KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Vaidas Dzedulionis

Prieigos teisių paskirstymas daug vartotojų turinčioje sistemoje, užtikrinant informacijos patikimumą

Magistro darbas

Darbo vadovas
doc. dr. E. Karčiauskas

Kaunas, 2009

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
PROGRAMŲ INŽINERIJOS KATEDRA

Vaidas Dzedulionis

Prieigos teisių paskirstymas daug vartotojų turinčioje sistemoje, užtikrinant informacijos patikimumą

Magistro darbas

Recenzentas
prof. R. Butleris
2009-05-25

Vadovas
doc. dr. E. Karčiauskas
2009-05-25

Atliko
IFM3/2 gr. stud.
Vaidas Dzedulionis
2009-05-25

Kaunas, 2009

Access rights management in multi-user system ensuring the reliability of the information

SUMMARY

One of the main things in development of software systems, which receive a lot of attention, is the maintenance and security of the data. In the systems that have a lot of users such as the various social networks, data protection and management issues are highly relevant. At such a systems it is very important to ensure certain procedures, important that users post correct information and it is also important that the users of system would behave according to predetermined set of rules for the system. All these issues need to manage very large number of human resources, especially in the social networks of hundreds or thousands of users, it is difficult to control each user.

In this paper all these issues are solved by using one of several access control models. After adjusting the model there was developed such a system, where simple users are becoming administrators and information publishers at the same time. System was created by the principle – the greater involvement of the user, the more system access rights it receives.

However, an allocation of rights to consumers also requires a lot of administration, whereas it is necessary to verify the extent of the user, to carry out the proper or improper actions, and to grant or take away various system rights.

These problems are solved in this work too, by refining model for the control of the distribution of rights, which automates the allocations of rights and at the same time this also reduces the time required by administration to a minimum.

Turinys

Paveikslėlių sąrašas

Lentelių sąrašas

1. Įvadas

1.1 Dokumento paskirtis

Šis dokumentas yra programų sistemų inžinerijos magistro baigiamasis darbas. Pagrindinis baigiamojo darbo tikslas yra rinkoje egzistuojančių prieigos teisių kontrolės modelių analizė ir projekto „Web finansinės skaičiuoklės“ metu sukurtos programos vartotojų prieigos teisių kontrolės sistemos optimizavimas. Prieigos teisių kontrolės optimizavimu buvo siekiama iki minimumo sumažinti sistemos prieigos teisų administravimui skiriamo laiko tarpą ir kaštus, užtikrinant patikimą informacijos apsaugą ir tvarką. Siekiant išsikelto tikslo buvo suformuluota keletas uždavinių:

- Išanalizuoti rinkoje esančias programavimo technologijas leidžiančias sukurti patogią ir lanksčia prieigos teisių valdymo sistemą;
- Išanalizuoti įvairius prieigos teisių valdymo modelius ir išrinkti labiausiai reikalavimus atitinkantį, kuris galėtų būti pritaikytas sistemoje;
- Ištirti esamos sistemos prieigos teisių modelį, rasti silpnąsias modelio vietas, pritaikyti tinkamesnį modelį arba pateikti būtinų modelio patobulinimų sąrašą;
- Pateikti siūlymus nustatymo modelio įdiegimo ir adaptavimo klausimais.

Visi paminėti uždaviniai yra sprendžiami ir aprašomi šiame dokumente.

Dokumentas yra sudarytas iš trijų pagrindinių dalių. Pirmojoje dalyje yra analizuojamos rinkoje esančios programavimo technologijos ir įvairūs prieigos teisių modeliai. Antrojoje dalyje pateikiami pagrindiniai sistemos reikalavimai ir projektiniai sprendimai. Trečiojoje dalyje aprašomas sistemos prieigos teisių tyrimas, pateikiamos ištirtos sistemos tyrimo išvados ir reikalingi patobulinimai, pasiūlomas optimizuotas prieigos teisių modelis.

1.2 Santrauka

Vienas iš pagrindinių dalykų programų sistemų kūrime, į kuriuos kreipiama labai daug dėmesio, yra duomenų saugumas. Daug vartotojų turinčiose sistemose, tokiose kaip įvairūs socialiniai tinklai, duomenų apsaugos ir tvarkymo problemos yra itin aktualios. Sistemoje esant daugeliui registruotų

vartotojų svarbu užtikrinti tam tikrą tvarką, svarbu, kad vartotojų skelbiama informacija būtų korektiška, taip pat svarbu, kad sistemos vartotojai elgtųsi pagal iš anksto nustatytas sistemos taisykles. Visiems šiems dalykams tvarkyti reikia labai daug administravimo resursų, ypač socialiniuose tinkluose esant šimtams ar tūkstančiams vartotojų, kiekvieną vartotoją sukontroliuoti yra itin sudėtinga.

Šiame darbe šios problemos yra sprendžiamos panaudojant vieną iš keleto prieigos teisių valdymo modelių. Pasirinktą modelį pakoregavus buvo sukurta sistema, kurią naudojant patys sistemos vartotojai tampa ir sistemos administratoriais, ir informacijos skleidėjais. Sukurtos sistemos principas – kuo aktyvesnis vartotojas, tuo daugiau sistemos prieigos teisių jis gauna. Tačiau pats teisių paskirstymo vartotojams procesas taip pat reikalauja labai daug administravimo ir laiko sąnaudų, kadangi būtina tikrinti, kiek kuris vartotojas atliko tinkamų ar netinkamų veiksmų ir pagal tai suteikti ar atimti įvairias teises.

Šioms problemoms spręsti, šiame darbe, sukuriamas patobulintas teisių paskirstymo kontrolės modelis, kuris automatizuoja teisių paskirstymą ir sumažina teisių administravimui reikalingo laiko tarpą iki minimumo, taip sutaupydamas brangu administratoriaus laiką ir nemažai sistemos administravimo kaštų.

2. Analitinė dalis

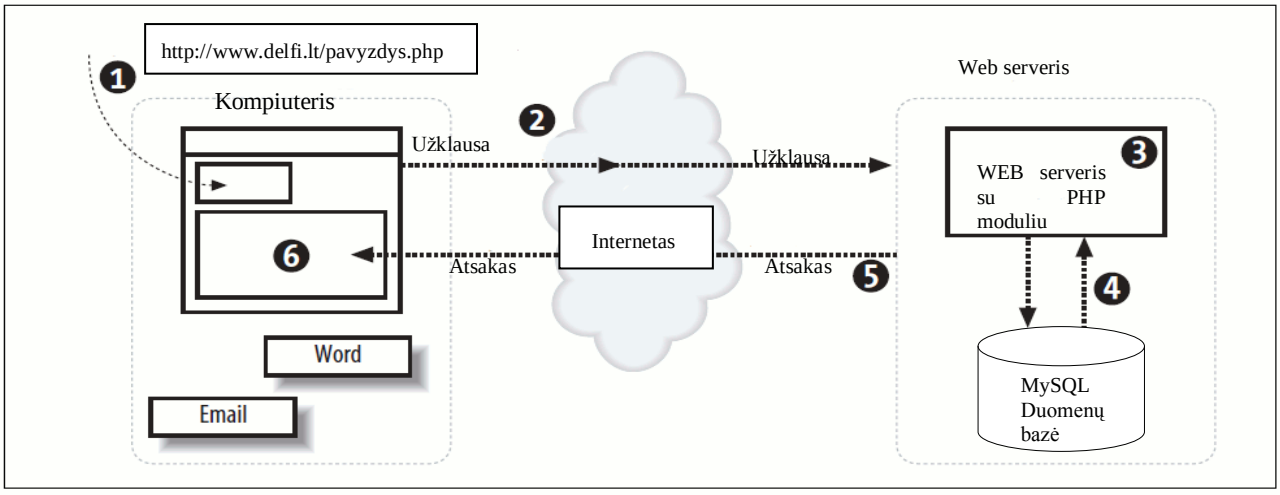
2.1 Technologijų analizė

2.1.1 PHP ir MySQL

PHP yra tinklapių programavimo kalba, parašyta pačių tinklapių kūrėjų. Kai tik PHP buvo sukurta ji originaliai buvo pavadinta „Personal Home Page Tools“, nuo šių angliškų žodžių ir kilo PHP pavadinimas. Šiuo metu PHP yra penktą kartą perrašyta iš pagrindų ir pavadinta PHP5 arba tiesiog PHP.

PHP yra serverio pusės programavimo kalba, kuri yra įterpiama į HTML kodą arba naudojama, kaip atskira programavimo kalba. Šiai nišai priklausantys produktai yra: Microsoft’s Active Server Pages, Macromedia’s ColdFusion, ir Sun’s Java Server Pages [1]. Kai kurie techniniai žurnalistai mėgsta PHP pavadinti kaip „Atviro kodo ASP“, todėl kad PHP funkcionalumas yra panašus į ASP, bet iš tikrųjų šis formulavimas yra klaidingas, kadangi PHP atsirado anksčiau nei ASP. Per pastaruosius kelis metus PHP ir serverio pusės Java įgijo daug pasekėjų, kai tuo pat metu ASP prarado [19], todėl šis palyginimas išvis nebe tinkamas.

Toliau plačiau pakalbėsiu apie PHP. PHP galima pavadinti super-HTML „tagų“ rinkiniu arba mažų programėlių rinkiniu, kurios yra vykdomos internetiniame puslapyje, serverio pusėje prieš juos parsiunčiant į naršyklę. Pavyzdžiui PHP galima panaudoti pridedant „headerius“ ir „footerius“ visiems puslapiams serveryje arba panaudojant PHP galima išsaugoti formos duomenis duomenų bazėje.



1. pav. PHP veikimo architektūra

Kai vartotojas įveda adresą naršyklėje ir nuspaudžia mygtuką vykdyti, įvyksta keltas veiksmų, kurie pavaizduoti 1. paveikslėlyje ir žemiau aprašyti:

1. Vartotojas įveda internetinio puslapio adresą;

2. Naršyklė išskaido įvestą adresą ir siunčia užklausą web serveriui. Pavyzdžiui., jei nurodytas adresas yra http://www.delfi.lt/pavyzdys.php tai naršyklė išskaido šį adresą į dvi dalis: delfi.lt (serverio domenas) ir pavyzdys.php (reikalaujamas puslapis);
3. Programa web serveryje pavadinta web serverio procesas priima užklausą pavyzdys.php puslapiui ir ieško atitinkamo failo;
4. Web serveris nuskaito failą iš serverio kietojo disko. PHP modulis peržiūri failą ir jei randa PHP kodo jį įvykdo ir sugeneruoja HTML kodą. Jei yra užklausų į duomenų bazę, jos įvykdomos;
5. Web serveris siunčia atsaką atgal į naršyklę HTML formatu;
6. Naršyklė HTML kodą atvaizduoja atitinkamais vaizdais.

Griežtai kalbant PHP mažai ką turi bendro su puslapio išvaizda, tiesioginiais įvykiais paremtais DOM manipuliavimu, ir tikrai neturi nieko bendra su tuo, kaip puslapis atrodo ar kokius garsus skleidžia. Iš tikrųjų didžioji dalis to ką daro PHP yra nematoma puslapio vartotojui. Kažkas kas žiūri į PHP puslapį galėtų pasakyti, kad jis yra parašytas vien iš HTML kodo, ir jis būtų visiškai teisus, kadangi viskas ką padarė PHP buvo atlikta serveryje ir vartotojas mato tik rezultatus [1]. O paprastai rezultatas ir yra HTML kodas.

PHP yra oficialus Apache HTTP serverio modulis. Apache yra rinkos lyderis tarp WEB serverių, kuris užima beveik 67% visos WWW (World Wide Web) rinkos (pagal Netcraft interneto tyrimus) [20]. Tai reiškia, kad PHP programavimo variklis gali būti įdiegiamas į WEB serverius ir lydimas greito vykdymo ir efektyvaus atminties paskirstymo, ir labai supaprastinto palaikymo. Kaip ir Apache serveris, PHP yra pilnai įvairiaplatformis (angl. „cross-platform“), o tai reiškia, kad PHP gali būti naudojamas Unix, Windows ar Mac OS X sistemose [3].

MySQL yra atviro kodo SQL reliacinė duomenų bazės valdymo sistema (Relational Database Management System RDBMS), kuri yra nemokama daugumai vartotojų. Anksčiau istorijoje MySQL susidurdavo su pasipriešinimu, dėl jos trūkumo palaikant keletą pagrindinių SQL savybių tokių, kaip sub užklausos (angl. subselects) ar svetimi raktai (angl. foreign keys) [1][3]. Bet greitai MySQL rado savo gerbėjus, dėl savo liberalios licencijavimo sistemos, stabilaus ir greito veikimo ir lengvo panaudojimo. MySQL išpopuliarėjimo priežastis taip pat buvo ir platus įvairių technologijų

palaikymas, tokių kaip PHP, Java Perl, Python, taip pat MySQL turėjo gerą visų savo modulių ir išplėtimų dokumentaciją. MySQL nenuvylė savo pasekėjų ir su 4.1 versija, buvo pridėtas sub užklausų palaikymas ir svetimų raktų (angl. foreign keys) palaikymas [22].

Bendrai paėmus duomenų bazės yra labai naudingos, lyginant su kitas programinės įrangos produktas, dėl savo ypatingai greitų skaičiavimo ir duomenų išrinkimo galimybių. Kaip ir visi kiti konkuruojantys produktai MySQL nėra duomenų bazė kol tu jai nepriskiri tam tikros duomenų struktūros ir formos.

Kaina

PHP yra nemokama. Nereikia mokėti nei norint parsisiųsti, nei norint ją naudoti visą kuriamos aplikacijos gyvavimo laiką. Apache/PHP/MySQL derinys veikia labai gerai ir pigiai, o jei paimtume „IIS/ASP/SQL server“ derinį, tai jis kainuoja labai daug pinigų, o jų kokybės skiriasi neypatingai.

Jei nagrinėtume MySQL kainą, tai MySQL yra šiek tiek kitoks produktas žiūrint į jo licencijavimo sąlygas. MySQL yra atviro kodo sistema daugeliui naudojimo atvejų, tačiau ši sistema niekada nebuvo kuriama plačios bendruomenės. MySQL AB yra komercinė įmonė su tam tikrais komerciniais tikslais [3]. Tai nėra tas atvejis, kai programuotojai dirba kitose įmonėse, o savo laisvu metu dirba prie juos dominančių atviro kodo projektų. MySQL programuotojai gauna pajamas iš darbo prie MySQL projekto. Tačiau yra labai daug aplinkybių, kai MySQL gali būti naudojama nemokamai, bet jei vykdomi projektai, kurie naudoja MySQL naujiems produktams kurti, kurie yra komerciniai, tuomet reikia pirkti licenciją arba pasirašyti palaikymo kontraktą. Tai tikrai labiau apsimoka nei tik nusipirkti tam tikros programos licenciją.

Norėdamas parodyti kainų palyginimus pateikiu lentelę, kurioje galima matyti rinkoje siūlomų produktų kainas, JAV doleriais [3]. Visos kainos pateiktos vieno procesoriaus serveriams.

1. lentelė PHP ir MySQL analogų kainos

Produktas	**ASP/SQL server**	**ColdFusion MX/SQL Server**	**JSQP/Oracle**	**PHP/MySQL**
Programavimo įrankiai	0 - 2499	599	0 - 2000	0 - 249
Serveris	999	2298	0 - 35000	0
RDBMS	4999	4999	15000	0 - 220

Dauguma žmonių galvoja, kad tai kas pigiausia yra prasčiausios kokybės. Tai ne visiškai tiesa, kai kalba eina apie atviro kodo programinę įrangą. Dažnai pagalvojame, kad negali būti kokybiškos ir gyvybingos programinės įrangos, kuri nieko nekainuoja. Šis požiūris atsirado iš kuriamų nemokamų programinės įrangos produktų, kurie vadinami „freeware“, „shareware“ ar nemoka programinė įranga. Ši programinė įranga buvo skirstoma į šias pagrindines grupes [3]:

- Programinė įranga užpildanti mažas ne komercines nišas;
- Programinė įranga, kuri vykdo neatsakingus, žemo lygio darbus;
- Programinė įranga skirta įvairioms socialinėms politinėms problemoms nagrinėti.

Šiuo metu yra pats įkarštis vykstančių pasikeitimų programinės įrangos kūrimo versle. Dauguma pagrindinių vartojamų programinės įrangos produktų pateikiami be jokio mokesčio. Tai yra el. pašto klientai, naršyklės, žaidimai taip pat netgi pilni ofiso programų paketai, įdiegiami kompiuteryje ar netgi pateikiami interneto naršyklės lange, visiškai nemokamai. Programinė įranga Kuriama vartotojams skatina pardavinėti daugiau techninės įrangos, operacinių sistemų, prijungiamų įrenginių, skelbti reklamą ar kelia įmonių akcijų kainas, o tai su kaupu atperka visas patirtas išlaidas kuriant nemokamą, patikimą programinę įrangą.

Serverio pusės atviro kodo programinė įranga tampa vis galingesnė. Ji sėkmingai konkuruoja su geriausiais komerciniais produktais, kartais atrodo, kad jie jau nurungė visus savo priešininkus.

PHP licencija

Atviro kodo programinės įrangos nemokamumas yra garantuojamas daugybės licencijavimo schemų iš kurių populiariausia yra GPL (Gnu General Public License) arba „copyleft“. PHP buvo publikuojama su dviem licencijom, tai yra GPL ir jos pačios licencija. Kiekvienas vartotojas galėjo pasirinkti, kurią jis nori naudoti. Ši tvarka neseniai pasikeitė. Visa programa išleidžiama su jos pačios PHP licencija, kuri paremta BSD licencija, kur Zend, kaip atskiras produktas yra išleidžiamas pagal Q Public License [1].

Dauguma žmonių parsisiunčia PHP ar MySQL nemokamai iš atitinkamų svetainių, tačiau kartai nutinka taip, kad tenka susimokėti už kai kuriuos produktus, tokiu kaip kad Linux distribucija, techninė knyga ar kiti produktai, kurie turi integruotus šiuos produktus. Todėl kartais gali pasirodyti, kad šie produktai yra mokami.

Paprastai atviro kodo programinės įrangos vartotojai gali laisvai pasirinkti optimalų kainos ir kokybės santykį kiekvienai situacijai: nemokama – be garantijų arba brangų ir su labai geru palaikymu, arba kažką tarp šių dviejų variantų. Iki šiol nebuvo nė vieno bandymo parduoti paslaugų susijusių su PHP arba pačio PHP palaikymo. MySQL AB pardavinėjo MySQL palaikymą, kaip dalį paslaugų nurodytų jų licencijų pakete [22]. Kiti atviro kodo produktai tokie kaip Linux už savęs turi tokias kompanijas, kaip „Red Hat“, kurios suteikia palaikymą ir kuria mokamus papildymus, tačiau PHP sukomercininmas yra vis dar ankstyvosiose stadijose.

Naudojimo paprastumas

Programuoti PHP kalba yra nesunku išmokti, lyginant su kitomis programavimo kalbomis norint pasiekti tą patį funkcionalumą. Ne taip kaip Java ar C pagrindu kuriamos CGI programos, PHP nereikalauja iš vartotojo gilaus programavimo supratimo norint padaryti kreipimąsi į duomenų bazę ar į kitą serverį. PHP turi sintaksę, kurią yra gana lengva suprasti, kitaip tariant ji yra draugiška vartotojui (angl. „human-friendly“) [1]. PHP yra stabili ir pasiruošusi išspręsti visas iškilusias problemas.

Dauguma iš naudingiausių specialių funkcijų (tokių kaip susijungimų su Oracle duomenų baze inicializavimo ar el. laiškų gavimo iš IMAP serverio) yra iš anksto paruoštos naudojimui. Daug užbaigtų skriptų yra paruošta naujiems vartotojams, kurie tik pradeda mokintis PHP. Iš tiesų kartais yra geriau pradėti mokytis programuoti PHP redaguojant jau kažkieno sukurtus skriptus, nei stengiantis juos nuo pat pradžių rašyti pačiam, bet kokiu atveju vartotojas turės suprasti programavimo PHP pagrindus, tik pasirinkus antrą variantą būtų galima išvengti galimų užtrukimų ir laiko švaistymo bereikalingiems aiškinimams dėl menkiausių klaidų.

Būtina pabrėžti tai, kad „lengvumas“ programuoti yra skirtingai suvokiamas skirtingų žmonių. Kai kurie web programuotojai lengvumą supranta, kaip grafinių elementų tampymo (angl. „drag and drop“) principu, arba „ką matai, tą gali paimti“ (angl. „What You See Is What You Get“) (WYSIWYG) programavimo aplinka. Norint tapti PHP programavimo profesionalu reikia mokėti redaguoti HTML kodą be jokių tam skirtų WYSIWYG redaktorių, o tiesiog ranka rašyti kodą. Nerealu yra tikėtis išmokti programuoti PHP net nežiūrint į programos kodą [3].

Labiausiai pažengę PHP vartotojai (įskaitant tuos kurie priklauso PHP programuotojų bendruomenei) visada programuoja „rankomis“ ir nenaudoja jokių WYSIWYG įrankių. Taip jie elgiasi dėl keleto priežasčių, svarbiausia iš jų yra ta, kad rašant kodą „rankomis“ kodas būna labiau suprantamas ir neužterštas nereikalingais dalykais, taip pat tokį kodą geriausiai interpretuoja

naršyklės. Kodas kurį gauname naudodami WYSIWYG įrankius gali būti tinkamas tik tam tikrom naršyklėms, o kitos atvaizduos puslapį visiškai ne taip, kaip mes tikimės. Taip pat jei programuojant iškyla problemų PHP bendruomenė yra pasiruošusi padėti, tačiau norint gauti pagalbą reikalingas kodas, kuris yra parašytas ranka. Vartotojai naudojantys WYSIWYG įrankius paprastai duoda nuorodas į klaidingus suprogramuotus puslapius, bet neduoda kodo, o tokiu atveju niekas tinkamos pagalbos negalės suteikti.

Apibendrinant galima pasakyti, kad programuoti su PHP yra iš tikrųjų nesudėtinga, ypač tiems, kurie turi bent šiek tiek patirties programuojant su C stiliaus sintakse veikiančiom programavimo kalbomis. PHP yra šiek tiek sudėtingesnė nei HTML, bet turbūt žymiai paprastesne nei JavaScript ir tikrai paprastesnė nei JSP ar ASP.NET.

MySQL gali pasirodyti kiek sudėtinga, jei vartotojas neturi darbo patirties su reliacinėmis duomenų bazėmis arba pereina nuo tokios aplinkos, kuri naudoja Microsoft Access duomenų bazę. MySQL valdymas komandine eilute ir vaizdinės struktūros trūkumas pradžioje gali pasirodyti kiek gluminantis. Ir vėl gi žodis „lengvai suprantamas“ yra reliatyvus. Vienam vartotojui tai gali pasirodyti labai paprasta, o kitam, gali pasirodyti labai sudėtinga ir jis net nenorės pažiūrėti į MySQL pusę. Tačiau, MySQL griežtai laikosi ANSI SQL-92 nustatytų standartų [22], taip pat yra keletas grafinių įrankių kurie padeda administruoti domenų bazę naudojant grafinę aplinka, vieni populiariausių tokių įrankių yra PHPMyAdmin ir MySQL Control Center [3]. Šie įrankiai net ir mažai nusimanančiam vartotojui leidžia sėkmingai administruoti duomenų bazes. Tačiau yra ir vienas jų trūkumas, šie įrankiai nepadės išmokti SQL kalbos ir neišmokys, kaip reikia kurti geros architektūros duomenų bazes.

Įvairiaplatformiškumas

PHP ir MySQL veikia kiekvienoje Unix tipo sistemoje (įskaitant ir Mac OS X) taip pat Windows sistemose. Didžioji dalis visų HTTP serveriu veikia šių dviejų tipų operacinėse sistemose.

PHP yra suderinamas su trimis lyderiaujančiais Web serveriais: Apache HTTP Server, Microsoft Internet Information Server ir Netscape Enterprise Server (iPlanet Server). PHP taip pat veikia ir keliuose mažiau žinomuose serveriuose, tokiuose kaip: Alex Belits' fhttpd, Microsoft's Personal Web Server, AOLServer ir Omnicentrix's Omniserver application server [3]. Specialus Web serverio suderinamumas su MySQL nėra būtinas, kadangi visą suderinamumo darbą atlieka PHP.

Žemiau pateiktoje lentelėje pateiksiu sąrašą operacinių sistemų ir Web serverių, kuriuose veikia PHP.

2. lentelė WEB serveriai, kuriuose veikia PHP

	UNIX	Windows
Operacinė sistema	AIX, A/UX, BSDI, Digital UNIX/Tru64, FreeBSD, HP-UX, IRIX, Linux, Mac OS X, NetBSD, OpenBSD, SCO UnixWare, Solaris, SunOS, Ultrix, Xenix ir daugiau	Windows 95/98/ME Windows NT/2000/XP/2003/Vista
WEB serveris	Apache, fhttpd, Netscape	IIS, PWS, Netscape, Apache, Omni

Šiuo metu PHP veikia ir ant Macintosh sistemų, todėl ji tampa visiškai universalia. Programuotojas gali programuoti, bet kokioje jam patinkančioje operacinėje sistemoje, naudodamas jam labiausiai patinkančius įrankius, tada įkelti suprogramuotus failus į serverį esantį taip pat bet kokioje OS ir viskas veiks puikiai.

