	Vartotojams patogu išsiųsti užklausas iš karto keletui įmonių.				
Šaltinis:	Visi sistemos vartotojai				
Tikimo kriterijus:	Paprastas vartotojas gali išsiųsti užklausą daugeliui įmonių vienu metu.				
Užsakovo tenkinimas:	2			Užsakovo netenkinimas:	4
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2012.10.14				

Reikalavimas #:	6	Reikalavimo tipas:	9a	Įvykis/panaudojimo atvejis #:	2

Aprašymas:	Sistema turi leisti administratoriams moderuoti sistemos vartotojus		
Pagrindimas:	Sistemoje gali užsiregistruoti įvairių vartotojų: nuo tokių, kuriems reikia pagalbos kaip naudotis sistema, iki tokių, kurie kenks sistemai. Todėl administratoriui būtina turėti galimybę moderuoti vartotojus: redaguoti jų informaciją, rolę, teises, blokuoti ar šalinti iš sistemos.		
Šaltinis:	Administratorius		
Tikimo kriterijus:	Prisijungęs prie svetainės administravimo sąsajos ir pasirinkęs vartotoją, administratorius gali keisti jo informaciją arba pašalinti jį.		
Užsakovo tenkinimas:	3	Užsakovo netenkinimas:	3
Priklausomybės	Nėra	Konfliktai:	Nėra
Papildoma medžiaga:			
Istorija:	Užregistruotas 2012.10.14		

Reikalavimas #:	7	Reikalavimo tipas:	9a	Įvykis/panaudojimo atvejis #:	10
Aprašymas:	Sistema turi leisti administratoriams moderuoti katalogo įrašus				
Pagrindimas:	Įmonių atstovai gali ne iki galo teisingai užpildyti įmonės registravimo formą. Administratoriui būtina turėti galimybę redaguoti, šalinti įmonės anketą iš katalogo				
Šaltinis:	Administratorius				
Tikimo kriterijus:	Prisijungęs prie svetainės administravimo sąsajos ir pasirinkęs katalogo įrašą, administratorius gali keisti jo informaciją arba pašalinti jį.				
Užsakovo tenkinimas:	3			Užsakovo netenkinimas:	3
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2012.10.14				

Reikalavimas #:	8	Reikalavimo tipas:	9a	Įvykis/panaudojimo atvejis #:	3
Aprašymas:	Sistema turi leisti naršyti katalogą				
Pagrindimas:	Katago pagrindinė funkcija yra leisti vartotojams peržiūrėti jame esančius įrašus, tai galima padaryti naršymo pagalba				
Šaltinis:	Visi sistemos vartotojai				
Tikimo kriterijus:	Įėjus į katalogą, pateikiamas nustatytas skaičius pirmųjų pagal rikiavimą katalogo įrašų. Puslapiavimo pagalba galima peržiūrėti tolesnius kataloge esančius įrašus. Paspaudus ant įrašo įrašų sąraše, pereinama į įrašo peržiūros puslapį su visa apie įrašą pateikiama informacija				
Užsakovo tenkinimas:	2			Užsakovo netenkinimas:	4
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:					

Istorija:	Užregistruotas 2012.10.14

Reikalavimas #:	9	Reikalavimo tipas:	9a	Įvykis/panaudojimo atvejis #:	3
Aprašymas:	Sistema turi leisti filtruoti katalogą				
Pagrindimas:	Efektyvus naršymas kataloge gali būti užtikrintas filtravimo pagalba				
Šaltinis:	Visi sistemos vartotojai				
Tikimo kriterijus:	Naršant kataloge ir pasirinkus vieną ar kelis filtravimo kriterijus, sistema automatiškai perkrauna rodomą katalogo turinį ir rodo tik tuos įrašus, kurie atinka pasirinktus filtravimo kriterijus				
Užsakovo tenkinimas:	3			Užsakovo netenkinimas:	4
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2012.10.14				

Reikalavimas #:	10	Reikalavimo tipas:	9a	Įvykis/panaudojimo atvejis #:	4
Aprašymas:	Turi veikti paieška kataloge ir ji turi būti tiksli				
Pagrindimas:	Efektyvi paieška kataloge yra viena esminių kuriamos sistemos funkcijų				
Šaltinis:	Visi sistemos vartotojai				
Tikimo kriterijus:	Įvedus įmonės, kuri jau registruota kataloge, pavadinimą, surandama ta įmonė.				
Užsakovo tenkinimas:	3			Užsakovo netenkinimas:	4
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2012.10.14				

Reikalavimas #:	11	Reikalavimo tipas:	9a	Įvykis/panaudojimo atvejis #:	5
Aprašymas:	Sistema turi leisti įvertinti katalogo įrašus				
Pagrindimas:	Katalogo įrašų įvertinimas yra viena iš priemonių, padėsianti apskaičiuoti katalogo įrašų reitingus ir pagal tai surikiuoti įmones				
Šaltinis:	Registruoti sistemos vartotojai				
Tikimo kriterijus:	Vartotojui pažymėjus savo įvertinimą, kuris vartotojo sąsajoje vaizduojamas žvaigždučių pavidalu, įvertinimas išsaugomas ir perskaičiuojamas vidutinis katalogo įrašo įvertinimų vidurkis.				
Užsakovo tenkinimas:	3			Užsakovo netenkinimas:	4
Priklausomybės	Nėra			Konfliktai:	Nėra

Papildoma medžiaga:	
Istorija:	Užregistruotas 2012.10.14

Reikalavimas #:	12	Reikalavimo tipas:	9a	Įvykis/panaudojimo atvejis #:	5
Aprašymas:	Sistema turi leisti keisti įmonių reitingavimo kriterijus ir jų svarbumo koeficientus.				
Pagrindimas:	Bėgant laikui gali keistis požiūris į įmonių reitingavimo politiką. Galimybė sistemos administratoriams patiems keisti reitingavimo kriterijus sumažina programuotojų apkrovimą ir klaidų galimybę.				
Šaltinis:	Sistemos administratorius				
Tikimo kriterijus:	Išsaugojus naujai pasirinktus kriterijus, įmonės perreitinguojamos. Leidžiama pasirinkti ar įmones kataloge perrikiuoti tuojau pat, ar pagal grafiką numatytu laiku				
Užsakovo tenkinimas:	3			Užsakovo netenkinimas:	4
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2012.10.14				

Reikalavimas #:	13	Reikalavimo tipas:	9a	Įvykis/panaudojimo atvejis #:	5
Aprašymas:	Sistema turi kas savaitę arba adminstratoriaus nurodytu metu perreitinguoti katalogo įmones				
Pagrindimas:	Įmonių reitingavimo kriterijai yra dinamiški ir nuolat kinta. Dėl to reikalingas automatinis įmonių perreitingavimas.				
Šaltinis:	Sistemos administratorius				
Tikimo kriterijus:	Atėjus pagal grafiką numatytam laikui įmonių reitingai yra perskaičiuojami be jokio programuotojo ar administratoriaus įsikišimo. Įmonių eiliškumas kataloge yra pakeičiamas pagal šį reitingą.				
Užsakovo tenkinimas:	3			Užsakovo netenkinimas:	4
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2012.10.14				

Reikalavimas #:	14	Reikalavimo tipas:	9a	Įvykis/panaudojimo atvejis #:	10
Aprašymas:	Sistema turi leisti administratoriui rankiniu būdu nustatyti įmonės eiliškumą kataloge				
Pagrindimas:	Gali būti, kad sistemos savininkai norės reguliuoti įmonių eiliškumą kataloge ne pagal jų reitingą, o, pavyzdžiui, pagal tai, kad įmonė sumokėjo už galimybę figuruoti aukštesnėse katalogo pozicijose.				
Šaltinis:	Sistemos administratorius				

Tikimo kriterijus:	Nurodžius konkrečios įmonės vietą kataloge, ji atsiduria būtent toje vietoje. Kitos įmonės pasislenka per vieną poziciją žemyn.		
Užsakovo tenkinimas:	3	Užsakovo netenkinimas:	4
Priklausomybės	Nėra	Konfliktai:	Nėra
Papildoma medžiaga:			
Istorija:	Užregistruotas 2012.10.14		

3.1.6 Nefunkciniai reikalavimai sistemai

Nefunkciniai reikalavimai nusako kokybines sistemos savybes, kuriomis ji turi pasižymėti. Patikrinti, ar įvykdytas funkcnis reikalavimas, yra sunkiau ir dažnai priklauso nuo vertintojo subjektyvaus požiūrio.

Reikalavimai sistemos išvaizdai

Reikalavimas #:	15	Reikalavimo tipas:	10	Įvykis/panaudojimo atvejis #:	1-10
Aprašymas:	Modernaus, šiuolaikiško dizaino ir patogi naudoti vartotojo sąsaja				
Pagrindimas:	Siekiant konkuruoti su konkurentais, būtina sudaryti kokybiškos, modernios sistemos įvaizdį				
Šaltinis:	Visi sistemos vartotojai				
Tikimo kriterijus:	Sąsajos dizainą teigiamai vertina 60% apklaustų žmonių. Sąsają teigiamai vertina 75% apklaustų vartotojo sąsajos dizaino specialistų				
Užsakovo tenkinimas:	4			Užsakovo netenkinimas:	4
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2012.10.14				

Reikalavimai panaudojamumui

Reikalavimas #:	16	Reikalavimo tipas:	11a	Įvykis/panaudojimo atvejis #:	1-10
Aprašymas:	Vartotojo sąsaja pritaikyta išmaniesiems mobiliems įrenginiams				
Pagrindimas:	Vis daugiau vartotojų naudojasi išmaniais telefonais, planšetiniais kompiuteriais ir t.t. Todėl aktualu, kad jie galėtų visavertiškai naudotis sistema				
Šaltinis:	Visi sistemos vartotojai				
Tikimo kriterijus:	Nepriklausomai nuo to, kokia įrenginio rezoliucija, vartotojo sąsaja visada telpa į ekraną esant mąsteliui 1:1, o tekstą yra nesunku įskaityti.				

Užsakovo tenkinimas:	5	Užsakovo netenkinimas:	3
Priklausomybės	Nėra	Konfliktai:	Nėra
Papildoma medžiaga:			
Istorija:	Užregistruotas 2012.10.14		

Reikalavimas #:	17	Reikalavimo tipas:	11a	Įvykis/panaudojimo atvejis #:	1-10
Aprašymas:	Vartotojo sąsaja kokybiškai atvaizduojama visose populiariausiose interneto naršyklėse				
Pagrindimas:	Lietuvoje vis dar daug vartotojų naudoja pasenusias interneto naršykles. Būtina į tai atsižvelgti ir užtikrinti, kad sistema būtų galima naudotis visomis populiariausiomis naršyklėmis. Visiškai identiškas dizaino atvaizdavimas tarp įvairių naršyklių nebūtinas				
Šaltinis:	Visi sistemos vartotojai				
Tikimo kriterijus:	Sistemos dizainas atvaizduojamas be iškraipymų ir visos funkcijos korektiškai veikia šiose interneto naršyklėse: IE6+, Mozilla Firefox 4+, Google Chrome 9+.				
Užsakovo tenkinimas:	4			Užsakovo netenkinimas:	4
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2012.10.14				

Reikalavimas #:	18	Reikalavimo tipas:	10	Įvykis/panaudojimo atvejis #:	1-10
Aprašymas:	Susigaudyti vartotojo sąsajoje ir rasti reikalingus mygtukus per trumpą laiką sugeba mažai internetu besinaudojantys asmenys				
Pagrindimas:	Vis dar labai daug interneto vartotojų turi tik bazines naudojimosi internetu žinias, yra pripratę prie pasenusių, statiškų vartotojo sąsajų				
Šaltinis:	Visi sistemos vartotojai				
Tikimo kriterijus:	80 proc. bazines interneto naudojimo žinias turinčių asmenų neužgaišo daugiau nei 1 min. 30 sek. paprašius neuiti į įmonių sąrašo puslapį ir išrikiuoti įrašus pagal jų reitingą				
Užsakovo tenkinimas:	2			Užsakovo netenkinimas:	2
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2012.10.14				

Reikalavimas #:	19	Reikalavimo tipas:	10	Įvykis/panaudojimo atvejis #:	1-10
Aprašymas:	Visos sistemoje esančios duomenų įvedimo formos yra validuojamos kliento pusėje, su apsauga iš serverio pusės esant išjungtam Javascript palaikymui. Vartotojo klaidos ir patarimai turi būti jam parodomi dar nepaspaudus formos vykdymo mygtuko.				
Pagrindimas:	Kliento pusės validavimas prisideda prie draugiškos vartotojo sąsajos įvaizdžio, didina				

	klientų pasitenkinimą sistema.				
Šaltinis:	Visi sistemos vartotojai				
Tikimo kriterijus:	Įvedinėjant duomenis į formas, vartotojui įvedus neteisingą el. pašto formatą, neteisingą datos formatą, jam rodomas klaidos pranešimas dar nepaspaudus formos išsaugojimo mygtuko. Jei palikti tušti privalomi laukai, vartotojui rodomi klaidų pranešimai paspaudus formos išsaugojimo mygtuką, o formos išsaugojimo procesas nepradedamas.				
Užsakovo tenkinimas:	4			Užsakovo netenkinimas:	2
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2012.10.14				

Reikalavimai vykdymo charakteristikoms

Reikalavimas #:	20	Reikalavimo tipas:	12a	Įvykis/panaudojimo atvejis #:	1-16
Aprašymas:	Sistema turi sparčiai užsikrauti naršyklėje				
Pagrindimas:	Sistemos užsikrovimo ir veikimo greitis labai svarbus vartotojų pasitenkinimui				
Šaltinis:	Visi sistemos vartotojai				
Tikimo kriterijus:	Pirmą kartą atsidarius sistemos namų puslapį, sistema užsikrauna ne ilgiau kaip per 5 sek., o toliau naršant po svetainę kiti puslapiai užsikrauna ne ilgiau kaip per 2 sek. atsiuntimo greičiui esant 40Mbps nesvarbu kiek tuo metu lygiagrečių vartotojų naudojasi sistema.				
Užsakovo tenkinimas:	2			Užsakovo netenkinimas:	4
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:					
Istorija:	Užregistruotas 2012.10.14				

Teisiniai reikalavimai

Reikalavimas #:	21	Reikalavimo tipas:	17a	Įvykis/panaudojimo atvejis #:	1-16
Aprašymas:	Produktas turi vadovautis duomenų apsaugos įstatymu.				
Pagrindimas:	Duomenys neturi būti prieinami bet kam.				
Šaltinis:					
Tikimo kriterijus:	Sistema skelbia duomenų privatumo politiką.				
Užsakovo tenkinimas:	3			Užsakovo netenkinimas:	4
Priklausomybės	Nėra			Konfliktai:	Nėra
Papildoma medžiaga:	Duomenų apsaugos įstatymas.				

Istorija:	Užregistruotas 2004 Sausio 15.

3.2 Architektūros specifikacija

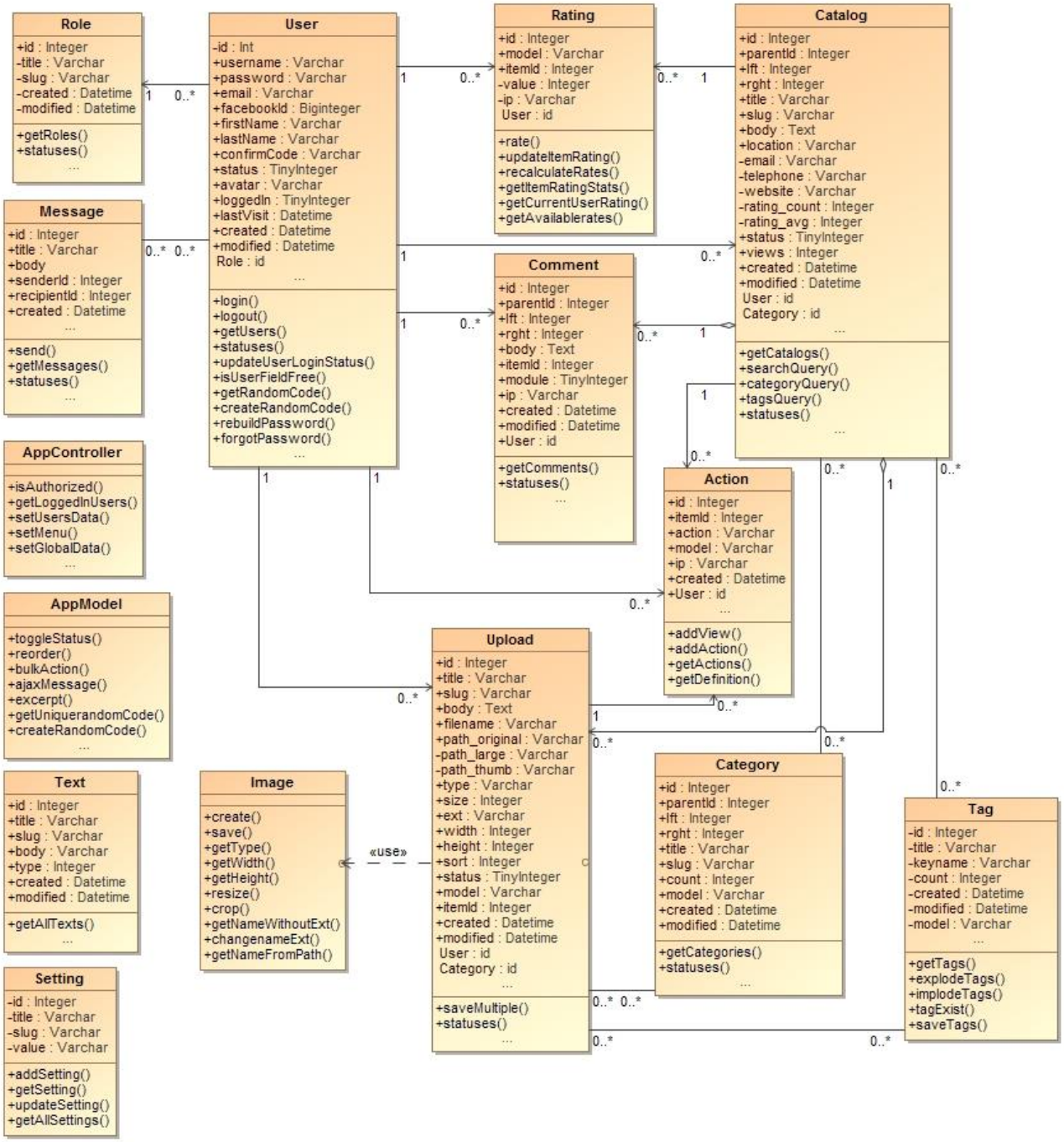
Šis dokumentas yra skirtas pateikti objektyvų ir konkretų kuriamos įmonių katalogo IS architektūrinį vaizdą. Jame surinkta ir pateikta visa svarbiausia medžiaga, sukaupta ruošiantis programuoti sistemą. Šiuo dokumentu galės naudotis programuotojai, realizuojantys sistemą, taip pat šis dokumentas turi būti nuolat tobulinamas lygiagrečiai su sistemos programavimu, kad kintant reikalavimams sistemai visuomet būtų galima vadovautis šiuo dokumentu kaip „atspirties tašku“.

3.2.1 Sistemos statinis vaizdas

Sistemos statiniui vaizdui pavaizduoti panaudota klasių diagrama. Klasė apibūdina objektų, kurie turi vienodą charakteristiką ir semantiką, struktūrą ir elgesį. Struktūra yra apibūdinama atributais, o elgesys - operacijomis [2]. Nors sistema yra kuriama remiantis MVC modeliu, vaizduojant klasių diagramą sistemą skaidyti išskiriant Model (duomenų bazės), Controller (kontrolerio) ir View (vaizdo) paketus yra nevisai logiška dėl šių priežasčių:

- sistema yra kuriama su PHP programavimo kalba. Joje, skirtingai nuo pvz. desktop aplikacijoms kurti skirtų programavimo kalbų, nėra išskiriamos View klasės. Sistemos vaizdas formuojamas elementais ne kaip klasės objektais, o panaudojant vaizdo šablonus (failus), kuriuose Html kodas persipina su PHP kodu;
- sistemos kūrime naudojamas karkasas, kuriame didžioji dalis dažniausiai naudojamų funkcijų, pvz sąveikai su DB, jau yra aprašyta. Todėl išskirti Model paketo taip pat nėra prasmės.

3 paveikslėlyje pateikta sistemos klasių diagrama, apjungianti visas būsimos sistemos unikalias esybes ir metodus, susijusius būtent su kuriamos sistemos logika:



Pav. 3. Įmonių katalogo IS klasių diagrama

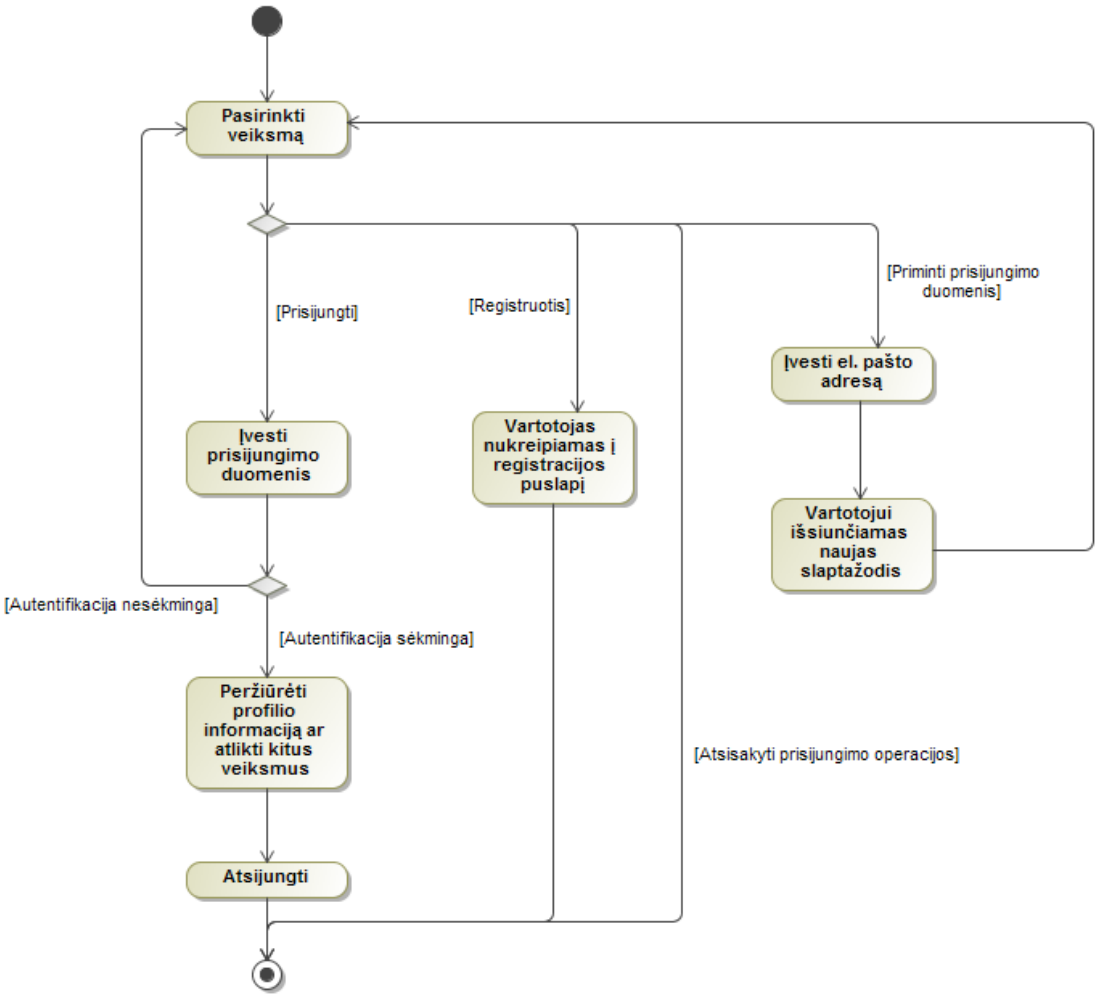
Modeliuojant sistemos architektūrą, stengtasi kurti universalius metodus, kaip įmanoma daugiau kodo perkeliant į modelius, tokiu būdu kontrolerius paliekant kaip įmanoma „liesesnius“. Pasikartojantys metodai, kurie gali būti panaudoti keliuose modeliuose, iškelti į tėvinę modelių klasę AppModel.

Gana dažnai pasikartojančiame metode statuses() yra aprašomi modelio esybės galimų būsenų, išreikštų skaičiais, žodiniai pavidalai, ikonos, atspindinčios kiekvieną statusą, statuso atitikmuo lotyniškomis raidėmis ir kita reikalinga informacija.

AppController klasėje laikomi metodai, naudojami globaliai visoje sistemoje, ir kurie negali būti aprašyti AppModel klasėje dėl to, kad yra tiesiogiai susiję su užklausos parametrais, pvz. puslapiavimu. Klasės Setting ir Text nėra tiesiogiai susietos su kitomis klasėmis. Klasė Setting skirta valdyti sistemos nustatymus, klasė Text - dirbti su statiniais sistemos tekstais. Klasę Image naudoja Upload klasė paveikslėliams apdoroti.