PHP yra tikra programavimo kalba. Jei palygintume su ColdFusion, tai ColdFusion yra „žymių“ (angl. Tags) rinkinys, toks kaip HTML [1]. Naudojant PHP galima apsibrėžti savo funkcijas, tokias kokias tik nori, užrašant funkcijos vardą ir pridedant atliekamus veiksmus. Naudodamas ColdFusion, programuotojas turi būtinai naudoti kitų programuotojų sukurtas „žymes“ arba turi pats susikurti reikiamas žymes, o tai yra gana sudėtinga ir užtrunka ne mažai laiko.

Stabilumas

Žodis „stabilumas“ (angl. Stable) yra suprantamas skirtingai:

- Nereikia dažnai perkrovinėti serverio;
- Programinė įranga radikaliai ir dažnai nesikeičia nuo vienos versijos iki kitos;

Iš tikrųjų, abi šias sampratas galima pritaikyti MySQL ir PHP.

Apache serveris yra laikomas vienas stabiliausių Web serverių, kurie yra naudojami šiuo metu internete. Nors jis ir nėra pats greičiausias ir nėra lengviausiai administruojamas, bet kai tik jis yra tinkamai suinstaliuojamas ir paleidžiamas jis beveik niekada „nelūžta“. Jis taip pat nereikalauja serverio perkrovimo po kiekvieno atlikto pakeitimo.

PHP ir MySQL taip pat bus stabilios sistemos ir ateityje, kadangi profesionalių programuotojų komandos su pasitenkinimu dirba prie šių projektų ir labai protingai ir nuodugniai apsvarsto kiekvieną vartotojų norą prieš pradedant jį programuoti [22]. Daugiausiai programuotojų dėmesio atitenka veikimo gerinimui, bendravimui tarp kitų pagrindinių duomenų bazių esančių rinkoje taip pat atliekant geresnius sesijų palaikymo sprendimus. PHP ir MySQL yra geros sistemos dar ir dėl to, kad jos palaiko senesnių sistemų suprogramuotus programinius failus naujesnėse sistemose. Pavyzdžiui kodas suprogramuotas ir paruoštas PHP3 sistemai puikiai veiks ir su PHP4, ir su PHP5 sistemomis. Taip pat ir su MySQL, dėka naudojamų SQL standartų MySQL 3.x duomenų bazės yra nesunkiai perkeliamos į MySQL4 ar MySQL5 [22].

Greitis

PHP kalba yra ypatingai greita, kai vykdomi ja parašyti skriptai, ypač šis greitis pastebimas, kai PHP yra Apache serverio modulis ir visa ši sistema yra suinstaliuota serveryje su Unix pagrindu veikiančia operacine sistema. Tik paleistas MySQL serveris, iš karto vykdo pačias sudėtingiausias užklausas rekordiniu greičiu. Taigi bendra PHP ir MySQL sistema veikia iš ties labai greitai.

Beveik visais atvejais PHP5 yra žymiai greitesnė nei CGI skriptai. Nors dauguma CGI skriptų yra parašyti C kalba, tai yra viena žemiausio lygio kalbų, ir dėl šių priežasčių turėtų būti vykdoma greičiausiai, tačiau problema yra tame, kad po kiekvieno kreipimosi į CGI turi būti sukurtas ir paleistas atskiras procesas [23]. Dėl šių priežasčių yra prarandamas brangus laikas ir daug resursų, taip pat kuo daugiau procesų paleidžiama tuo lėčiau pradeda dirbti visa sistema, kadangi visiem procesams yra išskiriama dalis atminties ir procesoriaus darbo. Dauguma internetinių tinklapių nusprendė nebenaudoti CGI skriptų, kadangi atsirado daug geresnių alternatyvų. PHP yra Apache modulis todėl, kai yra kreipiamasi į PHP modulį, serveris nekurdamas naujų procesų, tiesiog iš karto vykdo atitinkamas operacijas su esamais procesais. Taip sutaupoma ne mažai laiko ir resursų.

Nors PHP veikia ne taip greitai dėl to, kad ji vykdymo metu yra interpretuojama, o ne kompiliuojama [1], bet visą tai su kaupu atperka tai, kad PHP yra HTTP serverio modulis. Kai PHP yra

vykdoma šiuo būdu (kai PHP yra HTTP serverio modulis), jai nereikia kreiptis į serverį ir laukti atsakymų (kaip tai yra su ColdFusion), visos užklausos gali būti įvykdomos akimirksniu.

Nors labai formalių testų nėra atlikta su PHP greičio matavimais, tačiau mažesni testai parodo, kad PHP nė kiek nenusileidžia ASP veikimui ir tikrai veikia greičiau nei ColdFusion ar JSP aplikacijos.

Daug išplėtimų

PHP palengvina komunikavimą su kitomis programomis ar protokolais. Atrodo, kad PHP komanda nusprendė patenkinti maksimalų vartotojų kiekį padarydama PHP labai lanksčia.

Duomenų bazių palaikymas yra ypatingai stipri PHP dalis, su PHP galima jungtis prie apie penkiolikos populiariausiu duomenų bazių taip pat galima jungtis panaudojant ODBC tvarkykles, kas padaro prisijungimą prie duomenų bazių neribotu. Priedo PHP palaiko didelį kiekį protokolų, tokių kaip POP3, IMAP ir LDAP. PHP4 taip pat pradėjo palaikyti Java ir kitų padalintų objektų architektūras (COM ir COBRA). PHP5 pažengė dar toliau suteikdama galimybę pilnai naudotis GD grafine biblioteka, taip pat apdoroti XML su DOM ir simpleXML palaikymu.

Greitas naujovių atsiradimas

Kitų internetinių programavimo technologijų programuotojai, kartais būna nepatenkinti dėl lėto naujovių atsiradimo po jų pripažinimo, kad tai yra naudingi ir vertingi pakeitimai. Su PHP tokių problemų nėra. Visa tai reikalauja bent vieno programuotojo, C kompiliatoriaus ir noro sukurti naująjį funkcionalumą. Aišku tai nereiškia, kad PHP komanda priims kiekvieną vartotojų užgaidą ar rimtą pasiūlymą ir iš karto puls programuoti. PHP pasaulyje viskas vyksta kitaip, kiekvienas PHP programuotojas gali sukurti savo išplėtimus ir jais pasidalinti su kitais PHP programuotojais. Vėliau PHP komanda peržiūri šiuos išplėtimus ir gal būt net prideda prie išleidžiamos naujos PHP versijos. Pavyzdžiui, Dan Libby programuotojo labai šaunus xmlrpc-epi [3] išplėtimas buvo iš karto pridėtas prie naujos PHP4.1 versijos ir šiam įvykiui prireikė tik dviejų mėnesių po šio išplėtimo atsiradimo.

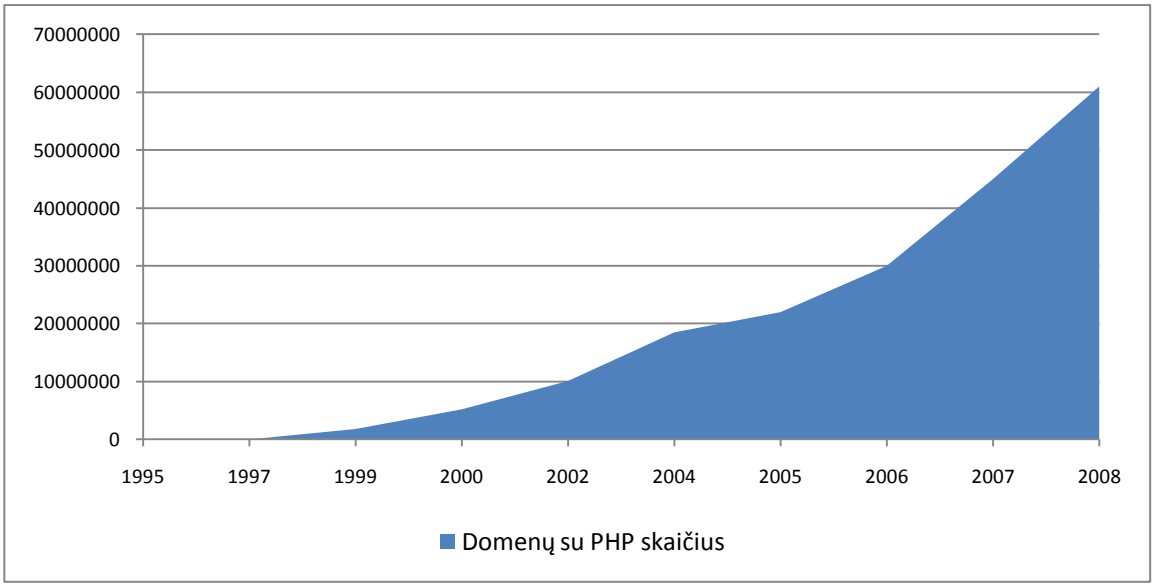
PHP atnaujinama pastoviai ir nuolatos. Nors ir nėra iš anksto žinoma, kada bus išleista nauja PHP versija, bet perėjimas nuo vienos versijos prie kitos su PHP būna visada sklandus. Tai užtikrina greitas patobulinimų išleidimas ir greita reakcija į pastebėtas klaidas. Nors kritinių klaidų būna tikrai nedaug, kadangi naujos versijos būna išleidžiamos su tokia filosofija „išleisk anksčiau, išleisk dažniau“ (angl. „release early, realease often“), kas suteikia programuotojams daug galimybių prisitaikyti prie pakeitimų, taip pat ištaisyti klaidas. Lyginant šį naujų versijų leidimo būdą su pavyzdžiui .NET perėjimais tarp versijų, galima matyti akivaizdų skirtumą. Microsoft po vienos versijos išleidimo

neišleidžia jokių atnaujinimų apie metus laiko ir tik po metų ištaiso kritines klaidas ir įdiegia reikiamus pakeitimus.

To paties kaip apie PHP nebūtų galima pasakyti ir apie MySQL pakeitimų įdiegimus. MySQL taip pat trūksta palaikymo ir naujovių greito atsiradimo pereinant nuo vienos versijos į kitą [3].

PHP ir MySQL populiarumas

PHP vis sparčiau tampa viena iš populiariausių Web programavimo kalbų [19].. Tai galima pamatyti pateiktame grafike 2. paveikslėlyje.



2. pav. PHP naudotojų skaičiaus augimas

2.1.2 Zend programavimo karkaso (Zend Framework) analizė

Maždaug dešimt metų PHP programavimo kalba buvo naudojama kurti dinaminiams internetiniams puslapiams. Iš tikrųjų PHP puslapiai buvo kuriami taip, kad PHP kodas buvo įterpiamas į HTML kodą, viskas būdavo suplakama į vieną krūvą. Toks programavimo būdas yra itin efektyvus, kai PHP kodo nėra daug ir jei žiūrėtumėm tik į puslapio vykdymo laiką. PHP labiausiai išpopuliarėjo su 3 ir 4 versijom todėl tuo pačiu buvo aišku, kad PHP kalba bus kuriami vis didesni ir didesni

projektai. Taip pat darėsi vis aiškiau, kad PHP ir HTML kodo dėjimas į vieną krūva neatneš nieko gero ypač dideliuose projektuose.

Pagrindinės tokio programavimo problemos yra dvi: palaikomumas ir išplečiamumas. Nors PHP sumaišius su HTML kodu gauname labai greitai vykdomą kodą, tačiau vėliau projektui plečiantis tampa vis sunkiau atlikti pakeitimus ar atnaujinimus. Vienas iš smagiausių puslapio talpinimo internete dalykų yra tai, kad puslapio turinys ir išdėstymas visada dinamiškai keičiasi. Dideli puslapiai keičiasi itin dažnai. Pastoviai talpinamas naujas turinys, reguliariai kuriamos naujos duomenų kategorijos. Dėl šių priežasčių buvo pradėti kurti sprendimai, kurie padėtų atskirti HTML nuo PHP ir tuo pačiu palengvintų puslapių atnaujinimą ir pakeitimų įdiegimą.

Zend Programavimo karkasas buvo sukurtas visoms anksčiau minėtoms problemoms spręsti. Jis palengvino PHP kalba parašytų puslapių palaikomumą ilgalaike prasme. Zend programavimo karkasas turi labai daug iš anksto paruoštų komponentų, kuriuos galima panaudoti daug kartų įvairiose puslapio dalyse. Zend Programavimo karkasas naudojamas panaudojant MVC (Model – View – Controller) architektūrą [2]. MVC architektūra taip pat prisideda prie puslapio palaikymo palengvinimo.

Zend Programavimo karkasas yra PHP biblioteka skirta PHP tinklapių kūrimui [2]. Jis suteikia visą rinkinį specialių komponentų, kurie leidžia lengvai sukurti PHP aplikacijas ir vėliau jas lengvai palaikyti ar išplėsti.

2.1.2.1 Zend programavimo karkaso komponentai

Zend Programavimo karkasas yra sudarytas iš daugelio atskirų komponentų sugrupuotų į modulių rinkinius. Kaip visumą programuotojas gauna viską, ko reikia sukurti pažangią internetinę aplikaciją. Programavimo karkasas yra labai lankstus, todėl programuotojas gali pasirinkti ir naudoti tik tuos komponentus, kurie reikalingi pasiekti išsikeltą tikslą. Žemiau esančioje lentelėje pateikiami visi komponentai esantys programavimo karkase [2].

3. lentelė Zend Programavimo karkaso komponentai

Branduolys: Zend_Controller Zend_View	**Bendravimas tarp aplikacijų** (angl. Inter-application communication): Zend_Json Zend_XmlRpc

Zend_Db Zend_Config Zend_Filter & Zend_Validate Zend_Registry **Autentifikacija ir pasiekiamumas** (angl. Authentication and Access): Zend_Acl Zend_Auth Zend_Session **Internacionalizacija** (angl. Internationalization): Zend_Date Zend_Locale Zend_Measure **Http**: Zend_Http_Client Zend_Http_Server Zend_Uri	Zend_Soap Zend_Rest **Web servisai** (angl. Web Services): Zend_Feed Zend_Gdata Zend_Service_Amazon Zend_Service_Flickr Zend_Service_Yahoo **Pažangios technologijos** (angl. Advanced): Zend_Cache Zend_Search Zend_Pdf Zend_Mail/Zend_Mime **Kiti įvairūs komponentai** (angl. Misc) Zend_Measure

Kiekviena grupė sudaryta iš daugelio komponentų, kurių pavadinimai paprastai yra tokie, kokie yra jų pagrindinių klasių pavadinimai. Pavyzdžiui, Zend_View yra aplikacijų naudojama konkreti view klasė. Kiekvienas komponentas taip pat sudarytas ir iš daugelio kitų klasių.

2.1.2.2 Struktūros pritaikymas PHP puslapiams

Norint išsigelbėti nuo betvarkės, kai naudojamas HTML ir PHP kodas kartu, reikia tam tikros puslapio struktūros, kuri padėtu išspręsti šią problemą. Akivaizdžiausias puslapio struktūrizavimo sprendimas būtų išskaidymas atskiras puslapio dalis pagal jų veikimo tikslus. Tai reiškia kad kodas,

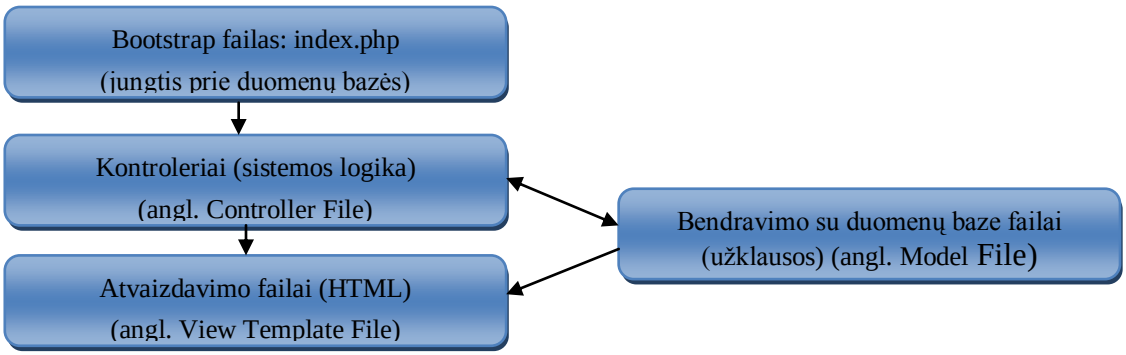
kuris nėra parodomas naršyklės lange ar jungiasi prie duomenų bazės ir ima iš ten duomenis, turi būti atskirtas kitame faile nuo kodo, kuris generuoja vaizdus.

Pirmieji puslapio struktūrizavimo žingsniai paprastai žengiami kiekvieno programuotojo nenaudojant jokių pagalbinių priemonių, visa tai daroma dėl tų pačių priežasčių. Paprastai pirmas dalykas, kuris būna atskiriamas, tai būna kodas, kuriuo jungiamasi prie duomenų bazės, šis kodas būna patalpinamas atskirame faile (pvz.: db.inc.php). Tuomet logiška tampa atskirti ir kodą, kuris prideda puslapyje antraštes (angl. header, footer) ir kuris yra atvaizduojamas kiekviename puslapyje. Tam panaudojamos tam tikros funkcijos, kurios padeda išvengti globalių kintamųjų naudojimo problemas [23].

Kai puslapis auga (didėja), bendras funkcionalumas yra sugrupuojamas į bibliotekas. Šis sprendimas žymiai palengvina puslapio palaikymą ir naujų funkcijų pridėjimą. Šis sprendimas yra geras iki tol kol puslapis netampa labai didelis ir visas bendras sistemos veikimo vaizdas nebetelpa į programuotojo galvą.

Kiekvienas PHP programuotojas tikrai pasakys, kad PHP suteikia lengvą prieigą prie bibliotekų, tokių kaip GD atvaizdų biblioteka, leidžia naudotis daugelio duomenų bazių klientų bibliotekomis ir net leidžia naudotis tokiomis konkrečiomis bibliotekomis kaip Windows COM [2]. Dėl visų šių dalykų buvo aišku, kad PHP turi tapti objektine programavimo kalba. PHP4 klasės nebuvo ypatingai patogios, kadangi neturėjo daugelio ypatybių, kurios būna kitos programavimo kalbose. PHP5 ištaisė šias klaidas ir padarė PHP pilnai objektine kalba.

Pagerintos objektinio programavimo galimybės paskatino sudėtingų bibliotekų, vadinamų programavimo karkasais, atsiradimą. Taip atsirado ir Zend Programavimo karkasas, kuris palaiko tokį puslapio failų grupavimą, kuris prašomas MVC dizaino architektūroje. MVC architektūra pavaizduota žemiau esančiame paveikslėlyje.



3. pav. Zend Programavimo karkase naudojama MVC architektūra

Aplikacija sukurta naudojant MVC architektūrą priverčia išskaidyti atskiras funkcijas į atskirus failus. Kiekvienas failas yra specializuotas, o tai ir padaro sistemos palaikymą daug lengvesniu darbu. Pavyzdžiui: visas kodas, kuris atlieka duomenų bazės užklausas yra saugomas klasėse, kurios kitaip vadinamos Modeliai [2]. Taip pat HTML kodas žinomas, kaip vaizdas (angl. View) (kuris taip pat gali turėti ir šiek tiek PHP kodo logikos) ir Kontrolerių failai kurie užtikrina sistemos logikos veiksmus ir suriša Modelių failus su Vaizdų failais.

Zend Programavimo karkasas ne vienintelis pasirinkimas, kuris gali padėti organizuoti puslapio kūrimą naudojant MVC principus. Rinkoje yra ne viena alternatyva.

2.1.2.3 Zend programavimo karkaso privalumai

Zend Programavimo karkasas pristato visą rinkinį standartizuotų komponentų, kurie žymiai palengvina dinamiško internetinio puslapio programavimo darbą. Tokie puslapiai lengvai gali būti programuojami, palaikomi ir išplečiami.

Pagrindinės galimybės, kurias suteikia Zend Programavimo karkasas yra:

- Viskas vienoje vietoje;
- Moderni architektūra;
- Lengva išmokti ir pritaikyti;
- Pilna dokumentacija;
- Paprastesnis programavimas;
- Greitesnis tikslo pasiekimas.

Viskas vienoje vietoje

Zend Programavimo karkasas yra pažangus pilnai sukomplektuotas framewokr‘as, kuris savyje turi viską ko reikia programuojant sudėtingą sistemą. Naudodamas Zend programavimo karkasą esi priverstas kurti tinklapį pagal visas geriausias tinklapių kūrimo praktikas ir naudoti MVC programavimo architektūrą. Šiame programavimo karkase yra iš anksto paruošti komponentai: autentifikavimo, paieškos, lokalizacijų, PDF kūrimo, el. pašto ir jungčių prie web servisų ir t.t.

Taip pat Zend Programavimo karkasas turi visas galimybes naudoti ir kitas bibliotekas. Pagrindiniai Zend Programavimo karkaso komponentai palengvina puslapio kūrimą, tačiau naudojant

šį programavimo karkasą yra visos galimybės naudoti tokias bibliotekas kaip PEAR, Doctrine ORM ar Smarty [1].

Moderni architektūra

Zend Programavimo karkasas yra parašytas naudojant objektinę PHP5 programavimo kalbą pagal moderniausias architektūros technikas, žinomas kaip architektūros šablonai (pavyzdžiui: Singletone, Factory, Decorator ir kt.) [23]. Programinės įrangos programavimo šablonai yra aukštos klasės programavimo sprendimai, kurie padeda spręsti ypatingai sudėtingas programinės įrangos kūrimo problemas. Zend Programavimo karkasas naudoja daugumą šių šablonų, o jo pačio architektūra buvo sukurta atsižvelgiant į tai, kad jis būtų maksimaliai lanksčiai panaudojamas visų programuotojų be ypatingų pastangų.

Programavimo karkasas leidžia naudoti visas PHP galimybes ir neverčia naudoti visų jau sukurtų komponentu, programuotojas gali laisvai pasirinkti, kurie komponentai padės pasiekti norimą tikslą, o ką jis pats susiprogramuos [23]. Pagrindinis šio programavimo karkaso privalumas yra tas, kad visi komponentai yra labai mažai susiję tarpusavyje, todėl naudojant vieną komponentą nebūtina naudoti kitų.

Lengva išmokti ir pritaikyti

Zend programavimo karkaso architektūra yra tokia, kad jis sudarytas iš atskirų paprastų modulių, kuriuos nesudėtinga perprasti ir išmokti per trumpą laiko tarpą. Kiekvienas komponentas nepriklauso nuo daugelio kitų komponentų todėl ir yra paprasta studijuoti ir mokytis dirbti su šiuo programavimo karkasu. Komponento architektūra yra tokia, kad programuotojui nebūtina suprasti kaip jis veikia, programuotojas gali tiesiog imti ir naudoti jį savo reikmėms be didesnio laiko tarpo praleidimo perprantant kaip šis komponentas veikia. Aišku tuo pačiu patyrę programuotojai gali nesunkiai pridėti reikiamą funkcionalumą išplėsdami kiekvieną komponentą pagal savo reikmes, tai taip pat nėra itin sudėtinga ir apie tai buvo galvojama kuriant šį programavimo karkasą.

Pavyzdžiui, konfigūracijos komponentas Zend_Config [24] yra naudojamas suteikti objektinio programavimo interfeisą konfigūracijos failui. Jis palaiko dvi pažangias ypatybes: „section overloading“ ir „nested keys“, bet nei viena iš šių ypatybių nebūtinai turi būti suprasta ir išmokta norint panaudoti šį komponentą. Kai tik programuotojas pradeda naudoti Zend_Config komponentą savo

kode, jo patirtis ir pasitikėjimas didėja ir vėliau jis laisvai perpras ir panaudos pažangias šio komponento ypatybes.

Pilna dokumentacija [24]

Nesvarbu koks geras programos kodas bebūtų, bet trūkumas dokumentacijos gali paversti visą darbą niekais. Zend Programavimo karkaso programuotojai norėdami patys pasilengvinti sau darbą ir padaryti taip kad kiti suprastų k aje daro visą laiką lygiagrečiai programavimui dokumentuoja visa tai ką jie daro. Jei tik atsiranda naujas kodas jis turi būti dokumentuojamas, kitaip jis nepridedamas prie programavimo karkaso. Tokia logika labai pravertė, nes visas Zend Programavimo karkasas yra detaliai dokumentuotas.

Zend Programavimo karkasas turi dviejų tipų dokumentacijas: API ir galutinio vartotojo (angl. end-user). API dokumentacijas yra sukurta panaudojant „PHPDocumenter“ ir yra automatiškai sugeneruojama panaudojant specialius „dockblock“ komentarus programos kode. Šiuos komentarus paprastai galima rasti virš kiekvienos klasės, funkcijos ar kintamojo inicializacijos.

Zend Programavimo karkasas taip pat palaiko ir pilną dokumentaciją, kuri yra talpinama internete adresu http://framework.zend.com/manual [24]. Ši dokumentacija pateikia detalius visų komponentų ir jų veikimo bei panaudojimo aprašymus. Taip pat pateikiami ir panaudojimo pavyzdžiai pradedantiesiems. Taip pat prie sudėtingesnių komponentų pateikiama ir veikimo teorija, tam kad programuotojas suprastų kodėl komponentas veikia būtent taip, o ne kitaip.