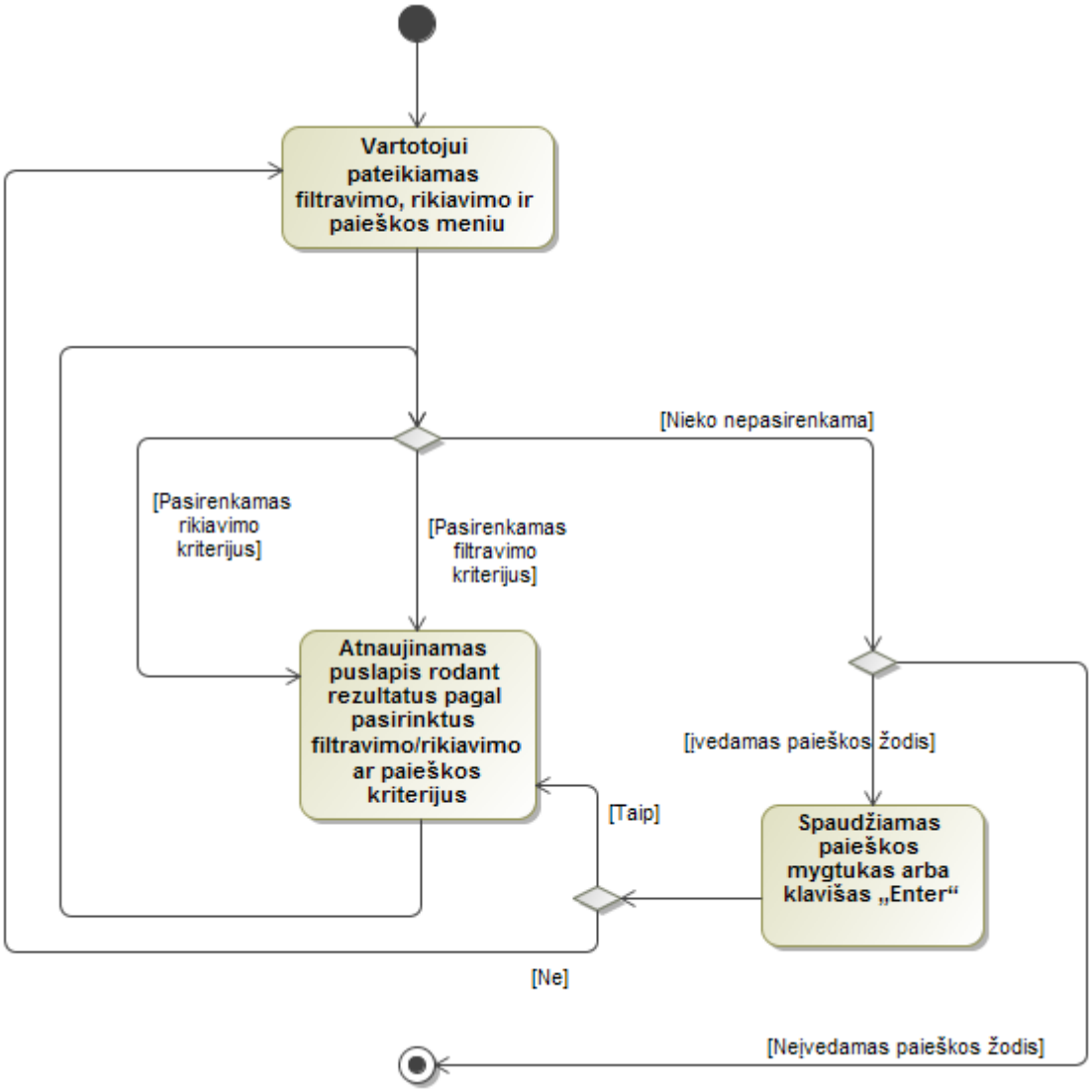
3.2.2 Sistemos dinaminis vaizdas

Šiame skyriuje pateikiamos svarbiausius sistemos panaudojimo atvejus vaizduojančios veiklos diagramos. Jos leis geriau atskleisti procesus, kurie vyks vartotojams naudojantis sistema.



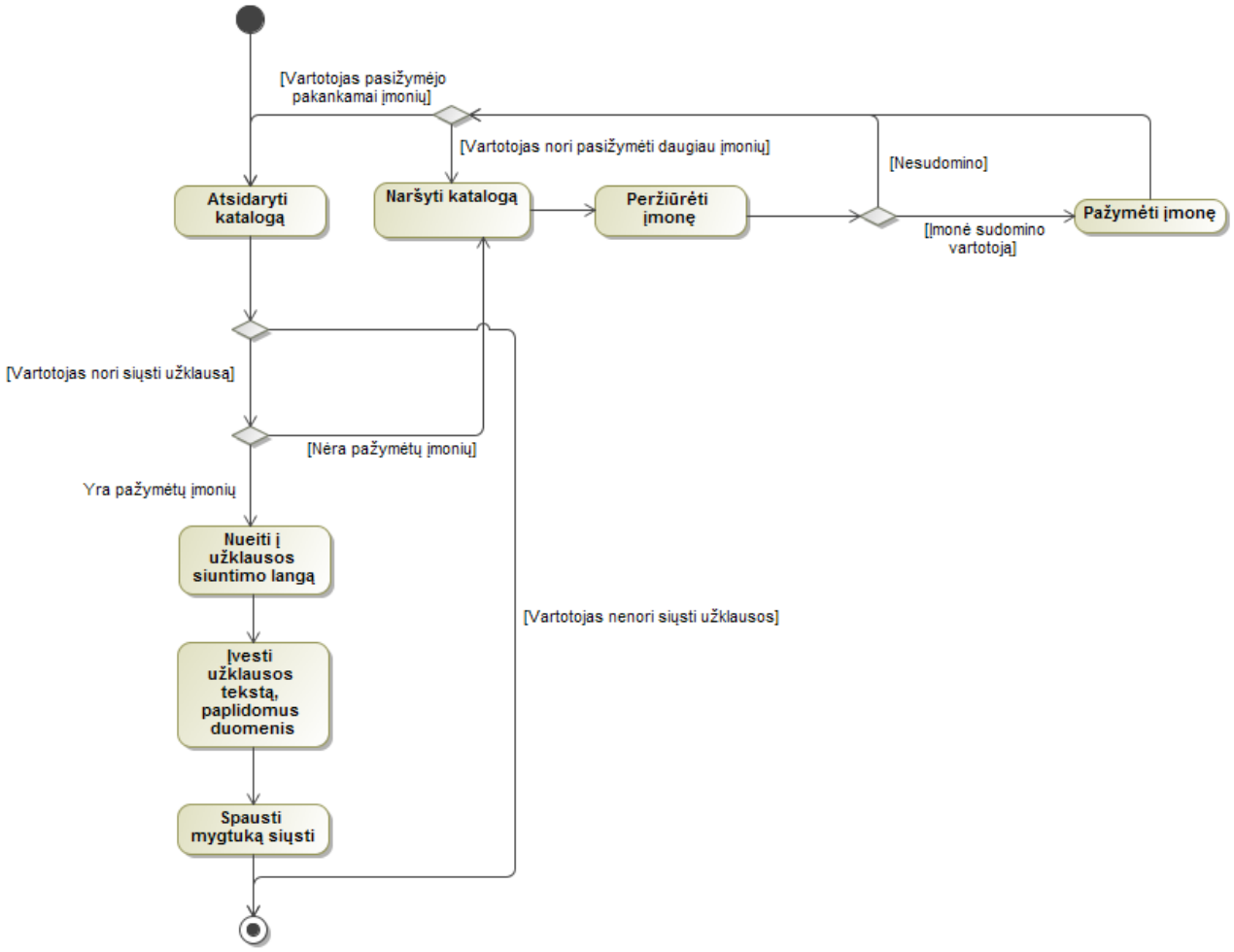
Pav. 4. Panaudojimo atvejo „Prisijungti“ veiklos diagrama

Vartotojo prisijungimo procesui bus skirtas atskiras sistemos puslapis su forma, skirta prisijungimo duomenims įvesti. Šalia formos bus pateikiami du mygtukai: „Registracija“ ir „Slaptažodžio priminimas“, kad vartotojas galėtų užsiregistruoti sistemoje jei dar nėra to padaręs ar gauti naują slaptažodį į savo el. pašto adresą, jei pamiršo senąjį.



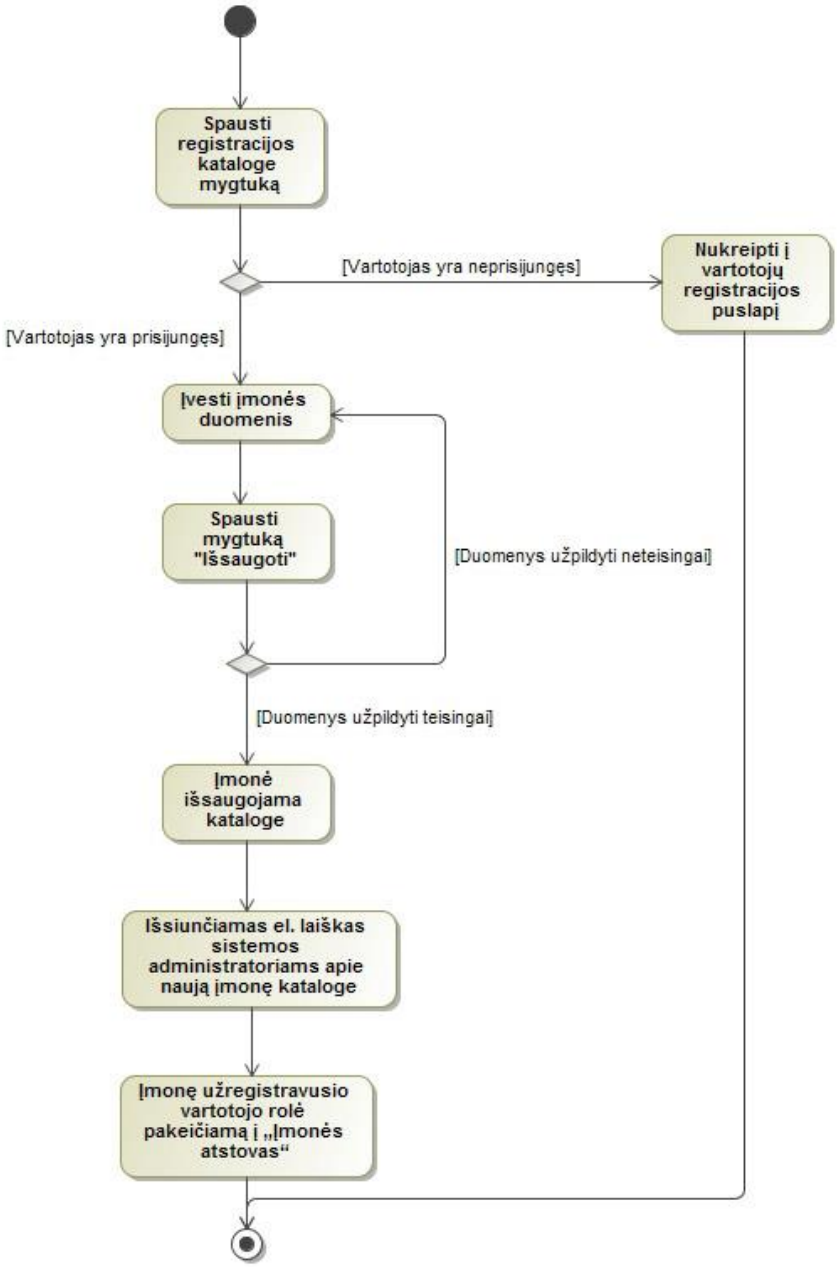
Pav. 5. Panaudojimo atvejo „Ieškoti kataloge“ veiklos diagrama

Paieška kataloge bus atliekami šoniniame meniu esančių filtravimo, rikiavimo ir paieškos parinkčių pagalba. Vartotojai galės filtruoti rezultatus pagal kategoriją ar žymę, o esant reikalui programuotojams bus nesunku įvesti papildomus filtravimo kriterijus. Bus galima surikiuoti įrašus pagal reitingą, registracijos datą, atnaujinimo datą, populiariumą, abėcėlę. Siekiant filtravimo ir rikiavimo funkciją padaryti kuo dinamiškesnę ir patogesnę vartotojui, rezultatai bus atnaujinami automatiškai, vartotojui pasirinkus kažkurį filtravimo ar rikiavimo kriterijų, kad nereikėtų papildomai spausti mygtuko „Ieškoti“. Paspaudus antrą kartą ant to paties kriterijaus, jis panaikinamas. Pažymėtą kriterijų leis atskirti suteikta ryškiai mėlyna spalva. Šios metodikos nepavyks įgyvendinti paieškai pagal raktažodį: įvedęs norimą žodį, vartotojui reikės paspausti klavišą „Enter“ arba mygtuką „Ieškoti“, automatiškai rezultatai neatsinaujins.



Pav. 6. Panaudojimo atvejo „Siųsti užklausą įmonėms“ veiklos diagrama

Naršant įmonių katalogą, prie kiekvienos įmonės pateikiamas mygtukas „Pažymėti“. Jei vartotojas yra pažymėjęs bent vieną įmonę, viršutiniame sistemos meniu jam bus rodomas pažymėtų įmonių skaičius. Paspaudęs ant jo, vartotojas atsidurs užklausos siuntimo lange, kur matys visų pažymėtų įmonių sąrašą, galės įvesti užklausos tekstą, savo kontaktinius duomenis ir išsiųsti užklausą.

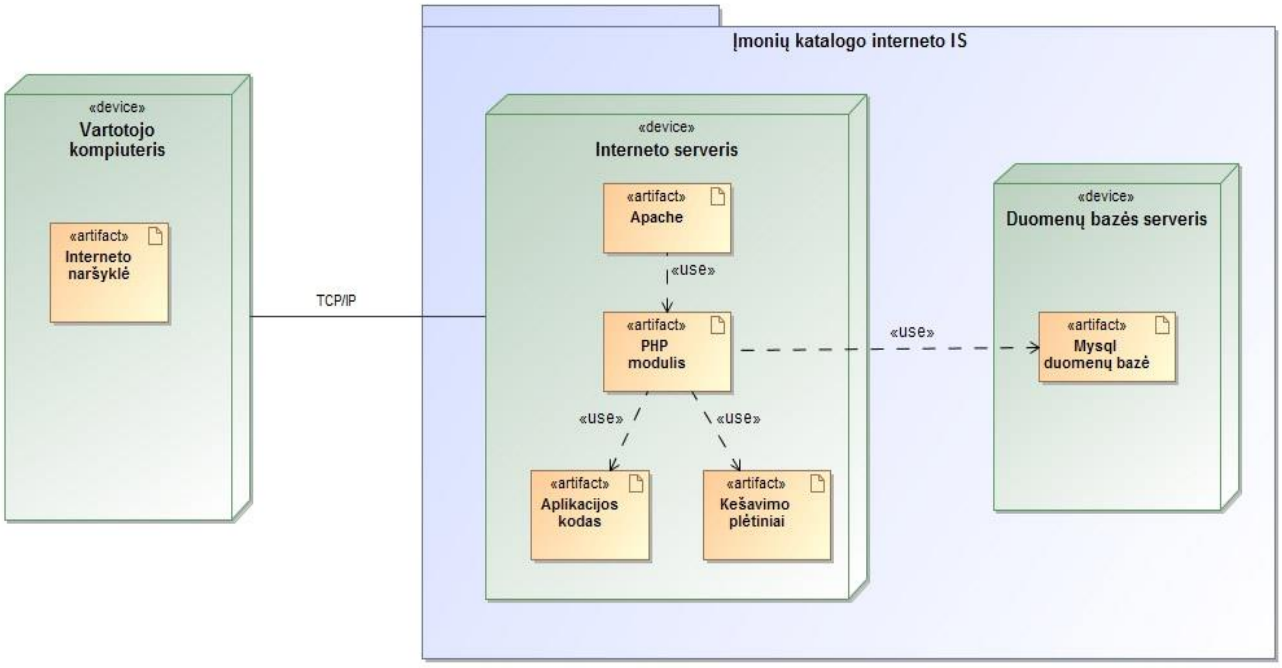


Pav. 7. Panaudojimo atvejo „Registruotis kataloge“ veiklos diagrama

Sistema turės gerai matomą mygtuką „Registruotis kataloge“. Jį matys tiek registruoti, tiek neregistruoti vartotojai, besinaudojantys sistema. Tai reikalinga, kad visi sistemos naudotojai gerai suprastų, jog sistema yra atvira ir visi gali užregistruoti įmones (jei tokių turi), ne tik sistemos administratoriai. Tačiau paspaudus šį mygtuką, neregistruoti vartotojai bus nukreipiami į vartotojų registracijos puslapį ir jiems pateikima informacinė žinutė, kad norėdami užregistruoti įmonę kataloge, pirmiausia jie turi būti užsiregistravę kaip sistemos vartotojai. Tai reikalinga, kad kiekvienas katalogo įrašas turėtų „savininką“ - registruotą vartotoją, su kuriuo vėliau būtų galima susisiekti sistemos administratoriams ar katalogo naudotojams.

3.2.3 Komponentų modelis

8 paveikslėlyje pavaizduotas sistemos architektūros komponentų modelis.



Pav. 8. Sistemos architektūros komponentų modelis

Sistemos veikimui užtikrinti bus panaudoti 2 atskiri aparatinės įrangos blokai-serveriai: interneto ir duomenų bazės. Du atskiri serveriai naudojami saugumo sumetimais.

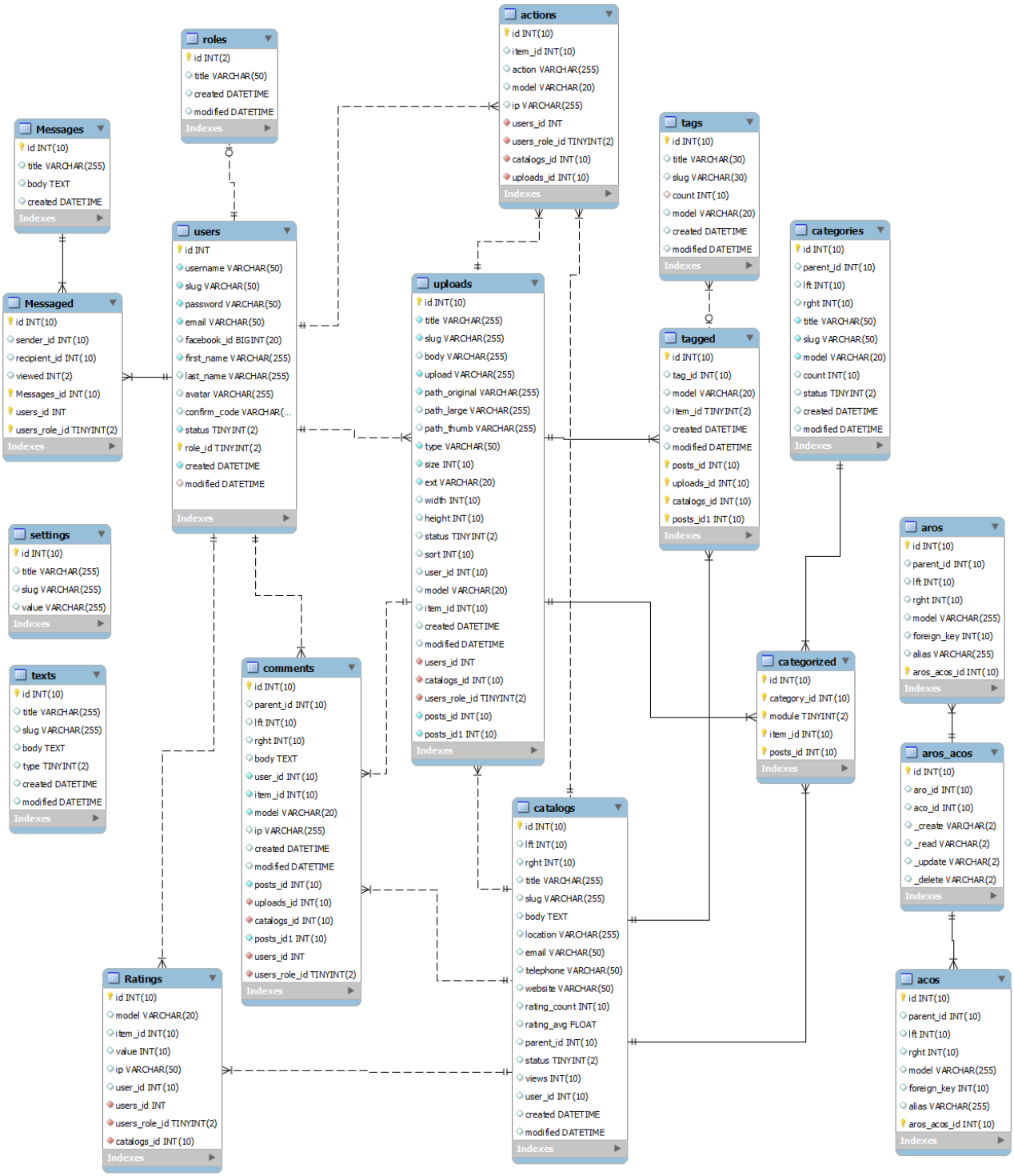
Interneto serveryje yra Apache http programinė įranga, atsakinga už http užklausos priėmimą ir atsakymo į ją sugeneravimą. PHP modulis vykdo sistemos algoritmus, parašytus PHP kalba. Jame, kaip papildomi plėtiniai, yra įrašytos kešavimo technologijos.

DB serveris skirtas Mysql duomenų bazės darbams vykdyti. Mysql pasirinkta todėl, kad tai populiariausia duomenų bazių sistema. Laikoma, kad MySQL stabilumas ir paplitimas dažnai yra tinkamesnis pasirinkimas daugeliui programų [12].

Vartotojai sistemą gali pasiekti interneto naršyklės pagalba. Sistema optimizuota veikti tokiose interneto naršyklėse: InternetExplorer 7+, Mozilla Firefox 2+, Google Chrome 9+, Opera 8+, Safari 5+. Jokios kitos papildomos programinės įrangos vartotojo kompiuteryje nereikia. Sistema optmizuota veikti Windows XP/Vista/7/8 ir Mac OS operacinėse ssitemose.

3.2.4 Duomenų bazės modelis

„Tinkamas duomenų bazės projektavimas yra vienintelis būdas užtikrinti, kad jūsų taikomoji programa bus naši, lengvai valdoma ir prižiūrima“ (Meloni J.C., 2007: 284). Todėl įmonių katalogo IS duomenų bazė buvo projektuojama itin atidžiai, stengiantis numatyti „keletą ėjimų į priekį“.



Pav. 9. Įmonių katalogo IS duomenų bazės modelis

Duomenų bazės modeliui sukurti panaudota „Mysql Workbench“ programinė įranga, kadangi ji leidžia konvertuoti sukurtą modelį į SQL užklausą, kuri leidžia automatiškai sugeneruoti lenteles MySQL duomenų bazėje. Kuriant modelį buvo stengiamasi užtikrinti, kad DB saugomiems duomenims būtų užtikrinamos savybės, kurias kaip svarbiausias nurodo V.Sekliuckis, S.Gudas ir G.Garšva [9]: vientisumas, nepertekliškumas, neprieštaringumas, saugumas, nepriklausomumas.

Kiekviena DB modelio esybė turi automatiškai didėjantį unikalų atributą "id", beveik kiekviena - atributą "slug", kuriame saugomas objekto pavadinimas lotyniškomis raidėmis. Jis bus naudojamas nuorodoms į objektus generuoti.

Atributas "model", esantis ratings, comments, uploads, categorized ir tagged lentelėse nurodo, kurį modelį, kitaip tariant esybę (pvz. katalogo įrašą, įkeltą failą ar vartotoją) apibūdina lentelės reikšmė. Atributas "item_id", tai raktas, nusakantis tos esybės įrašo "id" reikšmę. Ši metodika leidžia trečios normalinės formos lenteles panaudoti daugiau nei tik dviems esybėms sujungti.

Comments ir categories lentelėse esantys "parent_id", "lft" ir "rght" atributai reikalingi siekiant suteikti šioms esybėms medžio elgsenos modelį.

Detaliau kiekvieną duomenų modelio lentelę aprašo 4 lentelė.