Dokumentacija pateikiama su programavimo karkasu nepaaiškina, kaip visi komponentai veikia su kitais komponentais, kaip juos naudoti bendroje krūvoje. Dėl šios priežasties labai daug internete atsirado labai daug įvairių pamokančių straipsnių (angl. tutorials) šiomis temomis, čia pasistengė visa programuotojų bendruomenė, kuri pamatė Zend Programavimo karkaso privalumus ir sėkmingai jį naudoja [2].

Paprastesnis programavimas

Kaip jau minėjau, viena iš Zend Programavimo karkaso stipriųjų pusių yra jo paprastumas programuojant dinaminiu internetinius puslapius. Taip padėjo daugeliui pasaulio vartotojų susikurti dinaminius internetinius puslapius, kurių seniau jie negalėjo susikurti. Zend Programavimo karkasas sukurtas taip, kad padėtų kurti internetinius puslapius ir pradedančiajam programuotojui ir pažengusiam.

Pagrindinis dalykas dėl ko Zend karkasas programavimą padaro paprastesniu, tai yra tai kad visos pagrindinės galimybės ir funkcijos, kurias suteikia programavimo karkasas yra pilnai ištestuotos ir visas kodas, kuris suteikia tam tikrą funkcionalumą yra 100% patikimas [2]. Tai reiškia, kad kodas, kurį rašo programuotojas yra tik tas kodas, kurio reikia pasiekti trūkstamam funkcionalumui, visą kitą „nuobodų“ kodą suteikia programavimo karkasas, taip palengvindamas sistemos kūrimą.

Greitesnis tikslo pasiekimas

Zend Programavimo karkasas padeda greičiau kurti internetinę sistemą ar pridėti naują funkcionalumą esamai sistemai [23]. Kadangi programavimo karkasas padengia labai daug įvairių funkcionalumų ir programuotojui nereikia jų programuoti, jis gali daugumą laiko skirti sistemos pagrindiniams funkcionalumams pasiekti. Taip pat yra paprasčiau pradėti darbą ir greičiau pamatyti norimą rezultatą.

Zend programavimo karkasas ir prieigos teisių valdymas

Zend programavimo karkase prieigos teisių valdymui yra sukurtas atskiras komponentas pavadintas `Zend_Acl`[24]. `Zend_Acl` įgalina programuotojus sukurti labai lankstų prieigos kontroliavimo mechanizmą panaudojant prieigos sąrašą (angl. Access control list) (ACL) skirtą vartotojų privilegijų kontroliavimui. Bendrai paėmus, ACL sąrašas įgalina prieigos kontroliavimą prie apsaugotų objektų nuo kitų reikalaujančių prieigos subjektų. Yra išskiriamos dvi objektų rūšys kalbant apie `Zend_Acl` komponentą [23]:

- *Resursas* yra objektas, prie kurio prieiga yra kontroliuojama.
- *Rolė* yra objektas, kuris gali reikalauti prieigos prie *Resurso*.

Kalbant paprastai, rolės reikalauja prieigos prie resursų. Pavyzdžiui, jei pirkėjas nori patekti į parduotuvę, tai pirkėjas yra reikalaujanti prieigos rolė, o parduotuvė yra resursas, kadangi įėjimas į parduotuvę gali būti apribojamas tam tikriems pirkėjams.

Resursai

Resursų sukūrimas naudojant `Zend_Acl` yra labai paprastas. `Zend_Acl` sukūrė atskirą klasę `Zend_Acl_Rsource_Interface`, kuri palengvina resursų kūrimą aplikacijoje. Klasei reikia tik išvystyti šį interfeisą, kuris susideda tik iš vieno metodo `getResourceId()`, taigi `Zend_Acl`

atpažįsta objektą, kaip resursą. Taip pat `Zend_Acl_Resource` yra pateikiamas tam, kad programuotojai galėtų papildyti `Zend_Acl` veikimą pagal savo poreikius.

`Zend_Acl` suteikia medžio struktūrą į kurią gali būti įtraukiama daugybė resursų. Kadangi resursai saugomi tokiame medžio struktūroje, jie gali būti rūšiuojami nuo pagrindinio (prie medžio šaknų) iki specifinio (prie medžio lapų). Konkrečių resursų užklausos automatiškai ieškos, resursų hierarchijoje, taisyklių, kurios priskirtos protėviniams resursams, leidžiantiems paprastai paveldėti taisykles. Pavyzdžiui, jei pagal nutylėjimą taisyklė pritaikoma kiekvienam pastatui mieste, viena taisyklė būtų priskirta visam miestui, vietoj tos taisyklės priskyrinėjimo atskiriems pastatams. Kai kurie pastatai gali būti išskirtiniai ir turėti kitokias taisyklės, toks atvejis yra numatytas `Zend_Acl`, tai gali būti pasiekiama pridedant netikėtų situacijų (angl. exceptions) rinkinius prie kiekvieno pastato, kuris reikalauja išskirtinių taisyklių. Resursas gali paveldėti taisykles tik iš vieno tėvinio resurso, o šis tėvinis resursas gali būti paveldėjęs taisykles iš jo pačio tėvinio resurso.

`Zend_Acl` taip pat palaiko ir resursų privilegijas (pavyzdžiui, „skaityti“, „rašyti“,“atnaujinti“,“trinti“), taigi programuotojai gali priskirti taisykles, kurios veikia vienus resursus vienaip pagal turimas privilegijas, o kitus, pagal jų turimas privilegijas, kitaip [24].

Rolės

Kaip ir su resursais, rolių kūrimas yra labai paprastas. Visos rolės turi išplėsti `Zend_Acl_Role_Interface` klasę. Šis interfeisas susideda iš vieno metodo `getRoleId()`. Taip pat, `Zend_Acl` pateikia ir `Zend_Acl_Role` objektą, kurį naudodami programuotojai gali išplėsti ar papildyti `Zend_Acl` pagal savo poreikius [24].

Naudojant `Zend_Acl` rolė gali būti paveldėta iš vienos ar daugiau rolių. Tai padaryta dėl taisyklių paveldėjimo iš kitų taisyklių palaikymo. Pavyzdžiui, vartotojo rolė „Jonas“, gali priklausyti vienai ir daugiau tėvinių rolių, tokių kaip „redaktorius“ ar „administratorius“. Programuotojai gali priskirti roles „redaktoriui“ ir „administratoriui“ atskirai ir „Jonas“ paveldėtų taisykles iš abiejų rolių, be tiesioginio abiejų rolių priskyrimo „Jonui“.

Nors ir galimybė paveldėti roles iš daugybės rolių yra galima, tačiau toks paveldimumo naudojimas gali padaryti programą labai sudėtingą.

Prieigos kontrolės sąrašų kūrimas

Prieigos kontrolės sąrašas (angl Acces Contro List (ACL)) gali atstovauti bet kokių fizinių ar virtualių objektų rinkinį. Norint sukurti naują ACL sąrašą, reikia inicijuoti ACL be jokių parametrų.

```
$acl = new Zend_Acl();
```

Iki kol vartotojas nesukuria, kokios nors „leidžiančios“ taisyklės, `Zend_Acl` neleis prieiti nei prie vieno objekto.

Rolių registravimas

Kuriamos didesnės sistemos beveik visada reikalaus, leidimų (angl permissions) hierarchijos tam kad būtų galima apibrėžti skirtingas skirtingų vartotojų galimybes [2]. Sistemoje gali būti „svečių“ grupė, leidžianti apribotą prieigą prie sistemos, „darbuotojų“ grupė, galinti atlikti paprastas - kasdienines sistemos operacijas, „redaktorių“ grupė galinti publikuoti, peržiūrėti, archyvuoti ir šalinti skelbiamą informaciją, taip pat „administratorių“ grupė, kurie galėtų atlikti visų aukščiu paminėtų grupių darbus ir taip pat valdyti jautrią informaciją, skirstyti vartotojų roles, daryti sistemos kopijas ir t.t. Toks rolių paskirstymas gali būti padarytas registruojant roles, leidžiant paveldėti kiekvienai grupei prieš ją buvusių grupių teises, taip pat užtikrinant unikalias teises, kurias turi tik ta vienintelė grupė. Leidimai gali būti paskirstomi taip kaip pavaizduota lentelėje.

4. lentelė Leidimų sistemoje paskirstymas

Rolės pavadinimas	Unikalūs leidimai	Paveldėti leidimai iš
Svečias	Peržiūrėti	-
Darbuotojas	Redaguoti, išsaugoti, revizuoti	Svečias
Redaktorius	Publikuoti, archyvuoti, šalinti	Darbuotojas
Administratorius	Suteikiamos visos teisės	-

Programoje toks paskirstymas atrodytų taip:

```
$acl = new Zend_Acl();
// Prideam grupes panaudodami Zend_Acl_Role
// Svečias nepaveldi jokių leidimų
$roleSvecias = new Zend_Acl_Role('svecias');
$acl->addRole($roleSvecias);
// Darbuotoji paveldi leidimus iš svečio
$acl->addRole(new Zend_Acl_Role('darbuotojas'), $roleSvecias);
// Redaktorius paveldi leidimus iš darbuotojo
$acl->addRole(new Zend_Acl_Role('redaktorius'), 'darbuotojas');
// Administratorius nepaveldi jokių teisių
$acl->addRole(new Zend_Acl_Role('administratorius'));
```

Priskyrus ACL sąrašui atitinkamas roles, reikia nustatyti, kaip resursai bus pasiekiami šių rolių atstovų. `Zend_Acl` klasė suteigia galimybę priskirti tik pačią žemiausią ir pačią aukščiausią roles, todėl kad resursai ir rolės paveldi taisykles, kurios buvo priskirtos jų protėviams, taip sumažinant reikalingų taisyklių skaičių. Dėl šių priežasčių galima sudaryti sudėtingą rinkinį taisyklių su minimaliu programinio kodo kiekiu. Programoje toks priskyrimas atrodytų taip:

```
// Svečiui leidžiama tik peržiūrėti turinį
$acl->allow('svecias', null, 'view');
// darbuotojas paveldi teises iš svečio, tačiau jam reikia dar ir kitų leidimų
$acl->allow('darbuotojas', null, array('edit', 'submit', 'revise'));

// taip pat ir redaktorius
$acl->allow('redaktorius', null, array('publish', 'archive', 'delete'));
// administratorius nepaveldi nieko, tačiau jam suteikiamos visos teisės
$acl->allow('administratorius');
```

Su šiais programinio kodo pavyzdžiais sudaromas ACL sąrašas, kuris gali būti naudojamas sužinojimui ar prieigos užklausėjas turi teisė atlikti veiksmus ar ne. Šių užklausų atlikimas yra labai supaprastinamas naudojant `isAllowed()` metodą:

```
echo $acl->isAllowed('guest', null, 'view') ?
     "allowed" : "denied";
// leidžiama
echo $acl->isAllowed('staff', null, 'publish') ?
     "allowed" : "denied";
// neleidžiama
echo $acl->isAllowed('administrator', null, 'update') ?
     "allowed" : "denied";
// leidžiama, nes administratorius turi visas teises
```

2.1.2.4 Zend programavimo karkaso (Zend Framework) alternatyvos

Kadangi PHP programavimo kalba suteikia labai daug galimybių, todėl tikrai nėra tokio programavimo karkaso , kuris tiktų ir patiktų kiekvienam programuotojui. PHP pasaulyje yra labai daug programavimo karkasų, kurie traukia programuotojų dėmesį ir kiekvienas jų turi savų pliusų ir minusų. Žemiau esančioje lentelėje pateiksiu keleto populiariausių PHP programavimo karkasų palyginimus [2].

5. lentelė Zend Framework alternatyvų apžvalga

	Zend Framewok	Cake	Code Igniter	Solar	Symfony
Pilnai naudoja PHP5 funkcionalumą	Taip	Ne	Ne	Taip	Taip
Nustatyta katalogų struktūra	Ne	Taip	Taip	Ne	Taip
Oficialus internacionalizacijos palaikymas	Taip	Taip	Ne	Taip	Taip
Konfigūruojant reikia naudoti komandinę eilutę	Ne	Ne	Ne	Ne	Taip
Reikalauja konfigūravimo	Taip	Ne	Ne	Taip	Taip
Patekta išsami ORM	Ne	Taip	Ne	Ne	Taip
Gera dokumentacija ir pamokantys straipsniai	Taip	Taip	Taip	Taip	Taip
Įmanomas Unit testavimas	Taip	Ne	Ne	Taip	Taip
Bendruomenės palaikymas	Taip	Taip	Taip	Taip	Taip
Licencija	New BSD	MIT	BSD-style	New BSD	MIT

2.2 Programinės įrangos sistemų saugumo ir prieigos teisių kontrolės analizė

Vienas iš svarbiausių sistemų reikalavimų dažnai būna informacijos ir resursų apsaugojimas nuo neteisėto jos peržiūrėjimo ar modifikavimo tuo pačiu užtikrinant galimybę teisėtiems vartotojams pasinaudoti šia informacija. Informacijos saugumo problemos egzistuoja nuo pat to laiko, kai pradėta koncentruotai kaupti, saugoti ir valdyti informaciją. Tačiau, nors technologijų ir informacijos valdymo sistemų pažangumas vis didėja ir tampa vis galingesniais, informacijos apsaugojimo problema taip pat darosi vis kritiškesnė. Šiuo metu yra daugiau nei bet kada įvairių kompanijų, kurios yra priklausomos nuo informacijos, kurią valdo ir kuria varijuodamos daro savo verslą. Saugumo pažeidimai gali sužlugdyti ar paralyžiuoti kompanijų darbą ir atnešti labai didelius finansinius nuostolius. Ligoninės, bankai, viešos administravimo įmonės ir privačios organizacijos, visi yra priklausomi nuo informacijos, kuria jie naudojasi, tikslumo, pasiekiamumo ir konfidencialumo.

Fundamentalus sprendimas padedantis užtikrinanti informacijos saugumą yra *prieigos teisių valdymas*, kurio užduotis yra kontroliuoti kiekvieną prieigą prie kompiuterių sistemos ir jos resursų ir užtikrinti, kad galimi prisijungimai yra tik autorizuoti. Šiam tikslui pasiekti, kiekviena informacijos valdymo sistema savyje turi prieigos kontrolės paslaugą, kuri padeda naudojant tam tikrą specifikavimo

kalbą nustatyti prieigos prie informacijos taisykles [7]. Naudodami pateikiamą interfeisą saugumo administratoriai gali apibrėžti prieigos kontroliavimo politiką (ar politikas), kurios turi būti laikomasi norint prieiti prie tam tikros informacijos.

Prieigos kontrolės politikos apibrėžimas nėra paprastas procesas. Vienas iš pagrindinių sunkumų yra realaus pasaulio saugumo politikų interpretavimas, kuris dažnai yra sudėtingas ir nevienareikšmiškas. Daugelis realaus pasaulio situacijų turi sudėtingas politikas, kai prieigos sprendimai priklauso nuo skirtingų taisyklių rinkinių, pavyzdžiui valstybinių įstatymų ar organizacijos vidinių taisyklių. Saugumo politikos turi apimti skirtingas tvarkas ir tuo pačiu turi įvertinti visas galimas grėsmes, kurios galėtų nutikti kompiuterių sistemoms. Šis sudėtingas scenarijus sudarytas, tam kad galima būtų patenkinti skirtingus reikalavimus, kurie gali iškilti skirtingose sistemose, tuo pačiu turi būti užtikrinamas ir sistemos kontroliavimo paprastumas ir išplečiamumas.

Prieigos kontrolė kompiuterių saugumo kontekste apima autentifikaciją, autorizaciją ir auditavimą. Ji taip pat apima ir techninius įrenginius tokius kaip biometriniai skeneriai, metalo detektoriai, skaitmeniniai parašai, kodavimas, socialiai kodai ar žmonių ir automatizuotų įrenginių priežiūra.

Kiekviename prieigos kontroliavimo modelyje esybės , kurios gali atlikti tam tikrus veiksmus sistemoje yra vadinamos *subjektais* ir esybės, prie kurių prieigos teisės yra kontroliuojamos yra vadinamos *objektais*. Subjektai ir objektai turi būti laikomi pagrindinėmis programinės įrangos prieigos kontroliavimo esybėmis, labiau nei žmogus – sistemos vartotojas, kadangi kiekvienas vartotojas gali paveikti ar kontroliuoti sistemą tik per šias esybes. Kai kurios sistemos nurodo subjektams vartotojo ID numerius, tam, kad vartotojo paleisti procesai pagal nutylėjimą iš karto turėtų tą pačią reikšmę. Tokia kontroliavimo sistema nėra aukšto lygio todėl ji neužtikrina apsaugos nuo nepageidaujamų prisijungimų.

Kai kuriuose modeliuose, pavyzdžiui „Object-capability“ modelyje, visos programos esybės potencialiai gali elgtis ir kaip subjektai ir kaip objektai [6].

Dabartinėse sistemose prieigos teisių modeliai paprasti būna dviejų tipų: paremti "*Gebėjimais*" (angl. Capabilities) ir tie, kurie paremti *prieigos teisių sąrašais* (angl. access control lists (ACL)).

"Gebėjimais" pagrįstas modelis yra grindžiamas pagrindiniu dalyku - "*gebėjimu*", kuris leidžia prieiti prie vieno objekto, o turint prieigą prie vieno objekto vartotojui suteikiama prieiga prie visų kitų susijusių objektų (grubiai tariant, kažkas panašaus į tai, kaip raktas suteikia prieiga prie buto ir visų jame esančių daiktų). Prieiga prie kitų objektų perduodama per saugų kanalą [6].

Prieigos sąrašais paremtose (ACL) sistemose, subjektas gali turėti prieigos teises prie objekto priklausomai nuo to ar subjekto ID yra įtrauktas į sąrašą susietą su tuo objektu [16] (pavyzdžiui, tai būtų panašu į tai, kai einant į organizuojamą vakarėlį būtų patikrinama tavo asmenybė ir ar tu esi svečių sąraše). Prieiga suteikiama papildant sąrašą. Skirtingos ACL sistemos turi įvairias sąrašų pildymo sistemas.

Abu "gebėjimais" pagrįstas ir ACL pagrįstas modeliai turi mechanizmus leidžiančius suteikti prieigos teises visiems subjektų grupės nariams (dažnai pati grupė yra laikoma kaip subjektas).

Prieigos teises kontroliuojančios sistemos suteikia svarbią pagalbą *identifikuojant ir autentifikuojant* (angl. identification and authentication), *autorizuojant* (angl. authorization) ir užtikrinant *atskaitomybę* kur [21]:

- *Identifikacija ir autentikacija* apibrėžia tai, kas gali prisijungti prie sistemos taip pat nurodo asociacijas su programinės įrangos dalimis, kurias prisijungę vartotojai gali valdyti.
- *Autorizacija* apibrėžia ką subjektas gali daryti;
- *Atskaitomybė* identifikuoja ką subjektas atliko.

Identifikacija ir autentikacija

Identifikacija ir autentikacija yra asmenybės patikrinimo procesai, užtikrinantys, kad asmuo turi priėjimą prie esybės. Šie procesai atlieka pradinį tapatumo patikrinimą, kurio metu nustatomas *autentikatorius*. Vėliau, esybės tapatumo nustatymas vykdomas kartu su autentikatoriumi, kaip priemonė validacijai. Vienintelis iškylantis reikalavimas identifikatoriui yra tai, kad jis turi būti unikalus savo saugumo srities viduje.

Paprastai autentikatoriai būna paremti vienu iš faktorių [21]:

- *Kažkas ką tu žinai*, pavyzdžiui slaptažodis, asmens identifikavimo numeris (PIN). Tai užtikrina, kad tik sąskaitos savininkas žino slaptažodį ar PIN kodą, reikalingą pasiekti sąskaitą.

- *Kažkas ką tu turi*, pavyzdžiui protingoji kortelė (angl. Smart card) ar saugumo žetonas (angl. Security token). Tai užtikrina, kad prie sąskaitos gali prieiti tik sąskaitos savininkas, kuris turi išmaniąją kortelę ar saugumo žetoną, kurie reikalingi sąskaitos atrakinimui.
- *Kažkas kuo tu esi,* pavyzdžiui pirštų antspaudai, balsas, akies rainelė ar kiti biometriniai duomenys.
- *Kur tu esi*, pavyzdžiui už kompanijos ugniasienės ar prieš ją, apytikslė jungimosi prie sistemos vieta, nustatoma GPS įrenginiais.

Autorizacija

Autorizacija labiau siejama su subjektais nei su vartotojais (asociacijos tarp vartotojo ir subjektų pagrinde yra kontroliuojamos vartotojų, kurie yra apibrėžti identifikacijos ir autentikacijos metu). Autorizacija apibrėžia ką subjektas gali daryti sistemoje [9]. Moderniausios operacinės sistemos nurodo leidimų rinkinius, kurie yra įvairios variacijos, pagrinde trijų tipų prieigų:

- *Skaitymo* (angl. Read (R)) – subjektas gali:
    - Skaityti failo turinį;
    - Peržiūrėti katalogo turinį;
- *Rašymo* (angl. Write (W)) – subjektas gali atlikti failo ar katalogo pakeitimus naudodamasis šiomis teisėmis:
    - Pridėjimo (angl. Add);
    - Sukūrimo (angl. Create)
    - Šalinimo (angl. Delete)
    - Pervadinimo (angl. Rename)
- *Vykdymo* (angl. Execute (X)) – jei failas yra programa, subjektas gali paleisti šią programą.

Šios teisės ir leidimai programuojant yra naudojami skirtingai sistemose paremtose savarankiškos prieigos kontrole ir privalomos prieigos kontrole.

Atskaitomybė

Atskaitomybė naudoja tokius sistemos komponentus, kaip *pakeitimų sekimas* (angl. audit trail) ir sisteminių *įrašų kaupimas* (angl. logs) tam, kad būtų galima susieti subjektus su veiksmais. Įrašyta informacija turi būti tinkama susieti subjektą su kontroliuojančiu vartotoju. Pakeitimų sekimai ir sisteminių įrašų kaupimas yra svarbūs:

- Aptinkant saugumo sutrikimus
- Atkuriant saugumo incidentus

Jei niekas reguliariai neperžiūri kaupiamų įrašų ir jie nėra saugiai ir pastoviai palaikomi, jie gali būti netinkami, kaip saugumo sutrikimų įrodymai.

Daugelis sistemų gali automatiškai generuoti ataskaitas paremtas iš anksto numatytais kriterijais, dar žinomais kaip "*karpymo lygiai*" (angl. clipping levels) [21]. Pavyzdžiui, karpymo lygis gali būti nustatytas sugeneruoti ataskaitą tam, kad ataskaitoje būtų matomi:

- Daugiau nei trys nepavykę prisijungimo bandymai per tam tikrą laikotarpį;
- Bet koks bandymas panaudoti išjungtą vartotojo sąskaitą;

Tokios ataskaitos padeda sistemos administratoriams lengviau identifikuoti galimus įsilaužimus.

2.2.1 Prieigos kontrolės koncepcijos

Atsakingumas ir patikimas informacijos įvedimas. Prieigos kontrolė turi būti paremta teisingu ir patikimu atpažinimo informacijos įvedimu [4]. Šio paprasto principo yra laikomasi ne visada, sistemose, kuriose prieigos kontrolės sistemos taisykles galima kurti remiantis nepatikima informacija. Tokios sistemos pavyzdys galėtų būti, sistema, kuri naudoja prieigos teisių kontrolę pagal buvimo vietą (angl. location-based) ir vietovei nustatyti naudoja IP adresą. Tokią sistemą galima nesunkiai apgauti paduodant netikrą IP adresą ir apgaunant prieigos kontrolės sistemą.

Mažų ir didelių specifikacijų palaikymas. Prieigos kontrolės sistema turi leisti kurti įvairaus sudėtingumo prieigos taisykles, kas leistų sukurti labai daug prieigos taisyklių (naudojant didelę reikalavimų specifikaciją) prie vieno objekto, tuo pačiu turi būti įmanoma kurti ir ne itin detalias taisykles (naudojant maža ir nesudėtingą specifikaciją) [4]. Reikia paminėti, kad kuriant labai daug įvairių taisyklių, kiekvienam vartotojui ar objektui, labai apsunkinamas administratoriaus darbas. Be to, vartotojų grupės ir objektų kolekcijos dažnai naudojasi tais pačiais prieigos kontrolės reikalavimais. Tokiu atveju prieigos kontrolė turi palaikyti autorizaciją skirtą vartotojų grupėms, objektų grupėms ar

net veiksmų grupėms. Taip pat daugelyje organizacijų prieigos gali būti paskirstomos pagal organizacijos veiklas, tokiu atveju prieigos kontrolės sistema turi palaikyti autorizaciją paremta organizacijos rolėmis.

Sąlyginė autorizacija. Gali būti reikalaujama kad apsauga priklausytų nuo tam tikrų sąlygų įvertinimo [4]. Sąlygos gali būti paprasčiausios formos sistemos predikatai, tokie kaip data ir prisijungimo vieta. Pavyzdžiui, gali būti tokia sąlyga, kad vartotojas gali prisijungti prie sistemos tik nuo 9.00 iki 17.00. Sąlygos gali taip pat priklausyti nuo informacijos, kurią bandoma pasiekti.

Mažiausio privilegijuotumo principas. Mažiausio privilegijuotumo principas yra tas, kai kiekvienas subjektas sistemoje turi veikti su galimai mažiausiu teisių rinkiniu reikiamo tikslo pasiekimui ar užduoties įvykdymui. Vartotojai autorizuoti su galingais teisių rinkiniais nenaudoja visų priskirtų teisių vykdydami savo užduotis iki tol kol kartais iš tiesų jiems prireikia visų teisių. Todėl įprastam sistemos vartojimui tokie vartotojai galėtų autorizuotis, kaip mažiau teisių turintys vartotojai, o tik reikalui esant jungtis prie sistemos su visomis turimomis teisėmis. Toks principas sumažina netyčinius duomenų sugadinimus arba sumažina tikimybę, kad bus perimti daug teisiu turinčio vartotojo prisijungimo duomenys [21].

Atsakomybių išskaidymas. Atsakomybių išskaidymo principas teigia, kad nei vienam vartotojui neturi būti suteikiama tiek teisių, kad būtų įmanoma naudoti sistemą netinkamai [8]. Nors atsakomybių išskaidymas geriau klasifikuojamas, kaip apribojimai politikos specifikacijoje, atsakomybių teisių išskaidymo palaikymas reikalauja, kad saugumo sistema būtų ganėtinai išraiškinga ir lanksti, tam kad galima būtų naudoti įvairius prieigos apribojimus.

Daugelis politikų ir išimčių. Tradiciškai, savarankiškos prieigos politikos yra išskaidomos į dvi klases: uždarą ir atvirą. Populiaresnėje, uždaroje politikoje yra apibrėžiama tik autorizavimo prieiga. Kiekviena užklausa yra autorizuojama ir tik pavykus autorizacijai yra vykdoma. Kitoje, uždaroje prieigos politikos klasėje, autorizacija apibrėžia tik tuos atvejus kai prieiga negali būti leidžiama. Visos prieigos užklausos, kurioms nėra priskirtų neigiamų autorizacijų yra vykdomos pagal nutylėjimą.

Politikų kombinacija ir konfliktų sprendimai. Jei apibrėžiant prieigos teisių taisykles egzistuoja daug modulių, prieigos teisių sistema turi apibrėžti, kaip skirtingi moduliai turi bendrauti tarpusavyje [4]. Pavyzdžiui jų sąjungoje arba, kai jie veikia atskirai. Dažnai atsitinka tokios situacijos, kai vieno

modulio prieigos teisės leidžia priėjimą, o kito modulio neleidžia, tokiu atveju turi būti apibrėžtos tam tikros taisyklės, kurios pagal nutylėjimą galioja visiems vartotojams, visuose moduliuose. Tačiau tokiu atveju visada atsiranda tokių situacijų, kad šios taisyklės veikia netinkamai. Ir šitoje vietoje nieko pakeisti negalima, nes jei pakeisi taisykles iš karto atsiras tokių kuriems šie pakeitimai bus netinkami.

Administravimo politikos. Administravimo politikos reikalingos tam, kad reguliuotų prieigos taisyklių sudarymą. Jos nurodo kas gali pridėti naujas, ištrinti ar modifikuoti egzistuojančias taisykles. Paprastai nesudėtinga administravimo politika gali tarnauti daugeliui skirtingų aplikacijų. Teorijoje rašoma, kad savarankiškos sistemos gali palaikyti skirtingas administravimo politikų rūšis [4]:

- *Centralizuota* – kai privilegijuotas vartotojas ar jų grupė yra rezervuoti privilegijai, kuri leidžia suteikti ir panaikinti privilegijas;
- *Hierarchinė* – kai grupė privilegijuotų vartotojų yra rezervuota privilegijai, kuri leidžia suteikti ar panaikinti privilegijas;
- *Nuosavybinė* – kai kiekvienas objektas yra asocijuotas su savininku (paprastai objekto sukūrėju), kuris gali suteikti ar atimti priėjimą prie jo objekto;
- *Decentralizuota* – kuri praplečia anksčiau aprašytus požiūrius ir leidžia objekto savininkui ar administratoriui deleguoti kitus vartotojus būti šio objekto administratoriais, kurie galėtų taip pat deleguoti kitus vartotojus.

Dėl savo paprastumo ir plataus pritaikomumo populiariausia yra nuosavybinė politika decentralizuota administravimo politika dažnai gali būti matoma duomenų bazių valdymo sistemose. Decentralizuota politika yra patogi, todėl kad ji leidžia vieniems vartotojams deleguoti administratoriaus teises kitiems vartotojams. Tačiau naudojant ją yra sunku atsekti, kas gali prieiti prie kokių objektų, taip pat laibai sudėtingas tampa autorizacijų panaikinimas.

2.2.2 Prieigos valdymo modeliai

Tradiciškai prieigos valdymo technikos yra skirstomos į *privalomo prieigos valdymo* (angl. Mandatory Access Control) (MAC) ir *savarankiško prieigos valdymo* (angl. Discretionary Access Control) (DAC) modelius. MAC modelį naudojančiose sistemose sistema nustato prieigos prie objektų kontrolės griežtumą remdamasi saugumo politika (angl. security policy) ir subjektu, kuris nori vykdyti tam tikrą operaciją su objektu. DAC modelį naudojančiose sistemose prieigos prie objekto

kontroliavimas yra paliekamas savarankiškam subjekto, kuriam priklauso objektas, kontroliavimui. Ne per seniausiai atsirado ir trečia alternatyva, kuri papildė prieigos kontrolės modelių gretas, tai yra *rolėmis paremtas prieigos kontrolės modelis* (angl. role-based access control) (RBAC), kuris yra pakankamai galingas ir gali simuliuoti abu prieš tai paminėtus modelius MAC ir DAC [6].

2.2.2.1 Privalomo prieigos valdymo modelis (MAC)

Sistemose, kuriose naudojamas privalomo prieigos valdymo modelis prieigos politika yra nustatoma pačios sistemos, o ne objekto savininko [21]. MAC modelis naudojamas daugiasluoksnėse sistemose, kurios dirba su labai jautriais, svarbiais duomenimis, tokiais kaip valdžios ar karinė informacija [6]. Daugiasluoksnė sistema yra vieno kompiuterio sistema, kuri valdo daugybę klasifikavimo lygių tarp subjektų ir objektų. Klasifikavimo lygių pavyzdys pateiktas 6. lentelė .

6. lentelė Bendros prieigos klasifikavimo antraštės mažėjančia tvarka

Klasifikacija
Labai slapta
Slapta
Konfidencialu
Neklasifikuota

- *Jautrumo etiketės* (angl. sensivity labels): MAC paremtose sistemose visi objektai ir subjektai turi turėti priskirtas jautrumo etiketes. Subjekto jautrumo etiketė apibrėžia pasitikėjimo lygmenį. Objekto jautrumo etiketė apibrėžia pasitikėjimo lygmenį reikalingą prieigai [21]. Tam, kad gautų prieigą prie duoto objekto, subjektas turi turėti lygų arba aukštesnį jautrumo lygmenį nei turi reikalautas objektas.
- *Duomenų importas ir eksportas*: Informacijos importavimo iš kitų sistemų ir eksportavimo į kitas (įskaitant ir spausdintuvus) kontroliavimas yra kritinė MAC paremtų sistemų funkcija,

kuri turi užtikrinti kad jautrumo etiketės yra tinkamai palaikomos ir įgyvendintos, tam kad jautri informacija būtų visada tinkamai apsaugota.

Paprastai yra naudojami du metodai, kurie naudoja MAC modelį:

- *Taisyklėmis pagrįstas prieigos valdymas*: šis prieigos kontroliavimo tipas apibrėžia specialias sąlygas reikalaujamų objektų pasiekimui. Visos MAC modeliu paremtos sistemos įgyvendina paprastą formą taisyklėmis paremto prieigos kontroliavimo apibrėžiančio tai ar turi būti suteikiama prieiga ar atsižvelgiant į:
    - Objekto jautrumo etiketę;
    - Subjekto jautrumo etiketę.
- *Grotelėmis pagrįstas* (angl. lattice-based) prieigos valdymas: šis metodas gali būti naudojamas sudėtingiems prieigos kontroliavimo uždaviniams spręsti įskaitant daugelį objektų ir subjektų. Grotelių modelis yra matematinė struktūra, kuri apibrėžia aukščiausias apatines ir mažiausias viršutines subjektų ir objektų porų reikšmes.

MAC modelis dažniausiai taikomas komercinėse operacinėse sistemose, tokiose kaip Linux, IBM AIX, ir FreeBCD, taip pat keletą metų šis modelis yra diegiamas ir „Trusted Solaris“ sistemose [12].

Nors formalūs MAC modeliai labai gerai apsaugo sistemas nuo tokių pavojų, kaip virusai ar Trojos arkliai, nuo informacijos nutekėjimo iš sistemos, daugeliu atveju MAC modeliai yra per daug sudėtingi ir reikalauja per daug kaštų, kad jie būtų naudojami praktiškai [6]. Operacijos, kurios turėtų būti paprastos, pavyzdžiui objektų kūrimas ir trynimas tampa labai sudėtingos.

2.2.2.2 Savarankiško prieigos valdymo modelis (DAC)

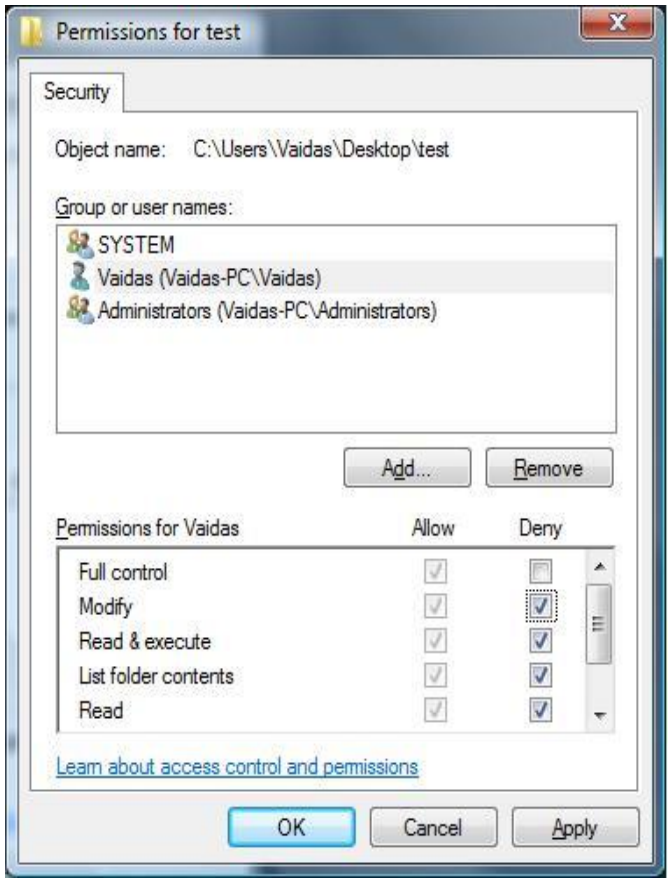
Daugelyje operacinių sistemų yra naudojamas DAC (angl. Discretionary Access Control) modelis. Savarankiško prieigos valdymo modelis yra toks, kai priėjimo prie objekto politika yra apibrėžta pačio objekto savininko. Savininkas nusprendžia kas turi teisę prieiti prie objekto ir kokias privilegijas jis turi.

Du svarbūs dalykai kalbant apie DAC yra:

- *Failo ir duomenų nuosavybė*: kiekvienas objektas sistemoje turi savininką. Daugumoje DAC sistemų kiekvienas objektas turi savo savininką, kurio pastangomis tas objektas ir buvo sukurtas. Prieigos prie objekto politika yra apibrėžiama pačio objekto savininko.
- *Prieigos teisės ir leidimai*: tai yra kontroliavimo įrankiai, kuriuos savininkas gali priskirti kitiem subjektams.

ACL arba gebėjimais paremtose sistemose gali būti naudojamas savarankiškas prieigos kontroliavimo modelis.

DAC modelis naudojamas daugelyje MS Windows sistemų. 4. paveikslėlyje galima matyti, kaip galima valdyti priėjimo prie sistemos katalogo leidimus. Pirma pasirenkamas sistemos vartotojas, tada uždedamos varnelės ant tų leidimų, kuriuos norime, kad turėtų pasirinktas vartotojas.



4. pav. Windows Vista priėjimo prie katalogo leidimų valdymas

Prieigos kontrolės matrica DAC modelyje

Prieigos teisės DAC modelį naudojančiose sistemose gali būti aprašinėjamos naudojant prieigos kontrolės matricą, kuri buvo pirmą karta pasiūlyta 1974m. Butlerio W. Lampsono [6]. Prieigos kontrolės matrica sudaryta iš eilučių, kurios atstovauja subjektus ir stulpelių, kurie atstovauja objektus. Lentelės langeliai apibrėžia operaciją, kuria subjektas gali atlikti su duotu objektu, kaip parodyta 7. lentelė .

7. lentelė Prieigos teisų matrica parodanti priėjimo prie skirtingų katalogų teises

	C:/katalogas1	**C:/katalogas2**	**C:/katalogas3**
Jonas	Skaityti, rašyti	Skaityti, rašyti	Skaityti
Petras	Skaityti	Skaityti, rašyti, vykdyti	Skaityti, rašyti
Marytė	Skaityti	Skaityti, rašyti	Skaityti, vykdyti

Prieigos teisių kontrolės matrica yra geras teorinis įrankis, tačiau jis retai naudojamas realiose programuojamose sistemose. Ši matrica bus retai naudojama sistemose, kuriose yra daugiau nei vienas vartotojas, kur objektai, prie kurių gauna prieigą vartotojai, retai persikloja. Pavyzdžiui, tipiškoje daug vartotojų turinčioje Unix sistemoje vartotojai turi savo failus, savose namų direktorijose. Vieninteliai failai, kurie yra naudojami bendrai su kitais vartotojais yra paleidžiamieji sistemos failai. Todėl, paprastai, prieigos kontrolė yra užtikrinama naudojant ACL sąrašus arba gebėjimus.

Prieigos kontrolės sąrašai (ACL)

Žiūrint atskirai į kiekvieną prieigos kontrolės matricos stulpelį, jis gali būti paverstas į prieigos kontrolės sąrašą (angl. Access control list) (ACL). ACL dažniausia yra saugomas su objektu, kuri atstovauja stulpelis. ACL sąrašas yra sudarytas iš kiekvieno subjekto ir subjekto galimų operacijų sąrašo, kaip ir matoma lentelėje.

ACL pagrindu kuriamose sistemose dažnai sunku matyti, kuriuos objektus gali pasiektas duotas

subjektas. Tai retai, kada būna problema, nes paprastai dažniau reikalaujama gauti subjektų sąrašą, kuriems yra leidžiama prieiga prie pasirinkto objekto.

8. lentelė ACL sąrašas naudojantis vieną prieigos kontrolės matricos stulpelį

	C:/Katalogas3
Jonas	Skaityti
Petras	Skaityti, rašyti
Marytė	Skaityti, vykdyti

Daugiadomeninėse sistemose naudojant centralizuotus ACL sąrašus iškyla keletas problemų. Daugiadomeninėse sistemose yra sunku užtikrinti, kad kiekvienas domeno kompiuteris turi naujausią ACL sąrašų rinkinį. Tai pat sunku išvardinti subjektus daugiadomeninėje sistemoje sudarinėjant ACL sąrašus ir garantuoti, kad subjektų vardai yra unikalūs [6].

Dar svarbesnis dalykas daugiadomeninėse sistemose yra tai, kad kyla sunkumų deleguojant prieigos teises kitiems subjektams. Deleguojantis vartotojas turės arba atnaujinti globalų ACL sąrašą arba turės prašyti objekto savininko atnaujinti jo ACL. Pirmu atveju ACL turės įtraukti leidimą atnaujinti ACL. Daugiadomeninėse sistemose pilnas ACL atnaujinimas būtų labai griozdiškas ir tai labai stipriai sulėtintų sistemos veikimą.

Literatūroje yra išskiriamas ir socialinis prieigos kontrolės sąrašas (angl. Social Access Control List), kuriame įrašomas objekto savininko viešas raktas, vieši raktai visų žmonių kurie gali šį objektą pasiekti ir socialiniai ryšiai. Objekto pasiekimui, žmonės turi turėti jų nuosava viešą raktą, kuris būtų ACL sąraše arba turi būti atestuoti savininko ir įtraukti kaip turintys patikimą ryšį įrašytą į ACL [7].

Operacinėse sistemose, dėl praktinių priežasčių, dažnai ACL sąrašai yra sutrumpinami. Pavyzdžiui Unix sistemose duomenų failo ACL sąrašas yra sudarytas tik iš šių subjektų: *vartotojas, grupė* ir *kiti [4]*. ACL sąrašai operacinėse sistemose yra sumažinami dėl keleto priežasčių. Pirma, prie daugelio Unix sistemos failų prieigą turi tik keli subjektai ar viena subjektų grupė. Kita priežastis, kad

pilnas ACL atnaujinimas turi būti atliktas kiekvieną kartai kai tik naujas subjektas yra pridedamas prie sistemos, o tai reiškia, kad programinė įranga turi pereiti per visų objektų visus ACL sąrašus.

Gebėjimai

Prieigos kontrolės sistema alternatyviai gali būti perrašoma ir imant eilutę iš prieigos kontrolės matricos. Šiuo atveju matricos eilutė yra laikoma kaip subjekto *gebėjimai*. Gebėjimai susideda iš objekto įrašų, kurie nurodo, ką subjektas gali atlikti su objektu. Tai galima matyti 9. lentelė .

9. lentelė Gebėjimai nurodo, ką subjektas gali atlikti su tam tikru objektu

	Katalogas1	**Katalogas2**	**Katalogas3**
Petras	Skaityti	Skaityti, rašyti, vykdyti	Skaityti, rašyti

Kadangi gebėjimai dažniausiai yra saugomi su subjektu ir gebėjimai suteikia prieigą, tai yra svarbu išsaugoti gebėjimų vientisumą su subjektu. Svarbu yra tai, kad gebėjimai negali būti pamirštami ir perduodami. Niekas negali atimti gebėjimų iš subjekto ir gebėjimai negali būti perleidžiami kitiems subjektams. Centralizuotose gebėjimų sistemose gebėjimai dažnai yra apsaugomi panaudojant techninę įrangą ir operacines sistemas. Padalintose sistemose gebėjimai dažnai būna apsaugomi panaudojant skaitmeninius parašus, į juos įtraukiami subjektų identifikavimo numeriai, tam kad būtų galima susieti gebėjimą su subjektu.

2.2.2.3 Rolėmis paremto prieigos valdymo modelis (RBAC)

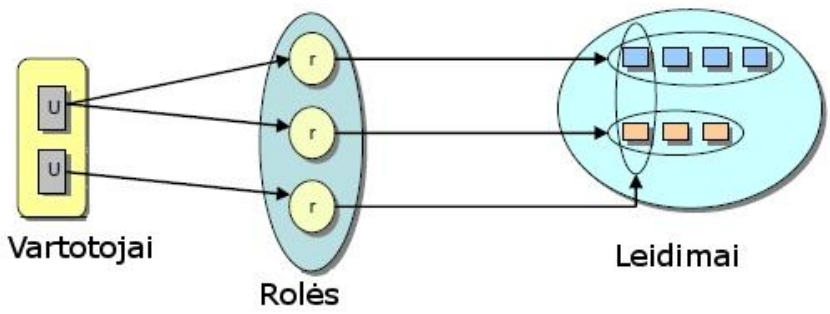
Rolėmis paremtas prieigos valdymo modelis yra prieigos kontrolės politika, kurią apibrėžia sistema, o ne tam tikro objekto savininkas. RBAC dažnai yra naudojama komercinėse aplikacijose ir karinėse sistemose, kur yra reikalaujamas daugelio lygių saugumo užtikrinimas.

5. pav. RBAC modelis

RBAC modelį sudaro sekančios trys pagrindinės struktūros [8]:

- *Vartotojai* – kurie atspindi tikruosius sistemos vartotojus;
- *Leidimai* (angl. permissions) – prieigos prie sistemos objektų aprašymas;
- *Rolės* – vartotojų funkcijų organizacijoje aprašymas.

Rolėmis paremtas prieigos kontrolės modelis yra labai lankstus sistemos privilegijų administravime [13]. Vartotojų prieigos teisės gali būti varijuojamos arba pagal atskirą autorizaciją arba netiesiogiai varijuojant rolės privilegijų rinkiniu. Didelis sistemos pliusas yra tada, kai vartotojai yra suskaidomi į grupes ir grupėms yra priskiriamos rolės, kad autorizuojamos būtų grupės, o ne atskiri vartotojai. Administravimas tampa paprastas, nes darbuotojų rolės organizacijoje yra statiški esybės, tuo tarpu vartotojai, kurie yra priskiriami prie šių rolių gali keistis dažnai [14]. Panašiai ir leidimų rinkiniai asocijuoti su rolėmis paprastai būna stabilūs, tačiau prieiga prie resursų gali būti kintanti. Toks vartotojų rolių ir leidimų surišimas gali būti toks kaip pavaizduota 6. paveikslėlyje.



6. pav. Pagrindiniai RBAC elementai ir jų sąveika

Egzistuoja ir keletas faktorių dėl, kurių RBAC administravimas gali tapti komplikuotas [18]:

- Sistemose, kuriose prieigos teisės gali būti nustatomos aplikacijų atliekamoms funkcijoms ir tų aplikacijų pasiekiamų duomenų, gali padidėti saugumo nevientisumas. Pavyzdžiui, rolė gali ne neleisti prieigos prie objekto, bet suteikti leidimą iškviesti funkciją, kuriai yra leidžiama prieiti prie to objekto.
- Sistemos administratorius diegiantis aplikaciją dažnai nėra ta pati persona, kuri suprogramavo tą aplikaciją, todėl sistemos administratorius gali nežinoti tiksliai, kurie duomenys yra pasiekiami ir modifikuojamo kiekvieno metodo.

RBAC nuo DAC skiriasi tuo, kad DAC leidžia patiems vartotojams kontroliuoti priėjimą prie resursų, o tuo tarpu RBAC prieiga yra kontroliuojama pačios sistemos, be vartotojo kontrolės. Nors RBAC modelis yra nesavarankiškas, jis gali būti atskiriamas nuo MAC pirmiausia pagal tai, kaip yra

apdorojami leidimai (angl. permissions). MAC kontroliuoja skaitymo ir rašymo leidimus paremtus vartotojo atvirumo lygiu ir papildomomis etiketėmis. RBAC kontroliuoja leidimų kolekcijas, kurios gali apimti sudėtingas operacijas, tokias kaip elektroninės komercijos sandoris, arba gali būti tokios paprastos, kaip skaitymas ar rašymas. Rolė RBAC modelyje gali būti matoma kaip leidimų rinkinys. RBAC modelio paprasčiausia schema pateikta 5. paveikslėlyje [8].

RBAC modelyje yra apibrėžiamos trys pradinės taisyklės [4]:

1. *Rolių priskyrimas*: Subjektas gali vykdyti transakciją, tik jei subjektas buvo pasirinktas arba jam priskirta rolė;
2. *Rolės autorizacija*: Subjekto aktyvi rolė turi būti autorizuota pačiam subjektui. Ši taisyklė kartu su pirma taisykle užtikrina, kad vartotojai gali paimti tik tas roles, kurioms jie yra autorizuoti.
3. *Transakcijos autorizacija*: Subjektas gali vykdyti transakciją tik jei transakcija yra autorizuota su subjekto aktyvia role. Kartu su pirmomis dviem taisyklėmis, ši taisyklė užtikrina, kad vartotojai gali vykdyti tik tas transakcijas, kurioms jie yra autorizuoti.

RBAC modelis turi iš keturias pagrindines ypatybes – *branduolio* (angl. core) RBAC, *hierarchinio* RBAC, *statiškai apriboto* RBAC ir *dinamiškai apriboto* RBAC [5]. RBAC branduolio ypatybės padengia visas pagrindines galimybes, kurios yra įtrauktos į visas RBAC sistemas. Branduolio ypatybės išskiria RBAC modelį iš visų kitų prieigos kontrolės modelių. *Hierarchinės* RBAC ypatybės į RBAC modelį įveda hierarchinių rolių koncepciją [17], kurios apibrėžia dalinį rolių tvarkymą, naudojant paveldimumo ryšį tarp atskirų rolių. Taip pat RBAC turi ir *apribojimo (angl. constrained)* ypatybių [17], nes jis yra sudarytas iš statinio ir dinaminio atsakomybių padalinimo (angl. Separation of Duties) ypatybių [15], kurios yra, atitinkamai, atsakomybių taisyklių padalinimas visam laikui, arba tik vienai sesijai. Taip pat RBAC yra apribojamas statiškai ir dinamiškai. *Statiškai apribotas* RBAC prideda ryšių apribojimus, kurie veikia rolių paskirstymo ryšius. *Dinaminio apribojimo* RBAC įveda apribojimus rolių rinkinių aktyvavimui, kurie gali būti įtraukiami, kaip vartotojo subjektų atributas.