Lentelė 4. Duomenų bazės modelio lentelių aprašymas

Lentelė	Aprašymas
actions	Saugoma informacija apie vartotojų atliktus veiksmus. Ši lentelė leis sistemos lankytojams pateikti tokią informacija: "Vardenis Pavardenis įkėlė naują nuotrauką" arba "Vardenis Pavardenis paliko atsiliepimą apie A įmonę" ir t.t.
aros	Lentelė skirta vartotojų teisių valdymo moduliui. Joje saugoma informacija objektus, kurie gali prašyti leidimo atlikti kokius nors veiksmus sistemoje. Lentelės pavadinimas tai angliško termino „Access request object“ trumpinys.
acos	Lentelė skirta vartotojų teisių valdymo moduliui. Joje saugoma informacija veiksmus, kurie kurie gali būti atliekami sistemoje. Lentelės pavadinimas tai angliško termino „Access Control object“ trumpinys.
aros_acos	Lentelė skirta vartotojų teisių valdymo moduliui. Tai trečios normalinės formos lentelė, kurioje saugoma informacija apie tai, kas leidžiama ir kas neleidžiama sistemos naudotojams.
catalog	Pagrindinė sistemos esybė, kurioje saugomi užregistruotų įmonių duomenys

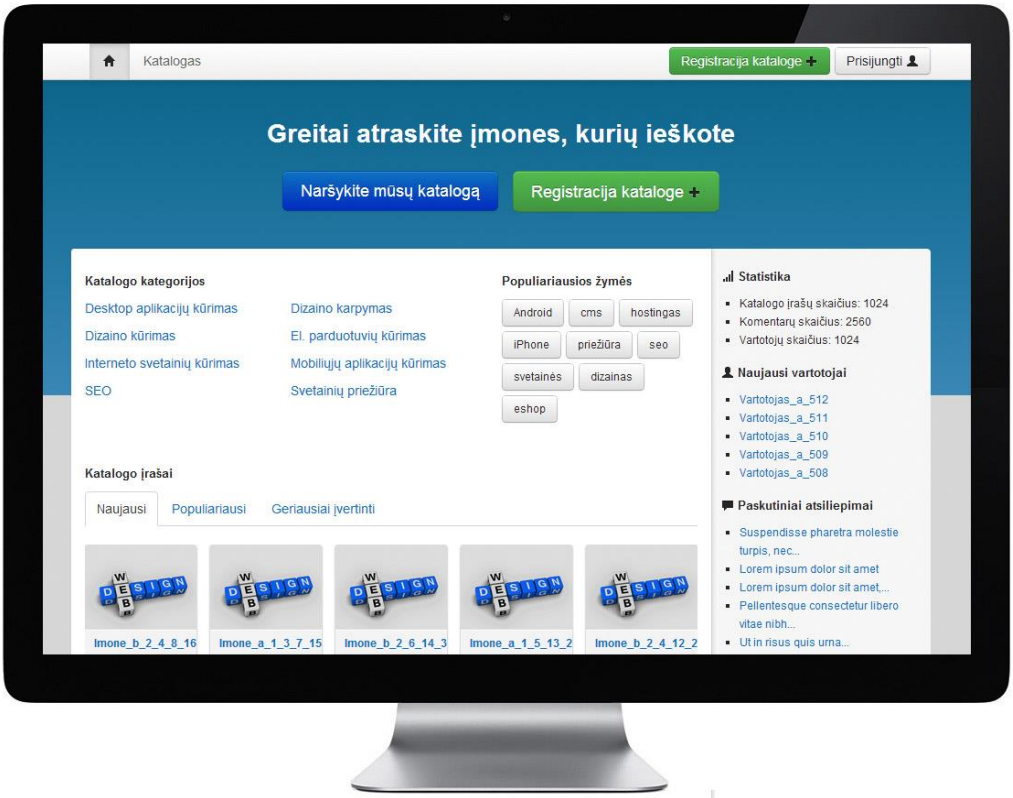
categories	Saugomos sistemos esybes kategorizuojančios kategorijos
categorized	Trečios normalinės formos lentelė, padedanti kategorizuoti sistemos esybes
comments	Saugomi vartotojų atsiliepimai
messages	Saugomos vartotojų parašytos žinutės
messaged	Trečios normalinės formos lentelė, leidžianti saugoti informaciją apie vienos žinutės išsiuntimą daugeliui gavėjų
ratings	Saugomi įmonių įvertinimai
roles	Saugoma informacija apie sistemos vartotojų roles. Numatoma, kad sistemoje bus tokios vartotojų rolės: administratorius, įmonės atstovas, paprastas vartotojas
settings	Su kitomis sistemos lentelėmis ryšių neturinti lentelė, kurioje saugoma informacija apie įvairius sistemos nustatymus
tags	Saugoma informacija apie sistemoje įvestas žymes
tagged	Trečios normalinės formos lentelė, padedanti priskirti žymes sistemos esybių objektams
texts	Su kitomis sistemos lentelėmis ryšių neturinti lentelė, kurioje saugomi statiniai sistemos tekstai
uploads	Saugoma informacija apie įkeltus failus (nuotraukas, word, pdf ir kt. dokumentus).
users	Saugoma informacija apie sistemos vartotojus.

3.3 Informacinės sistemos realizacija

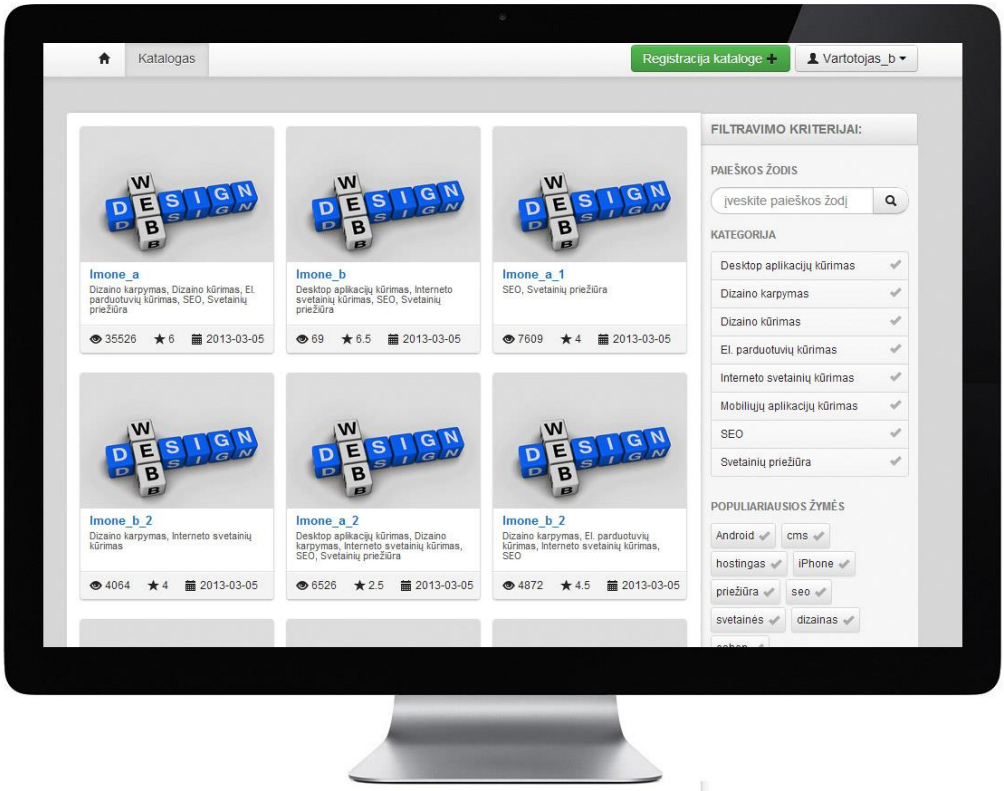
Atsižvelgiant į suprojektuotą sistemos architektūros modelį bei suformuluotus reikalavimus sistemai, buvo sukurta moderni, tačiau paprasta ir neperkrauta vartotojo sąsaja ir sistema realizuota programiškai. Vartotojo sąsaja kurta remiantis šiais principais:

- modernus, aiškus, neperkrautas dizainas;
- HTML5, CSS3 technologijų naudojimas;
- švarūs CSS stiliai, galimybė nesunkiai pakeisti sąsajos dizainą neliečiant pagrindinių CSS stilių;
- „Reaguojantis“ (angl. Responsive) dizainas. Ta pati sąsaja turi prisitaikyti ir tikti įvairaus dydžio ekranams.

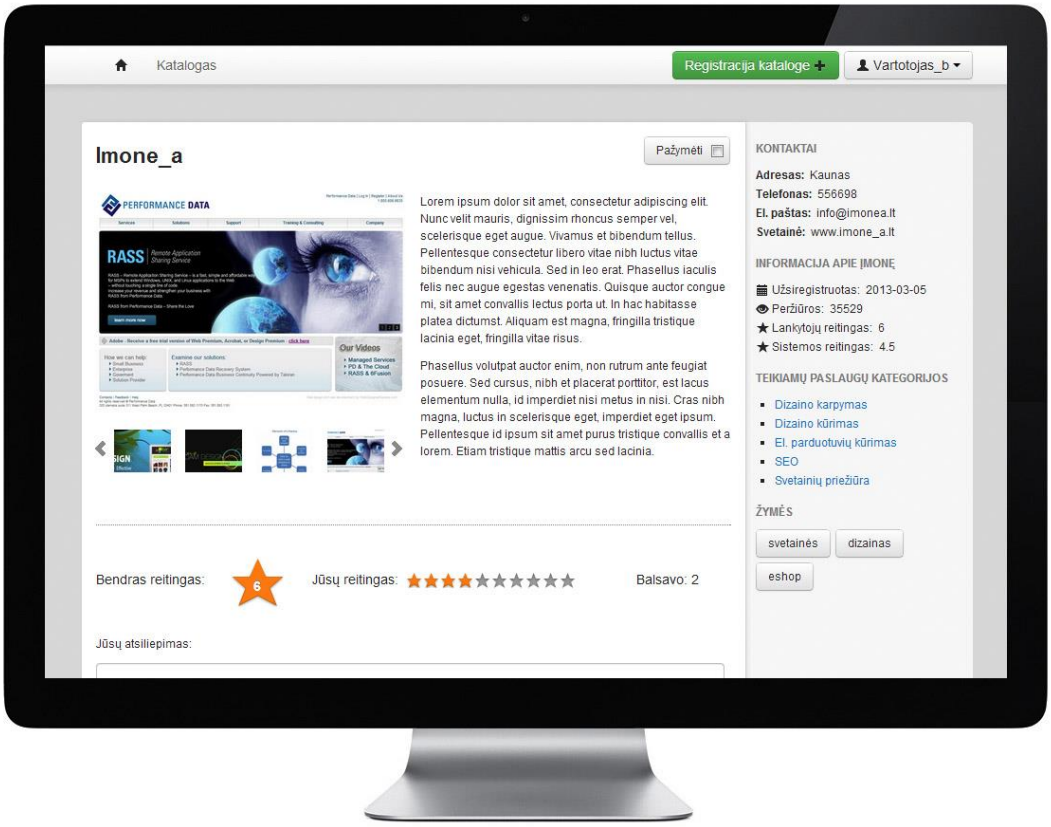
Toliau pateikiami svarbiausių sistemos puslapių vartotojo sąsajos langai.



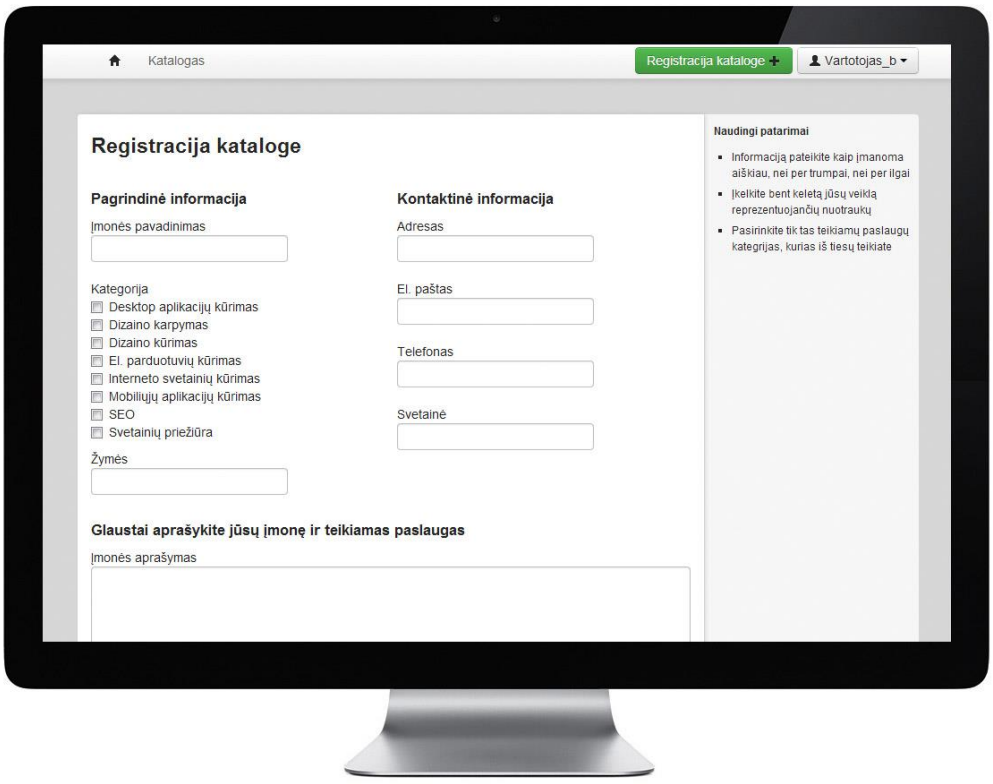
Pav. 10. Įmonių katalogo IS pradžios puslapis



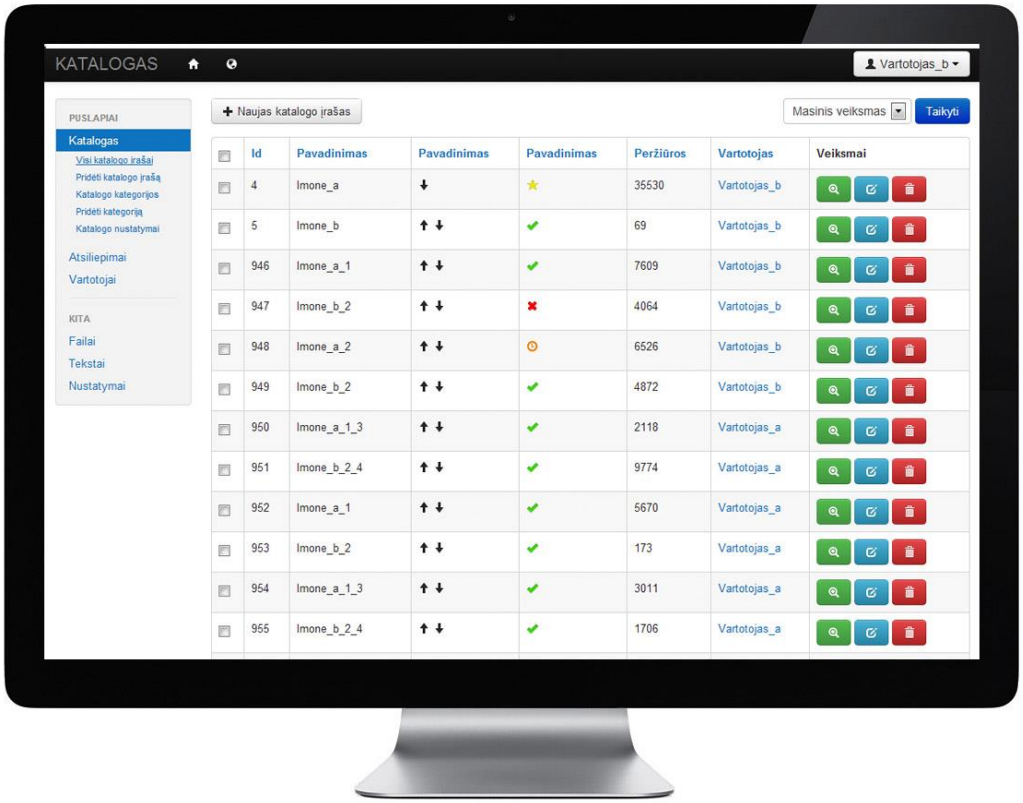
Pav. 11. Įmonių katalogo IS įmonių sąrašo puslapis



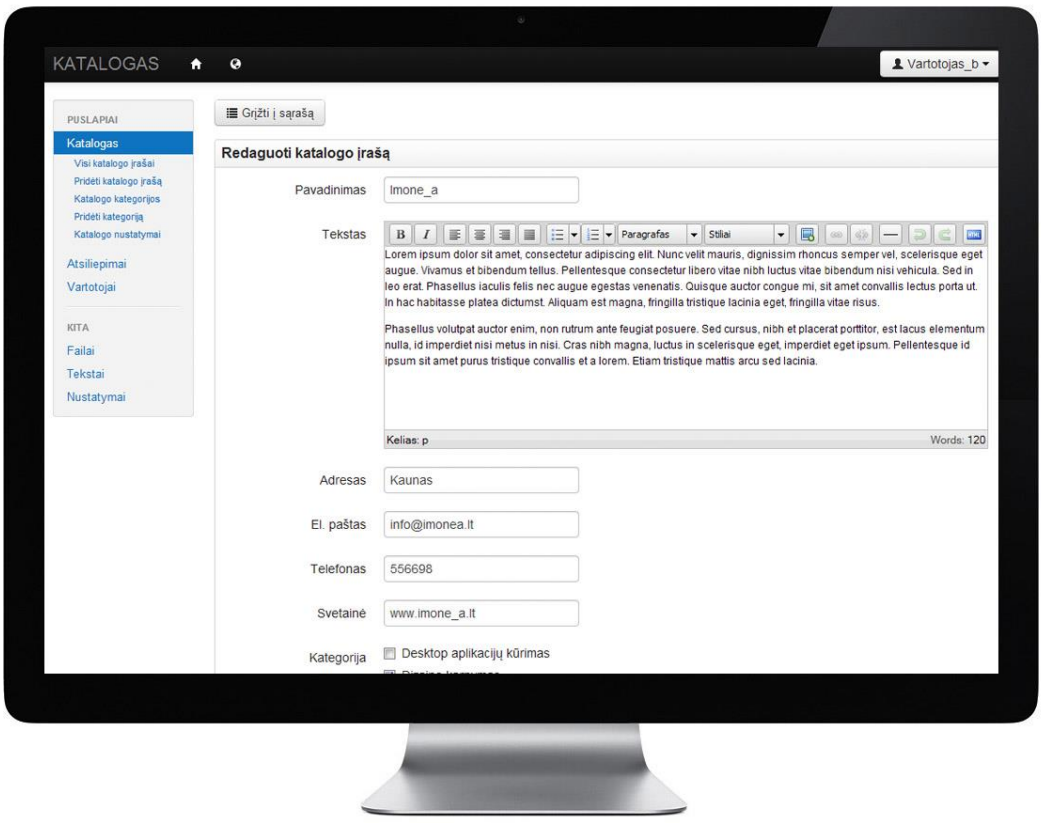
Pav. 12. Įmonių katalogo IS vidinis įmonės puslapis



Pav. 13. Įmonių katalogo IS registracijos kataloge puslapis



Pav. 14. Įmonių katalogo IS administracijos puslapis: katalogų sąrašas



Pav. 15. Įmonių katalogo IS administracijos puslapis: katalogo redagavimas

3.4 Apibendrinimas

Išstudijavus pirminius užsakovo poreikius, atlikta sistemos funkcinių ir nefunkcinių reikalavimų analizė bei parengta reikalavimų specifikacija.

Remiantis MVC modeliu suprojektuota sistemos architektūra leis efektyviai prižiūrėti sistemą ateityje, programuototjai ir techniniai dizaineriai galės atlikti savo darbus neliesdami vienas kito kodo.

Iki trečio normalinio lygio išskaidytas reliacinės duomenų bazės modelis suprojektuotas taip, kad, atsiradus poreikiui įvesti papildomas esybes į sistemą (pvz, skelbimai), jos galės būti susietos su komentarų, kategorizavimo, žymių, failų klasėmis trečia normaline forma nebekuriant naujų duomenų bazės lentelių, nerašant naujų metodų. Tai labai sumažins kodo kiekį ir padidins jo patikimumą.

Pasirinkta techninė (ang. Hardware) įranga yra labai plačiai naudojama viso pasaulio interneto sistemose, todėl yra patikrinta ir patikima.

Tos pačios vartotojo sąsajos pritaikymas iš karto kompiuteriams su dideliu ekranu ir mobiliesiems įrenginiams, turintiems mažesnį ekraną, nekuriant dviejų atskirų sąsajų, leidžia išvengti didelių laiko ir finansinių sąnaudų ir skirti daugiau laiko programinės įrangos tobulinimui.

4. KEŠAVIMO PRIEMONĖS IR JŲ TYRIMAS

4.1 Kešavimo priemonių analizė

4.1.1 Kešavimo samprata ir taikymas ir praktikoje

Kešavimo samprata

Kešavimo (angl. *caching*) sąvoką pirmą kartą įvedė kompanija IBM 20 a. 7-ajame dešimtmetyje. Kešavimas yra „subrendusi“ technologija, ilgą laiką naudota pagrinde operacinėse sistemose ir duomenų bazėse. Tačiau pastaruoju metu „pasaulinis interneto tinklas tampa dar viena populiaria kešavimo pritaikymo sritimi“ (Sammie, 2009: 96). Toliau šiame darbe kalbant apie kešavimą, bus turimas galvoje būtent **interneto** sistemų kešavimas.

Literatūros šaltiniuose galima rasti įvairių interneto sistemose naudojamo kešavimo apibūdinimų:

- tai originalių duomenų kopijos išsaugojimas atmintyje, iš kurios vėliau nuskaityti duomenis galima daug greičiau, nei norint gauti tuos pačius duomenis iš originalaus šaltinio [15];
- web kešavimo idėja yra pateikti reikalingą informaciją klientui mažesnėmis sąnaudomis [21];
- kešavimas leidžia išsaugoti populiarius objektus arčiau kliento ir yra laikomas efektyviu sprendimu išvengti web sistemų butelio kakliuko bei sumažinti serverio apkrovą [7].

Aiškumo, kas yra kešavimas, galėtų suteikti toks kasdieniškas pavyzdys:

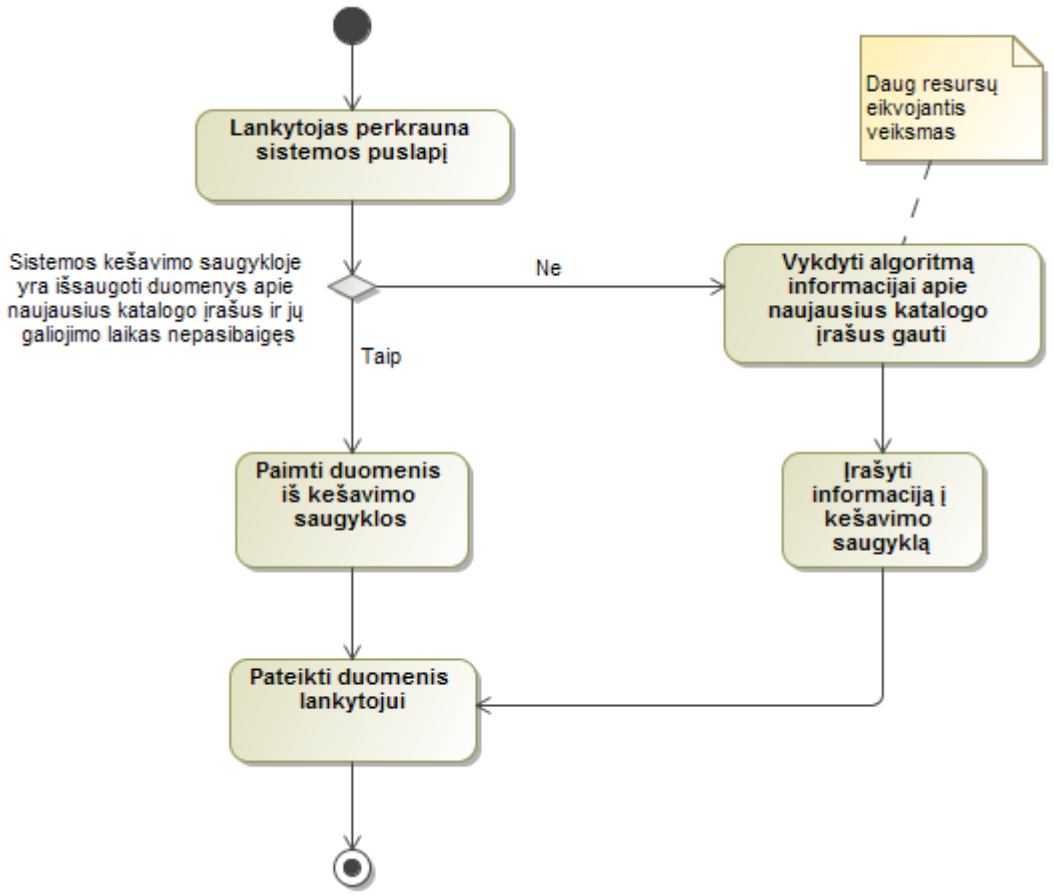
Įvaizduokite, kad jums paskambina draugas ir paprašo sužinoti, kokia šios dienos „Lietuvos ryto“ antraštė. Jūs nueinate iki artimiausio spaudos kiosko, grįžtate namo ir pranešate draugui antraštę. Tada jums paskambina antras draugas ir paprašo to paties. Ar vėl eisite iki kiosko? Be abejo, kad ne. Jūs atsimenate antraštę ir dar bent kurį laiką saugosite ją savo atmintyje. Deja, kompiuteris nėra toks protingas ir turbūt labai dažnai „laksto į spaudos kioską“ (tarkim, duomenų bazę).

Interneto projektuose galima rasti labai daug analogiškų situacijų aukščiau aprašytam pavyzdžiui. Tarkime, vartotojui atėjus į sistemą, jam pateikiami naujausi katalogo įrašai. Kad įrašų sąrašas būtų išvestas į ekraną, turėjo būti atlikta viena ar keletas užklausų į DB, įvykdyti PHP algoritmai. Vartotojui toliau vaikštant po sistemą, kaskart vis iškviečiama ta pati funkcija, naudojanti serverio resursus. Įsivaizduokime kas būtų, jei vienu metu prisijungtu keli šimtai ar

tūkstančiai vartotojų ir visiems jiems reikėtų parodyti tą patį sąrašą – būtų visiškai nenaudingai eikvojami serverio resursai.

Vertėtų pagalvoti apie tokios informaciją kešavimą – pavyzdžiui, įrašyti naujausių katalogo įrašų sąrašą į kešavimo atmintį 30 minučių. Ir vietoj to, kad būtų kviečiama ta pati funkcija vėl ir vėl, į ekraną tiesiog būtų išvedami duomenys, paimti iš kešavimo saugyklos, o pasibaigus šių duomenų galiojimo laikui, rezultatai būtų vėl sugeneruojami ir užkešuojami.

Šios situacijos algoritmą grafiškai vaizduoja 16 paveikslėlis (algoritmas nubraižytas pagal Abhijit Gadkari laikino kešavimo grafinį modelį [17]):



Pav. 16. Naujausių katalogo įrašų sąrašo kešavimo algoritmo veiklos diagrama

Taigi kešavimo procesas yra ganėtinai paprastas: serveris, gavęs užklausą pateikti naujausių katalogo įrašų sąrašą, iš pradžių patikrina, ar tokios užklausos duomenys neegzistuoja kešavimo saugykloje ir jų galiojimo laikas nėra pasibaigęs. Jeigu nėra arba galiojimo laikas pasibaigęs, tuomet jis vykdo algoritmą ir sugeneruoja naują naujausių katalogo įrašų sąrašą. Šiuos duomenis pateikia vartotojui ir išsaugo kešavimo saugykloje nustatytam laikui. Kitą kartą gavęs tokią pačią užklausą, patikrina ar saugykloje saugomų duomenų galiojimo laikas dar nepasibaigęs, ir, jeigu ne,

pateikia duomenis vartotojui tiesiai iš saugyklos. Pasibaigus duomenų galiojimo laikui, gavęs užklausą duomenis vėl sugeneruoja ir išsaugo kešavimo saugykloje.

Tačiau ką daryti, jei norima, kad užregistravus naują įmonę kataloge, naujausių katalogo įrašų sąraše ji atsirastų tuojau pat ir nereikėtų laukti 30 min.? Tokiu atveju galima kešuoti nenurodant galiojimo laiko, kitaip tariant, padaryti jį begalinį. O vartotojui užregistravus naują įmonę, šiuos kešuotus duomenis ištrinti. Toks būdas leidžia turėti labiau dinamiškus duomenis, tačiau reikalauja didesnių programavimo pastangų. Plačiau apie tai bus kalbama „Kešuotų duomenų invalidacijos“ skyrelyje.

Kešuojant dažniausiai naudojamos operacijos pateiktos 5 lentelėje.

Lentelė 5. Dažniausiai pasitaikančios operacijos kešavime

Operacija	Dažniausiai pasitaikantys angliški funkcijų pavadinimai
Išsaugoti duomenis į kešą, nurodant raktą ir galiojimo laiką:	add($key, $time), store($key, $time)
Gauti duomenis iš kešo, nurodant raktą	fetch($key), get($key)
Ištrinti duomenis iš kešo, nurodant raktą	delete($key)
Ištrinti visus užkešuotus duomenis	clear(), clear_cache(), flush()

Kešavimo taikymas praktikoje

Jei kešavimo operacijų funkcijų sintaksė yra pakankamai paprasta, tai nuspręsti ką, kaip ir kada kešuoti yra daug sudėtingiau. Kešuojant, susiduriama su tokiais klausimais ir problemomis:

- kada reikėtų taikyti kešavimą;
- ką kešuoti;
- duomenų įrašymo į atmintį kaina;
- duomenų paėmimo iš atminties kaina;
- invalidacija (kešuotų duomenų atnaujinimas)

Kada ir ką kešuoti

Kešuoti apsimoka tik tuomet, kai nauda dėl greitesnio duomenų gavimo viršija tikrinimo, ar duomenys nepasenę, ir kešuotų duomenų nuskaitymo sąnaudas. Kešuojami yra tie duomenys, apie kuriuos yra žinoma, kad jie nesikeis labai dažnai. Dažniausiai kešuojamos yra tos operacijos, kurios vykdomos dažnai, o gaunami duomenys keičiasi rečiau. Taip pat tokios, kurioms atlikti reikia daug resursų:

- operacijos su duomenų baze;
- failų nuskaitymas;
- dideli skaičiavimų algortimai;
- operacijos, susijusios su darbu su lėtais resursais ir kt.

Yra žinoma, kad pagrindinis „butelio kakliukas“ daugelyje PHP aplikacijų yra DB. Todėl kešuojant užklausoms į DB turi būti skirtas ypatingas dėmesys.

Ko negalima kešuoti

Dažname projekte yra tokių dalių, kurias užkešuoti būtų klaida. Dažniausiai tai susiję su vartotojais ir jų duomenimis. Pvz., kiekvienam prisijungusiam varotojui turėtų būti parodomi tik jam vienam būdingi jo sąskaitos duomenys. Ypač vengtina kešuoti duomenis susijusius su asmens privatumu - pavyzdžiui, banko kortelės duomenys.

Kešuotų duomenų invalidacija

Kešuotų duomenų invalidacija – tai logikos valdymas, kuomet sprendžiama, kada pasenusius kešuotus duomenis reikia pakeisti naujais. Tai yra bene didžiausia kešavimo problema ir nuo kešuotų duomenų invalidacijos priklausys, ar kešavimas atneš projektui naudos, ar pridarys žalos. Tarp programuotojų yra gerai žinoma Filo Karltono (Phil Karlton) frazė:

„Programavime yra tik du sunkūs dalykai: pavadinimų kintamiesiems, klasėms sugalvojimas ir kešuotų duomenų invalidacija“ (angl. „There are only two hard things in Computer Science: cache invalidation and naming things“).

Skiriami du kešuotų duomenų invalidacijos tipai:

1. Kešuojama tam tikram nustatytui laikui

- pavyzdžiui, kešuojama dešimčiai minučių, kurioms praėjus kešuoti duomenys ištrinami ir vietoj jųišsaugomi nauji;
- negalime būti tikri, kad iš atminties paimami duomenys bus teisingi;
- tinka naudoti, kur šiek tiek pasenusi informacija nėra didelė bėda.

2. Kešuojama nenurodant galiojimo laiko (amžinai), bet kešuoti duomenys ištrinami įvykus nurodytiems įvykiams

- pavyzdys – įrašyti naujausius katalogo įrašus į atmintį visam laikui. Kai vartotojas užregistruoja naują įmonę kataloge, ištrinti kešuotus duomenis ir sugeneruoti naujausių katalogo įrašų sąrašą iš naujo;
- naudojama, kai duomenys yra naudojami dažniau, nei keičiasi; kai yra svarbu, kad duomenys būtų teisingi;
- reikalauja didesnių programavimo sąnaudų nei kešuojant nustatytam laikui.

4.1.2 Kešavimo technologijų skirstymas

Nors apibrėždami kešavimą iki šiol daugiausia minėjome, kad jis leidžia išvengti serverio ir duomenų bazės etapo užklausos procese, iš tiesų kešavimas apima platesnę sritį ir yra kešavimo būdų, mažiau susijusių su serverio procesu. Web kešavimo technologijų skirstymas sufomuluotas ir pateiktas 6 lentelėje:

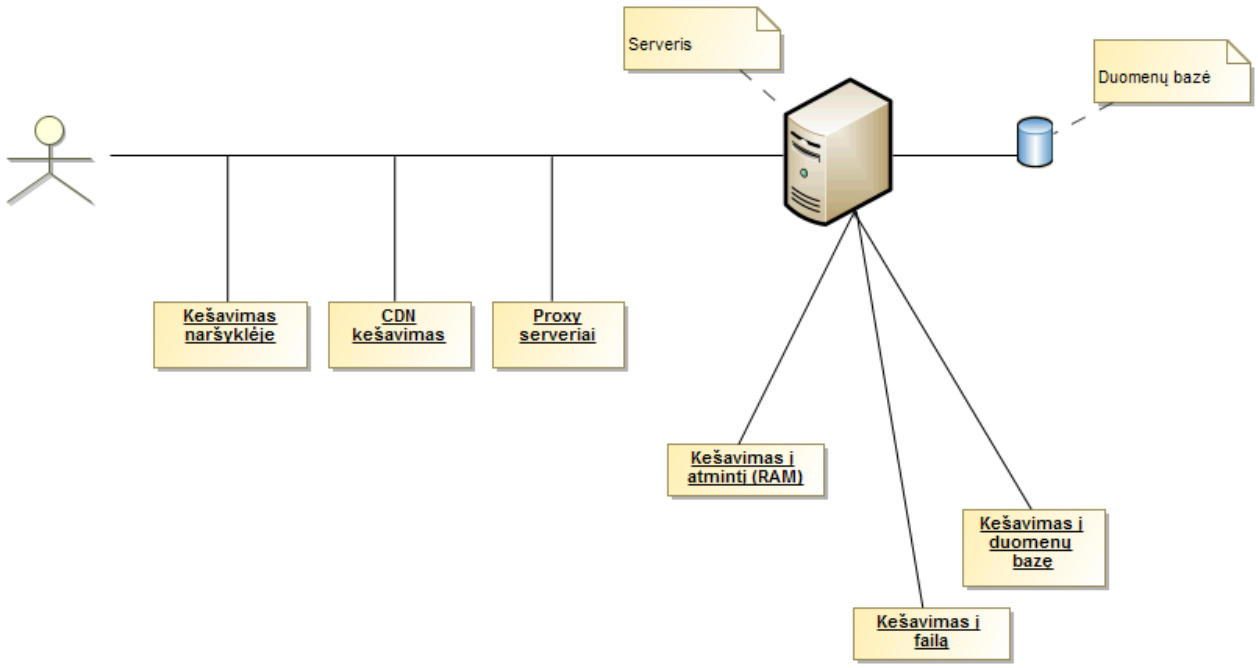
Lentelė 6. Kešavimo technologijų skirstymas

Pagal tai, kur saugomi kešuojami duomenys	**Pagal kešavimo apimtį**	**Pagal kešuojamų duomenų pobūdį**
• kešavimas į failą; • kešavimas į atmintį (RAM); • kešavimas į duomenų bazę; • proxy serveriai;	• reikšmės; • puslapio.	• aplikacijos kodo; • duomenų bazės užklausų; • opkodo.

• kešavimas tinklo duomenų saugyklose (Content Network Delivery, toliau - CDN); • kešavimas naršyklėje.		

Kešavimo technologijos pagal duomenų saugojimo vietą

17 paveikslėlis parodo, kurioje vietoje tarp vartotojo ir serverio yra pagrindiniai kešavimo būdai. Kuo arčiau vartotojo (užklausos iniciatoriaus) yra kešavimo technologija, tuo greičiau ji veikia, tačiau tuo mažiau programuotojui lieka lankstumo ją kontroliuoti.



Pav. 17. Kešavimo technologijos pagal nuotolį nuo vartotojų

Kešavimas naršyklėje, CDN kešavimas ir proxy serveriai yra greičiausi būdai užkešuotiems duomenims pasiekti. Tačiau juos galima laikyti tik kaip papildomus būdus greitesniam puslapio veikimui pasiekti. Pavyzdžiui, programuotojas beveik negali įtakoti naršyklės kešavimo. CDN kešavimas daugiausia skirtas tik vaizdo dalies užsikrovimo laikui paspartinti.

Kešavimas į failą

Naudojant šį metodą duomenys dažniausiai yra serializuojami ir išsaugojami į failą kataloge, skirtame kešuotiems failams laikyti. Paprasčiausią PHP klasę, aprašančią metodus įrašymo, nuskaitymo ir trynimo operacijas iš failo, nesunku susikurti pačiam panaudojant fopen(), fwrite(), file_get_contents(), serialize(), unserialize(), unlink() ir kitas standartines PHP funkcijas.Dažnai naudojamos ir PHP išeigos kontrolės funkcijos. Plačiau jos detalizuojamos 7.4 priede.

Įrašymo į failą kešavimo būdas patogus tuo, kad nereikalauja jokių papildomų PHP plėtinių, todėl gali būti naudojamas „shared“ serveriuose, kurių administravimo galimybės yra ribotos. Tai pakankamai nesudėtingas būdas kešuoti ir nors veikia ne taip greitai kaip kešavimas į atmintį, tačiau vis tiek ženkliai padidina sistemos veikimo greitį palyginus su atveju, jei kešavimas išvis nebūtų taikomas.

Kešavimas į atmintį

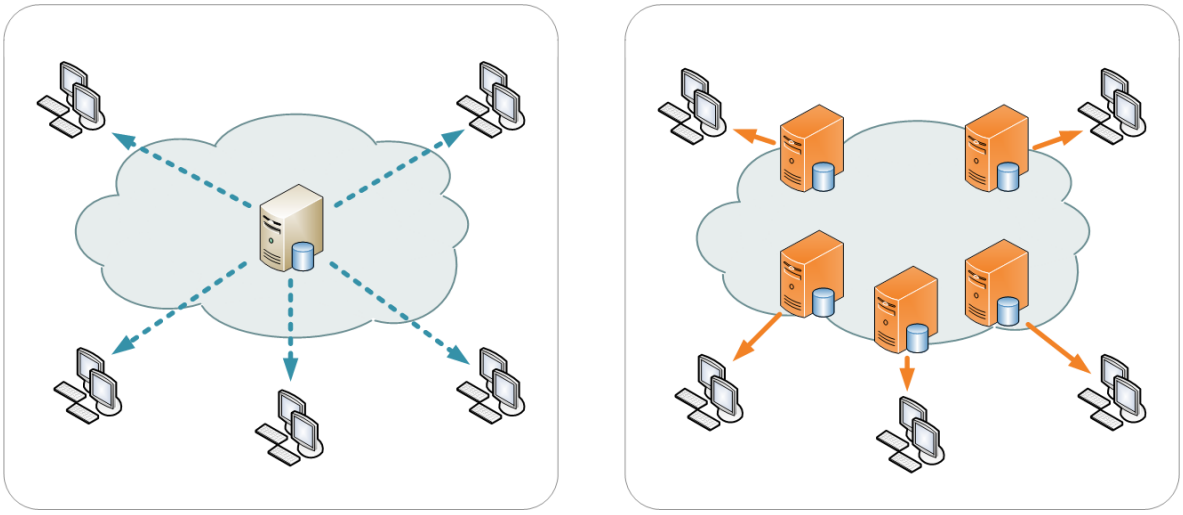
Kešavimas į atmintį yra pats greičiausias dinaminis būdas kešuoti duomenis [103]. Tačiau šis būdas nėra toks prieinamas kaip kešavimas į failą. RAM vis dar yra gerokai brangesnis atminties tipas nei paprastas diskas. Dėl to dažnai naudojantkešavimo į atmintį technologiją tenka bandyti išsiversti su ribotu atminties kiekiu. Šis būdas turėtų būti naudojamas dinaminėse sistemose kešuojant labiausiai kintančius duomenis, kurių gavimas eikvoja daug serverio resursų.

Kešavimas į duomenų bazę

Nors operacijos su DB PHP sistemose yra didžiausias „butelio kakliukas“, egzistuoja kešavimo būdas, kuomet kešuojami duomenys yra rašomi į tą pačią DB. Gauti užkešuotus duomenis per vieną užklausą iš DB vis tik yra geriau nei visai nekešuoti ir atlikinėti daug tokių užklausų į DB. Tačiau šis būdas yra mažai naudojamas. Bet yra atvejų kada jis gali būti naudingas, pvz. saugoti sesijos, ar kitą svarbią informaciją, kurią laikyti faile būtų per daug nesaugu.

CDN (angl. Content delivery network) kešavimas

Užklausą pateikiančio vartotojo nutolimas nuo serverio įtakoja užklausos įvykdymo greitį. Paskirstant sistemos duomenis tarp keleto geografiškai išsisklaidžiusių serverių galima sumažinti užklausos įvykdymo laiką [3]. Tai ypač aktualu sistemose, kuria naudojasi vartotojai esantys įvairiose pasaulio vietose. CDN koncepciją puikiai atspindi 18 paveikslėlis:



Pav. 18. Duomenų paskirstymas vartotojams be ir su CDN

CDN kešavimo paslaugas dažniausiai už mėnesinį mokestį teikia privatūs tiekėjai, turintys serverius įvairiose pasaulio vietose. Užsisakius paslaugą papildomo darbo atlikti praktiškai nereikia - tiekėjo technologija pati atrenka kuriuos failus kešuoti priklausomai nuo to kaip dažnai jie naudojami.

CDN teikia ir papildomų privalumų: sistema gali išlikti prieinama internete net jei pagrindinis sistemos serveris nustojo veikti, blokuojami nepageidaujami vartotojai ir kt.

Proxy serveriai. Šie serveriai veikia kaip tarpininkai tarp http užklausos iniciatoriaus ir pagrindinio sistemos serverio. Jie išsaugo kešuotas tinklalapių kopijas ir atėjus atitinkamai http užklausai, ją nutraukia ir pateikia kešuotą failą. Proxy serveriai dažniausiai būna arčiau vartotojo nei pagrindinis sistemos serveris, o tai taip pat padeda sumažinti užklausos įvykdymo laiką [20]. Naudojant šį kešavimo būdą resursų sunaudojimo sumažėjimas gali siekti iki 80 proc. [1].

Kešavimas į failą ir virtualiąją atmintį yra vieni pagrindinių kešavimo būdų, kuriems skirsime daugiausia dėmesio šiame darbe, todėl apibendrinkime ir palyginkime šių dviejų technologijų savybes.

Lentelė 7. Į failą ir atmintį duomenis saugančių kešavimo technologijų apibendrinimas

Vieta, kur saugomi kešuojami duomenys	**Failas**	**Atmintis**
Duomenų įrašymo, paėmimo, trynimo greitis	Vidutinis	Labai greitas

Privalumai	Pigus būdas, nereikalingos jokios išorinės sistemos	Duomenys įrašomi, paimami ir trinami labai greitai
Trūkumai	Nėra pats grečiausias būdas	Brangus būdas
Kur naudotinas	Shared serveriuose, mažuose projektuose, kuriuose informacija keičiasi retai	Dinaminėse sistemose kešuojant labiausiai kintančius ir serverio resursus eikvojančius duomenis.

Kešavimo technologijos pagal kešavimo apimtį

Reikšmės kešavimas

Reikšmės kešavimo technologija leidžia į atmintį įrašyti pasirinktą informaciją, o ne visą puslapį. Dažniausiai tai būna nedidelės apimties informacija – kintamieji, masyvai, objektai – kuriems gauti reikia išnaudoti daug serverio resursų. Praverčia saugant konfiguracijos kintamuosius.

Šis kešavimo būdas yra struktūrizuotas ir pagrįstas rakto/reikšmės saugojimo principu. Saugomus duomenis kešavimo saugykloje galėtume įsivaizduoti kaip vieną didelį asociatyvų masyvą, kuriame kiekvienas masyvo elementas (saugomų duomenų grupė) yra indeksuotas tam tikra simbolių eilute (raktu). Raktas yra naudojamas saugant, paimant ar trinant duomenis iš kešo.

Privalumai: visas kešavimo procesas yra programuotojo rankose. Jis gali nuspręsti ką ir kada kešuoti, kada ištrinti kešą.

Trūkumai: didelis imlumas darbui, praktiškai kiekvieną metodą reikia sukontroliuoti kaip vykdomas jo kešavimas

Šio tipo kešavimą atlieka šios bibliotekos: APC, XCache, Memcache, Wincache ir kt.

Viso puslapio į html failą kešavimas

Ši metodika puikiai tinka tuomet, kai visas puslapio turinys keičiasi gana retai, ir nėra atskirų jo dalių kurios neturėtų būti kešuojamos. Jei visi puslapio lankytojai ilgą laiką mato tą patį puslapį, tuomet tikslinga tiesiog jį visą užkešuoti.

Privalumai: paprastas naudoti ir labai efektyvus.

Trukumai: nelankstus būdas – norint, kad atskira puslapio dalis liktų dinaminė, susiduriama su problemomis, netinka dinamiškiems interaktyviems puslapiams, reaguojantiems į vartotojų veiksmus.

Kešavimo technologijos pagal kešuojamų duomenų pobūdį

Aplikacijos kodo kešavimas

Šis būdas apima didžiąją dalį aukščiau aprašytų kešavimo priemonių. Tai būdas kuomet į atmintį yra įrašomi aplikacijos algoritmų vykdymo metu gauti duomenys kodo pavidalu.

Duomenų bazės užklausų kešavimas

Tai būdas kuomet kešuojami tiktai užklausų į duomenų bazę metu gauti duomenys. Jei kelis kartus įvykdoma visiškai identiška užklausa ir yra įjungtas duomenų bazės užklausų kešavimas, gauti rezultatai bus įrašomi į atmintį ir, kitą kartą inicijuojant tą pačią užklausą, ji nebebus vykdoma, o tik paimami duomenys iš atminties. Tačiau šis metodas turi keletą apribojimų: negali būti kešuojamos sub-užklausos, užklausose negali būti naudojami kintamieji ir kt.

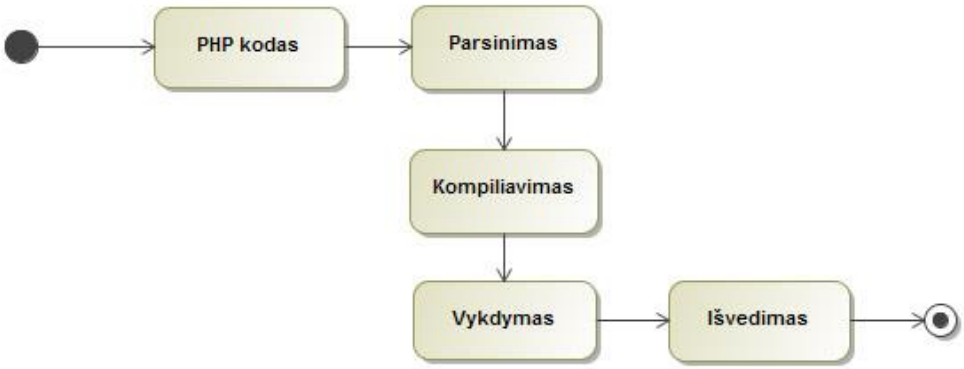
Kaip ir kiti kešavimo būdai, ši technologija daro prielaidą, kad bent tam tikrą dalį laiko informacija sistemoje nesikeis, taigi bus nuolatos vykdomos tospačios užklausos.

Opkodo kešavimas

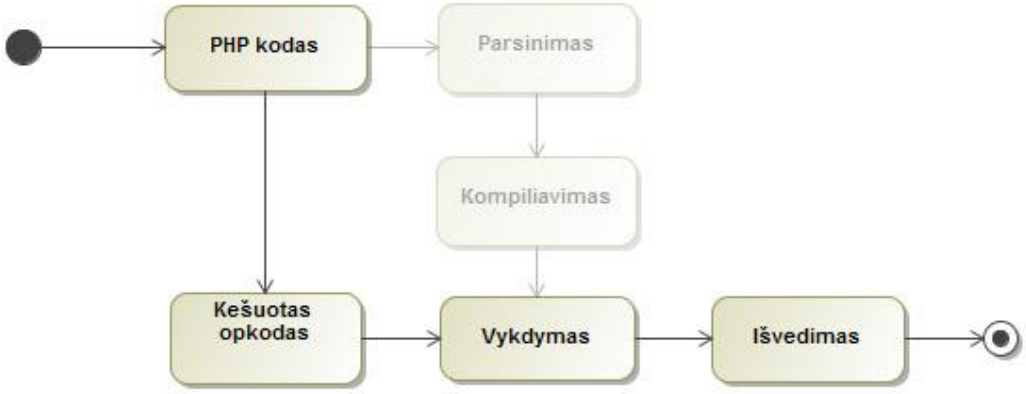
Programuojant su PHP kešavimas yra ypač svarbus. PHP yra interpretuojama aukšto lygio programavimo kalba, o tai reiškia, kad vykdant kiekvieną užklausą, visi PHP failai iš pradžių turi būti sukompiliuojami į mašinoms suprantamą kodą – vadinamąjį opkodą, ir tik tada gali būti vykdomi mašinų, kitaip sakant - serverio [18]. Kešavimo technologijos, kurios išsaugo sukompiliuotą opkodą atmintyje, padeda išvengti pakartotinio kompiliavimo etapo ir taip labai pagreitina užklausos įvykdymą. Be to, opkodas yra saugomas atmintyje (RAM), todėl jo nuskaitymas yra daug greitesnis nei iš disko.

Logiška, kad šiandien didžioji dalis programuotojų dirba ne su žemo lygio programavimo kalbomis, kuriose, kad gauti tą patį rezultatą, kodo reikia parašyti keletą kartų daugiau nei dirbant su aukšto lygio programavimo kalbomis. Todėl galimybė aukšto lygio parašytą kodą užkešuoti mašinoms suprantamo opkodo pavidalu labai pagreitina skriptų vykdymo greitį.

19 ir 20 paveikslėliai iliustruoja kaip įprastai vykdomas PHP kodas ir kaip PHP kodas vykdomas naudojant opkodo kešavimą.



Pav. 19. Įprastas PHP kodo vykdymo algoritmas



Pav. 20. PHP kodo vykdymo algoritmas naudojant opkodo kešavimą

Kaip matome, opkodo kešavimas leidžia išvengti PHP kodo parsinimo ir kompiliavimo etapų.

Opkodą kešuoja šios kešavimo bibliotekos: APC, XCache, Wincache ir kt. Plačiau opkodas ir kiti kodo tipai aprašyti 8.2 priede „Kodo tipai“. Taip pat opkodo veikimas bus plačiau analizuojamas aprašant APC kešavimo biblioteką.

Http užklausų mažinimas - dar vienas būdas pagreitinti sistemos veikimą

Yahoo! kūrėjai teigia, kad net iki 80% užklausos laiko, per kurį vartotojas laukia, kol užsikraus puslapis, sudaro ne serverio, o vaizdo dalies užkrovimo laikas: tai paveikslėliai, css ir javascript failai [19]. Sumažinus kiekvieno iš šių komponentų skaičių, mažėja ir http užklausų skaičius. O kiekvienos http užklausos panaikinimas reiškia greitesnį puslapio užsikrovimo laiką.

Keletas būdų, sumažinti http užklausų skaičių:

- paveikslėliams naudoti „Sprite“ techniką, kuomet visi dizaine naudojami paveiksliukai sumaketuojami į vieną paveikslėlį, o vėliau html elementams reikiamas paveikslėlis priskiriams naudojant CSS atributą „background-position“;
- minimizuoti CSS ir javascript failus, pašalinant komentarus, tarpų ir naujų eilučių simbolius, bei visus failus sudedant į vieną;
- naudoti aukščiau aprašytą CDN kešavimą.

4.1.3 Esamų kešavimo technologijų lyginamoji analizė

Kaip ir daugelyje programavimo sričių, taip ir kešavime yra sukurta įvairių klasių, bibliotekų ir plėtinių, padedančių įgyvendinti kešavimą. Internete galima rasti nuo paprasčiausių klasių ar sudėtingesnių bibliotekų (pvz. „Cache Lite“), aprašančių kešavimo į failą metodus, iki PHP bibliotekų, vykdančių duomenų saugojimą į RAM atmintį ir opkodo kešavimą. Toliau bus analizuojami pastarosios, kadangi jos, bent jau teoriškai, leidžia pasiekti geriausių kešavimo rezultatų.

Programuojant su PHP dažniausiai naudojamos kešavimo bibliotekos: APC, XCache, Memcache. Taip pat populiarios Zend Platform (dirbantiems su Zend karkasu) ir Windows cache.