RBAC branduolys

RBAC branduolys susideda iš penkių administravimo rinkinių: vartotojai, rolės, leidimai, operacijos ir objektai [5]. Kur leidimai yra sudaryti iš operacijų pritaikytų objektams. Rolės sąvoka yra RBAC pagrindas [17]. Rolė yra semantinė konstrukcija, aplink kurią yra formuojama prieigos politika.

Leidimai yra asocijuojami su rolėmis, o vartotojai tampa sudarytų rolių nariais, tuo pačiu gaudami ir visus rolės leidimus. Vienam vartotojui gali būti prisikirta viena arba daugiau rolių ir viena rolė gali turėti vieną ir daugiau priskirtų vartotojų. Toks rolių tvarkymas, kai leidimai priskiriami rolėms, o rolės vartotojams, suteikia labai daug sistemos prieigos valdymo lankstumo ir detalumo.

RBAC hierarchinės ypatybės

Naudojant RBAC rolės gali turėti persidengiančias atsakomybes ir privilegijas, tai yra vartotojams priklausantiem skirtingoms rolėms, gali reikėti įvykdyti bendras operacijas. Kai kurios bendros operacijos gali būti vykdomos visų darbuotojų. Šioje situacijoje, tai būtų neefektyvu ir administravimo požiūriu – griozdiška, pakartotinai apibrėžinėti bendras operacijas kiekvienai rolei, kurios yra sukurtos [5]. Rolių hierarchijos gali būti sudarinėjamos naudojant realią įmonės struktūrą. Rolių hierarchija apibrėžia roles, kurios turi unikalius požymius ir gali turėti savyje kitas roles, tai yra viena rolė gali netiesiogiai apimti operacijas, kurios yra priskirtos kitai rolei. Pavyzdžiui, sveikatos priežiūros įmonėje, rolė pavadinta „Specialistas“ gali savyje turėti „gydytojo“ ir „rezidento“ roles. Tai reiškia, kad rolės „Specialistas“ nariai yra automatiškai susieti su operacijomis, kurios įeina į „Gydytojo“ ir „Rezidento“ roles ir administratoriui nebereikia iš naujo priskyrinėti šių operacijų „Gydytojams“ ir „Rezidentams“. Taip pat rolės „Kardiologas“ ir „Traumatologas“ gali savyje turėti specialisto roles. Rolių hierarchijos yra natūralus rolių organizavimo kelias, kuris atspindi valdžią, atsakomybę ir kompetenciją: rolė, kurią gauna vartotojas nėra bendrai išskirtinė iš visų kitų rolių, kuriose narystę turi vartotojas. Šios operacijos ir rolės gali būti priklausomi nuo organizacinės politikos ir apribojimų. Hierarchinis rolių paskirstymas gali būti pradėtas naudoti, kai operacijos pradeda persidenginėti.

Statinio apribojimo ypatybės RBAC modelyje

Užuot diegusios brangią prieigos tikrinimo sistemą organizacijos gali naudoti prieigos apribojimus naudodamos RBAC [5]. Pavyzdžiui, gali būti pakankama leisti gydytojams turėti laisvą prieigą prie visų pacientų duomenų įrašų, jei gydytojų prieiga yra pakankamai įdėmiai stebima. Bet taip dažnai nebūna. Su RBAC, apribojimai gali būti naudojami prisijungimo apribojimui, tam kad gydytojai galėtų prieiti tik prie tų duomenų, su kuriais gydytojas ar jo pacientai yra susiję. Organizacijos gali nustatyti taisyklės pagal kurias būtų susiejamos operacijos ir rolės. Pavyzdžiui, sveikatos paslaugų tiekėjas gali nuspręsti, kad laboratorijos darbuotojo rolė, turi būti apribota taip, kad jis galėtų tik skelbti atliktų testų rezultatus, o ne kitaip juos pateikti gydytojui, kur galėtų atsirasti klaidų ar nukentėtų

paciento privatumas. Operacijos taip pat gali būti apribotos tam, kad atitiktų išleistus valstybinius įstatymus ir taisykles. Pavyzdžiui, farmacininkas turėtų būti apribotas taip, kad galėtų paskirstyti vaistus po vaistines, bet negalėtų jų išrašyti.

Operacija atstovauja valdymo vienetą, kuriam gali duoti nurodymus atskira rolė, priklausanti nuo reguliavimo apribojimų RBAC struktūros viduje. Operacija gali būti panaudojama tam, kad surinktų sudėtingą saugumui tinkamą, išsamią informaciją ar apribojimus, kurie negali būti nustatyti su paprasta prieiga. Pavyzdžiui, yra skirtumas tarp prieigos poreikių kasininkui ir apskaitos vykdytojui banke. Bankas apibrėžia tokias prieigos teises kasininkui, kurių jam užtektų paprastų operacijų su sąskaitomis atlikimui. Tai reikalauja skaitymo ir rašymo teisių specifiniame sąskaitų faile. Organizacija taip pat gali apibrėžti sąskaitų prižiūrėtojo rolę, kuriam leidžiama atlikti korekcijų operacijas. Šios operacijos reikalauja skaitymo ir rašymo prieigos teisių prie tų pačių laukų sąskaitų faile, kaip ir kasininkui. Tačiau, sąskaitų prižiūrėtojui gali būti apribota teisė leidžianti inicijuoti depozito pakeitimus ar grynųjų pinigų išėmimus. Skirtumas tarp šių dviejų rolių yra operacijos, kurios yra vykdomos skirtingų rolių ir reikšmės kurios yra įrašomos į transakcijų atsekimo (angl. log) failą.

Dinamiško apribojimo ypatybės RBAC modelyje

RBAC struktūra suteikia galimybę administratoriams reguliuoti operacijas, kurios turi būti vykdomos tam tikros rolės narių. Narystės rolėje suteikimas gali būti apribotas. Kai kurios rolės gali būti užimtos tik tam tikro skaičiaus darbuotojų, tam tikrą laiko tarpą. Pavyzdžiui administratoriaus rolė gali būti suteikiama tik vienam darbuotojui vienu metu. Nors kitas darbuotojas gali taip pat veikti toje rolėje tačiau tik vienas žmogus gali būti atsakingas kaip administratorius tuo metu. Vartotojas gali tapti nauju rolės nariu tik tuomet jei leidžiamas narių skaičius tai rolei yra neviršytas.

RBAC modelio panaudojimo galimybės

Tinkamai valdoma RBAC sistema įgalina vartotojus atlikti daugelį įgaliotų operacijų ir suteikia didelį prieigos kontroles sistemos lankstumą. Sistemos administratoriai gali valdyti prieigas abstrakcijos lygyje, kuris natūraliai atspindi organizacijos veiklas. Tai pasiekiama naudojant statinį ir dinaminį vartotojų veiksmų reguliavimą, apibrėžiant roles, rolių hierarchijas, ryšius ir apribojimus. Tokiu būdu, kai tik RBAC struktūra yra pritaikyta organizacijai, pagrindiniai administraciniai veiksmai būtų: vartotojų priskyrimas ir išmetimas iš tam tikrų rolių. Tai yra pagrindinis skirtumas nuo labiau tradicinio ir mažiau intuityvaus bandymo administruoti žemo lygio prieigos kontrolės mechanizmus

tiesiogiai (pavyzdžiui, prieigos kontrolės sąrašai (ACL), prieigos kontrolės matricos), objektas su objektu principu. Toliau, galima asocijuoti RBAC operacijų koncepciją su objektiškai orientuotų technologijų koncepcijomis. Šios asociacijos veda prie požiūrio kur objektiškai orientuotos technologijos gali būti panaudojamos RBAC operacijų vystymui įvairiose aplikacijose ir operacinėse sistemose.

RBAC modelis yra tinkamas įvairioms pramonės ir vyriausybinių organizacijų saugumo politikoms patenkinti. Tačiau tik keletas komercinių sistemų yra prieinamos paprastam vartotojui naudojančios RBAC modelį. Windows Server 2003 pristatė papildantįjį autorizacijos interfeisą pavadintą „autorizacijos vadovas“ (angl. Authorization Manager), kuris naudoja RBAC modelį, kuris pagrįstas ACL sąrašais. Jis apima pagrindines esybes: roles, užduotis, operacijas, apimtis (angl. scopes), pagrindines aplikacijų grupes ir „Lengvasvorio“ katalogų prieigos protokolo (angl. Lightweigth Directory Access Protocol) (LDAP) užklausų grupes. Su šiomis esybėmis vartotojas gali kurti ir diegti autorizacijos taisykles ir išplėsti autorizacijos vadovo programavimo interfeisą. Populiarios duomenų bazių valdymo sistemos, tokios kaip Informix Online Dynamic Server, Oracle Enterprice Server ir Sybase Adaptive Server taip pat suteikia galimybes naudotis pagrindinėmis RBAC galimybėmis.

RBAC modelio trūkumai

RBAC modelis teigia, kad visi leidimai (angl. permissions) reikalingi funkcijos atlikimui gali būti nesunkiai identifikuoti ir sukurtos atitinkamos rolės. Iš tikrųjų, pasirodo, kad rolės sukūrimas yra labai sudėtinga užduotis [10]. RBAC deklaruoja, kad jis užtikrina stiprų saugumą ir lengvą administravimą. Stipriam saugumui geriau būtų sukurti kuo daugiau rolių, kad kiekvienam vartotojui būtų įmanoma priskirti kelias roles. Lengvesniam administravimui būtų geriau turėti kuo mažiau rolių, tuo pačių būtų mažiau darbo jas administruojant. Organizacijos turi laikytis privatumo ir kitų taisyklių ar įgaliojimų taip pat pagerinti saugumo politiką tuo pačiu sumažinant rizikas ir administravimo kainą. Tuo tarpu rinkoje plinta internetiniai ir kiti naujų programų tipai, taip pat web servisai įveda dar daugiau sudėtingumo veikdami internete kartu su atskirais komponentais ir teikdami įvairias paslaugas įvairioms aplikacijoms. Be to, failų ir serverių paskirstymas gali būti nesuderinamas su organizacijos struktūra, kuri reikalauja iš vartotojų daugiausiai dėmesio skirti į praktinius dalykus, tokius kaip sąskaitų atidarymas ir sąskaitų apmokėjimas.

Nors prieigos valdymo lankstumas ir yra pagerintas lyginant su DAC ir MAC modeliais, RBAC modelis sieja vartotojus su rolėmis ir asocijuoja tas roles su privilegijomis objektų autorizacijos metu, jis nepalieka galimybės kontroliuoti, vartotojų ar rolių, savo nuožiūra [11]. Todėl, RBAC yra sunku naudoti įgyvendinant DAC politiką. Yra įrodyta, kad yra įmanoma, įgyvendinti DAC politiką naudojant RBAC modelį naudojant daug rolių susietų su kiekvienu sistemos objektu. Yra keltas DAV variacijų, kurios galėtų būti palaikomos naudojant šį metodą. Deja, tokiems veiksmams reikia labai daug išteklių: kiekvienam sistemos objektui turi būti sukurta po keturias roles ir aštuonis leidimus. Taigi, DAC modelio įgyvendinimas naudojant RBAC yra labiau teorinis nei praktinis. Žiūrint iš kitos pusės, realūs RBAC įgyvendinimo apribojimai iš tikrųjų nesukelia jokių reikšmingų problemų, nors ir daugelyje operacinių sistemų yra naudojamas DAC modelis, kaip pagrindine prieigos valdymo forma.

3. Projektinė dalis

3.1 Sistemos aprašymas

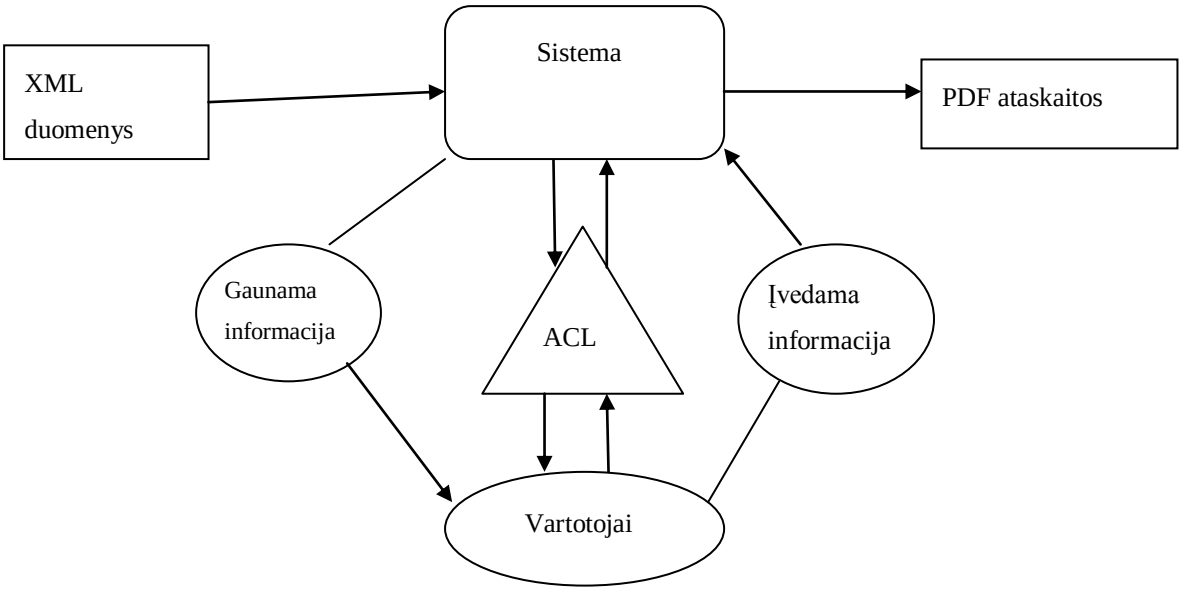
Kuriama sistema – socialinis portalas, kuriame įvairūs suinteresuoti žmonės skelbia įvairių produktų kainas. Sistema veikia interneto tinkle ir vartotojų pasiekiama naudojant interneto naršykles. Sistema skirta sunkiau besiverčiantiems ar šiaip norintiems sutaupyti pinigų Lietuvos gyventojams.

Prie sistemos prisijungęs vartotojas, priklausomai nuo turimų prieigos teisių gali skelbti produktų kainas, ar pačius produktus, taip pat gali peržiūrėti kitų skelbiamas kainas, gali susidaryti prekių krepšelį, kuriame stebės kainų kitimus, matys kuriose parduotuvėse jį dominančios prekės yra pigiausios.

Sistemos vartotojai yra kelių lygių. Vartotojo lygis priklauso nuo pačio vartotojo aktyvumo. Kuo aktyvesnis vartotojas, tuo daugiau teisių jis gauna. Taip pat jei vartotojas atlieka netinkamus veiksmus iš jo atimamos turimos teisės. Tai įgyvendinama naudojant RBAC prieigos teisių paskirstymo modelį, kai vartotojams yra priskiriamos tam tikros rolės, su atitinkamomis teisėmis.

Kuriama sistema turi penkių lygių rolių sistemą.

3.2 Sistemos veiklos kontekstas



7. pav. Sistemos veiklos kontekstas

- Į sistemą įvedami duomenys arba XML formato duomenų failais arba vartotojų pildant specialiai parengtas formas.
- Sistema generuoja PDF ataskaitas.
- Vartotojai autorizuojami panaudojant ACL sąrašus.

3.3 Sistemos vartotojai ir jų funkcijos

Sistemos vartotojai yra paprasti žmonės, norintys sutaupyti pinigų ir visada žinoti, kur kokia prekė yra pigiausia.

Išskiriami pagrindiniai penki sistemos vartotojų lygiai:

1. **neregistruoti vartotojai** (bendrai *„NV funkcionalumas“*)
    - registracija
    - prekių bei kainų peržiūra
    - prekės kainos informacijos peržiūrėjimas
    - naujienų peržiūra
    - išplėstinė paieška
2. **registruoti vartotojai** (bendrai *„RV funkcionalumas“*)
    - *NV funkcionalumas*
    - prisijungimas prie sistemos
    - asmeninių duomenų (profilio) redagavimas
    - prekių krepšelio sudarymas
    - sudarytų krepšelių saugojimas bei administravimas
    - prekių atitikimo realybei įvertinimas
    - komentarų apie prekes ir kainas rašymas
    - atsijungimas nuo sistemos
3. **redaktoriai** (*bendrai „RD funkcionalumas“)*
    - *RV funkcionalumas*
    - prekių kainų skelbimas
4. **moderatoriai** (bendrai „*MR funkcionalumas*“)
    - *RD funkcionalumas*
    - naujų prekių registravimas

- prekių duomenų koregavimas (pavadinimų, aprašymų ir t.t.)
- prekių grupių ir pogrupių administravimas
- naujienų administravimas
- prekių ir kainų komentarų redagavimas
- parduotuvių administravimas

5. **administratoriai**
   - *MR funkcionalumas*
   - Vartotojų teisių administravimas
   - Prekių iš XML duomenų failų importavimas
   - Ataskaitų peržiūrėjimas
   - PDF ataskaitų generavimas

3.4 Sistemos nefunkciniai reikalavimai ir apribojimai

Reikalavimai išvaizdai

- Lengvai skaitoma, intuityvi sąsaja;
- Paprastas (nesudėtingas) panaudojimas;
- Prieinamumas, kad vartotojas nesivaržytų naudodamas sistemą;
- Akį nerėžiančios, tarpusavyje derančios spalvos;
- Visas pagrindinis tekstas rašomas naudojant šriftą „Tahoma“, 12 dydžio, antraštėse pastorintas šriftas.
- Įdomi, pritraukianti lankytojus išvaizda;
- Išvaizda turi būti unikali, nepanaši į jau egzistuojančias sistemas;

Reikalavimai panaudojamumui

- Galimybė prisijungti prie sistemos naudojant Internet Explorer 6, Internet Explorer 7, Mozilla Firefox 2, Mozilla Firefox 3, Opera 9 naršykles;
- Atsiskaitymų pervedimas naudojant kreditines korteles ir bankų paslaugas;
- Paprastai panaudojamas, bet kokio asmens be specialių apmokymų;

- Anglų kalbos naudojimas (tarptautiniai klientai);
- Greitas naujai prisiregistravusių klientų peržiūrėjimas ir aktyvavimas;
- Prisijungus sistemos administratoriui pirmame puslapyje matomi neapmokėtų sąskaitų savininkai;
- Pirmame puslapyje matomi įvykusių klaidų pranešimai;

Reikalavimai vykdymo charakteristikoms

- Visos vykdomos komandos turi būti įvykdomos be uždelsimų;
- Visi klientų duomenys duomenų bazėse yra patikimai saugomi, slaptažodžiai užkoduojami.
- Duomenų bazėje turi tilpti 2GB duomenų.
- Sistema turi pilnavertiškai funkcionuoti naudojant stabilų interneto ryšį su 256kbps pralaidumu;
- Sistemos pagrindinis langas (horizontaliai) turi tilpti į 1024 (1024x768) taškų vaizduoklio rezoliuciją;

Reikalavimai veikimo sąlygoms

- Prie sistemos galima prisijungti iš bet kurios pasaulio vietos, kurioje yra Interneto ryšys, kompiuteris su įdiegta interneto naršykle.
- Sistema turi kokybiškai veikti kai kompiuteris prijungtas prie nemažesnio kaip 256kbps interneto ryšio pralaidumo kanalo;

Reikalavimai sistemos priežiūrai

- Galimybė greitai pakeisti sistemos kalbą;
- Galimybė greitai pakeisti apskaičiuojamų mokesčių procentą (pasikeitus įstatymams);
- Galimybė prijungti prie sistemos naujus modulius arba išplėsti esančius (nauji funkcionalumai);

Reikalavimai saugumui

- Konfidencialumas - sistemoje esantys duomenys apsaugoti nuo neteisėtos prieigos;
- Visi sistemos naudotojų slaptažodžiai užkoduojami ir nėra žinomi niekam (net sistemos administratoriui);
- Norint vartotojui pakeisti slaptažodį sistema sugeneruoja naują, atsitiktinį slaptažodį, kurį sistema išsiunčia vartotojui registracijos metu nurodytu el. pašto adresu. Vartotojui prisijungus reikalaujama, kad sugeneruotas naujas slaptažodis būtų nedelsiant pakeistas nauju.
- Keičiant slaptažodį būtina naują slaptažodį nurodyti du kartu;

- Prisijungimo duomenys nėra saugomi tekstiniuose failuose („loginami“);
- Visi vykdomi veiksmai (išskyrus prisijungimą prie sistemos) yra („loginami“) išsaugomi tekstiniuose failuose;
- Visi piniginiai atsiskaitymai naudojami jungiantis per saugu (https) protokolą;
- Vientisumas - sistemos duomenys vienareikšmiškai atitinka šaltinio perduotus (iš jo gautus) duomenis, kartu užtikrinant jų panaudojimo teisėtumą;
- Pasiekiamumas - galimybė pasinaudoti duomenimis per fiksuotą laiką teisėtiems vartotojams.

Kultūriniai-politiniai reikalavimai

- Sistemos išvaizda, joje naudojami terminai neturi prieštarauti galiojantiems įstatymams, taip pat neturi įžeisti ar kitaip sukelti religinių bendruomenių nepasitenkinimą;
- Sistema turi atrodyti estetiškai, atitikti kultūros normas.
- Sistemos išvaizda neturi būti panaši į jau egzistuojančias panašaus pobūdžio sistemas.

Teisiniai reikalavimai

Sistema turi laikytis vartotojo duomenų apsaugos įstatymo reikalavimų.

3.5 Sistemos funkciniai reikalavimai

- Registracija
    - Duomenų įvedimas
    - Duomenų korektiškumo tikrinimas
        - Slaptažodžių sutapimas
        - El. pašto adreso egzistavimo tikrinimas
        - Apsauga nuo automatinės registracijos
- Penkių (skirtingomis prieigos teisėmis) lygių prisijungimas prie sistemos
    - Neregistruoto vartotojo lygis (informacijos peržiūra)
    - Registruoto vartotojo lygis (kainų skelbimas)
    - Kainų skelbėjo lygis (produktų įtraukimas)
    - Moderatoriaus lygis (produktų šalinimas, redagavimas)
    - Administratoriaus lygis (visos teisės)
- Atsijungimas nuo sistemos
- Produktas

        - Produktų talpinimas
            - Importuojant XML duomenų failą
            - Produkto informacijos suvedimas rankomis
        - Šalinimas
        - Redagavimas
        - Naujos kainos įvedimas
        - Kainos komentavimas
        - Produkto nuotraukos įkėlimas
        - Produkto peržiūrėjimas
            - Išsamios produkto informacijos peržiūrėjimas
            - Produkto kainų peržiūrėjimas
                - Kainos informacijos peržiūrėjimas
                - Kainų archyvo peržiūrėjimas
        - Produkto nuotraukos peržiūrėjimas
        - Produkto aprašymo įvertinimas
        - Produkto kainos korektiškumo įvertinimas
        - Produkto komentavimas
- Produktų krepšelis
    - Sudarymas
    - Išsaugojimas
    - Redagavimas
        - Papildomų prekių įtraukimas
        - Nereikalingų prekių pašalinimas
    - Krepšelio komentaro išsaugojimas
- Produktų katalogo sudarymas
    - Produktų grupės
        - Įtraukimas
        - Redagavimas
        - Šalinimas
    - Produktų pogrupiai
        - Įtraukimas
        - Redagavimas
        - Šalinimas
- Naujienos
    - Redagavimas
    - Šalinimas
- Statistinių duomenų kaupimas ir peržiūrėjimas
- PDF ataskaitų generavimas

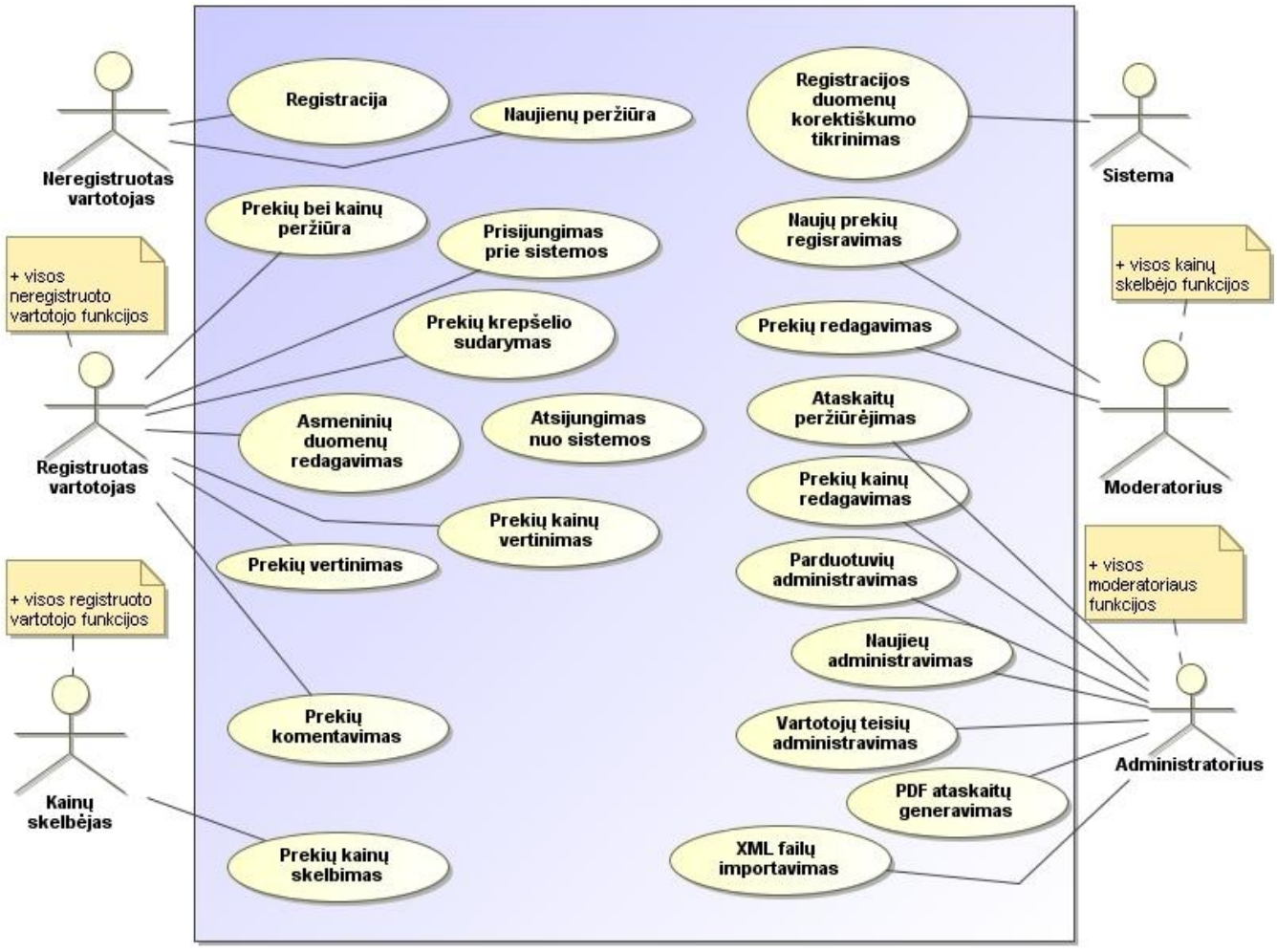
3.6 Sistemos architektūra

Sistemos architektūra aprašoma pateikiant sistemą keliais skirtingais vaizdais:

- Panaudojimo atveju vaizdas;
- Loginis vaizdas;
- Procesų vaizdas;
- Išdėstymo vaizdas.