Lentelė 8. Dažniausiai PHP naudojamos kešavimo bibliotekos

Bibliotekos pavadinimas	**APC**	**XCache**	**Memcached**
Naujausia versija ir jos paleidimo data	3.1.13 2012.09.03	3.0.1 2013.01.11	1.4.15 2012.09.03
Oficiali svetainė:	http://pecl.php.net/package/APC	http://xcache.lighttpd.net	http://memcached.org/

APC

APC – tai angliško termino „Alternative PHP Cache“ (alternatyvus PHP kešavimas) trumpinys [5]. APC bibliotekos kūrėjai – ta pati kompanija, kuri kuria ir PHP programavimo kalbą. Vis dažniau APC yra laikoma kaip de facto standartine PHP kešavimo biblioteka, o nuo PHP 6 versijos planuojama, kad ji bus įtraukta standartinį PHP paketą, t.y. nebereikės jos įrašinėti kaip

plėtinio. Šiuo metu APC yra stabili, aktyviai palaikoma kešavimo biblioteka. APC yra PECL plėtinys ir jį yra palyginti nesunku įdiegti į serverį. APC turi opkodo kešavimo funkciją.

Kada naudoti APC?

Geriausius rezultatus APC duoda kešuojant nedidelės apimties sistemos kodo dalis, kurios kinta dažnai. Tačiau žinoma galima APC naudoti ir kitais atvejais. Svarbu tik kad užtektų laikinosios atminties kešuojamų duomenų saugojimui.

APC funkcijos

APC funkcijų aprašymas pateiktas 8.3 priede „Kešavimo bibliotekų funkcijų aprašymas“.

Toliau pateiktas 16 paveikslėlyje pavaizduoto algoritmo, kuomet kešuojamas naujausių katalogo įrašų sąrašas, kešavimo pavyzdys, naudojant APC funkcijas:

```
$newCatalogs = apc_fetch('newCatalogs');
if(!$newCatalogs){
    $newCatalogs = $this->Catalog->get('new');
    apc_add('newCatalogs', $newCatalogs, 1800);
}
$this->set('newCatalogs', $newCatalogs);
```

Šiame pavyzdyje iš pradžių bandoma paimti naujausių katalogų sąrašo masyvą su apc_fetch() funkcija iš atminties. Jei atmintyje jo nerandama, vykdoma Catalog modelio get() funkcija ir sugeneruojamas naujausių katalogo įrašų masyvas. Tuomet su apc_add() funkcija jis įrašomas į atminti 1800 sek., kitaip tariant, 30 min. Galiausiai masyvas paruošiamas išvedimui vaizdo faile su funkcija set().

APC opkodo kešavimas

Įdiegus APC, opkodo kešavimas būna įjungtas automatiškai. Norint jį išjungti, php.ini konfiguracijos faile reikėtų pakeisti nustatymo apc.cache_by_default reikšmę į nulį. Funkcija apc_cache_info() leidžia sužinoti kokia informacija yra išsaugota kaip opkodas. Pavyzdžiui, išjungus APC opkodo kešavimą, apc_cache_info() funkcija gražina tokią informaciją įmonių katalogo IS namų puslapyje:

```
Array
(
    [num_slots] => 1031
    [ttl] => 0
    [num_hits] => 0
    [num_misses] => 0
    [num_inserts] => 0
    [expunges] => 0
    [start_time] => 1365411557
    [mem_size] => 0
    [num_entries] => 0
    [file_upload_progress] => 1
    [memory_type] => IPC shared
    [locking_type] => file
    [cache_list] => Array
        (
        )

    [deleted_list] => Array
        (
        )
)
```

Įjungus opkodo kešavimą, funkcija apc_cache_info() pateikia tokį įmonių katalogo IS namų puslapio užkešuotų duomenų masyvą (pateikta tik masyvo pradžia):

```
Array
(
    [num_slots] => 1031
    [ttl] => 0
    [num_hits] => 140
    [num_misses] => 113
    [num_inserts] => 113
    [expunges] => 0
    [start_time] => 1365412146
    [mem_size] => 16229016
    [num_entries] => 111
    [file_upload_progress] => 1
    [memory_type] => IPC shared
    [locking_type] => file
    [cache_list] => Array
        (
            [0] => Array
                (
```

```
        [filename] => E:\ wamp\www\cat\app\Model\Action.php
        [device] => 1954457318
        [inode] => 35156
        [type] => file
        [num_hits] => 0
        [mtime] => 1363968589
        [creation_time] => 1365412175
        [deletion_time] => 0
        [access_time] => 1365412175
        [ref_count] => 0
        [mem_size] => 33032
    )

[1] => Array
    (
        [filename]=>E:\ wamp\www\cat\lib\Cake\Configure\PhpReader.php
        [device] => 1954457318
        [inode] => 39248
        [type] => file
        [num_hits] => 2
        [mtime] => 1362244990
        [creation_time] => 1365412167
        [deletion_time] => 0
        [access_time] => 1365412183
        [ref_count] => 1
        [mem_size] => 33032
    )

[2] => Array
    (
        [filename]=>E:\wamp\www\cat\lib\Cake\Configure\ConfigReaderInterface.php
        [device] => 1954457318
        [inode] => 39246
        [type] => file
        [num_hits] => 2
        [mtime] => 1362244990
        [creation_time] => 1365412167
        [deletion_time] => 0
        [access_time] => 1365412183
        [ref_count] => 1
        [mem_size] => 8360
    )

[3] => Array
    (
```

```
        [filename]=>E:\ wamp\www\cat\app\View\Helper\AppHelper.php
        [device] => 1954457318
        [inode] => 53670
        [type] => file
        [num_hits] => 0
        [mtime] => 1364324030
        [creation_time] => 1365412175
        [deletion_time] => 0
        [access_time] => 1365412175
        [ref_count] => 0
        [mem_size] => 28936
    )
...
```

Kaip matome, masyvo „cache_list“ elemente parodomi užkešuoti sistemos PHP failai: nuo karkaso „core“ klasių iki pačios sistemos klasių. Taigi kraunant sistemos puslapį, PHP moduliui jau nebereikia kiekvieną kartą nuskaityti šių failų iš disko, vykdomas opkodas esantis RAM atmintyje. Tačiau prieš vykdant opkodą, APC kiekvieną patikrina, ar užkešuotų failų originaluose nėra pasikeitimų. Ir, jeigu jų yra, perkompiliuoja opkodą. Toks tikrinimas, žinoma, kainuoja šiek tiek laiko, todėl jei yra žinoma, kad sistemos kodas ilgą laiką nesikeis, šį tikrinimą galime išjungti su php.ini parametru apc.stat = 0.

XCache

XCache - atviro kodo kešavimo biblioteka, skirta PHP serveriams. Kaip rašoma oficialiame puslapyje, XCache leidžia sumažinti puslapių generavimo laiką iki 5 kartų [6]. Nemažai šaltinių, atlikusių kešavimo testus, teigia, jog XCache veikia greičiau nei APC. Kaip ir APC, turi opkodo kešavimo funkciją.

XCache funkcijos

XCache funkcijų aprašymas pateiktas 8.3 priede „Kešavimo bibliotekų funkcijų aprašymas“.

XCache funkcijų naudojimo pavyzdys užrašant 16 paveikslėlyje pavaizduotą naujausių katalogo įrašų kešavimo algortimą:

```
$newCatalogs = xcache_get('newCatalogs');
if(!$newCatalogs){
    $newCatalogs = $this->Catalog->get('new');
```

```
        xcache_set('newCatalogs', $newCatalogs, 1800);
    }
    $this->set('newCatalogs', $newCatalogs);
```

XCache opkodo kešavimas

Kaip ir APC, XCache turi opkodo kešavimo funkcija, tačiau instaliavus XCache plėtinį ši funkcija būna išjungta. Norint ją įjungti, php.ini konfiguracijos faile reikia įrašyti nustatymą xcache.cacher = 1.

Memcached

Memcached - dar vienas kešavimo būdas, kuomet duomenys yra saugojami virtualioje atmintyje (RAM), ir iš kurios vėliau paimti duomenis yra daug greičiau nei kviečiant skaičiavimo funkcijas [22]. Skirtingai nuo APC ar XCache, tai visiškai atskiras savarankiškas modulis, veikiantis atskirai nuo PHP serverio. Memcached turi pritaikytus klientus visoms pagrindinėms programavimo kalboms: C, C++, Java, PHP, Python, Ruby ir kt.

Naudojant Memcached, informacija, suskirstyta pagal atitinkamus raktus, užkešuojama ir saugoma RAM atmintyje serializuotu pavidalu. Serverio perkrovimas ar nulūžimas išvalo RAM atmintį, taigi dingta ir visa iki tol užkešuota informacija.

Kada naudoti Memcached?

Memcached naudojimas šiek tiek apkrauna pačią sistemą, todėl šią technologiją tikslinga naudoti tik didelėse sitemose, kuomet reikia kešuoti didelius objektus, reikalaujančius daug atminties. Memcached įdiegimas nedidelėje sistemoje neatneš pastebimų teigiamų rezultatų.

Memcached atminties kiekį riboja tik sistemos, kurioje modulis veikia, atminties kiekis. Kadangi Memcached gali būti naudojamas su įvairiomis programavimo kalbomis, jis ypač naudingas sistemose, kurios sukurtos panaudojant kelias programavimo kalbas. Be to, Memcached naudoja paskirstyto kešavimo metodiką, kuri įgalina panaudoti užkešuotus duomenis keliose aplikacijose arba aplikacijoje, išdėstytoje skirtinguose serveriuose [8]. Tokiu būdu tie patys kešuoti duomenys bus pasiekiami užklausoms iš įvairių serverių, taigi nereikės jų duplikuoti, gerokai sumažės programavimo ir atminties resursų sąnaudos.

Memchace ir Memcached

PHP turi dvi atskiras Memcached modulio realizacijas labai panašiais pavadinimais, kurie klaidina pradedančius dirbti su šiuo plėtiniu: Memcache ir Memcached. Tai vienas nuo kito nepriklausomi moduliai, iš kurių pirmasis yra senesnis ir laikomas kaip labiau patikimas, o antrasis yra naujesnis PHP plėtinys, turintys tokias pačias funkcijas kaip ir Memcache bei keletą naujų, tačiau nėra visiškai stabilus. Memcached modulis yra paremtas ne Memcached, o libMemcached biblioteka.

Lentelė 9. PHP plėtinių Memcache ir Memcached palyginimas

Plėtinys	**PECL/Memcache**	**PECL/Memcached**
Pirmo pasirodymo data	2004.06.08	2009.01.29 (beta)
Aktyviai palaikomas?	Taip	Taip
Automatinė raktų korekcija	Taip	Ne
Append/Prepend	Ne	Taip
Automatinė serializacija	Taip	Taip
Binarinis protokolas	Ne	Pasirenkamas
CAS	Ne	Taip
Kompresija	Taip	Taip
Communication timeout	Connect only	Various options
Consistent hashing	Taip	Taip
Atidėti gavimai	Ne	Taip
Multi-gavimai	Taip	Taip
Sesijos palaikymas	Taip	Taip
Set / Get specifiniam serveriui	Ne	Taip
Saugo skaitiniu pavidalu	Konvertuoja į eilutę	Taip

Memcached funkcijos

Memcached funkcijų aprašymas pateiktas 8.3 priede „Kešavimo bibliotekų funkcijų aprašymas“.

Memcache funkcijų panaudojimo pavyzdys užrašant 16 paveikslėlyje pavaizduotą naujausių katalogo įrašų sąrašo kešavimo algortimą:

```
$memcache = new Memcached();
$memcache->connect("localhost", 11211) or die ("Could not connect");

$newCatalogs = $memcache->get('newCatalogs');
if(!$newCatalogs){
    $newCatalogs = $this->Catalog->get('new');
    $memcache->set('newCatalogs', $newCatalogs, 1800);
}
$this->set('newCatalogs', $newCatalogs);
```

Kadangi Memcache yra atskiras sisteminis modulis, pirmiausia turime prie jo prisijungti nurodydami serverį ir porto numerį. Paskui pagal nurodytą raktą su get() funkcija bandome gauti duomenis iš atminties. Jei nepavyksta, vykdome pirminę duomenų gavimo funkciją ir gautus duomenis užkešuojame su funkcija set().

Smarty šablonų variklis

Smarty – šablonų variklis, turintis kešavimo funkciją. Tai vienas populiariausų PHP šablonų variklių, plačiai naudojamas įvairiuose interneto projektuose ir laikomas bene geriausiu iš šiuo metu esančių konkurentų. Smarty pasižymi patikimu veikimu ir dideliu funkcionalumu. Smarty yra vienas seniausių šablonų variklių, skirtų PHP. Trečioje versijoje Smarty visiškai perrašytas pagal PHP5 principus [23].

Smarty bibliotekoje, be programavimo logikos ir vaizdo atskyrimo funkcijos, dar yra šios funkcijos:

- kešavimas;
- šablonų paveldimumas;
- įskiepių architektūra;
- duomenų apdorojimo funkcijos.

Kaip veikia Smarty?

Visa kodo logika aprašoma kontroleryje. Sukūrus Smarty objektą, būtina jam nurodyti kelią iki katalogų, kuriuose bus saugomi vaizdo failai (su tpl plėtiniu) ir jau sukompiliuoti vaizdo failai. Kintamieji, kurie bus išvedinėjami vaizdo failuose, priskiriami vaizdo kintamiesiems su funkcija assign().

```
include('libs/Smarty.class.php');
$smarty = new Smarty;
$smarty->template_dir = "templates/";
$smarty->compile_dir = "templates_c/";

$smarty->assign('page_title', 'Paprastas Smarty puslapis');
$smarty->assign('article_name', 'Pirmojo straipsnio pavadinimas');
$smarty->display('index.tpl');
```

Vaizdo failuose (šablonuose), kurie saugomi su tpl plėtiniu, kontroleryje aprašyti ir priskirti kintamieji yra prieinami juos užrašant riestiniuose skliaustuose. Taip galėtų atrodyti elementarus index.tpl vaizdo šablonas su jam perduotais kintamaisiais:

```
<html>
    <head>
        <title>{$page_title}</title>
    </head>
    <body>
        <h1>{$article_name}</h1>
    </body>
</html>
```

Šablono failus puslapio krovimo metu Smarty sukompiliuoja ir išsaugo kaip paprastą PHP skriptą šablonų failų kopijose nurodytame sukompiliuotų vaizdo failų kataloge. Ši kompiliacija vyksta pirmą kartą kai yra iškviečiamas tpl failas. Nuo antro karto jau yra naudojama sukompiliuota vaizdo šablono kopija. Sukumpiliuotas index.tpl.cache.php failas atrodo taip:

```
<?php /* Smarty version Smarty-3.1.10, created on 2012-06-28 18:30:02
         compiled from "application\views\templates\index.tpl" */ ?>
<?php /*%%SmartyHeaderCode:296784feca2aaaf79d5-
48462719%%*/if(!defined('SMARTY_DIR')) exit('no direct access allowed');
$_valid = $_smarty_tpl->decodeProperties(array (
  'file_dependency' =>
  array (
    '7dd4df7bd0ecb27b6fd234da8271ae822ae86bc1' =>
```

```
    array (
      0 => 'application\\views\\templates\\index.tpl',
      1 => 1340908021,
      2 => 'file',
    ),
  ),
  'nocache_hash' => '296784feca2aaaf79d5-48462719',
  'function' =>
  array (
  ),
  'variables' =>
  array (
    'page_title' => 0,
    'article_name' => 0,
  ),
  'has_nocache_code' => false,
  'version' => 'Smarty-3.1.10',
  'unifunc' => 'content_4feca2aacadce6_98449934',
),false); /*/%%SmartyHeaderCode%%*/?>
<?php if ($_valid && !is_callable('content_4feca2aacadce6_98449934')) {functioncontent_4feca2aacadce6_98449934($_smarty_tpl) {?><html>
    <head>
        <title><?php echo $_smarty_tpl->tpl_vars['page_title']->value;?>
</title>
    </head>
    <body>
        <h1><?php echo $_smarty_tpl->tpl_vars['article_name']->value;?>
</h1>
    </body>
</html>
<?php }} ?>
```

Smarty turi sukompiliuotų šablonų kešavimo į html failus funkciją. Užkešavus, vaizdo failai saugomi jau kaip paruošti failai ir pakraunami be jokių papildomų veiksmų iš PHP pusės – tiesiog paprasti html failai.

Smarty kešavimas

Norint įjungti kešavimą, kreipiamės į setCaching funkciją, taip pat nurodome katalogą, kuriame bus saugomi kešuojami failai.

```
$smarty->setCaching(Smarty::CACHING_LIFETIME_CURRENT);
$smarty->cache_dir = "cache/";
```

Esant tokiems nustatymams, funkcijos display('index.tpl')iškvietimas pirmąjį kartą ne tik sugeneruos šabloną, bet jo kopiją išsaugos nurodytame kešavimo kataloge. Kitą kartą kviečiant tą pačią funkciją, bus pateikta užkešuota šablono kopija.

```
if(!$smarty->is_cached('index.tpl')) {
       $data = $this->get_results();
        $smarty->assign(data, $data);
}
$smarty->display('index.tpl');
```

Kešuotas index.tpl.php failas atrodo taip:

```
<?php /*%%SmartyHeaderCode:296784feca2aaaf79d5-
48462719%%*/if(!defined('SMARTY_DIR')) exit('no direct access allowed');
$_valid = $_smarty_tpl->decodeProperties(array (
  'file_dependency' =>
  array (
    '7dd4df7bd0ecb27b6fd234da8271ae822ae86bc1' =>
    array (
      0 => 'application\\views\\templates\\index.tpl',
      1 => 1340908021,
      2 => 'file',
    ),
  ),
  'nocache_hash' => '296784feca2aaaf79d5-48462719',
  'variables' =>
  array (
    'page_title' => 0,
    'article_name' => 0,
  ),
  'has_nocache_code' => false,
  'version' => 'Smarty-3.1.10',
  'unifunc' => 'content_4feca2aacb81d4_48284598',
  'cache_lifetime' => 3600,
),true); /*/%%SmartyHeaderCode%%*/?>
<?php if ($_valid && !is_callable('content_4feca2aacb81d4_48284598')) {functioncontent_4fec
a2aacb81d4_48284598($_smarty_tpl) {?><html>
    <head>
        <title>Paprastas Smarty puslapis</title>
```

```
    </head>
    <body>
        <h1>Pirmojo straipsnio pavadinimas</h1>
    </body>
</html>
<?php }} ?>
```

Smarty kešavimo funkcijų aprašymas pateiktas 8.3 priede „Kešavimo bibliotekų funkcijų aprašymas“.

Smarty leidžia turėti kelias vaizdo failo kešuotas versijas. Pavyzdžiui, tai gali būti naudojama skirtingoms vartotojų grupėms pateikiant skirtingą kešuotą tiklalapio turinį. Jei funkcijai display() antruoju parametru perduosime rakto reikšmę, tai sugeneruotas kešo failas bus priskirtas šiam raktui ir vėliau pagal jį bus pateikiama būtent ta kešuoto failo versija:

```
$smarty->display('index.tpl', $my_cache_id);
```

Esamų kešavimo priemonių taikymo prielaidos

Apibendrinant esamų kešavimo priemonių analizę galima teigti, kad dinaminiams puslapių duomenims kešuoti geriausia naudoti APC ar XCache bibliotekas, o itin didelėse sistemose, paskirstytose tarp keleto serverių, logiška naudoti ir potencialiai didesnį efektą galėtų duoti Memcache biblioteka. Tačiau neesant poreikio išlaikyti sistemą ypač dinamišką, vertėtų pagalvoti apie viso puslapio į html failą kešavimo technologiją, kadangi ši taip pat gali suteikti labai didelį sistemos pagreitėjimo efektą, be to, nereikalauja jokių papildomų bibliotekų instaliavimo į serverį. Kita vertus, net ir naudojant šią technologiją, APC ar XCache gali būti naudingi dėl opkodo kešavimo funkcijos.

4.2 Įmonių katalogo IS greitaveikos spartinimo eksperimentas

4.2.1 Pasiruošimas eksperimentui

Eksperimente naudojami duomenys

Tam kad ištirti, kurios kešavimo technologijos duoda geriausius greitaveikos rezultatus įmonių katalogo IS, sistemos DB pirmiausia užpildyta netikrų įmonių duomenimis, kad būtų galima įvertinti sistemos veikimą esant dideliam įrašų skaičiui. Sukurtas metodas, kurio pagalba importuota

10.000 įmonių įrašų. Kiekviena įmonė turi skirtingas kategorijų, žymių kombinacijas, skirtingą atsiliepimų skaičių, įvertinimą ir poziciją kataloge.

Bus kešuojami du pagal savo pobūdį visiškai skirtingi sistemos puslapiai. Detalūs kešuojami duomenys pateikti 10 lentelėje:

Lentelė 10. Eksperimento metu kešuojami duomenys

Puslapis	Namų	Įmonės profilio
Aprašymas	Šiame puslapyje atvaizduojami duomenys yra surenkami iš daugelio sistemos modelių, todėl užklausų skaičius į duomenų bazę yra didelis. Tačiau šis puslapis yra pakankamai statinis, beveik nepriklauso nuo vartotojų atliekamų veiksmų;	Šis puslapyje informacijos nėra tiek daug, tačiau jis turi būti dinaminis ir reaguoti į vartotojo atliekamus veiksmus: išsaugoti ir parodyti vartotojo įvestą atsiliepimą, perskaičiuoti ir parodyti naują reitingo balą vartotojui pabalsavus;
Duomenys, kurie paimami iš DB ir kešuojami	• naujausi katalogo įrašai (15) • populiariausi katalogo įrašai (15) • aukščiausiai įvertinti katalogo įrašai (15) • parinktieji katalogo įrašai (15) • naujausi atsiliepimai (15) • naujausi vartotojai (5) • katalogo įrašų skaičius • komentarų skaičius • vartotojų skaičius • paskutiniai vartotoju veiksmai (25) • prisijungę vartotojai (visi) • statiniai tekstai	• visa informacija apie katalogo įrašą (įrašo duomenys, nuotraukos, paslaugų kategorijos, žymės, reitingas, atsiliepimai) • panašūs įrašai (15) • prisijungę vartotojai (visi) • statiniai tekstai (visi)

Eksperimentas bus atliekamas dviem aspektais:

- Užkešuojami puslapiai ir tikrinama, kaip greitai jie užsikrauna. Taip pat įjungiamas ir išjungiamas opkodo kešavimas. Lyginama su puslapio užsikrovimo laiku jei kešavimas nenaudojamas;

- Kiekvienam puslapiui sukuriami metodai kurių metu kiekvienas puslapis yra įrašomas į atmintį ir iškarto ištrinamas. Tokiu būdu, vykdant daug užklausų, bus nustatyta, kaip greitai kiekviena technologija leidžia įrašyti ir ištrinti kešuojamus duomenis;

Eksperimentui pasirinktos kešavimo priemonės

Sistemos greitaveikai spartinti pasirinktos šios kešavimo bibliotekos: APC, XCache ir Memcache. Taip pat bus kešuojama CakePHP karkase įdiegtu kešavimo metodu, kuris leidžia įrašyti kešuojamus duomenis į failą. Taip pat CakePHP įdiegto metodo pagalba bus kešuojami ne tik atskiri objektai, bet ir visas puslapis (viso puslapio kešavimas į html failą), taip paverčiant jį visiškai statiniu.

Tyrimui naudojama įranga

Techninė įranga

- Stalinis kompiuteris Intel Core i-3470 3.2G 6M, Corsair DDR3-1600 2*4GB RAM, Samsung SSD830128GB

Programinė įranga

- Windows 7 operacinė sistema

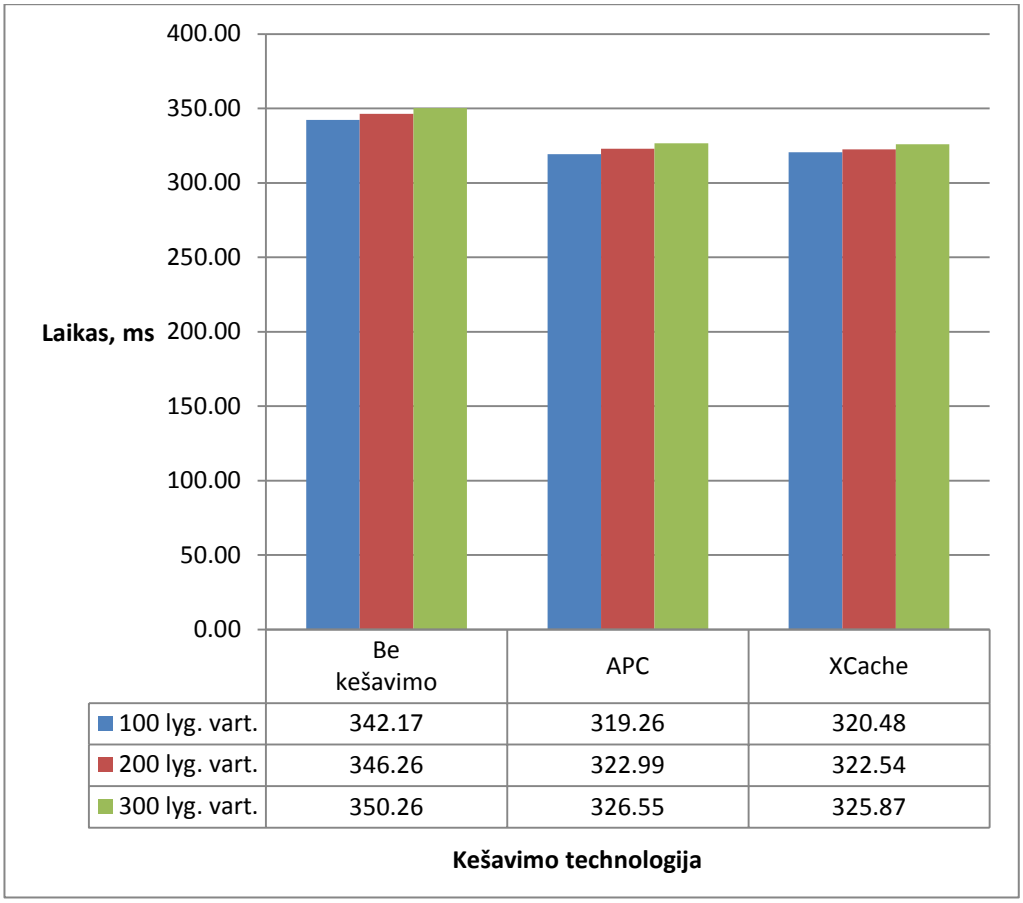
- WampServer programa, atstojanti PHP serverį ir duomenų bazę (Apache 2.2.22, PHP 5.3.13, MySQL 5.5.24)

- Apache Benchmark įrankis. Šis įrankis leidžią testuoti sistemą atliekant nustatytą kiekį užklausų simuliuojant lygiagrečių vartotojų skaičių (užklausų kiekį vienu metu). AB įrankis gali parodyti, kiek užklausų per sekundę Apache serveris įvykdė, kiek laiko truko viena užklausa, koks klaidų skaičius ir kt. [101]. Plačiau šis įrankis aprašytas 8.5 priede.

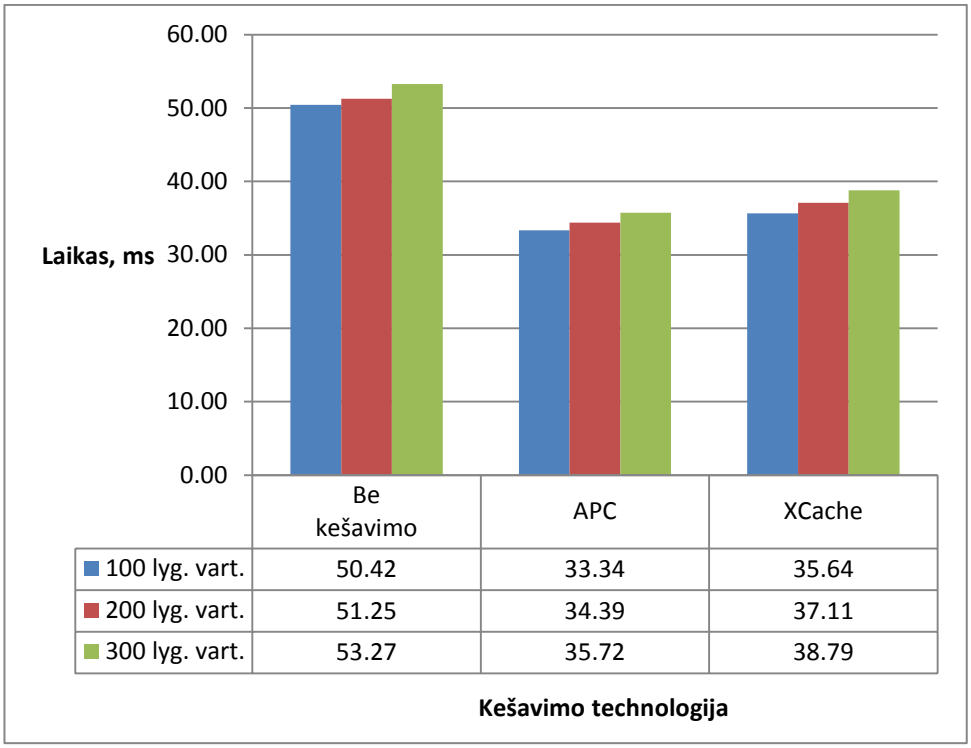
4.2.2 Eksperimento rezultatai

Puslapių užsikrovimo rezultatai, taikant tik opkodo kešavimą

21 ir 22 paveikslėliuose pateikti namų ir įmonės profilio puslapių užsikrovimo laikai naudojant tik opkodo kešavimą, atlikus 1000 testų su lygiagrečių vartotojų skaičiais 100, 200, 300. Rodomas vieno testo vidutinis laikas milisekundėmis.



Pav. 21. Namų puslapio kešavimo rezulatai, naudojant tik opkodo kešavimą



Pav. 22. Įmonės profilio puslapio kešavimo rezulatai, naudojant tik opkodo kešavimą

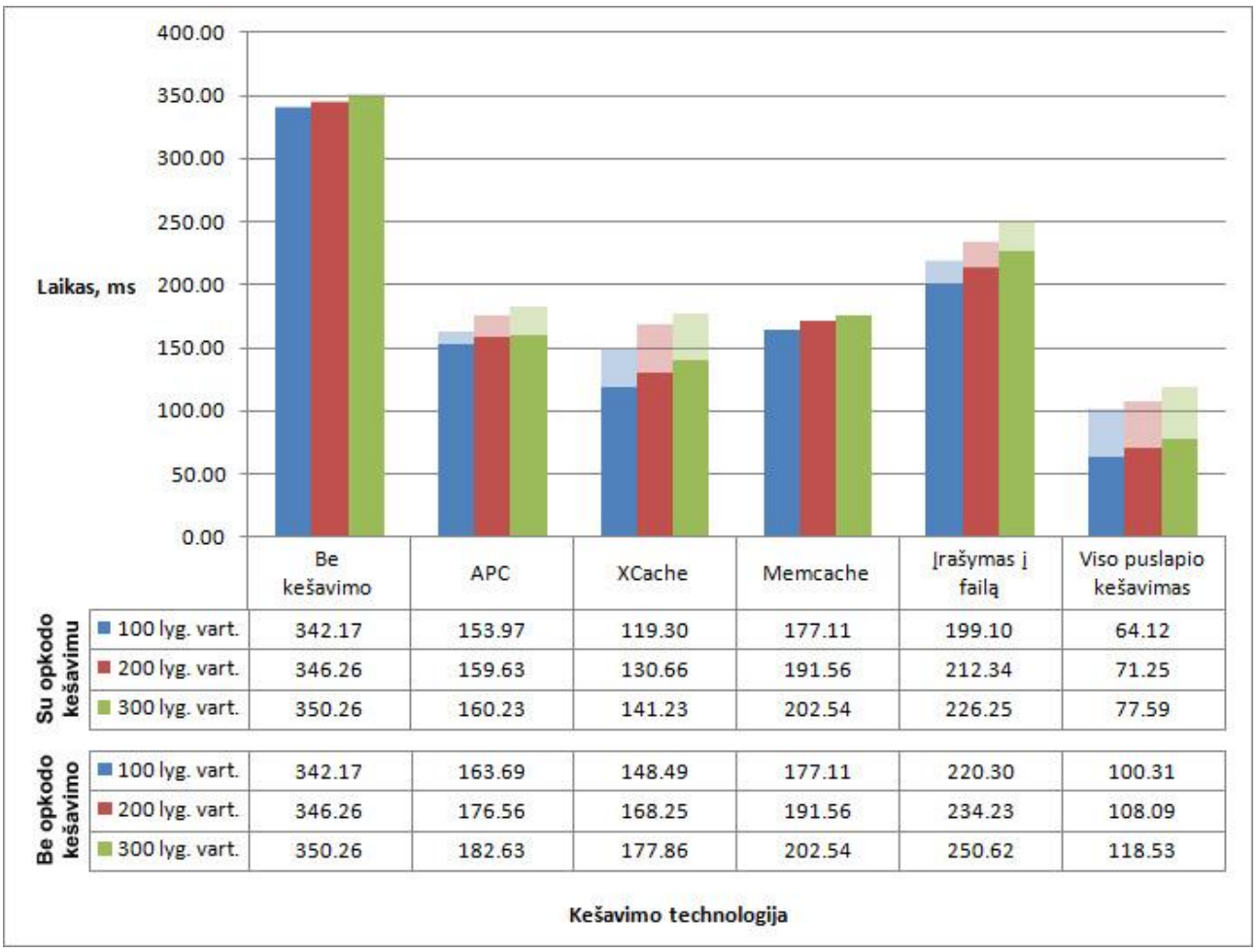
Opkodo kešavimo rezultatai namų puslapio ir įmonės puslapio užsikrovimo greičiams turėjo skirtingą įtaką. Įmonės profilio puslapyje, kuriame užklausų į DB kiekis yra nedidelis, opkodo kešavimas leido ženkliai pagreitinti užsikrovimo laiką. APC padidino puslapio užsikrovimo greitį 49-51 proc., XCache 37-41 proc. Tuo tarpu namų puslapyje, kuriame didžiają dalį užkrovimo laiko sudaro užklausos į DB, opkodo kešavimo nauda yra tik šiek tiek pastebima: puslapio užkrovimas naudojant tiek APC, tiek XCache padidėjo apie 7 proc. Stebime tendenciją, kad APC opkpdą kešuoja šiek tiek greičiau, o didesnis lygiagrečių vartotojų skaičius mažina kešavimo teikiamą naudą.

Šie rezultatai buvo pasiekti visiškai nekeičiant sistemos programinio kodo. Kad veiktų opkodo kešavimas, užtenka instaliuoti APC ar XCache technologiją ir įjungti opkodo kešavimo funkciją (jei ji būna automatiškai neįjungta). Taigi matome, kad įjungus opkodo kešavimą, be jokio papildomo programavimo buvo padidintas puslapių veikimo greitis 7 - 50 proc.

Puslapių užsikrovimo rezultatai, taikant aplikacijos kodo kešavimą kartu su įjungtu ir išjungtu opkodo kešavimu

23 ir 24 paveikslėliuose pateikti namų ir įmonės profilio puslapių užsikrovimo laikai naudojant **aplikacijos kodo** kešavimą, esant įjungtam ir išjungtam opkodo kešavimui, atlikus 1000 testų su lygiagrečių vartotojų skaičiais 100, 200, 300. Rodomas vieno testo vidutinis laikas milisekundėmis. Blyškūs stulpeliai rodo rezultatus opkodo kešavimui esant išjungtam, ryškūs - opkodo kešavimui esant įjungtam.

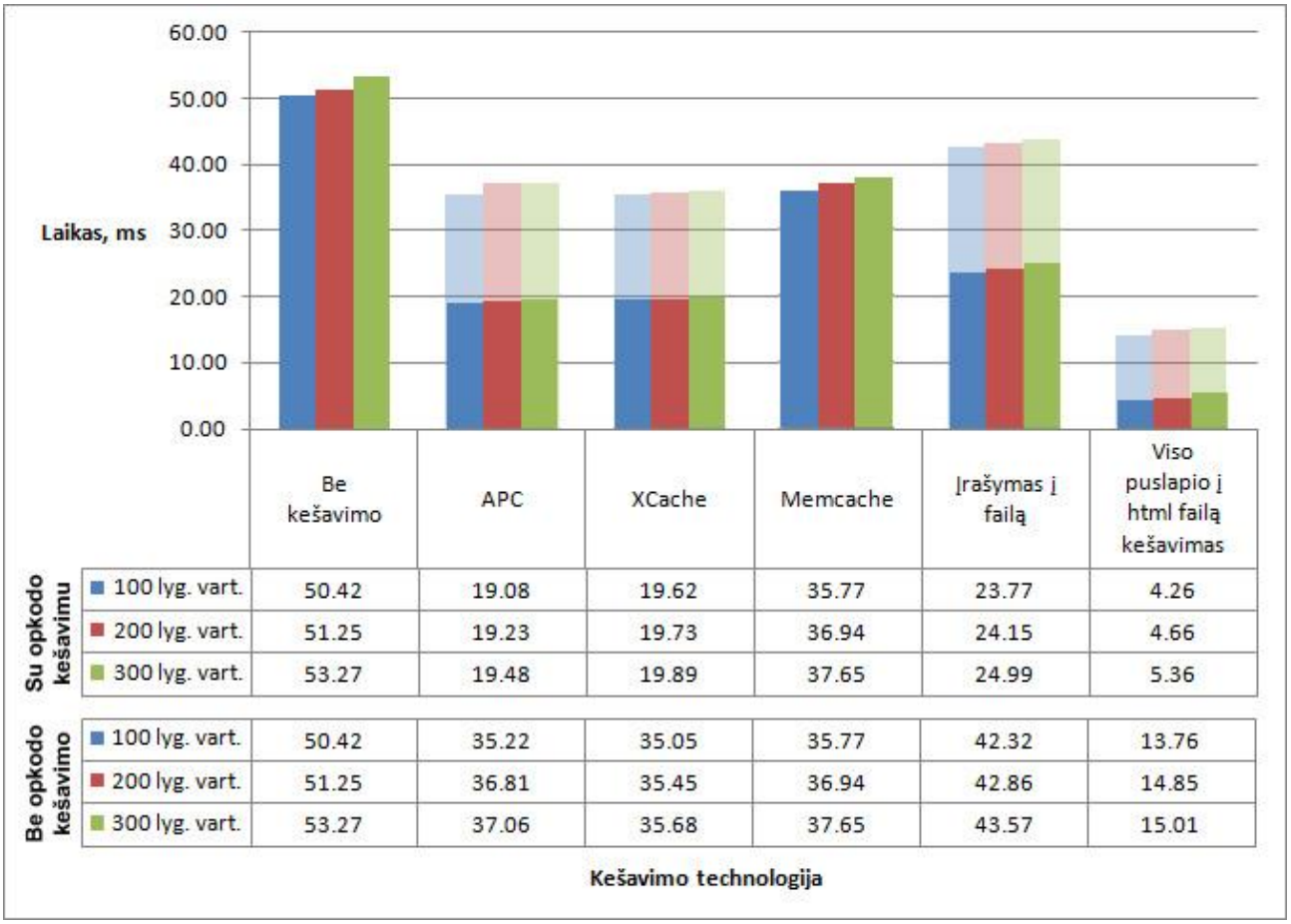
Pažymėtina, kad Memcache technologija neturi opkodo kešavimo funkcijos, o suderinti jos su APC ir XCache taip pat nepavyko. Todėl 23 ir 24 pav. pateikti Memcache skaičiai yra tik duomenų kešavimo. Duomenų kešavimas su įrašymo į failą ir viso puslapio įhtml failą kešavimo technologijos buvo apjungtos su APC opkodo kešavimu. O XCache duomenų kešavimas apjungtas su XCache opkodo kešavimu.



		Be kešavimo	APC	XCache	Memcache	Įrašymas į failą	Viso puslapio kešavimas
Su opkodo kešavimu	100 lyg. vart.	342.17	153.97	119.30	177.11	199.10	64.12
	200 lyg. vart.	346.26	159.63	130.66	191.56	212.34	71.25
	300 lyg. vart.	350.26	160.23	141.23	202.54	226.25	77.59
Be opkodo kešavimo	100 lyg. vart.	342.17	163.69	148.49	177.11	220.30	100.31
	200 lyg. vart.	346.26	176.56	168.25	191.56	234.23	108.09
	300 lyg. vart.	350.26	182.63	177.86	202.54	250.62	118.53

Pav. 23. Namų puslapio kešavimo rezulatatai, naudojant aplikacijos kodo ir opkodo kešavimą[1]

[1]Blyškūs stulpeliai rodo duomenų kešavimo rezultatus nenaudojant opkodo kešavimo. Kešuojant duomenis su Memcache, opkodo kešavimas nenaudotas.



		Be kešavimo	APC	XCache	Memcache	Įrašymas į failą	Viso puslapio į html failą kešavimas
Su opkodo kešavimu	100 lyg. vart.	50.42	19.08	19.62	35.77	23.77	4.26
	200 lyg. vart.	51.25	19.23	19.73	36.94	24.15	4.66
	300 lyg. vart.	53.27	19.48	19.89	37.65	24.99	5.36
Be opkodo kešavimo	100 lyg. vart.	50.42	35.22	35.05	35.77	42.32	13.76
	200 lyg. vart.	51.25	36.81	35.45	36.94	42.86	14.85
	300 lyg. vart.	53.27	37.06	35.68	37.65	43.57	15.01

Kešavimo technologija

Pav. 24. Įmonės profilio puslapio kešavimo rezulatai, naudojant aplikacijos kodo ir opkodo kešavimą[2]

Iš gautų rezultatų matome, kad greičiausiai veikia viso puslapio į html failą kešavimo būdas. Opkodo kešavimui esant išjungtam, namų puslapiui jis padeda užsikrauti 3 - 3,4 karto, įmonės vidiniam puslapiui - apie 3,5 karto greičiau nei nenaudojant kešavimo. Įjungus opkodo kešavimą, užsikrovimo greičiai namų ir įmonės profilio puslapiams padidėja atitinkamai 4,5 - 5,3 ir 9,9 - 11,8 karto (priklausomai nuo lygiagrečių vartotojų skaičiaus).

Palyginus su reikšmės kešavimo technologijomis, kurios kešuoja tik atskirus duomenis, esant išjungtam opkodo kešavimui, viso puslapio į html failą kešavimas veikia apie 1,5 karto greičiau namų puslapyje ir apie 2,5 karto greičiau įmonės profilio puslapyje. Įjungus opkodo kešavimą, šis pranašumas padidėja iki maždaug 2 kartų namų puslapyje ir 3,7 - 4,7 kartų įmonės profilio puslapyje.

[2]Blyškūs stulpeliai rodo duomenų kešavimo rezultatus nenaudojant opkodo kešavimo. Kešuojant duomenis su Memcache, opkodo kešavimas nenaudotas.

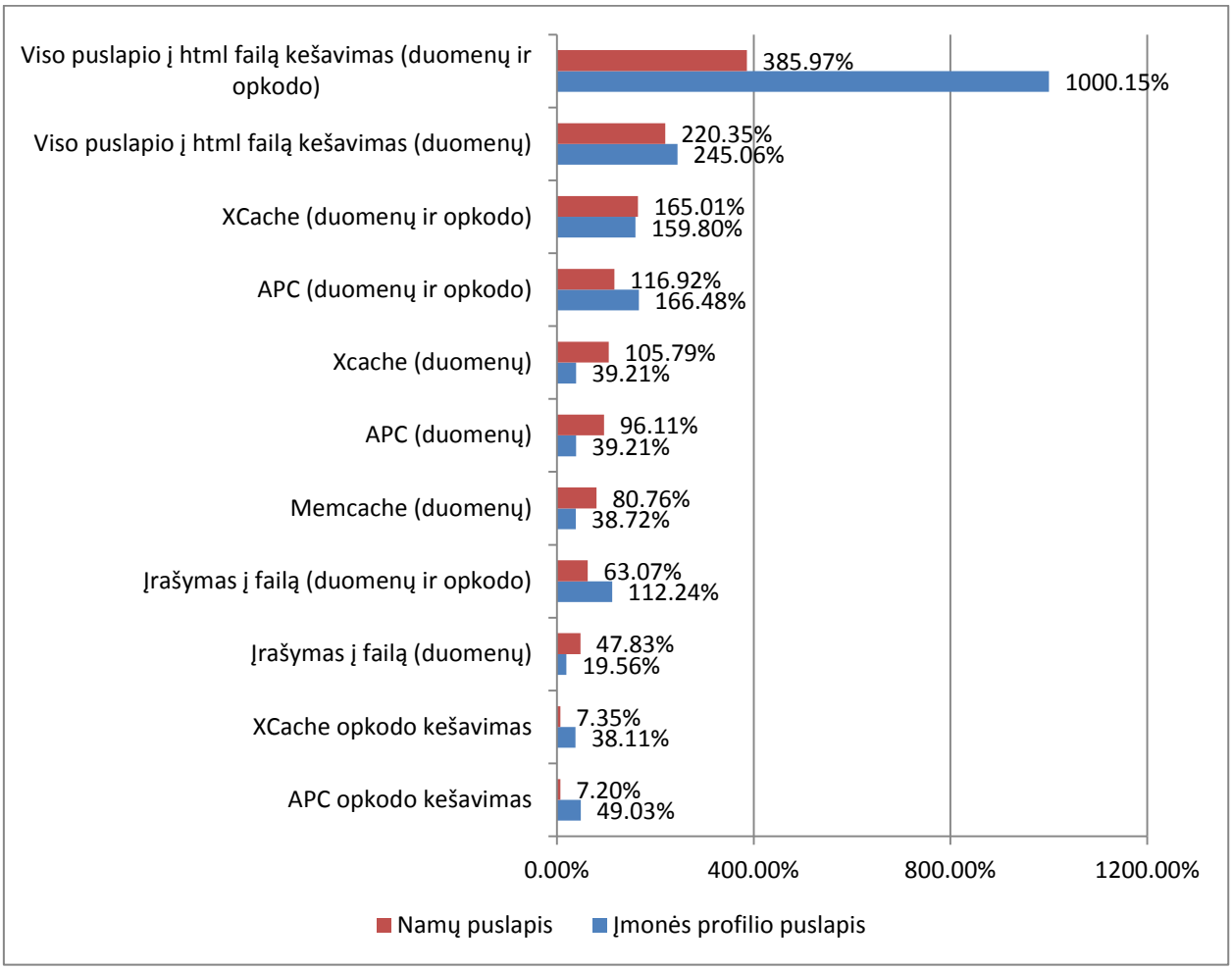
Jei lyginsime tik reikšmės kešavimo technologijas, geriausius laikus parodė APC ir XCache. Įmonės profilio puslapyje abiejų technologijų parodyti rezultatai beveik nesiskiria, o namų puslapyje, kur duomenų kiekis gerokai didesnis, gana ženklų pranašumą parodė XCache. Memcache nors ir nežymiai, tačiau veikia šiek tiek lėčiau už APC ir XCache.

Lėčiausiai, kaip ir tikėtasi, veikia įrašymo į failą technologija. Ji neprilygsta technologijoms, duomenis saugančius RAM, tačiau šis atsilikimas nėra ypatingai didelis, o nauda, palyginus jei išvis nebūtų kešuojama, yra akivaizdi.

Lygiagrečių vartotojų skaičius daro tiesioginę įtaką puslapio užkrovimo greičiui. Kuo didesnis lygiagrečių užklausų skaičius, tuo ilgiau užtrunka įvykdyti užklausą. Tačiau nepastebima, kad kažkuri kešavimo technologija su tuo susitvarkytų geriau ar prasčiau, visų rezultatai, keičiantis lygiagrečių vartotojų skaičiui, kinta maždaug proporcingai vienodai.

Sistemos pagreitėjimo, naudojant skirtingas kešavimo technologijas, palyginimas

Palyginkime, kuris kešavimo būdas - aplikacijos kodo ar opkodo - atnešė geresnius rezultatus įmonės profilio puslapiui, bei koks buvo procentinis pagreitėjimas naudojant šias dvi technologijas kartu. 25 paveikslėlyje rodomas procentinis pagreitėjimas lygiagrečių vartotojų skaičiui esant 200. Rezultatai išrikiuoti didėjimo tvarka pagal namų puslapio kešavimo rezultatus.



Pav. 25. Puslapių užkrovimo laiko pagreitėjimas dėl kešavimo naudojant įvairias aplikacijos kodo ir opkodo kešavimo technologijų kombinacijas

Kaip matome, minimalus pagreitėjimas buvo namų puslapyje naudojant tik opkodo kešavimą: APC ir XCache suteikė atitinkamai 7,20 proc. ir 7,35 proc. pagreitėjimą. Jei žiūrėsime tik į duomenų kešavimą, minimalus pagreitėjimas buvo 19,56 proc. naudojant įrašymo į failą technologiją įmonės profilio puslapyje.

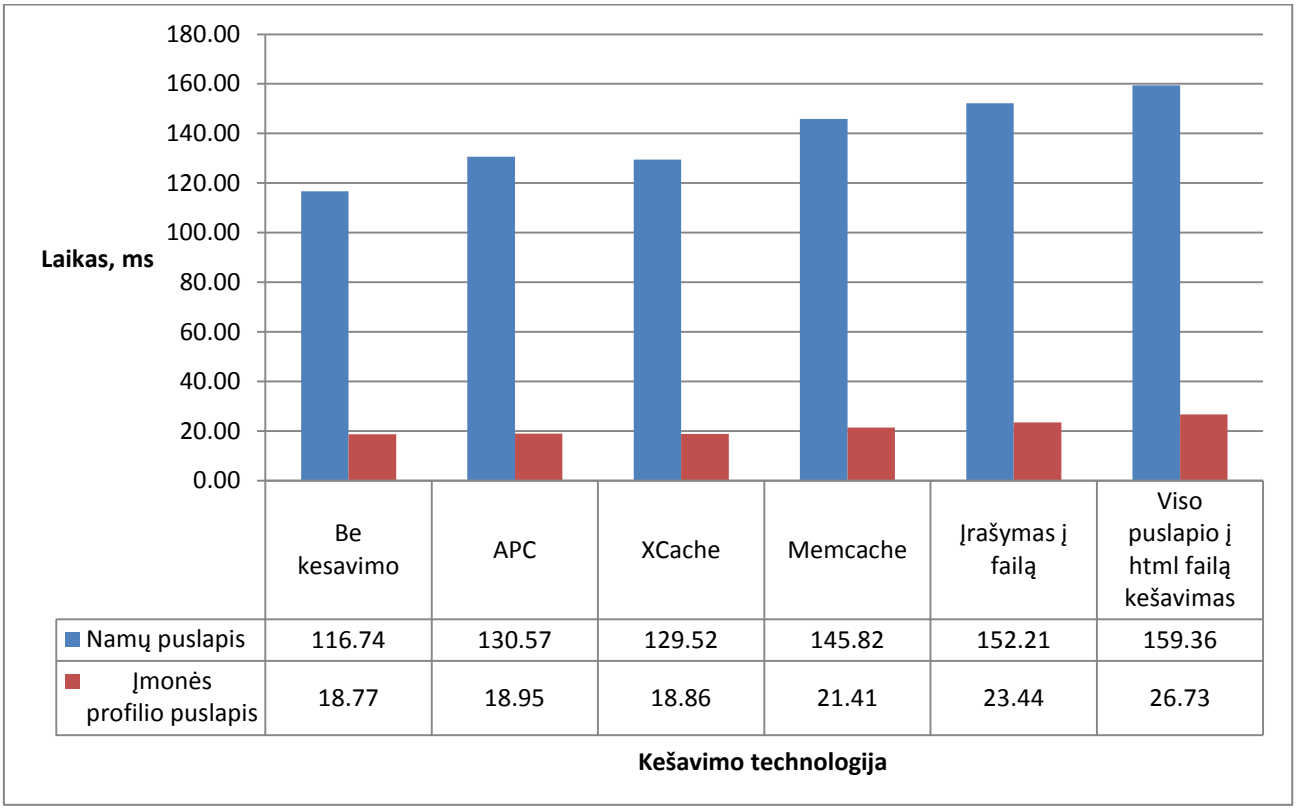
Maksimalus - net 1000,15 proc - stebėtas įmonės profilio puslapyje naudojant viso puslapio į kešavimo į html failą kartu su APC opkodo kešavimo technologiją.