3.6.1 Panaudojimo atvejų vaizdas

Panaudojimo atvejų vaizde pateikiama panaudojimo atvejų diagrama, kurioje matomos pagrindinės sistemos vartotojų funkcijos.

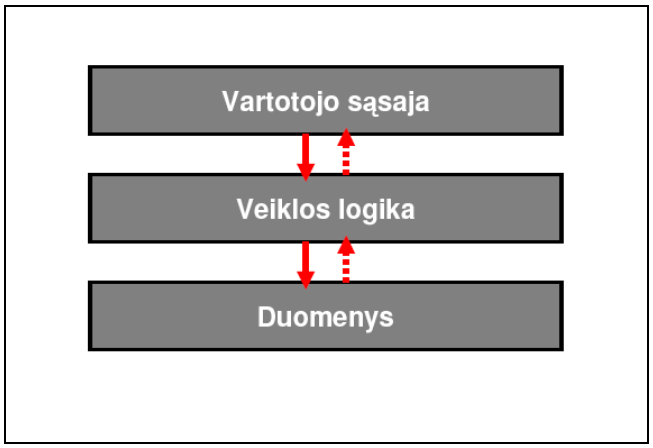


8. pav. Panaudojimo atvejų diagrama

3.6.2 Loginis vaizdas

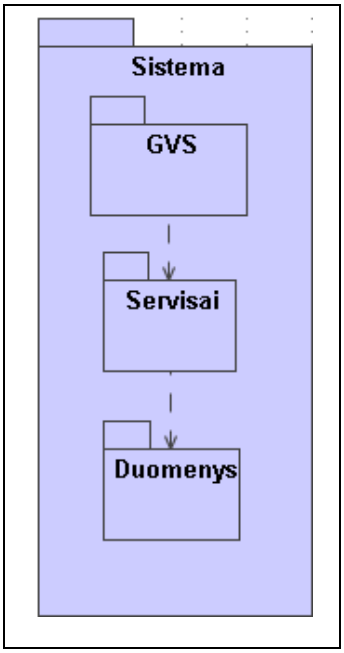
Sistemos programinė įranga realizuojama moduliais su griežtai apibrėžta sąsaja. Šiame skyriuje pateiksiu loginę sistemos architektūrą, sistemos skaidymą į paketus, bei svarbiausias paketų klasių diagramas.

Sistemos loginė architektūra kuriama vadovaujantis MVC šablonu:



9. pav. Veiklos logikos šablono MVC principas

Sistemą planuojama realizuoti naudojant tokį paketų išskaidymą:



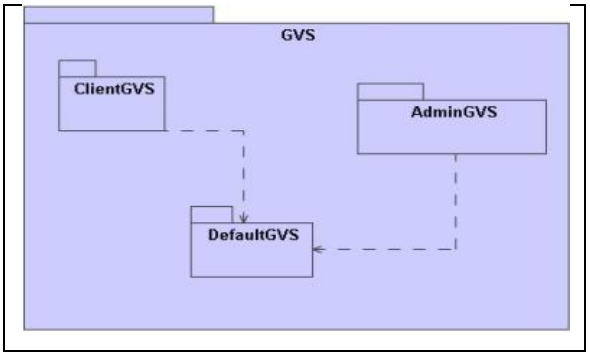
10. pav. Sistemos skaidymas į paketus

Sistemoje atskirtos trys pagrindines sritys:

- GVS – vartotojo sąsajos ir grafiniai elementai (langai, mygtukai ir t.t), padedantys pasiekti norimą funkcionalumą;
- Servisai – veiklioji sritis, atliekanti konkrečius veiksmus, aktyvuojama vartotojo sąsajos elementų;
- Duomenys – konkrečios duomenų struktūros (DB įrašai ar sąskaitos);

Paketų detalizavimas

GVS Paketas

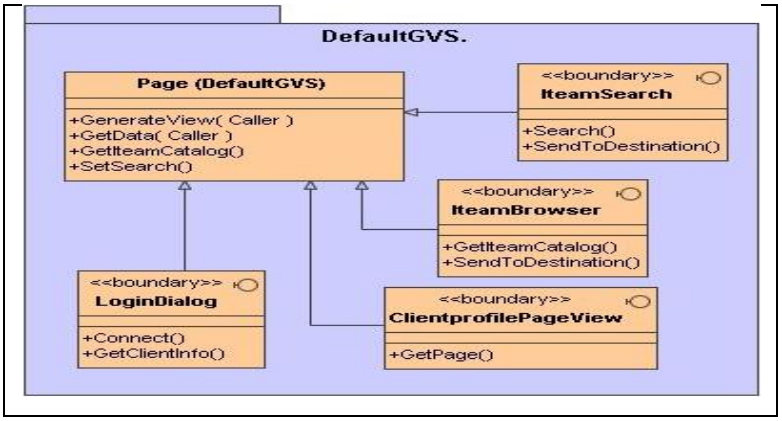


11. pav. Paketo GVS detalizavimas

Pakete saugomi visi bendriniai (default) vaizdai, "layout" šablonai, spalvos, meniu punktai ir t.t. Atitinkamas servisas generuoja vaizdą vartotojui pagal pasirinktą funkcionalumą. Kiekvienam tipui vartotojų priskiriamas skirtingas paketas.

DefaultGVS paketas

Paketas pasiekiamas visų tipų vartotojams. Jame saugomas bendras išdėstymas (default layout) bei komponentai atvaizduojami visuose generuojamuose puslapiuose.

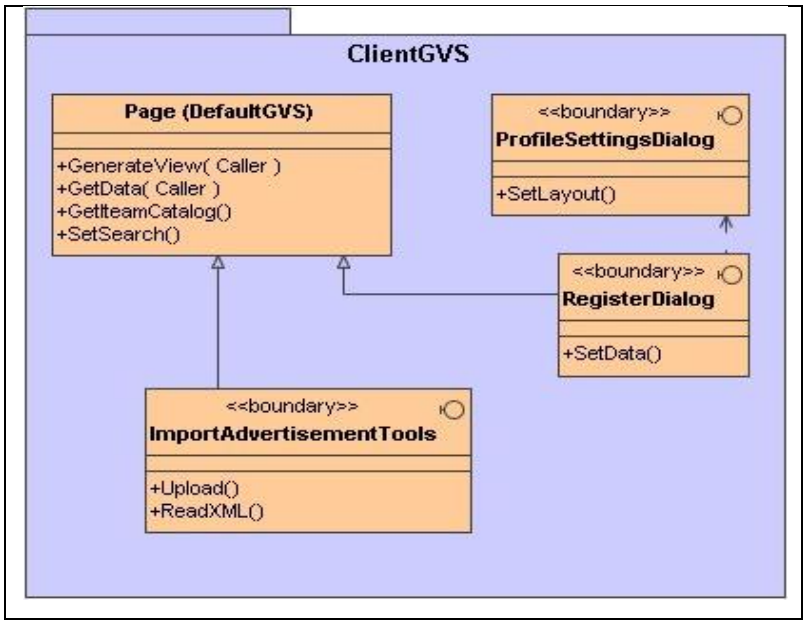


12. pav. Paketą DefaultGVS sudarančios klasės

- **LoginDialog** – dialogas skirtas prisijungimams;
- **ClientProfilePageView** – vartotojų nustatymų langas;
- **IteamSearchDialog** – langas, kurio pagalba galima atlikti norimo produkto paiešką;
- **IteamBrowser** – Interfeisas, kurį naudojant galima atlikti paiešką pagal produktų kategorijas;
- **Page (DefaultGVS) –** puslapių bendrų metodų ir išdėstymo komponentų viršklasė;

ClientGVS paketas

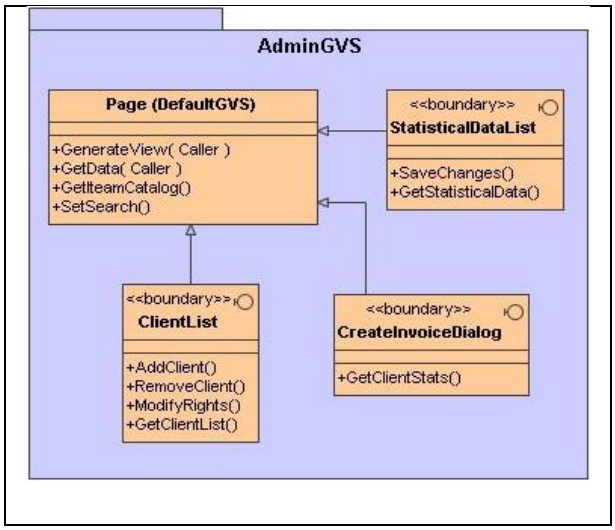
Paketas atspindintis pagrindines sistemos vartotojo funkcijas – registruotis, sukelti duomenis, nustatyti asmeninius nustatymus ir parametrus. Nors funkcionalumas "užsiregistruoti" pasiekiamas visiem, laikoma, kad lankytojas sėkmingai užbaigęs registracijos procesą virsta registruotu vartotoju.



13. pav. Paketą ClientGVS sudarančios klasės

- **Page (DefaultGVS) –** bendrinė puslapių struktūra, pasiekiama iš **DefaultGVS**
- **RegisterDialog** – registracijos interfeisas. Pats procesas susideda iš trijų etapų todėl nuo šio interfeiso neatsiejami:
- **ProfileSettingsDialog** – asmeninių nustatymų interfeisas: pasirenkamas vaizdo ir išdėstymo šablonas, spalvos ir t.t.
- **ImportAdvertisementTools** – pageidaujamos reklaminių produktų informacijos bei priemonių įkėlimo interfeisas.

AdminGVS paketas



14. pav. Paketą AdminGVS sudarančios klasės

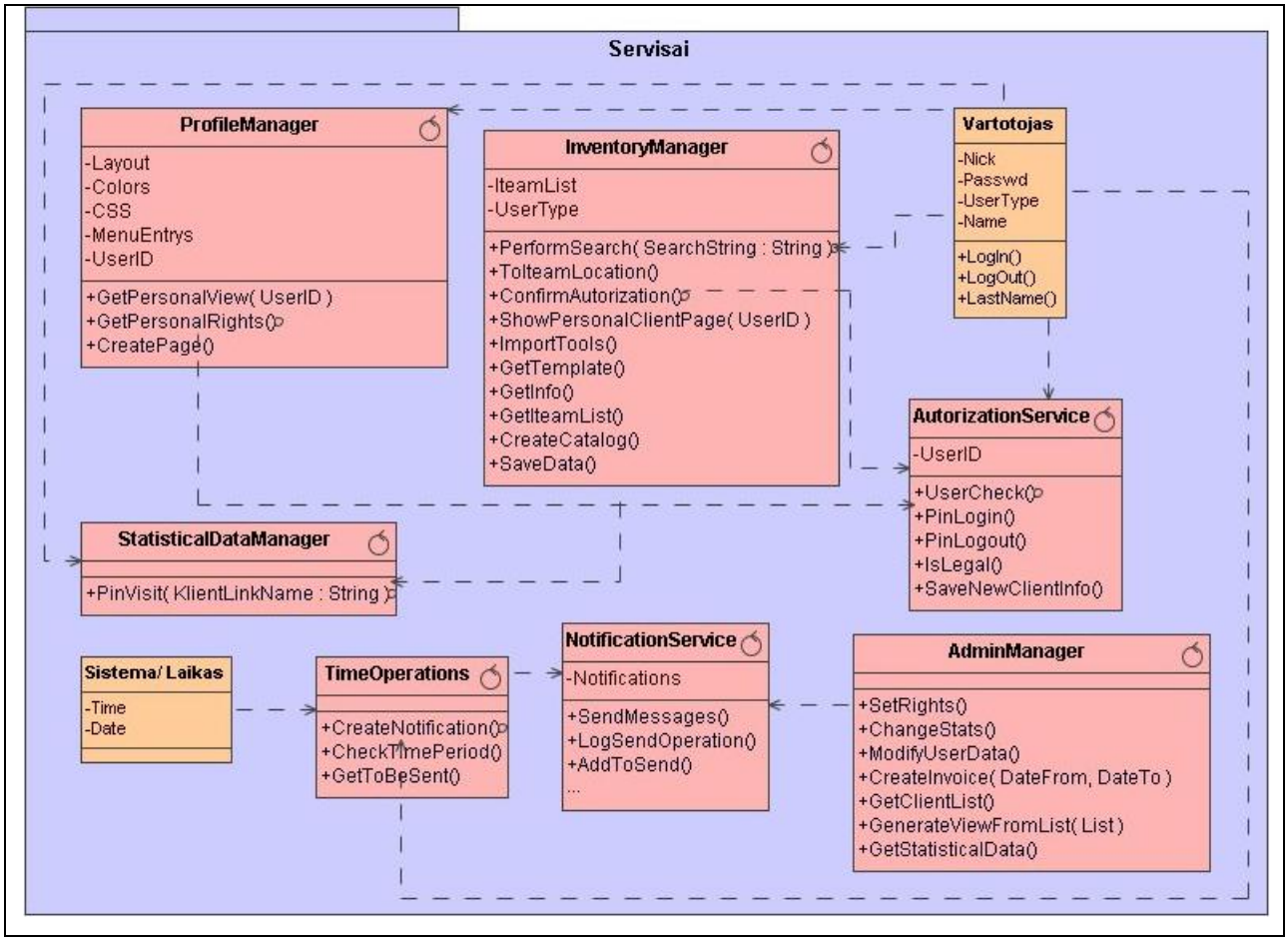
- **Page (DefaultGVS) –** bendrinė puslapių struktūra, pasiekiama iš **DefaultGVS**
- **ClientList** – interfeisas leidžiantis administratoriui peržvelgti klientų sąraša, suteikti/apriboti teisias ir pan;
- **CreateInvoiceDialog** – interfeisas leidžiantis administratoriui sukurti klientų ataskaitas;
- **StatisticalDataList** – interfeisas leidžiantis administratoriui peržvelgti statistinius duomenis;

Paketas servisai

Pakete saugomas visas pagrindinis funkcionalumas. Čia generuojami asmeninių nustatymų (profilių) puslapiai, saugomi registracijos duomenys, atliekama autorizacija, bei įrašinėjami duomenys į DB. Kadangi visi procesasi glaudžiai sąveikauja tarpusavyje nėra prasmes dar labiau juos skaldyt. Pvz.: **ProfileManager** kreipiasi į **AutorizationService** vienokio ar kitokio veiksmo legalumui nustatyti (ar turi vartotojas teisias konkrečiam veiksmui ar ne).

Schemoje pavaizduoti **Vartotojo** bei **Sistemos/Laiko** klasės tik simbolinės bendram kontekstui

išryškinti.



15. pav. Paketą Servisai sudarančios klasės

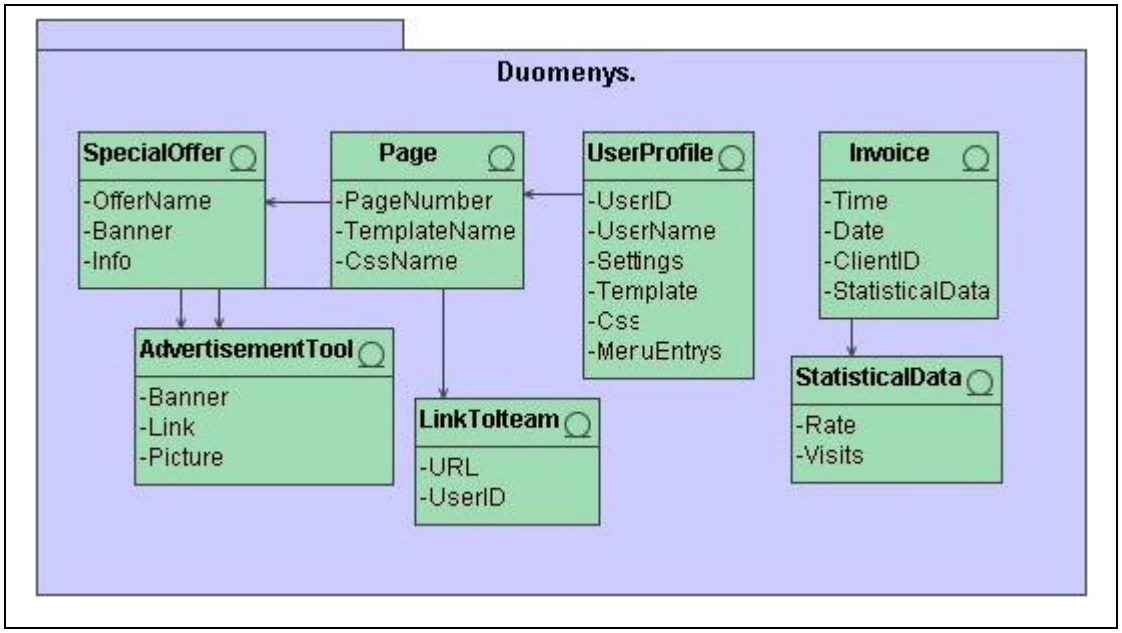
- **ProfileManager –** klasė atsakinga už individualių parametrų atvaizdavima generuojamame (kliento skelbiamame) puslapyje, teisių nuskaitymą ir nustatymą. Viskas kas siejama su neinformacine dalimi yra šios klasės žinioje;
- **InventoryManager -** valdo visa informacinę dalį (nuotraukas, aprašus, nuorodas, banner'ius). Užpildo asmeninius puslapius informacija.
- **StatisticalDataManager –** įvikiais paremta klasė skaičiuojanti kiekvieno kliento visų produkto alankymus.
- **TimeOperations –** nuo laiko priklausančios funkcijos, kuriančios automatiškai tam tikro periodo ataskaitas, tikrinančios išsiuntimui skirtų ataskaitų buvimą (nebuvimą) bei periodą (ar laikas siųsti).
- **NotificationService -** servisas paverčiantis ataskaitą į reikiamą formatą ir pridedantis ją prie "siųsti" ataskaitų.

- **AutorizationService –** atlieka prisijungimo duomenų teisingumo patikrinimą, taip pat atliekamų veiksmų legalumą.
- **AdminManager –** administratoriaus funkcijų servisas (suteikti teisias, ištrinti/pridėti vartotoją, sugeneruot ataskaitą ir t.t.)

Paketas duomenys

Arčiausiai duomenų bazės esantis servisas. Jame saugomi duomenų modeliai atitinkantys DB esančias struktūras, t.y. jų atributai atitinkamai užpildomi DB laukais.

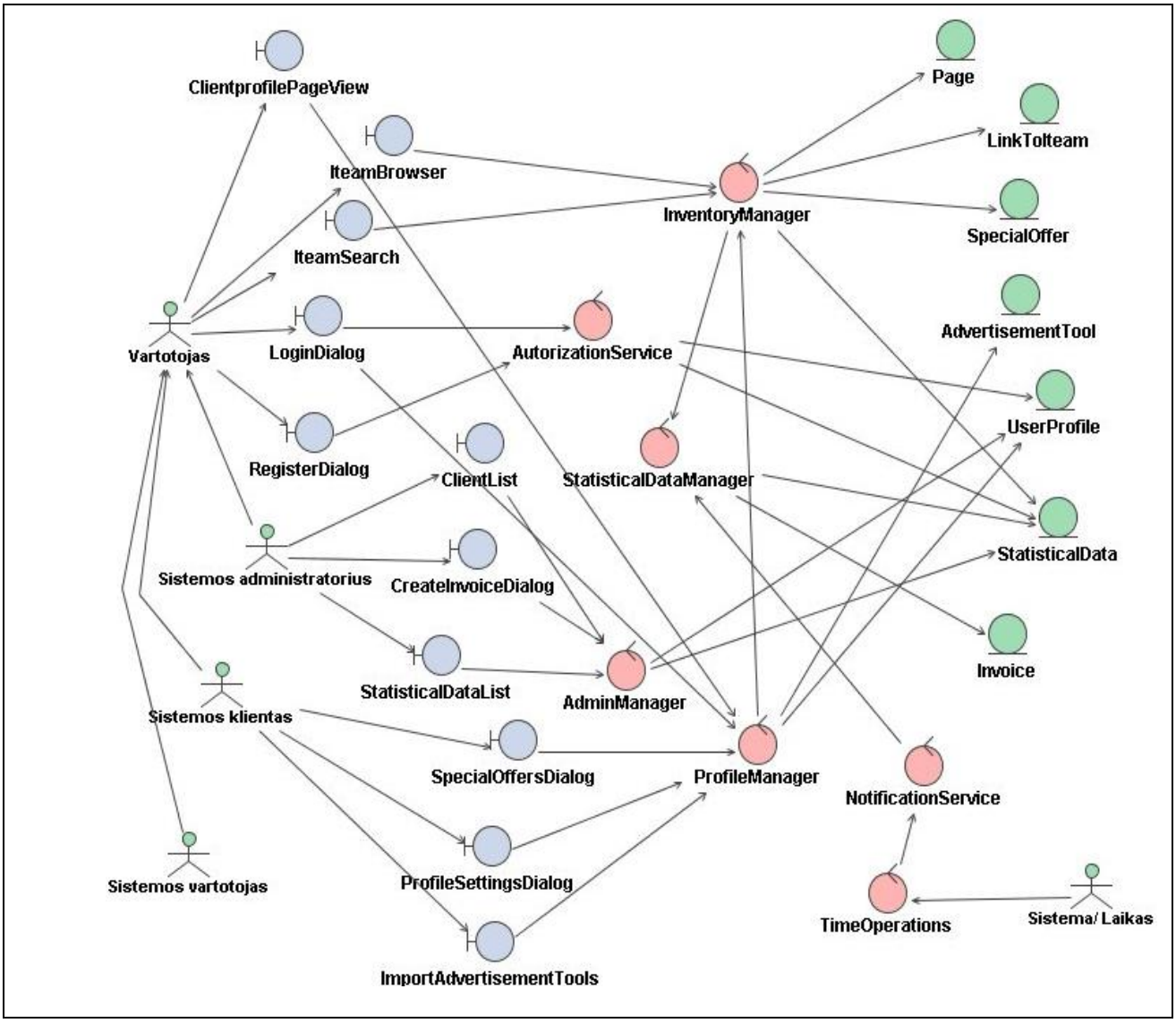
Atitinkamai saugant duomenis duomenų bazėje, šios klasės bus transformuojamos į DB lentelių laukus.



16. pav. Paketo duomenys detalizavimas

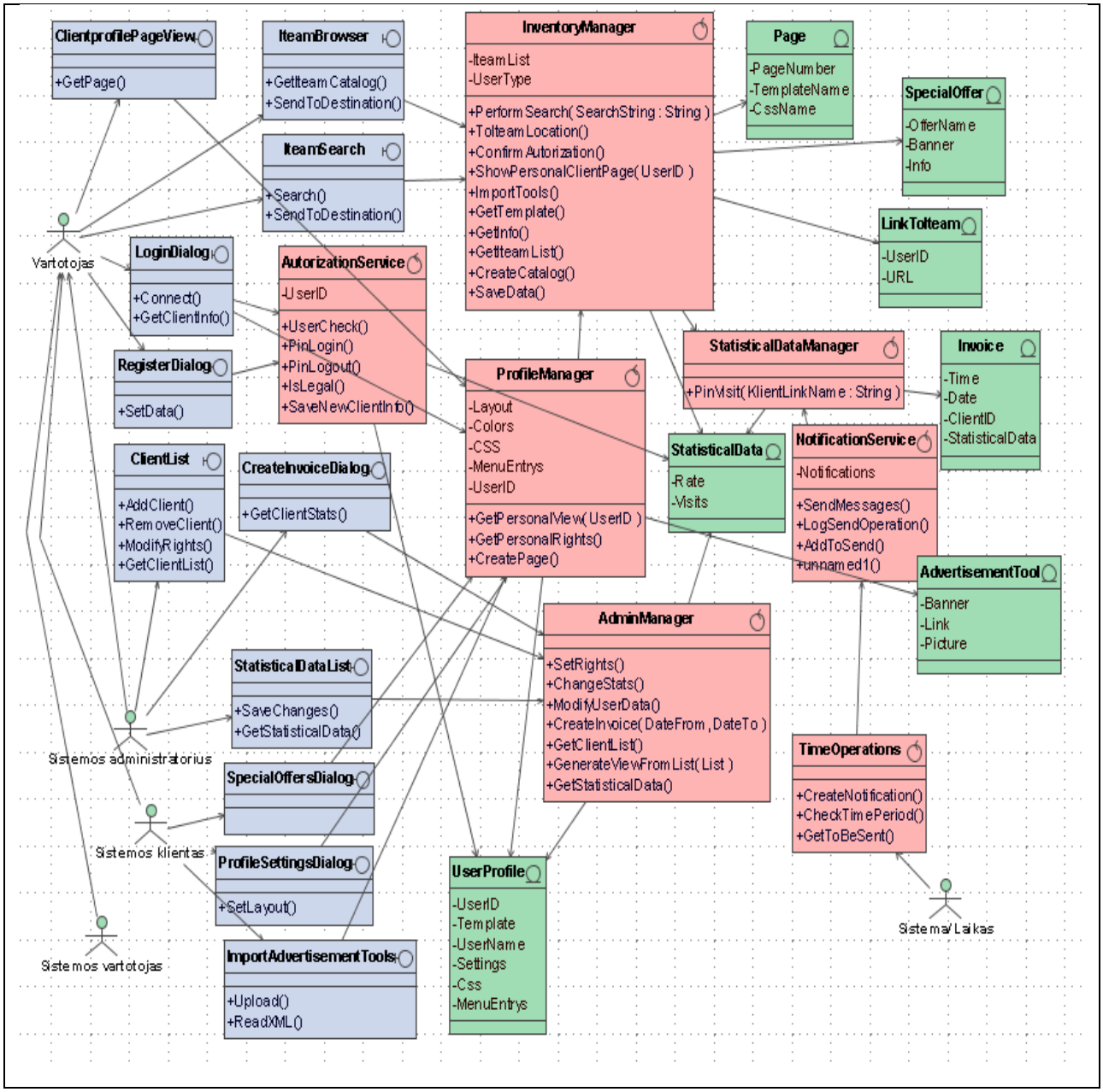
- **SpecialOffer** – specialių pasiūlymų klasė.
- **Page** – klasė sauganti puslapių skaičių, naudojamo šablono pavadinimą ir pan. Dalyvauja generuojant galutinį vaizdą vartotojui.
- **UserProfile** – informacijai apie vartotoją saugoti skirta klasė.
- **Invoice** – atsakaitų klasė (sukūrimo data, periodas, apsilankymai)
- **AdvertisementTool** – reklaminių priemonių ir kitų pagalbinių priemonių saugojimui skirta klasė (nuotraukos, aprašai, baner'iai ir);
- **LinkToIteam** – informacija apie galutinę reklamuojamo produkto vietą (kokiam klientui priklauso, koks galutinio tikslo adresas)
- **StatisticalData** – statistinių duomenų klasė.

Sistemos dinaminis vaizdas pateiktas 17. pav



17. pav. Sistemos išbaigtiems (angl. robustness) diagrama

Sistemos veiklos logikos diagrama, bendrojo panaudojimo kontekste, atvaizduojanti klientų funkcionalumo prieinamumą, serviso paketų bendradarbiavimą bei duomenų sluoksnio valdymą. Išskleista bendrojo atvejo sąveikos diagrama atrodo taip, kaip pavaizduota 18. paveikslėlyje.

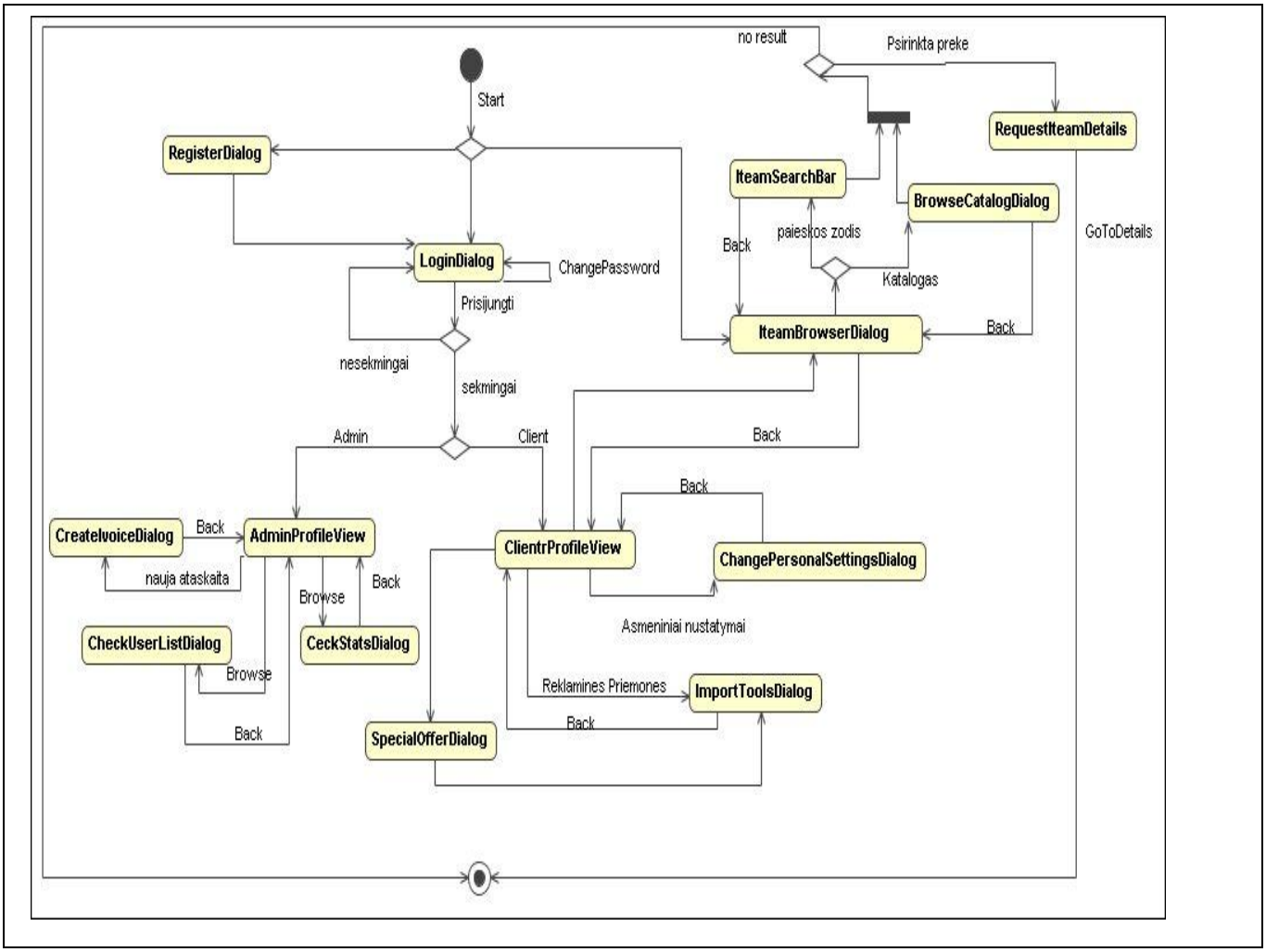


18. pav. Bendra visos sistemos išbaigties diagrama

3.6.3 Procesų vaizdas

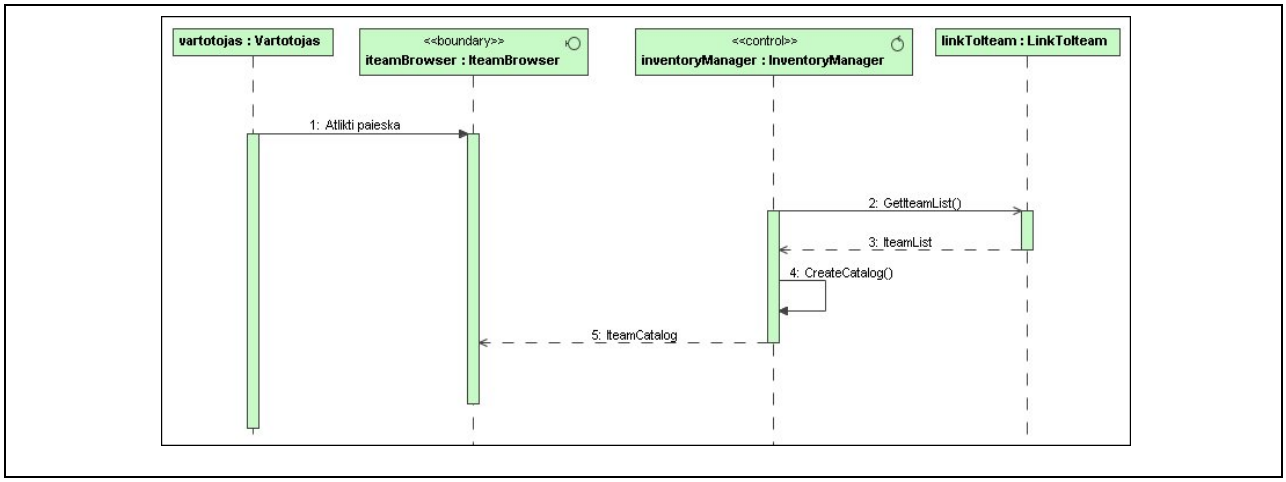
Būsenų diagrama

Bendra būsenų diagrama atvaizduoja sistemos būsenas bendruoju atveju, sistemos vartotojams atliekant įvairius veiksmus. Būsenų diagrama galima pamatyti 19. paveikslėlyje.

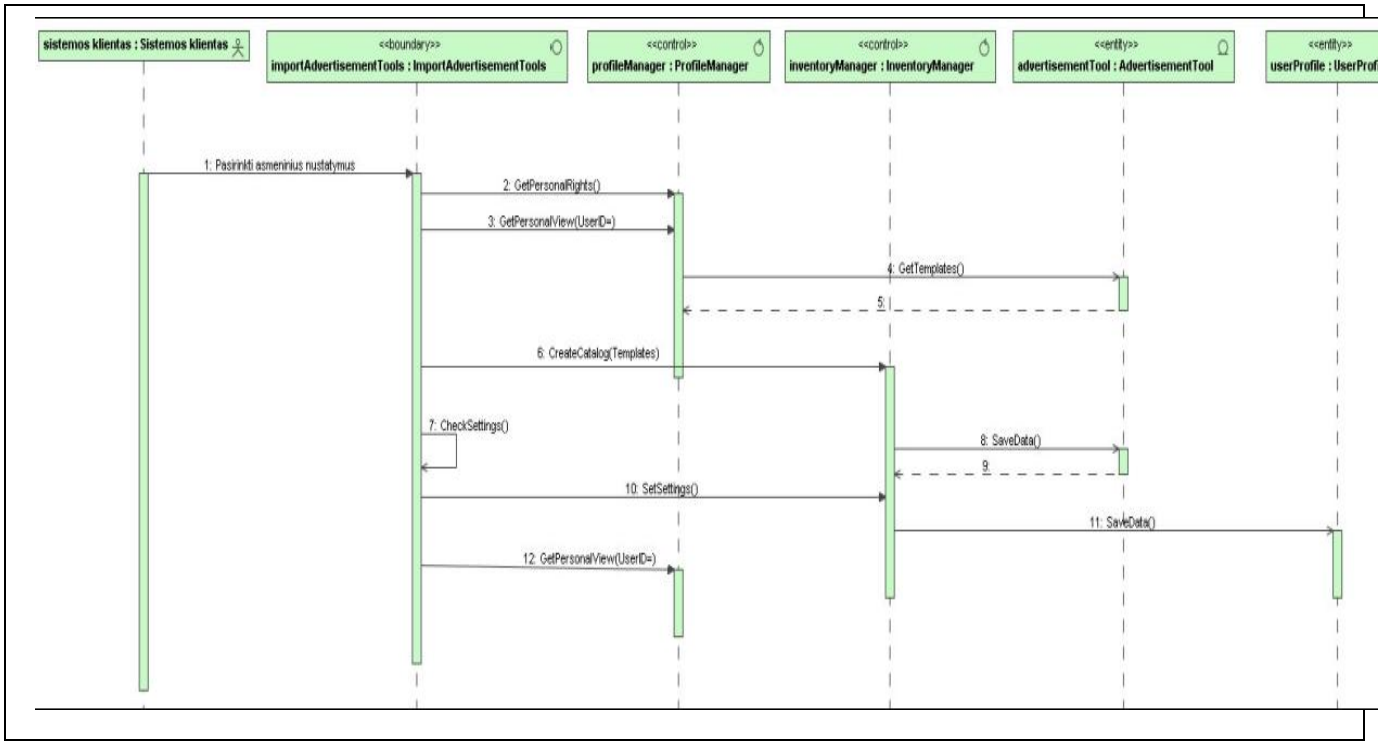


19. pav. Bendra sistemos būsenų diagrama

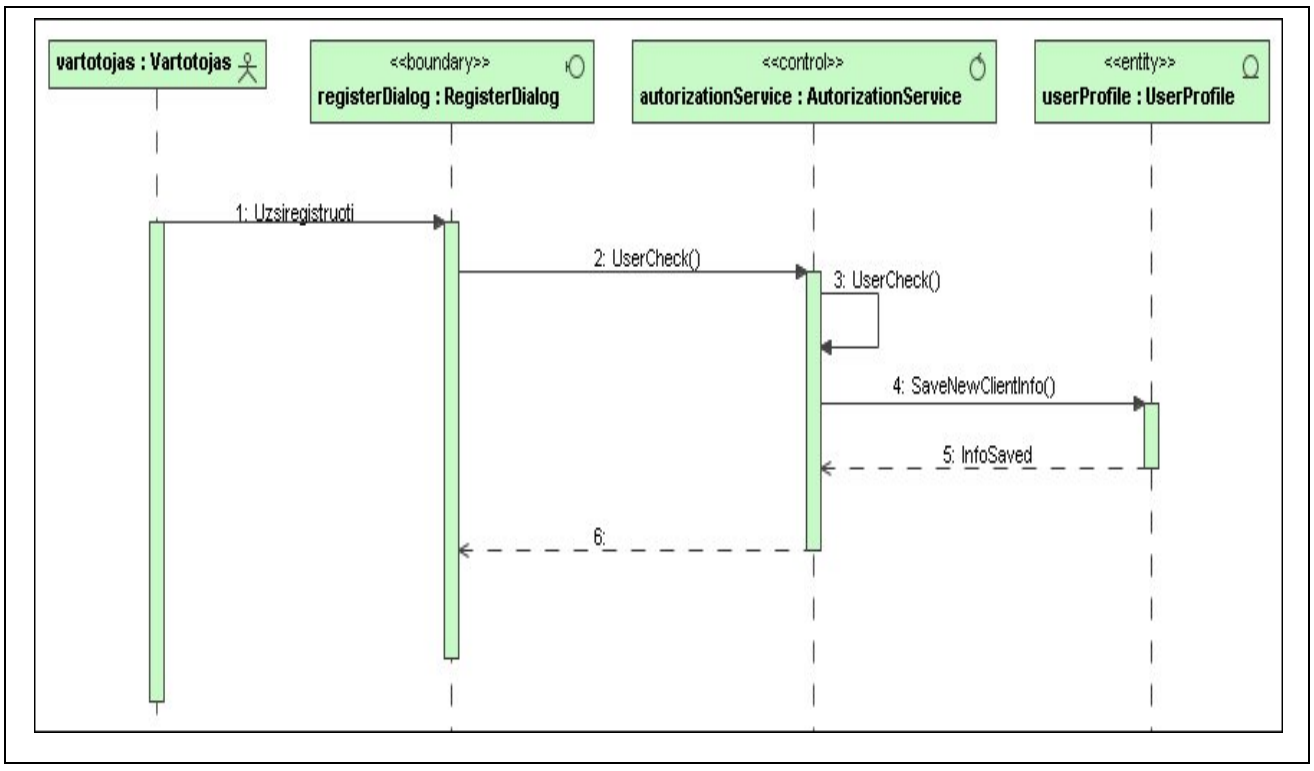
Pagrindinės sistemos sekų diagramos



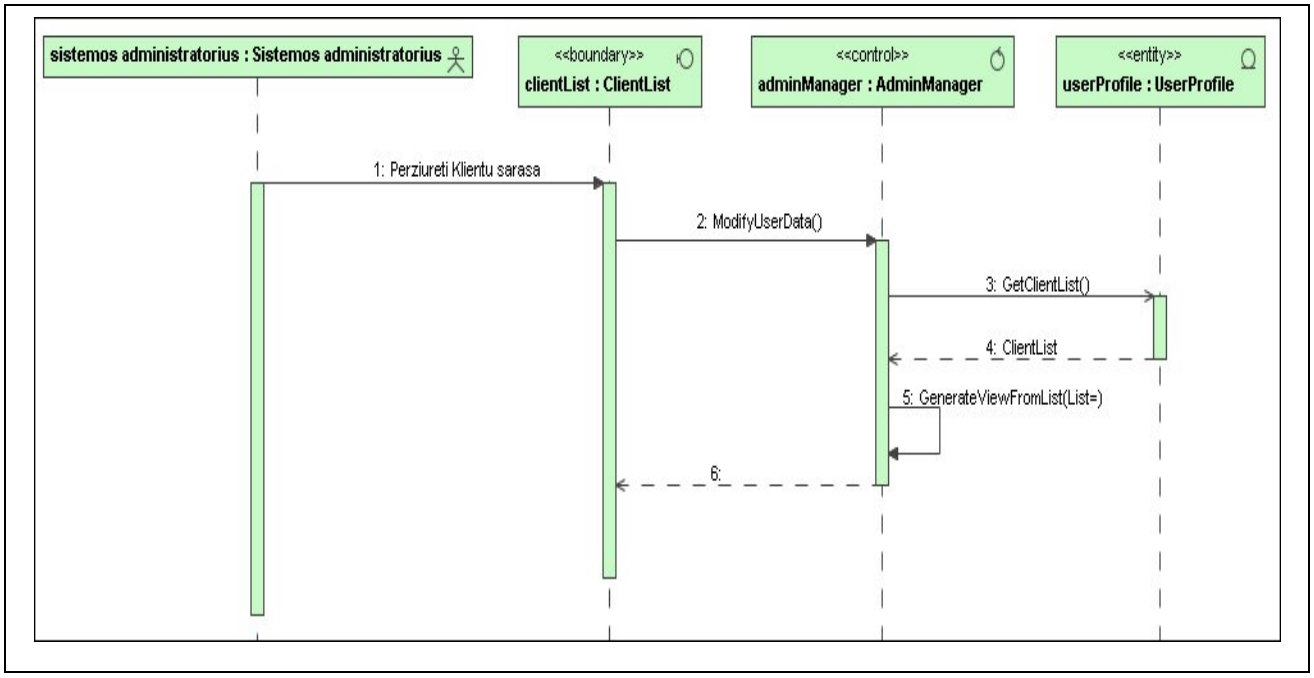
20. pav. Produktų katalogo peržiūra



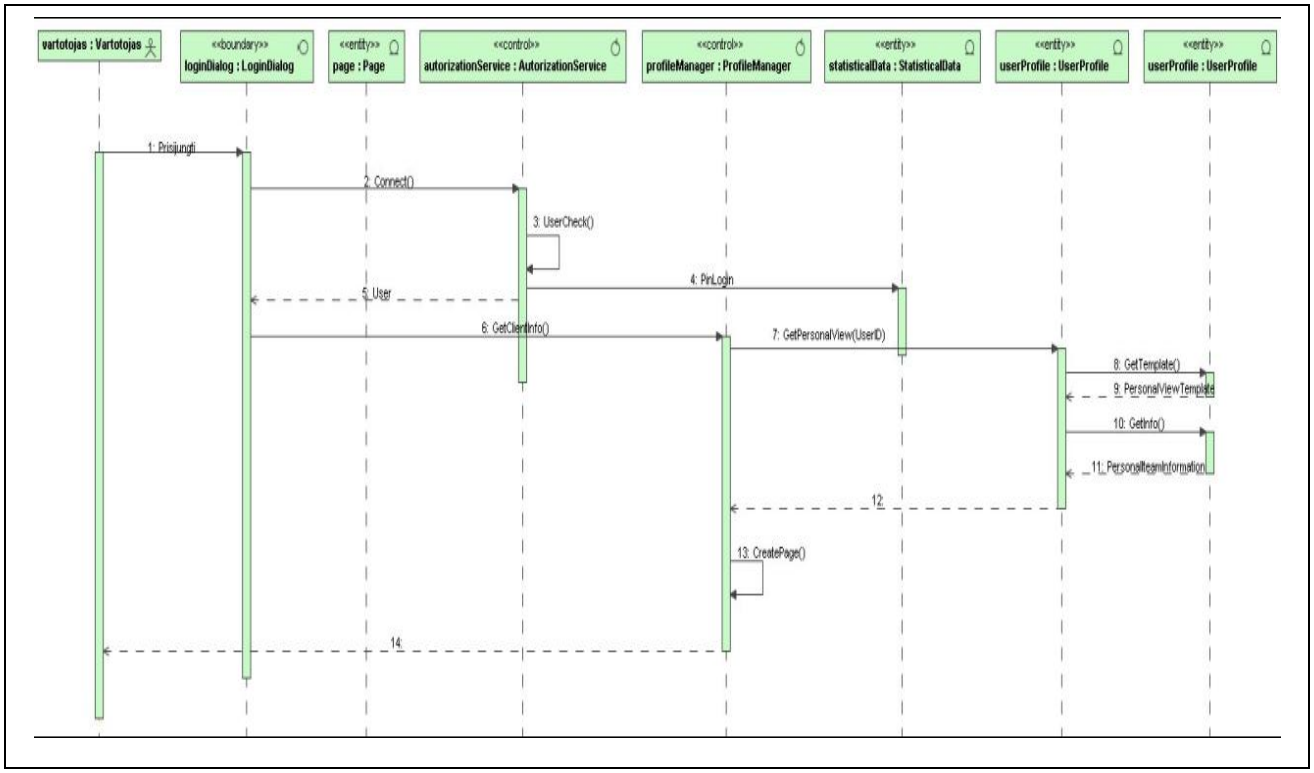
21. pav. Asmeninių nustatymų pasirinkimas