Opkodo kešavimas suteikė didesnį puslapio užkrovimo pagreitėjimą nei duomenų kešavimas įmonės profilio puslapyje, tačiau duomenų kešavimas davė geresnius rezultatus namų puslapyje. Taigi galime daryti išvadą, kad kešuojant puslapius, kuriuose yra pateikiama daug dinaminės informacijos iš DB ir todėl vykdomas didelis užklausų į DB skaičius, opkodo kešavimas negali būti

laikomas visaverčiu sistemos greitaveikos spartinimo įrankiu. Jis gali būti tik kaip papildomas „bonusas“ siekiant kuo labiau pagreitinti sistemos veikimą.

Kešuojamų duomenų įrašymo ir ištrynimo greičio palyginimas

Norint visavertiškai įvertinti kešavimo technologijų veikimo greitį, aktualu pasižiūrėti ne tik kaip greitai jos sugeba nuskaityti informaciją iš įrašytų į atmintį duomenų, bet ir kaip greitai gali juos įrašyti į atmintį ar ištrinti iš atminties. 26 paveikslėlis rodo namų ir įmonės profilio puslapių užkrovimo laikus, kuomet kiekvienoje užklausoje duomenys yra įrašomi į atmintį ir tuoj pat ištrinami iš jos. Duomenų skaitymas iš atminties nevyksta. Įvykdyta 1000 testų esant 1 lygiagrečiam vartotojui, rodomas vienos užklausos vidutinis laikas milisekundėmis. Testai su daugiau nei vienu lygiagrečiu vartotoju nebuvo atliekami, nes tuomet galėtų atsitikti taip, jog vienas testas sugeneravo kešuotus duomenis, ir, jų nespėjus ištrinti, kito testo vykdymo metu duomenys būtų paimami jau iš kešuotų duomenų, o ne generuojami iš naujo.



Pav. 26. Namų ir įmonės profilio puslapių užkrovimo laikai simuliuojant kešuojamų duomenų įrašymą ir trynimą

Gauti rezultatai rodo, kad greičiausiai duomenis į atmintį įrašo ir ištrina APC ir XCache kešavimo technologijos. Namų puslapyje, kuriame yra kešuojamas didelis duomenų kiekis,

kešuojamų duomenų įrašymo ir trynimo procesas sulėtina puslapio veikimą 11,85 proc. APC atveju, ir 11,27 proc. XCache atveju. Įmonės profilio puslapyje, kuriame kešuojamų duomenų apimtis nedidelė, sulėtėjimas yra labai nežymus: atitinkamai 0,96 proc. ir 0,43 proc.

Sunkiausiai su duomenų įrašymo ir trynimo operacijomis susidoroja viso puslapio į html failą kešavimo technologija. Ji puslapio veiko greitį sumažina 1,4 karto namų ir įmonės profilio puslapiuose. Taigi jei nuskaitant duomenis šis būdas pasiteisino labiausiai, tai įrašant ir trinant duomenis šis būdas užtrunka daugiausiai laiko. Tačiau šis sulėtėjimas nėra toks ryškus palyginus su šios technologijos duodamu pagreitėjimu. Be to, įrašymo ir trynimo operacijos paprastai yra vykdomos daug rečiau nei nuskaitymo.

Galime teigti, kad pasirinkęs APC ar XCache kešavimo technologijas programuotojas gali sau leisti dažniau atnaujinti kešuotą informaciją ją ištrindamas bei vėl įrašydamas į atmintį. Tuo tarpu pasirinkus viso puslapio kešavimo metodiką, reikėtų stengtis kuo ilgiau išlaikyti kešuotą informaciją nepaliestą, nes duomenų atnaujinimas pareikalaus daugiau resursų.

Įrašymo ir trynime teste tarp reikšmės kešavimo technologijų aiškiai atsilieka Memcache, kurios pagrindiniai privalumai išryškėtų dirbant tik su itin didelėmis sistemomis, kuri būtų laikoma atskiruose nutolusiuose serveriuose.

4.2.3 Eksperimento išvados

- Didžiausią sistemos pagreitėjimo efektą suteikia viso puslapio į html failą kešavimas, netgi nepaisant to, kad tokiu atveju naudojamas kešavimas į failą, o ne į greitesnę RAM atmintį.
- Viso puslapio kešavimo į html failą būdas stipriai apriboja sistemos interaktyvumo galimybes. Jis turėtų būti naudojamas ten, kur informacija dažnai nesikeičia, ypač dėl to, kad kešuojant visą puslapį dažnas duomenų įrašymas ir trynimas labiau eikvoja resursus, nei naudojant reikšmės kešavimo technologijas.
- Siekiant pasinaudoti ypač efektyvia viso puslapio kešavimo į html failą technologija, išeitis galėtų būti dinaminių puslapio dalių atnaujinimas naudojant Ajax technologiją. Ji leidžia paimti duomenis iš serverio puslapiui jau užsikrovus;
- Tarp reikšmės kešavimo bibliotekų geriausius rezultatus parodė APC ir XCache. APC veikia šiek tiek greičiau kešuojant opkodą, XCache - aplikacijos kodą. Atsižvelgiant į tai, kad APC technologiją sukūrė ir palaiko ta pati kompanija, kuri kuria ir PHP, bei PHP 6 versijoje APC

bus įdiegtas kaip standartas,o XCache dokumentacija yra labai silpna, dinaminiams duomenims kešuoti optimalus sprendimas yra APC duomenų ir opkodo kešavimo technologijos;

- Būtina naudoti ne tik aplikacijos kodo, bet ir opkodo kešavimą dėl šių priežasčių:
    - jis nereikalauja jokių papildomų programavimo veiksmų, o sistemos veikimas pagreitėja nuo 5 iki 50 proc. ar net daugiau, priklausomai nuo to, kokią dalį puslapio užkrovimo laiko sudaro užklausos į DB, kurių opkodo kešavimas neįtakoja;
    - opkodo kešavimas apima ne tik tuos sistemos failus, su kuriais programuotojas tiesiogiai dirba. Pavyzdžiui, užkešuojamos ir naudojamo karkaso „core“ klasės, jų objektai, kurių pats programuotojas įtakoti negali ar bent jau neturėtų;
- Kešuojant reikia atsižvelgti į sąnaudas: jei pagreitėjimas nedidelis, o papildomo darbo programuotojui daug, geriau išlaikyt dinamiškumą ir nekešuoti, arba apsiriboti opkodo kešavimu.

4.2.4 Kešavimo priemonių realizavimas įmonių katalogo IS

Remiantis eksperimento išvadomis, suformuluoti sprendimai kaip kešavimas bus realizuotas įmonių katalogo informacinėje sistemoje.

Namų, įmonių ir vartotojų profilio puslapiams kešuoti nuspręsta pasinaudoti neįtikėtinu viso puslapio į html failą kešavimo efektyvumu. Joks kitas būdas neleidžia taip ženkliai paspartinti puslapių užsikrovimo laiko. Šis būdas bus naudojamas namų, įmonės ir vartotojo profilių puslapiams kešuoti.

Kadangi šių puslapių sistemoje bus labai daug, o dalis jų, galimas daiktas, bus aplankomi ne itin dažnai, laikyti retai naudojamus kešuotus duomenis ribotoje RAM atmintyje būtų neišmintinga.

Siekiant, kad sistema išliktų interaktyvi ir esant poreikiui pateiktų skirtingą informaciją skirtingiems vartotojams, bus naudojama Ajax technologija, kurios pagalba atskiros užkešuotų html failų dalys bus atnaujinamos asinchroniškai.

Kituose puslapiuose bus apsiribota globalių sistemos duomenų (statinių tekstų, nustatymų) kešavimu, panaudojant APC duomenų kešavimą bei opkodo kešavimą. Pastarasis bus įjungtas apskritai visame projekte, todėl bet kokiu atveju suteiks sistemai bent 10 proc. pagreitėjimą.

11 lentelė apibendrina kešavimo realizavimo rekomendacijas įmonių katalogo informacinėje sistemoje.

Lentelė 11. Kešavimo realizavimo rekomendacijos įmonių katalogo IS

Puslapis	**Kešavimo metodas**	**Prognozuojamas pagreitėjimas**	**Kešuotų duomenų invalidacija**
Namų puslapis	• Viso puslapio kešavimas įrašant į failą • APC opkodo kešavimas	Apie 386 proc.	Duomenys kešuojami nenurodant jų galiojimo laiko. Invaliduojami įvykus tam tikriems įvykiams. Su Ajax atnaujinami visi atskiri namų puslapio moduliukai (naujausi įrašai, prisijungę vartotojai ir kt.)
Įmonės profilio, vartotojo profilio puslapiai	• Viso puslapio kešavimas įrašant į failą • APC opkodo kešavimas	Apie 1000 proc.	Duomenys kešuojami nenurodant jų galiojimo laiko. Invaliduojami vartotojui atnaujinus profilio puslapį. Su Ajax atnaujinami: reitingas, panašūs katalogo įrašai
Kiti puslapiai	• APC duomenų (kešuojami tik sistemos nustatymai, statiniai tekstai • APC opkodo	10-20 proc.	Duomenys kešuojami nenurodant jų galiojimo laiko. Invaliduojami įvykus tam tikriems įvykiams (pvz. atnaujinus statinį tekstą ar nustatymą)

4.3 Apibendrinimas

Kešavimas yra efektyvus ir plačiai naudojamas būdas interneto sistemų našumui didinti, ypač aktualus didelėse interneto sistemose. Tačiau vykdyti kešavimą ir tuo pačiu užtikrinti informacijos korektišką atnaujinimą yra problematiška. Kešavimo svarba padidėja dirbant su PHP, nes kešavimas leidžia išvengti PHP kodo kompliavimo į opkodą etapo.

Kešavimo nauda:

- padidėjęs sistemos veikimo greitis;
- sumažėjęs serverio resursų naudojimas.

Kešavimo trūkumai:

- rizika, kad vartotojams bus pateikta pasenusi informacija;
- didesnės sistemos kūrimo ir palaikymo sąnaudos.

Atlikta esamų kešavimo priemonių analizė parodė, kad jas tikslinga grupuoti pagal kešuojamos informacijos saugojimo vietą, apimtį ir pobūdį. Nuspręsta detaliau išanalizuoti bei atlikti įmonių katalogo IS spartinimo eksperimentą su aplikacijos kodą bei opkodą kešuojančiančiomis kešavimo bibliotekomis: APC, XCache, Memcache bei CakePHP kešavimo į failą metodais. Atliktas eksperimentas parodė, kad didžiausią sistemos pagreitėjimo efektą suteikia viso puslapio į html failą kešavimas, tačiau jis stipriai apriboja sistemos interaktyvumo galimybes. Siekiant pritaikyti šį kešavimo būdą įmonių katalogo IS, dinaminės sistemos puslapių dalys bus atnaujinimos naudojant Ajax technologiją.

5. IŠVADOS

1. Atlikta rinkoje esančių analogiškų informacinių sistemų palyginimo analizė parodė, kad nėra sukurtų universalių atviro kodo sistemų, skirtų ūkio subjektams publikuoti katalogo pavidalu. Daugelis sistemų yra orientuotos į produktų, bet ne įmonių publikavimą.
2. Detalizavus funkcinius ir nefunkcinius reikalavimus sistemai bei suprojektavus jos architektūrą bei komponentus, buvo programiškai realizuota modernaus dizaino, šiuolaikines interneto technologijas naudojanti įmonių katalogo informacinė sistema, kurią galima pritaikyti įvairių ūkio subjektų paieškai ir publikavimui internete.
3. Atlikta esamų kešavimo priemonių analizė parodė, kad trūksta susistemintos informacijos apie kešavimo priemonių klasifikavimą, nurodymų, kada kokią kešavimo priemonę naudoti, jų taikymą interneto sistemose, todėl buvo atlikta kešavimo priemonių klasifikacija bei suformuotos teorinės jų taikymo prielaidos.
4. Remiantis atlikta kešavimo priemonių analize, atliktas sistemos greitaveikos eksperimentas testuojant įmonių katalogo informacinės sistemos greitaveiką su kiekvienu pasirinktu kešavimo sprendimu atskirai. Išanalizavus gautus rezultatus pateiktos kešavimo priemonių taikymo rekomendacijos:
    - interaktyviose, tampriai su vartotojo veiksmais susijusiose sistemose geriausia naudoti APC aplikacijos kodo kešavimą;
    - sistemose, kuriose didžioji dalis turinio yra statinis ir nekinta, geriausai naudoti viso puslapio į html failą kešavimo būdą;
    - siekiant pritaikyti viso puslapio į html failą kešavimo būdą dinaminėse sistemose, verta naudoti Ajax technologiją;
    - visais atvejais būtina naudoti opkodo kešavimą, kuris gali ženkliai padidinti sistemos veikimo greitį, tačiau nereikalauja beveik jokių papildomų programuotojo veiksmų.
5. Kadangi didžioji dalis įmonių katalogo informacinės sistemos puslapių yra statiniai, sistemoje buvo realizuotas viso puslapio į html failą kešavimo būdas kartu su APC opkodo kešavimo technologija. Dinaminės puslapio dalys atnaujinamos su Ajax technologija.
6. Atliktas kešavimo priemonių tyrimas bei išvados suteikia prielaidas praplėsti bei tęsti tyrimus atliekant kešavimo priemonių realizaciją sistemose sukurtose naudojant kitas programavimo kalbas tokias kaip .NET, Java, Ruby ar kt.

6. LITERATŪRA

1. Sulaiman S., Shamsuddin S.M. Intelligent Web Caching Using Adaptive Regression Trees, Splines, Random Forests and Tree Net// Data Mining and Optimization - 2011: tarptautinės konferencijos pranešimų medžiaga [Skudai, Malaizija, 2011 m. birželio 28-29 d.], p. 108-114.
2. Suzumura T., Scott T. ir kt. Performance of Web Service Engines in PHP, Java and C// Web Services: tarptautinės konferencijos pranešimų medžiaga [Tokijas, 2008 m. rugsėjo 23-26 d.], p. 385-392.
3. Santoro A., Ciciani B. Two-Tier Cooperation: A Scalable Protocol for Web Cache Sharing // Network Computing and Applications: tarptautinės konferencijos pranešimų medžiaga [Roma, 2001 m.], p. 186-193.
4. Samiee K. A. Replacement Algorithm Based on Weighting and Ranking Cache Objects// International Journal of Hybrid Information Technology. 2009 04, Nr. 2, p. 374-395.
5. Shim J., Scheuermann P., Vingralek R. Proxy Cache Design: Algorithms Implementation and Performance// Knowledge and Data Engineering. 1999, nr.4, p. 549-562.
6. Sosa V.J. Building a flexible Web caching system// Computer Science, 2003. ENC 2003. Proceedings of the Fourth Mexican International Conference: tarptautinės konferencijos pranešimų medžiaga [Vašingtonas, 2003 m.], p. 60-65.
7. Ali W., Shamsuddin S.M. Integration of Least Recently Used Algorithm and Neuro-Fuzzy System into Client-side Web Caching// International Journal of Computer Science and Security. ISSN 1985-1553. 2009, Nr.1, p. 1-15.
8. Zou Q. Transparent Web Caching with Minimum Response Time// Performance, Computing, and Communications Conference: tarptautinės konferencijos pranešimų medžiaga [Kingstonas, Kanada, 2003 m. balandžio 9-11 d.], p. 379-385.
9. Sekliuckis V., Gudas S., Garšva G. Informacijos sistemos ir duomenų bazės: Vadovėlis. Kaunas, Technologija, 2003. 349 p.
10. Gudas S., Lopata A. Žiniomis grindžiama sistemų inžinerija: Mokomoji knyga. Kaunas, Technologija, 2011. 228 p.
11. Butkienė R., Lopata A. Informacinių sistemų inžinerijos metodai ir modeliai: Paskaitų konspektas. Kaunas, Technologija, 2013. 139 p.
12. Boudreaux T.J. PHP 5 vaizdžiai. Kaunas: Smaltija, 2007. 308 p.
13. Meloni J.C. PHP, MySQL ir Apache. Kaunas: Smaltija, 2007. 602 p.

14. Čepolienė L., Nemuraitė L. Projektų vadybos informacinės technologijos. Mokomoji knyga. Mokomoji knyga. Vitae Litera, 2008. 132 p.
15. Padilla A., Hawkins T. Pro PHP Application Performance. Tuning PHP Projects for Maximum Performance. Apress, 2010. 244 p.
16. Tim Weilkens. Systems Engineering with SysML/UML. Modeling. Analysis. Design. Amsterdam, 2007. 307 p.
17. Abhijit Gadkari. Caching in the Distributed Environment. The Architecture Journal, 2008 [žiūrėta 2013 05 04], prieiga internete http://msdn.microsoft.com/en-us/library/dd129907.aspx
18. Walberg S.A. Tuning LAMP systems, Part 2: Optimizing Apache and PHP. IBM, 2007. [žiūrėta 2013 05 04], prieiga internete http://www.ibm.com/developerworks/linux/library/l-tune-lamp-2/
19. Best Practices for Speeding Up Your Website [žiūrėta 2013 05 04], prieiga internete http://developer.yahoo.com/performance/rules.html
20. Alternative PHP Cache dokumentacija [žiūrėta 2013 05 04], prieiga internete http://php.net/manual/en/book.apc.php
21. XCache dokumentacija [žiūrėta 2013 05 04], prieiga internete http://xcache.lighttpd.net/
22. Memcached dokumentacija [žiūrėta 2013 05 04], prieiga internete http://memcached.org/
23. Smarty dokumentacija [žiūrėta 2013 05 04], prieiga internete http://www.smarty.net

7. SANTRUMPŲ ŽODYNAS

IS - informacinė sistema

DB - duomenų bazė

UML - standartinė modeliavimo kalba (angl. Unified Modeling Language)

MVC - objektinio programavimo proncipu paremtaarchitektūra (angl. Model View Controller), leidžianti atskirti programavimo logiką nuo vaizdo pateikimo. Ši architektūra leidžia lengvai keisti programinio kodo funkcionalumą atskirai nuo vaizdo pateikimo, užtikrina tvarką projekte, leidžia su vaizdo failais lengviau dirbti dizaineriams, neišmanantiems programavimo.

KARKASAS- paruoštas klasių su dažniausiai naudojamomis funkcijomis ir bibliotekomis rinkinys, leidžiantis programuotojui išvengti monotoniško ir pasikartojančio darbo, suteikiantis projektui griežtą struktūrą, sudarantys sąlygas kurti patikimesnes, lengviau palaikomas sistemas, dažniausiai grindžiamas MVC architektūra. Naudojantis karkasu, sistema kuriama išplečiant karkaso klases, pačio karkaso nemodifikuojant.

TVS - turinio valdymo sistema.

CRUD funkcijos - baziniai metodai įrašų redagavimui: sukūrimui, nuskaitymui, atnaujinimui, trynimui (angl. create, read, update, delete),

RAM- aparatinės kompiuterio įrangos dalis, kurioje saugojami duomenys (angl.Random-access memory). Dar vadinama kompiuterio operatyviąja atmintimi, laisvosios kreipties atmintimi, virtualiąja atmintimi. Pasižymi tuo, kad duomenys saugojami atsitiktine tvarka, o ne iš eilės, kaip tradiciniuose kietuosiuose diskuose, o duomenų nuskaitymas iš šios atminties yra labai greitas.

AB - Apache serveriuose įdiegtas įrankis, leidžiantis testuoti puslapius simuliuojant serverio apkrovą (užklausų kiekį) ir lygiagrečių vartotojų skaičių (užklausų kiekį vienu metu).

AJAX - visuma technologijų (angl. Asynchronous Javascript and XML), leidžiančių interneto aplikacijose kliento pusei asinchroniškai bendrauti su serveriu.

8. PRIEDAI

8.1 Panaudojimo atvejų detalus aprašymas

Panaudojimo atvejai bendrose sistemos ribose:

Panaudojimo atvejis 1: Prisijungti prie sistemos

Vartotojo/aktoriaus pavadinimas:

Užsiregistravęs vartotojas, įmonės atstovas, sistemos administratorius

Aprašas:

Paprastas sistemosvartotojas (neprisijungęs lankytojas) prisijungia prie sistemos

Panaudojimo atvejo scenarijus:

Įvedamas vartotojo vardas

Įvedamas slaptažodis

Spaudžiamas mygtukas "Prisijungti"

Prieš sąlyga:

Lankytojas ateina į sistemą, bet nėra prisijungęs

Sužadinimo sąlyga:

Vartotojas nori pasinaudoti tik registruotam vartotojui prieinamomisfunkcijomis

Po sąlyga:

Vartotojas prisijungia prie sistemos, informaciniame vartotojo sąsajos blokelyje rodomi vartotojo sąskaitos duomenys.

Panaudojimo atvejis 2: moderuoti vartotojus

Vartotojo/aktoriaus pavadinimas:

Sistemos administratorius

Aprašas:

Registruojami, redaguojami, šalinami sistemos vartotojai, keičiamos jų rolės ir teisės

Panaudojimo atvejo scenarijus:

Administratorius susiranda vartotoją ir atsidaro jo sąskaitą

Pakeičia norimus duomenis

Spaudžia mygtuką "Išsaugoti"

Prieš sąlyga:

Vartotojas nėra arba yra registruotas sistemoje

Sužadinimo sąlyga:

Reikia užregistruoti naują arba redaguoti esamą vartotoją

Po sąlyga:

Užregistruotas naujas vartotojas arba pakeista esamo vartotojo informacija

Panaudojimo atvejai katalogo modulio ribose:

Panaudojimo atvejis 3: Naršyti katalogą

Vartotojo/aktoriaus pavadinimas:

Paprastas vartotojas, registruotas vartotojas, įmonės atstovas, sistemos administratorius

Aprašas:

Vartotojas ieško įmonių teikiamų paslaugų/produktų naršydamas po katalogą

Panaudojimo atvejo scenarijus:

Tam skirto meniu punkto pagalba įeinama į katalogą

Peržiūrimi įrašai pateikti pirmajame katalogo lape

Puslapiavimo pagalba nueinama į kitus katalogo puslapius

Filtravimo pagalba paliekami rodyti tik nurodytus kriterijus atitinkantys įrašai

Radus patinkantį įrašą, spaudžiama ant to įrašo ir patenkama į įrašo puslapį

Peržiūrima išsami įrašo puslapyje pateikiama informacija

Prieš sąlyga:

Vartotojas ieško paslaugų/produktų

Sužadinimo sąlyga:

Vartotojas pamato, kad gali peržiūrėti paslaugas teikiančias įmones naršydamas po katalogą

Po sąlyga:

Vartotojas išnaršė dalį arba visą katalogą, peržiūrėjo dalį ar visus įrašus

Panaudojimo atvejis 4: Ieškoti kataloge

Vartotojo/aktoriaus pavadinimas:

Paprastas vartotojas, registruotas vartotojas,įmonės atstovas, sistemos administratorius

Aprašas:

Vartotojas ieško paslaugų naudodamasis paieškos funkcija

Panaudojimo atvejo scenarijus:

Įvedamas paieškos žodis ir spaudžiamas mygtukas „Ieškoti“

Arba pasirenkami filtravimo kriterijai (pasirinkus filtravimo kriterijų, rezultatai pateikiami automatiškai, nebereikia spausti mygtuko „Ieškoti“

Arba pasirenkami rikiavimo kriterijai (pasirinkus rikiavimo kriterijų, rezultatai pateikiami automatiškai, nebereikia spausti mygtuko „Ieškoti“

Prieš sąlyga:

Vartotojas nori kuo greičiau surasti jam reikalingą paslaugą

Sužadinimo sąlyga:

Vartotojas pamato, kad lanksčios paieškos galimybės leis norimą paslaugą surasti greičiau

Po sąlyga:

Vartotojui parodomos jo paieškos kriterijus atitikę rezultatai

Panaudojimo atvejis 5: Reitinguoti katalogo įrašą

Vartotojo/aktoriaus pavadinimas:

Registruotas vartotojas

Aprašas:

Vartotojas gali įvertinti įmonę suteikdamas jai jo manymu teisingą reitingą

Panaudojimo atvejo scenarijus:

Naršomas katalogas

Įeinama į įmonės, kurią norima reitinguoti, išsamios peržiūros puslapį

Pasirenkama viena iš 10 žvaigždučių ir paspaudžiama ant jos. Kuo daugiau žvaigždučių, tuo įvertinimas geresnis

Perskaičiuojamas vidutinis įmonės įvertinimų vidurkis

Prieš sąlyga:

Vartotojas naršo kataloge

Sužadinimo sąlyga:

Vartotojas pamato įmonę, kurią nori įvertinti

Po sąlyga:

Įmonė įvertinta vartotojo. Perskaičiuotas vidutinis įvertinimų vidurkis.

Panaudojimo atvejis 6: Palikti atsiliepimą apie katalogo įrašą

Vartotojo/aktoriaus pavadinimas:

Paprastas vartotojas

Aprašas:

Vartotojas gali palikti atsiliepimą (parašyti komentarą) apie įmonę

Panaudojimo atvejo scenarijus:

Naršomas katalogas

Įeinama į įmonės, apie kurią norima palikti atsiliepimą, išsamios peržiūros puslapį

Įrašomas atsiliepimo tekstas

Spaudžiamas mygtukas "Palikti atsiliepimą"

Prieš sąlyga:

Vartotojas naršo kataloge

Sužadinimo sąlyga:

Vartotojas pamato įmonę, apie kurią nori palikti atsiliepimą

Po sąlyga:

Atsiliepimas apie įmonę išsaugotas ir matomas įmonės išsamios peržiūros puslapio atsiliepimų skiltyje

Panaudojimo atvejis 7: Siųsti užklausą įmonei (įmonėms)

Vartotojo/aktoriaus pavadinimas:

Paprastas vartotojas, registruotas vartotojas

Aprašas:

Vartotojas, suradęs kataloge įmonę (įmones), kuri tenkina jo reikalavimus, išsiunčia užklausą dėl išsamesnės informacijos

Panaudojimo atvejo scenarijus:

Naršant katalogą, pažymima įmonė (įmonės), kuriai bus siunčiama užklausa

Pasirinkus norimas įmones, spaudžiamas užklausos siuntimo mygtukas

Įvedamas klausimas

Įvedama papildoma (pvz, vartotojo kontaktinė) informacija

Spaudžiamas mygtukas "Išsiųsti"

Prieš sąlyga:

Vartotojas naršo kataloge

Sužadinimo sąlyga:

Vartotojas randa įmonę (įmones), kurių norėtų paklausti jam kilusio konkretaus klausimo

Po sąlyga:

Užklausa įmonei / įmonėms išsiųsta

Panaudojimo atvejis 8: Registruotis kataloge

Vartotojo/aktoriaus pavadinimas:

Registruotas vartotojas

Aprašas:

Registruotas vartotojas užregistruoja įmonę, kad ji taptų matoma kataloge

Panaudojimo atvejo scenarijus:

Įvedami įmonės rekvizitai

Įvedamas įmonės aprašymas

Įkeliamos įmonės veiklą apibūdinančios nuotraukos

Sužymimos paslaugų kategorijos, kurias gali suteikti įmonė

Įvedama kita reikalinga informacija

Pažymimos varnelės, kad įmonė sutinka su sistemos paslaugų teikimo sąlygomis ir pateikiami duomenys yra teisingi

Spaudžiamas mygtukas "Registruotis"

Sistemos administratoriui išsiunčiamas laiškas apie naujai registruotą įmonę

Registruotui vartotojui suteikiama vartotojo rolė „Įmonės atstovas“

Prieš sąlyga:

Registruotas vartotojas apsilanko sistemoje, jo įmonė nėra registruota kataloge

Sužadinimo sąlyga:

Registruotas vartotojas nori, kad jo įmonė atsidurtų kataloge

Po sąlyga:

Registruotas vartotojas įmonė užregistruojama kataloge ir tampa matoma vartotojams. Registruotui vartotojui suteikiama vartotojo rolė „Įmonės atstovas“

Panaudojimo atvejis 9: Moderuoti savo katalogo įrašą

Vartotojo/aktoriaus pavadinimas:

Įmonės atstovas

Aprašas:

Vartotojas savarankiškai redaguoja arba ištrina jam priklausantį katalogo įrašą

Panaudojimo atvejo scenarijus:

Pasirenkamas katalogo įrašas

Redaguojama informacija

Spaudžiamas mygtukas "Išsaugoti" arba "Ištrinti"

Prieš sąlyga:

Vartotojas yra įkėlęs katalogo įrašą

Sužadinimo sąlyga:

Vartotojas nori redaguoti arba ištrinti jam priklausantį katalogo įrašą

Po sąlyga:

Katalogo įrašas atnaujintas / ištrintas

Panaudojimo atvejis 10: Moderuoti katalogo įrašus

Vartotojo/aktoriaus pavadinimas:

Sistemos administratorius

Aprašas:

Administratorius redaguoja arba pašalina iš sistemos katalogo įrašus

Panaudojimo atvejo scenarijus:

Pasirenkamas katalogo įrašas

Redaguojama informacija

Spaudžiamas mygtukas "Išsaugoti" arba "Ištrinti"

Prieš sąlyga:

Kataloge yra daug įrašų

Sužadinimo sąlyga:

Reikia redaguoti arba ištrinti vieną iš katalogo įrašų, nes jame pateikiama informacija neatitinka sistemos naudojimo sąlygų arba įrašo savininkas pats to padaryti negali

Po sąlyga:

Katalogo įrašas atnaujintas / ištrintas

8.2 Kodo tipai

Mašinų kodas (angl. machine code). Šis kodas yra vadinamas „machine code“ arba „binary code“ - iš esmės bet kuris vykdomasis failas (pvz. su plėtiniu .exe) yra binary kodas - kodas, susidedantis iš nuliukų ir vienetukų - dvejetainių skaičių. Šį kodą supranta tik kompiuterio procesorius ir jis nėra suprantamas žmonėms (angl. human readable). Tačiau mašininis kodas nėra universalus kodas tinkantis vykdymui bet kuriam kompiuterio ar įrenginio procesoriui. Pvz., failo su .exe plėtiniu nepavyks įvykdyti linux ar unix sistemose, nes konkretus mašininis kodas - tai instrukcijos konkretaus tipo procesoriui, o procesorių ir įrenginių įvairovė didelė ir jie vienas nuo kito skiriasi.

Assembler kodas (angl. asssembly code). Natūralu, kad rašyti mašinoms suprantamą kodą programuotojams būtų per sudėtinga, todėl kompiuterių vystymosi pradžioje atsirado Assembler kalba. Šia kalba parašytas kodas po kompiliavimo etapo virsta mašininiu kodu. Kadangi nėra vieno universalaus mašininio kodo (vienoks mašininis kodas yra Windows 32 bitų tipo procesoriams, kitoks windows 64 bitų procesoriams), tai ir Assembler kalba turi skirtingus variantus - konkretus Assembler kalbos variantas būna skirtas konkrečios architektūros procesoriui, todėl Assembler kalba ir Assembler kodas - tai bendrinis terminas skirtas apibūdinti žmonėms suprantamą programuojamą kodą.

Assembler kalba parašyto kodo pavyzdys:

```
mov eax
77 jmp anywhere
```

Mašinoms suprantamos kalbos kodo pavyzdys:

```
5F 3A E3 F1
```

Baitkodas (angl. bytecode). Tam tikros kalbos, pavyzdžiui Java, programuotojų tekstu parašytą kodą kompiliuoja ne iš karto į mašininį kodą, o į tarpinį kodą - baitkodą. Tam, kad baitkodą paversti mašininiu kodu, yra naudojama virtualios mašinos programinė įranga. Pavyzdžiui, norint, kad kompiuteryje veiktų su Java programavimo kalba parašytos programos, į jį reikia instaliuoti JVM (angl. Java virtual machine) programinę įrangą. Tokiu būdu tą patį kodą, parašytą naudojant vieną operacinę sistemą, galima vykdyti visiškai kitame kompiuteryje su kitokia operacine sistema, kuriame taip pat yra instaliuota JVM. Na o JVM jau transformuoja baitkodą į kalbą, kurią supranta tos operacinės sistemos mašinos.

Opkodas (angl. opcode). Tai instrukcijos iš mašininio kodo dalies. Pavyzdžiui, „mov eax“ yra viena mašinų kodo instrukcija. Sukompiliavus šią vieną instrukciją bus gautas vienas opkodas (operation code). Taigi mašininis kodas susideda iš daug atskirų sukompiliuotų instrukcijų - opkodų. Opkodas, kaip ir mašinų kodas, skiriasi tarp įvairių platformų. Tas pats opkodas netiks dviems skirtingų architektūrų kompiuterio procesoriams. Kadangi PHP skriptas gali susidėti iš daugelio atskirų PHP failų, kompiliavimo metu yra padaromi keletas opkodo kodų, kurie vykdymo metu gali būti apjungiami ir vykdomi kaip mašininis kodas.

Opkodas yra jau paruoštas vykdymui procesoriuje kodas, todėl jis yra daug greičiau įvykdomas negu PHP kodas, nes jo nebereikia transformuoti į mašinoms suprantamą kodą. Be to, opkodas yra saugomas atmintyje (RAM), todėl jo nuskaitymas yra daug greitesnis nei iš disko.

8.3 Kešavimo bibliotekų funkcijų aprašymas

Lentelė 12. Pagrindinių APC funkcijų aprašymas

Funkcija	Aprašymas
apc_add($key, $value, [$ttl])	Naudojama duomenų įrašymui į saugyklą. Pirmas parametras $key –raktas, kuriam bus priskirta kešuojama reikšmė ir pagal jį vėliau bus galima paimti tą reikšmę. Raktui sugeneruoti dažnai būna paranki md5() funkcija, leidžianti sukurti unikalius raktus iš bet kokios eilutės. Antras funkcijos parametras $value – tai duomenys, kuriuos kešuojame. Trečiasis parametras – laikas sekundemis, kiek ilgai duomenys bus saugomi kešavimo saugykloje, kol pasibaigs jų galiojimo laikas. Jei šis parametras nenurodomas, duomenys priskirti raktui bus saugojami tol, kol nebus rankiniu būdu ištrinti.
apc_store($key, $value, [$ttl])	Veikia lygiai taip pat kaip apc_add() funkcija. Vienintelis skirtumas – jei pagal nurodytą raktą jau egzistuoja duomenys keše, tai apc_store() funkcija juos perrašys, o apc_add() nutrauks funkcijos vykdymą.
apc_fetch(`$key`)	Duomenų paėmimui iš kešavimo saugyklos skirta funkcija, kuriai kaip parametras perduodamas raktas.
apc_delete(`$key`)	Ištrinami kešavimo duomenys, priskirti nurodytam raktui. Ši funkcija reikalauja nemažai resursų, nes turi būti restruktūrizuojami kešavimo saugyklos duomenys. Todėl patartina suprogramuoti kešavimą taip, kad kešavimo resursai patys išsitrintų pasibaigus jų galiojimo laikui, ir jei įmanoma, patartina naudoti funkciją apc_clear_cache("user")
apc_clear_cache([$cache_type])	Ištrinami visi sistemos kešuoti failai. Jei perduodamas parametras „user“, bus ištrintas vartotojo kešas
apc_cache_info([$cache_type, $limited])	Gražina informaciją masyvo pavidalu apie sistemoje kešuotus failus
apc_exists($keys)	Patikrina ar nurodytas keše egzistuoja. Kaip parametrą galima nurodyti raktų masyvą.

Lentelė 13. Pagrindinių XCache funkcijų aprašymas

Funkcija	Aprašymas
xcache_set($key, $value, [$ttl])	Analogiška funkcija apc_add() funkcijai
xcache_get($key)	Analogiška funkcija apc_fetch() funkcijai
xcache_isset($key)	Analogiška funkcija apc_exists() funkcijai
xcache_unset($key)	Analogiška funkcija apc_delete() funkcijai

Lentelė 14. Pagrindinių Memcache funkcijų aprašymas

Funkcija	Aprašymas
memcache::connect($host, [$port], [$timeout])	Užmezgia ryšį su Memcache serveriu. Funkcijai perduodami parametrai: serveris, portas, laikas sekundėmis, skirtas prisijungimui prie modulio
memcache::add($key, $value, [$flag], [$ttl])	Išsaugo kešuojamus duomenis atmintyje, priskiriant jiems nurodytą raktą. Trečiasis parametras eidžia pasirinkti ar kešuojant naudoti MEMCACHE_COMPRESSED metodą
memcache::set($key, $value, [$flag], [$ttl])	Veikia analogiškai kaip ir add() funkcija, išskyrus tai, kad jei pagal nurodytą raktą jau egzistuoja duomenys keše, tai set() funkcija juos perrašys, o apc_add() nutrauks funkcijos vykdymą.
memcache::get($key)	Duomenų paėmimui iš kešavimo saugyklos skirta funkcija, kuriai kaip parametras perduodamas raktas.
memcache::replace($key, $value, [$flag], [$ttl])	Naudojama pakeisti kešuotus duomenis nurodytam raktui. Veikia panačiai kaip ir set() funkcija, tačiau gražina FALSE reikšmę jei nurodytas raktas neegzistuoja.
memcache::delete($key)	Ištrina kešą priklausantį nurodytam raktui

memcache::flush()	Pažymi visą kešą kaip nebegaliojantį. Funkcija neištrina kešo iš atminties ir neatlaisvina resursų. Tačiau nauji kešo duomenys bus užrašomi ant šių nebegaliojančių duomenų.

Lentelė 15. Pagrindinių Smarty kešavimo funkcijų aprašymas

Funkcija	Aprašymas
`setCaching(Smarty::CACHING_LIFETIME_SAVED)`	Funkcija, įjungianti Smarty kešavimą. Perduodamas parametras, kiek laiko bus saugojami užkešuoti failai.
setCacheLifetime($ttl)	Nurodomas užkešuoto failo galiojimo laikas, kurį nusako perduodamas parametras $ttl. Funkcija gali būti kviečiama prieš kiekvieną šablono atvaizdavimo funkciją display(), tokiu būdu skirtingiems vaizdo failams nurodant skirtingą galiojimo trukmę ir perrašant globalią užkešuotų failų galiojimo trukmę.
setCompileCheck(false);	Jei funkcijai perduodamas parametras true, rieš kraudamas kešuotą vaizdo failą, Smarty patikrins, ar tikrasis vaizdo failas nebuvo keistas. Jei buvo keistas, tuomet kešuotą failą sugeneruos iš naujo. Šis tikrinimas sunaudoja daugiau resursų, todėl nesant būtinybės, patartina jo neįjungti
clear_cache('index.tpl')	Ištrina nurodyto šablono kešuotą failą
clear_all_cache()	Ištrina visus kešuotus failus

8.4 Kešavimas naudojant PHP išeigos kontrolės funkcijas

PHP kešavimo į failą technologijos dažnai remiasi išeigos kontrolės funkcijomis. Šios funkcijos leidžia kontroliuoti, kada sugeneruotas kodas bus siunčiamas atvaizdavimui. Pradėjus išeigą su funkcija ob_start(), tolesnis kodo išvedinėjimas į ekraną yra blokuojamas, o vietoj to viskas išsaugoma atmintyje. Su funkcija ob_get_contents() galima įrašyti į failą buferyje išsaugotą informaciją, išvesti ją į ekraną ar atlikti kitus veiksmus.

Įjungus išeigos buferį, puslapio antraštės bus išsiųstos naršyklei, bet puslapio turinys išsiųstas nebus, kada tai padaryti galės nurodyti programuotojas.

Lentelė 16. Pagrindinių išeigos kontrolės funkcijų aprašymas

Funkcija	Aprašymas
bool ob_start ([callable $output_callback [, int $chunk_size = 0 [, bool $erase = true]]])	Įjungia išeigos buferį. Kol jis yra įjungtas, naršyklei siunčiamas kodas yra sulaikomas ir išsaugojamas buferyje.
string ob_get_contents (void)	Gauna išeigos buferio turinį, bet jo nepašalina iš buferio
void ob_clean (void)	Ištrina išeigos buferio turinį, bet neužbaigia įrašymo į buferį sesijos
bool ob_end_clean (void)	Ištrina išeigos buferio turinį ir užbaigia įrašymo į buferį sesiją.
void ob_flush (void)	Išsiunčia buferio turinį į naršyklę ir ištrina turinį iš buferio, bet pačio buferio nepašalina
bool ob_end_flush (void)	Išsiunčia buferio turinį į naršyklę ir pašalina buferį

Išeigos kontrolės funkcijos yra patogus būdas kešuoti puslapius – iškvietus ob_get_contents() funkciją sugeneruotas puslapio kodas įrašomas kaip string ir vėliau gali būti įrašytas į statinį html failą. Antrą kartą generuojant tą patį puslapį, pateikiamas jau sugeneruotas html failas.

Užrašykime 16 paveikslėlyje pavaizduotą algoritmą tai PHP kodo pavidalu, panaudojant php ob_start(), ob_get_contents() ir ob_end_flush() funkcijas:

```php
<?php
// failas, į kurį bus įrašomi duomenys:
$cache_file = „cache/newest_catalog_records.txt“;
// failo saugojimo laikas (30 minučių):
$cache_time = 30*60;

//jeigu jau yra sukurtas failas su duomenimis ir dar nepasibaigė jo galiojimo laikas, naudoti šį failą:
if (file_exists($cache_file) && ((time() - $cache_time) < filemtime($cache_file)){
    include($cache_failas);

//kitu atveju gauti prisijungusiu vartotoju sarasa atliekant užklausą i duomenu baze ir įrašyti užklausos duomenis į failą:
} else {
    ob_start();
    $newCatalogs = $this->Catalog->get('new');
    echo '<ul>';
    foreach($newCatalogs as $key => $val{
        echo '<li><a href="'.$val['url'].'">'.$val['title'].'</a></li>';
    }
    echo '</ul>';


    //įrašome užklausos duomenis į failą
    $f = fopen($cache_file, 'wb');
    $content = ob_get_clean();
    fwrite($f, $contant);
    fclose($f);
    echo $content;
}
?>
```

8.5 Apache Benchmark (AB) įrankio aprašymas.

Puslapių testavimui yra naudojami vadinamieji Benchmark testai. Apache serveriai, naudojami tinklapiams, parašytiems PHP kalba, turi įdiegtą AB įrankį, kuris leidžia pratestuoti puslapį simuliuojant serverio apkrovą (užklausų kiekį) ir lygiagrečių vartotojų skaičių (užklausų kiekį vienu metu). AB įrankis gali parodyti, kiek užklausų per sekundę Apache serveris įvykdė, kiek laiko truko viena užklausa, koks klaidų skačius ir kt.

Norint paleisti Apache Benchmark įrankį, per komandinę eilutę reikia iškviesti ab.exe failą, esantį Apache serverio bin kataloge. Jei dirbama lokaliai su WampServer programa, kelias iki šio failo bus toks: E:\wamp\bin\apache\apache2.2.22\bin\

Komandinėje eilutėje nuėjus į šį katalogą ir įvedus komandą ab, bus išmetama klaida, kad AB komandai reikalingi parametrai, bei išvedami ab funkcijos naudojimo parametrai:

```
E:\programos (portable)\wamp\bin\apache\apache2.2.22\bin>ab
ab: wrong number of arguments
Usage: ab [options] [http://]hostname[:port]/path
Options are:
    -n requests     Number of requests to perform
    -c concurrency  Number of multiple requests to make
    -t timelimit    Seconds to max. wait for responses
    -b windowsize   Size of TCP send/receive buffer, in bytes
    -p postfile     File containing data to POST. Remember also to set -T
    -u putfile      File containing data to PUT. Remember also to set -T
    -T content-type Content-type header for POSTing, eg.
                    'application/x-www-form-urlencoded'
                    Default is 'text/plain'
    -v verbosity    How much troubleshooting info to print
    -w              Print out results in HTML tables
    -i              Use HEAD instead of GET
    -x attributes   String to insert as table attributes
    -y attributes   String to insert as tr attributes
    -z attributes   String to insert as td or th attributes
    -C attribute    Add cookie, eg. 'Apache=1234. (repeatable)
    -H attribute    Add Arbitrary header line, eg. 'Accept-Encoding: gzip'
                    Inserted after all normal header lines. (repeatable)
    -A attribute    Add Basic WWW Authentication, the attributes
                    are a colon separated username and password.
    -P attribute    Add Basic Proxy Authentication, the attributes
                    are a colon separated username and password.
    -X proxy:port   Proxyserver and port number to use
    -V              Print version number and exit
    -k              Use HTTP KeepAlive feature
    -d              Do not show percentiles served table.
    -S              Do not show confidence estimators and warnings.
    -g filename     Output collected data to gnuplot format file.
    -e filename     Output CSV file with percentages served
    -r              Don't exit on socket receive errors.
    -h              Display usage information (this message)
```

AB funkcijos parametrai išsamiai paaiškinti 17 lentelėje. Visi jie nėra privalomi.

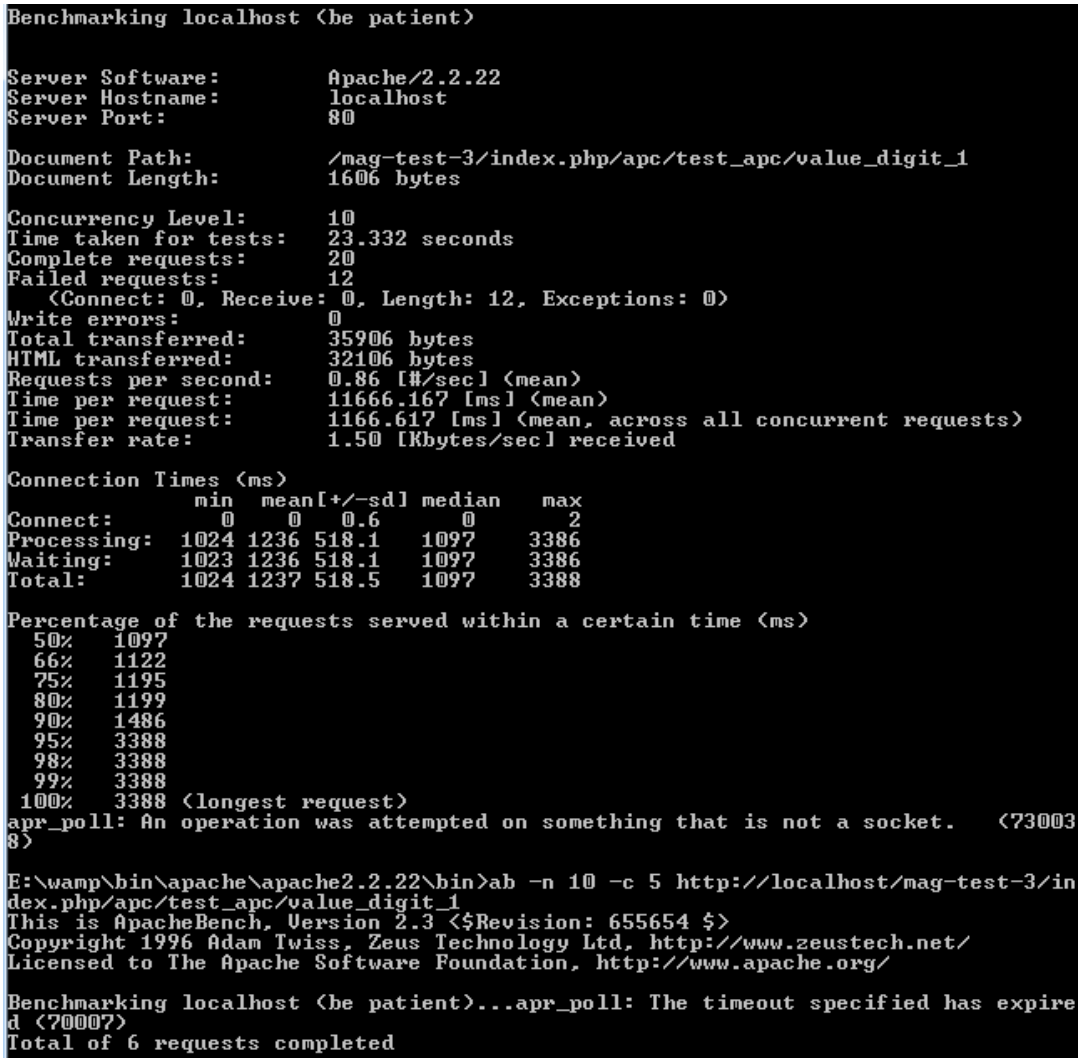
Lentelė 17. Svarbiausi AB funkcijos parametrai

Parametras	**Paaiškinimas**
-A	leidžia puslapiui perduoti vartotojo vardą ir slaptažodį, jei testuojamam puslapiui reikalingas prisijungimas. Vardas ir slaptažodis skiriamas dvitaškiu
-c	Kiek lygiagrečių užklausų bus vykdoma vienu metu. Nenurodžius parametro bus vykdoma viena užklausa
-C	leidžia nurodyti sausainėlį (???)
-e	išveda rezultatus į csv failą nurodytu pavadinimu
-h	parodoinformaciją kaip naudotis funkcija
-H	pridėti papildomas antraštes prie užklausos
-i	vykdyti HEAD užklausas vietohe GET
-k	įjungti KeepAlive savybę, t.y. atlikti daug užklausų naudojant vieną testavimo sesiją
-n	nurodyti, kiek užklausų bus vykdoma atliekant testą

Tarkime, norint pratestuoti http://localhost/cache_test puslapį, reikėtų įvesti tokią komandą:

ab –n 10 –c 5 http://localhost/cache_test

Parametras –n nurodo, kiek užklausų bus įvykdyta. Parametras –c nurodo, kiek lygiagrečių vartotojų vykdys užklausas vienu metu. Įvedus šią komandą, gaunami tokie rezultatai:



Kaip matome, testui įvykdyti prireikė 23.332 sekundžių. Per sekundę buvo įvykdomos vidutiniškai 0.86 užklausos

Testo rezultatai išsamiai paaiškinti 18 lentelėje:

Lentelė 18. Apache Benchmark testo rezultatai

Parametras (angl.)	**Parametras (liet.)**	**Rezultatas**
Time taken for tests	Laikas, kurio prireikė įvykdyti testui	23,332 s.
Complete requests	Įvykdytų užklausų skaičius	20
Failed requests	Neįvykdytų užklausų skaičius	12
Total transferred	Perduotas duomenų kiekis	35906
HTML transferred	Perduotas html duomenų kiekis	32106

Time per request	Vienos užklausos laiko vidurkis	11666.167 ms
Time per request across all concurrent requests	Vienos užklausos laiko vidurkis vienam lygiagrečiam vartotojui	1166.617 ms
Connections times	Prisijungimo laikai	Pateikiami prisijungimo, laukimo, vykdymo ir bendras laikas milisekundėmis: minimali, maksimali reikšmė, vidirkis ir mediana.
Percentage of the request served within a certain time	Užklausos nuo trumpiausios iki ilgiausios	Ilgiausia užklausa truko 3388 ms, trumpiausia – 1097 ms.

Apache Benchamrk leidžia neblogai įvertinti tą patį puslapį esant kokiems nors pasikeitimams, pavyzdžiui, kešuojant jį su skirtingomis bibliotekomis. Tiesa, reikia nepamiršti, kad realūs vartotojai „neatakuoja“ serverių tokiomis pačiomis užklausomis tūkstančius kartų, todėl realios serverio apkrovimo salygos ir su tuo susijusios problemos negali būti pilnai atspindėtos Apache Benchmark testo.

8.6 Lentelių sąrašas

8.7 Paveikslėlių sąrašas