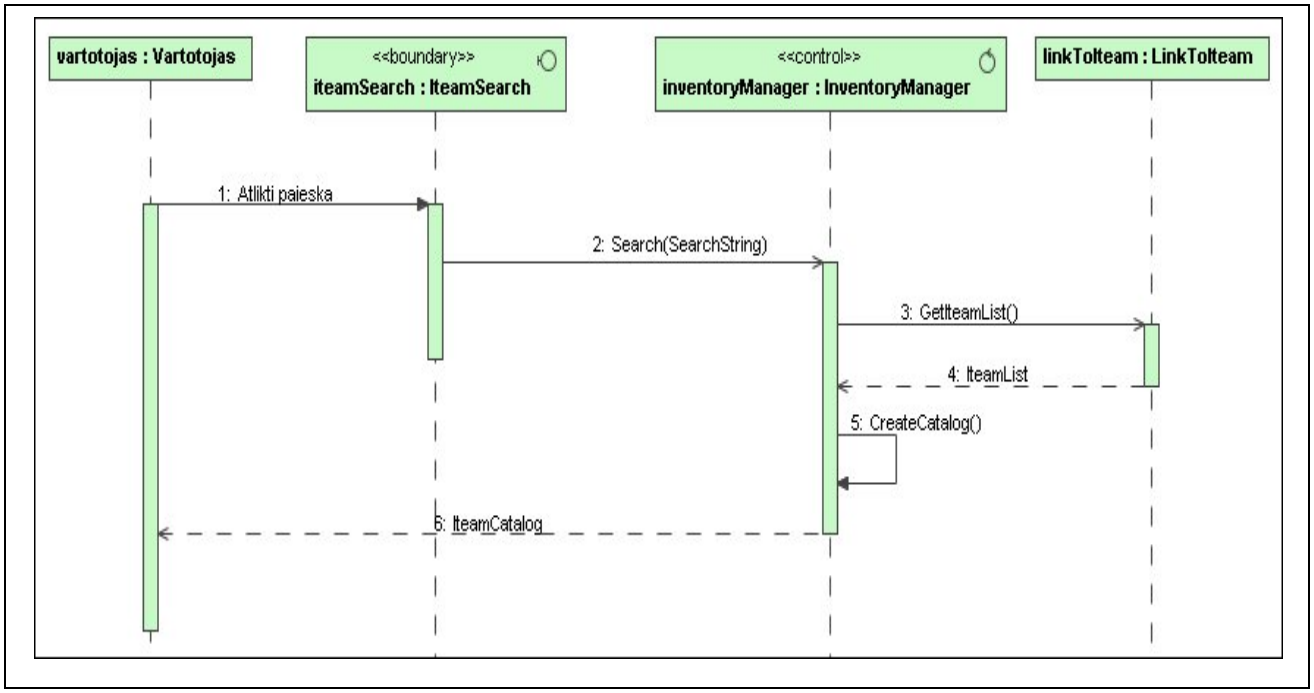
22. pav. Naujo profilio registravimas



23. pav. Ataskaitų generavimas

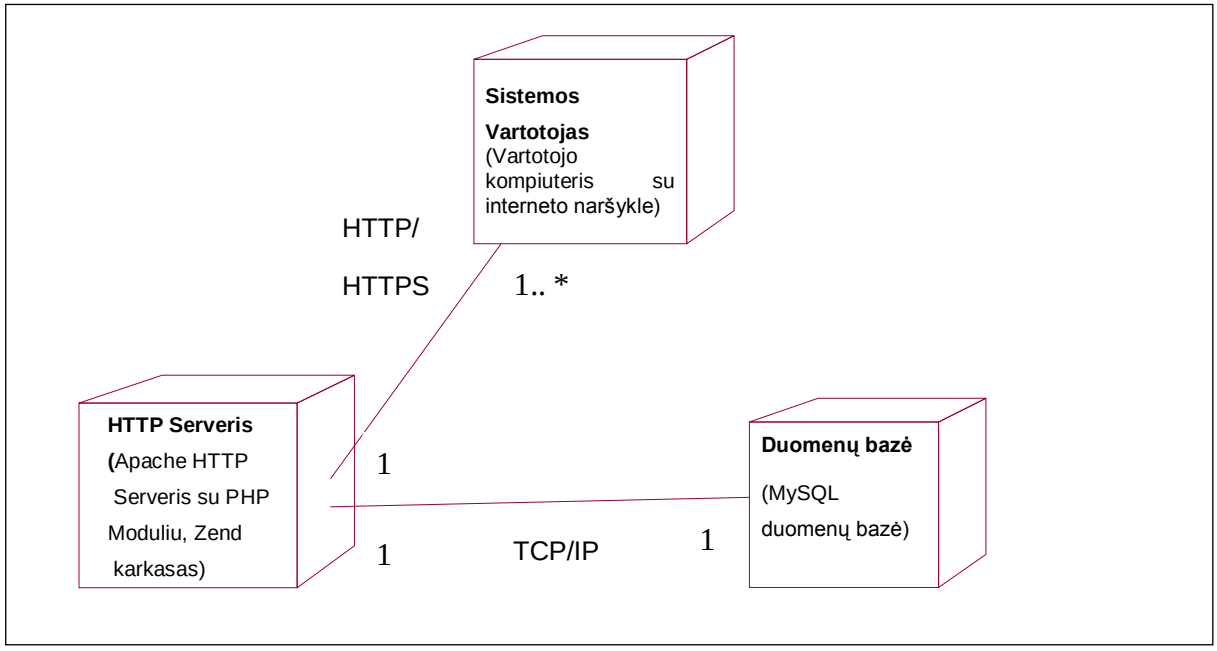


24. pav. Prisijungimas prie sistemos



25. pav. Dominančio produkto paieška

3.6.4 Išdėstymo vaizdas



26. pav. Sistemos išdėstymo vaizdas

- Sistemos klientas / vartotojas jungiasi prie sistemos naudodamas asmeniniame kompiuteryje suinstaliuotą interneto naršyklę. Klientas / vartotojas jungiasi prie sistemos iš bet kurios pasaulio vietos, kurioje yra interneto prieiga. Daugiau jokiu reikalavimų klientui / vartotojui nėra;
- Serveris palaikantis PHP programavimo kalbą, turintis 10mbps tarptautinio interneto srautą (reikalingą planuojamam vartotojų skaičiui tinkamai aptarnauti). Serveris turi palaikyti HTTPS protokolą. Serveris bus nuomojamas, tiksli jo buvimo vieta neturės įtakos sistemai. Visi pagrindiniai procesai vyks serveryje.
- Duomenų bazė – MySQL. MySQL duomenų bazėje bus saugomi visi sistemos duomenys. Duomenų bazė bus nuomojama, jos buvimo vieta sistemai įtakos neturės. Duomenys tarp internetinio serverio ir duomenų bazės keliaus naudojant TCP/IP duomenų perdavimo protokolą. Laba tikėtina, kad HTTP serveris ir MySQL duomenų bazė bus viename serveryje.

4. Tyrimo dalis

4.1 Tyrimo objektas ir tikslas

Tyrimo objektas – magistrantūros studijų metų vykdyto projekto „Web finansinės skaičiuoklės“ (kuriamos sistemos) prieigos teisių valdymo sistema.

Tyrimo tikslas – išanalizuoti esančius dabartinės prieigos valdymo sistemos trūkumus, pritaikyti vieną iš teorijoje aprašytų prieigos teisių valdymo modelių ir pateikti sukurtai sistemai šio modelio adaptavimo pasiūlymus

Laukiamas tyrimo rezultatas – tikimasi, kad tyrimo metu nustatyto prieigos teisių valdymo modelio adaptavimas sistemoje padės optimizuoti sistemos administravimą, sutaupys daug administratoriaus laiko ir administravimui skiriamų kaštų.

4.2 Kuriamos sistemos prieigos teisių valdymo tyrimas

Kuriant socialinį portalą, prie kurio jungsis daugelis nesusijusių žmonių ir redaguos informaciją būtina užtikrinti tam tikrą tvarką ir skelbiamų duomenų korektiškumą. Šiems tikslams pasiekti reikia sukurti sistemą, kuri skirtingiems vartotojams suteiktų skirtingas teises, teisių paskirstymas būtų vykdomas patogiai ir nereikalautų didelių pastangų iš administratorių, kadangi socialiniai tinklai gali būti labai dideli ir tokiuose tinkluose fiziškai neįmanoma prižiūrėti kiekvieno vartotojo. Teisių suteikimas ir atėmimas turėtų būti vykdomas automatiškai, žinoma neatmetant rankinio teisių paskirstymo galimybės.

Tam, kad kuriama sistema veiktų tinkamai ir joje skelbiama informacija būtų korektiška reikia, kad atsirastų daug sistemos vartotojų, kadangi tik daug sistemos vartotojų plačiai pasiskirsčiusių po miestą ar net visą šalį gali priversti kuriamą sistemą veikti pagal jos numatytą paskirtį. Kadangi sistemoje turėtų būti daugybė registruotų vartotojų, būtina užtikrinti informacijos suvedimo tvarką.

Šiuo metu sistemos prieigos teisių kontrolė vykdoma naudojant DAC modelį (plačiau modelis nagrinėjamas 2.2.2.2 skyrelyje) ir teisės vartotojams priskiriamos asmeniškai kiekvienam skirtingos pagal administratoriaus nuožiūrą. Toks teisių paskirstymas esant daugeliui vartotojų yra neefektyvus, kadangi kiekvienam subjektui reikia suteikti prieigą prie kiekvieno subjekto, tokiems veiksmams administratorius sugaišta begalę laiko. Žinoma jei sistemoje būtų nedaug vartotojų (iki 50) toks

modelis gal ir pasiteisintų, kadangi jis suteiktų administratoriui labai lanksčias galimybes skirstant prieigą prie daugelio objektų.

Remiantis išanalizuota literatūra analitinėje šio magistro darbo dalyje, tinkamiausias prieigos teisių valdymo modelis tiriamai sistemai būtų rolėmis paremtas prieigos teisių valdymo modelis RBAC (angl. Role-Based Access Control). Šis modelis yra tinkamiausias tiriamai sistemai dėl keleto priežasčių:

- Planuojama, kad sistema turės daug vartotojų, kurių prieigos teises būtina valdyti;
- Iš anksto žinomos vartotojų atsakomybės, kurios galėtų būti sugrupuotos į grupes ir priskirtos rolėms;
- Lankstaus administravimo galimybės;
- Galimybė automatizuoti rolių paskirstymą.

4.2.1 RBAC modelio pritaikymas tiriamai sistemai

Norint sistemoje naudoti RBAC modelį pirmiausia sistemą reikia išskaidyti į atskirus objektus, prie kurių būtų ribojamas ar leidžiamas subjektų priėjimas. Sistemą į atskirus objektus galima būtų skaidyti pagal sistemos architektų sudarytą techninę architektūrą išrenkant reikiamus, svarbius objektus prie kurių norima apriboti arba leisti priėjimą. Tokie objektai galėtų būti:

- Naujienos
- Svečių knyga
- Forumas
- Portalo nustatymai
- Vartotojų teisės
- Kaina
- Produktas
- Katalogas ir t.t.

Kiekvienam objektui galima būtų išskirti keturias pagrindines prieigos teises (leidimus), kurios leistų arba ribotų tam tikrų operacijų atlikimą:

- Skaitymas
- Rašymas
- Redagavimas
- Šalinimas

Vėliau reiktų išskirti sistemos subjektus. Sistemos subjektai – tai esybės kurioms yra ribojama prieiga prie anksčiau išskirtų sistemos objektų. Subjektus galima išskirti pagal suplanuotas sistemos

vartotojų veiklas, suskaidant vartotojus į tam tikras roles. Rolės galėtų būti skirstomos pagal tai, kiek kiekvienai rolei suteikiama teisių atlikti aukščiau minėtas operacijas ar veiksmus su objektais. Pagal paminėtus kriterijus galima būtų išskirti tokias sistemos roles:

- Neregistruotas vartotojas;
- Registruoto vartotojas;
- Kainų skelbėjas;
- Moderatorius;
- Administratorius.

Kai jau yra išskirti pagrindiniai sistemos objektai, subjektai (nustatytos rolės) ir išskirti leidimai galima sudaryti prieigos kontrolės sąrašą (angl. Access Control List) (ACL). Sąrašą galima sudaryti sudarant visų rolių ir objektų lentelę, kurioje stulpeliai atitiktų sistemos roles, o eilutės atitiktų sistemos objektus. Lentelės viduje turėtų būti surašomi atitinkamų rolių leidimai – atitinkamiems objektams. Žemiau 10. lentelė pateikiamas tokios lentelės pavyzdys.

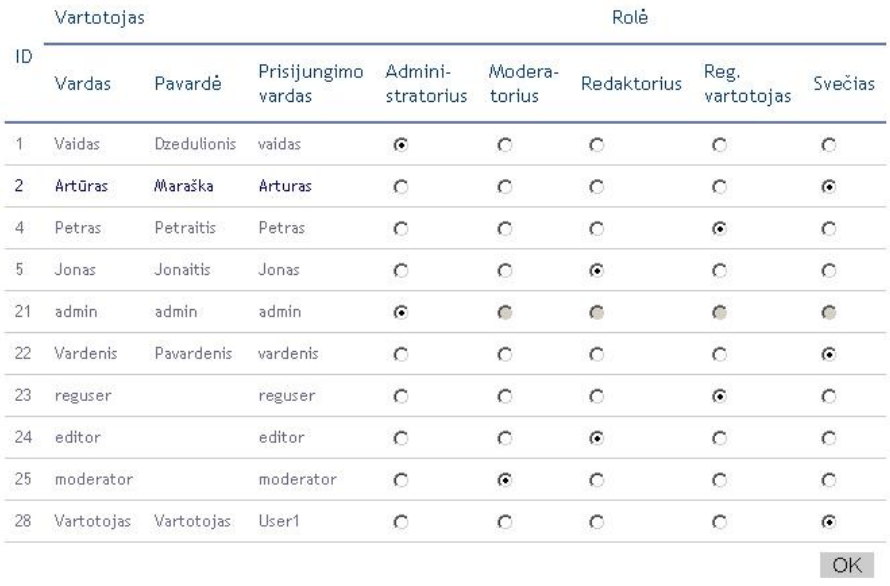
10. lentelė Siūlomas prieigos valdymo sąrašas

Rolės / Objektai	Neregistruotas vartotojas	Registruotas vartotojas	Redaktorius	Moderatorius	Administratorius
Naujienos	Skaityti	Skaityti	Skaityti, rašyti	Skaityti, rašyti, redaguoti, šalinti	Visos teisės
Svečių knyga	Skaityti	Skaityti, rašyti	Skaityti, rašyti	Skaityti, rašyti, redaguoti, šalinti	Visos teisės
Forumas	Skaityti	Skaityti, rašyti	Skaityti, rašyti	Skaityti, rašyti, redaguoti, šalinti	Visos teisės
Nustatymai				Skaityti, rašyti, redaguoti	Visos teisės
Vartotojų teisės					Visos teisės
Kaina		Skaityti	Skaityti, rašyti, redaguoti	Skaityti, rašyti, redaguoti, šalinti	Visos teisės
Produktas	Skaityti	Skaityti	Skaityti	Skaityti, rašyti, redaguoti, šalinti	Visos teisės
Katalogas	Skaityti	Skaityti	Skaityti	Skaityti, rašyti, redaguoti	Visos teisės

Kiekvienai iš šių rolių priklausantis vartotojas turėtų atitinkamus iš anksto suplanuotus ir priskirtus teisių rinkinius (roles). Vartotojų teisių rinkiniai turėtų būti apibrėžti iš anksto ir turėtų būti sudaryti

paveldimumo principu. Kiekvienas aukštesnis lygis paveldėtų visų žemesnių lygių teises. Pateiksiu pavyzdį aukščiau paminėtų vartotojų rolių ir jiems priskirtų teisių rinkinių:

1. Neregistruoto (svečio) vartotojo rolė:
    a. Naujienų peržiūra
    b. Forumo peržiūra
    c. Svečių knygos peržiūra
    d. Registracijos atlikimas
2. Registruoto vartotojo rolė:
    e. visos prieš tai buvusio lygio teisės (a, b, c ir d)
    f. naujos žinutės forume paskelbimas
    g. naujos žinutės svečių knygoje paskelbimas
3. Redaktoriaus rolė:
    h. visos prieš tai buvusio lygio teisės (a, b, c, d, e, f ir g)
    i. kainų skelbimas ir redagavimas
4. Moderatoriaus rolė:
    j. Visos prieš tai buvusių lygių teisės (a, b, c, d, e, f, g, h ir i)
    k. Forumo žinučių redagavimas ir šalinimas
    l. Svečių knygos įrašų redagavimas ir šalinimas
5. Administratoriaus rolė:
    m. Visos prieš tai buvusių lygių teisės (a, b, c, d, e, f, g, h, i, j, k ir l)
    n. Vartotojų teisių redagavimas
    o. Portalo nustatymų redagavimas



ID	Vartotojas			Rolė				
	Vardas	Pavardė	Prisijungimo vardas	Administratorius	Moderatorius	Redaktorius	Reg. vartotojas	Svečias
1	Vaidas	Dzedulionis	vaidas	●	○	○	○	○
2	Artūras	Maraška	Arturas	○	○	○	○	●
4	Petras	Petraitis	Petras	○	○	○	●	○
5	Jonas	Jonaitis	Jonas	○	○	●	○	○
21	admin	admin	admin	●	●	●	●	●
22	Vardenis	Pavardenis	vardenis	○	○	○	○	●
23	reguser		reguser	○	○	○	●	○
24	editor		editor	○	○	●	○	○
25	moderator		moderator	○	●	○	○	○
28	Vartotojas	Vartotojas	User1	○	○	○	○	●

OK

27. pav. Hierarchinis rolių paskirstymas

Naudojant tokią hierarchinę iš anksto sudarytą teisių paskirstymo sistemą būtų užtikrinama, kad teisės vartotojams bus paskirstytos be ypatingų pastangų. Ir tuo pačiu apsaugotų portalą nuo nepageidaujamų įvykių, tokių kaip sunaikinta svarbi informacija arba klaidingos informacijos paskelbimas. Sistemos rolių paskirstymą vartotojams galima matyti 27. paveikslėlyje. Administratorius norėdamas priskirti tam tikrą rolę vartotojui, tiesiog pasirenka vartotoją ir ties rolės stulpeliu pažymi mygtuką. Vartotojui iš karto suteikiamos visos pasirinktos rolės teisės. Plačiau apie RBAC modelio hierarchinę struktūrą galima paskaityti analitinėje dalyje, 2.2.2.3 skyriuje.

Aišku, šiame pavyzdyje yra pateikta tik keletas objektų, teisių rinkinių ir vartotojų rolių, kurie būtų tinkami tiriamai sistemai. Didesnėse sistemose tokių objektų, teisių rinkinių ir vartotojų lygių gali būti šimtai. Šių rinkinių skaičius priklauso nuo sistemos dydžio ir sistemoje esančių objektų, kuriuos reikia apsaugoti skaičiaus.

4.2.2 RBAC modelio patobulinimas ir adaptavimas

Vien tik pritaikius RBAC modelį sistemai išsprendžiamos tik kelios iš daugelio problemų, tačiau pagrindinės problemos – sistemos administracinio darbo sumažinimo RBAC modelis neišsprendžia. Tam kad būtų galima sumažinti administravimui skiriamo laiko tarpą, būtina patobulinti klasikinį RBAC modelį ir adaptuoti jį pagal kylančius poreikius.

Pagrindinis modelio patobulinimo tikslas yra sukurti tokią sistemą, kurioje prieigos teisės būtų paskirstomos automatiškai. Tiriamoje sistemoje turėtų būti pritaikytas automatinis teisių paskirstymas vartotojams, nes fiziškai gali būti sunku sukontroliuoti daugelio vartotojų skirtingas teises. Teisių paskirstymo automatizavimas padėtų sutaupyti daug administravimui skiriamo laiko. Automatinis teisių paskirstymas galėtų būti vykdomas pagal keletą kriterijų. Pavyzdžiui, vartotojui gali būti suteikiama daugiau teisių jei jis:

- paskelbia, tam tikrą iš anksto apibrėžtą, korektišką informacijos kiekį;
- atlieka tam tikrus pagalbinius portalo priežiūros darbus;
- padeda pašalinti portalo klaidas, pranešdamas apie jas administratoriams;
- suteikia naudingos informacijos apie tam tikru kitų vartotojų pažeidimus ar nesilaikymus bendrai priimtų taisyklių;
- būtų rekomenduojamas, kitų portalo vartotojų jau turinčių aukštesnes teises;

Trumpai galima būtų pasakyti, kad vartotojas turėtų įgyti tam tikrą pasitikėjimo lygį atsižvelgiant į aukščiau išvardintas sąlygas. Šie lygiai turėtų būti iš anksto išmatuoti, apibrėžti ir skelbiami portalo taisyklėse. Kiekvienam pasitikėjimo lygiui galima priskirti tam tikrą teisių rinkinį - rolę. Kuo didesnį pasitikėjimo lygmenį įgyja vartotojas, tuo daugiau teisių, naudojantis sistema, jis gauna. Toks teisių paskirstymo principas skatintų vartotojus elgtis pagal galiojančias portalo taisykles ir atrinktų tuos vartotojus, kurie norėtų prisidėti prie socialinio portalo tolimesnio vystymo, tvarkos palaikymo ir korektiškos informacijos skelbimo.

Pasitikėjimo lygio skalę galima būtų įvertinti 1000 balų. Šiuos balus vartotojai galėtų uždirbti atlikdami anksčiau paminėtus veiksmus. 1000 balu reikia suskaidyti į 4 dalis, tam kad būtų aišku, kiek balų turi surinkti vartotojas, norėdamas gauti aukštesnę sistemos rolę. Pasitikėjimo indeksas ir rolės galėtų būti susiejamos, taip kaip parodyta 11. lentelė . Žinoma, administratoriaus rolė turėtų būti suteikiama tik pačio administratoriaus, todėl administratoriaus rolė 11. lentelė nepateikiama. Vartotojai turintys 0 balų laikomi neregistruotais vartotojais, sistemoje gali būti tokių atvejų, kad administratorius sumažina bet kurio registruoto vartotojo pasitikėjimo indeksą iki 0 už tam tikrus nusižengimus, taip apribodamas jo veiklą sistemoje. Tik prisiregistravęs vartotojas gauna 1 pasitikėjimo indekso balą ir jam suteikiama registruoto vartotojo rolė. Tam kad vartotojas gautų redaktoriaus rolę, jam reikia surinkti 251 pasitikėjimo indekso balą. Tam kad gautų moderatoriaus rolę reikia surinkti dar papildomus 500 balų. Toks, nelygus balų pasiskirstymas yra todėl, kad moderatoriaus rolė suteikia gana nemažai prieigos teisių vartotojui ir žmogus gavęs šią rolę turi būti patikimas.

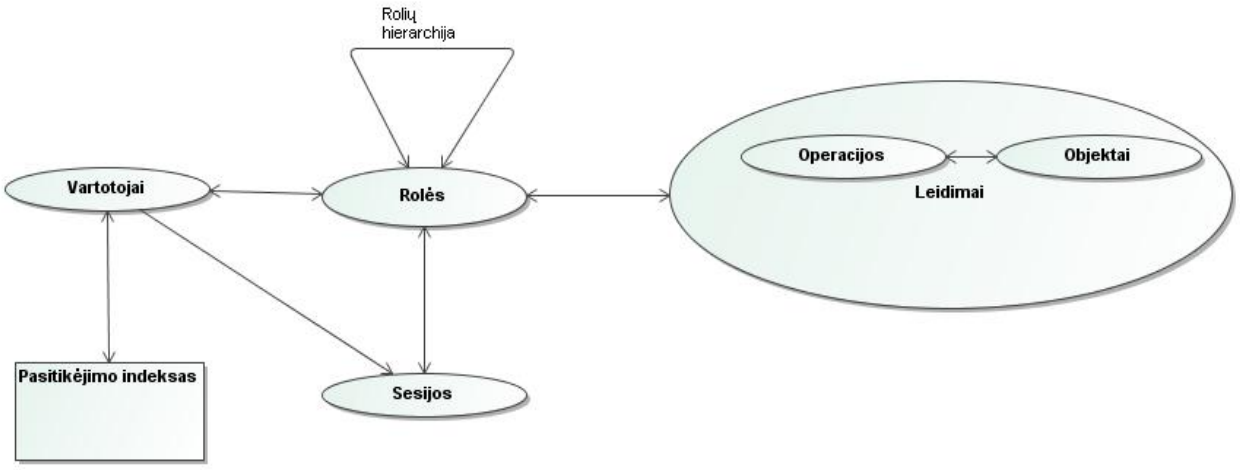
11. lentelė Pasitikėjimo indekso ir rolės ryšys

Pasitikėjimo indeksas	Rolė	Pasitikėjimo indeksas didėja	Pasitikėjimo indeksas mažėja
0	**Neregistruotas vartotojas (svečias)**	Registracija sistemoje	-
1 - 250	**Registruotas vartotojas**	Kainų teisingumo tikrinimas, klaidų raportavimas, prisidėjimas prie tvarkos palaikymo	Klaidingos informacijos skelbimas, taisyklių nesilaikymas
251 - 750	**Redaktorius**	Kainų skelbimas, aktyvumas, pranešimai apie akcijas, pranešima apie klaidas	Klaidingos informacijos skelbimas, taisyklių nesilaikymas
751 <	**Moderatorius**	Pasiekus šį lygį, pasitikėjimo indeksas nebeauga	Klaidingos informacijos skelbimas, taisyklių nesilaikymas, netinkamai naudojama priskirta rolė

Administratoriui turėtų būti suteikiama teisė redaguoti kiekvieno vartotojo pasitikėjimo indekso lygį (taip, kaip parodyta 27. paveikslėlyje), tam kad būtų galima automatiškai suteikti vartotojui, kuriuo pasitikima aukštesnę rolę, arba, kad būtų galima vartotojui, kuris užsitarnavo aukštą pasitikėjimo indeksą ir pradėjus sistemoje elgtis nekorektiškai, sumažinti jo pasitikėjimo indekso reikšmę, taip suteikiant jam žemesnę rolę arba suteikus vartotojui nulinį pasitikėjimo indeksą uždrausti jam naudotis sistema.

Po sistemos tyrimo nustatyta, kad kuriamai sistemai tinkamiausias prieigos teisių valdymo modelis yra rolėmis paremtas prieigos teisių valdymo modelis (angl. Role-Based Access Control) (RBAC). Plačiau šis modelis analizuojamas analitinėje šio darbo dalyje 2.2.2.3 skyriuje.

Šis modelis tinkamai užtikrina skirtingas prieigos teises, skirtingiems vartotojams, tačiau naudojant vien tik šį modelį, pastoviai reikia administruoti daugelio vartotojų prieigos teises. Šiam administravimui sugaištama itin daug laiko. Šiai problemai spręsti siūlomas patobulintas dinaminis RBAC modelis. Modeliui patobulinti įvedamas naujas sistemos objektas – pasitikėjimo indeksas, pagal kurio reikšmę bus dinamiškai priskiriamos atitinkamos vartotojo prieigos teisės. Sukurtas modelis pateikiamas 28. paveikslėlyje.



28. pav. Siūlomas patobulintas dinaminis RBAC modelis

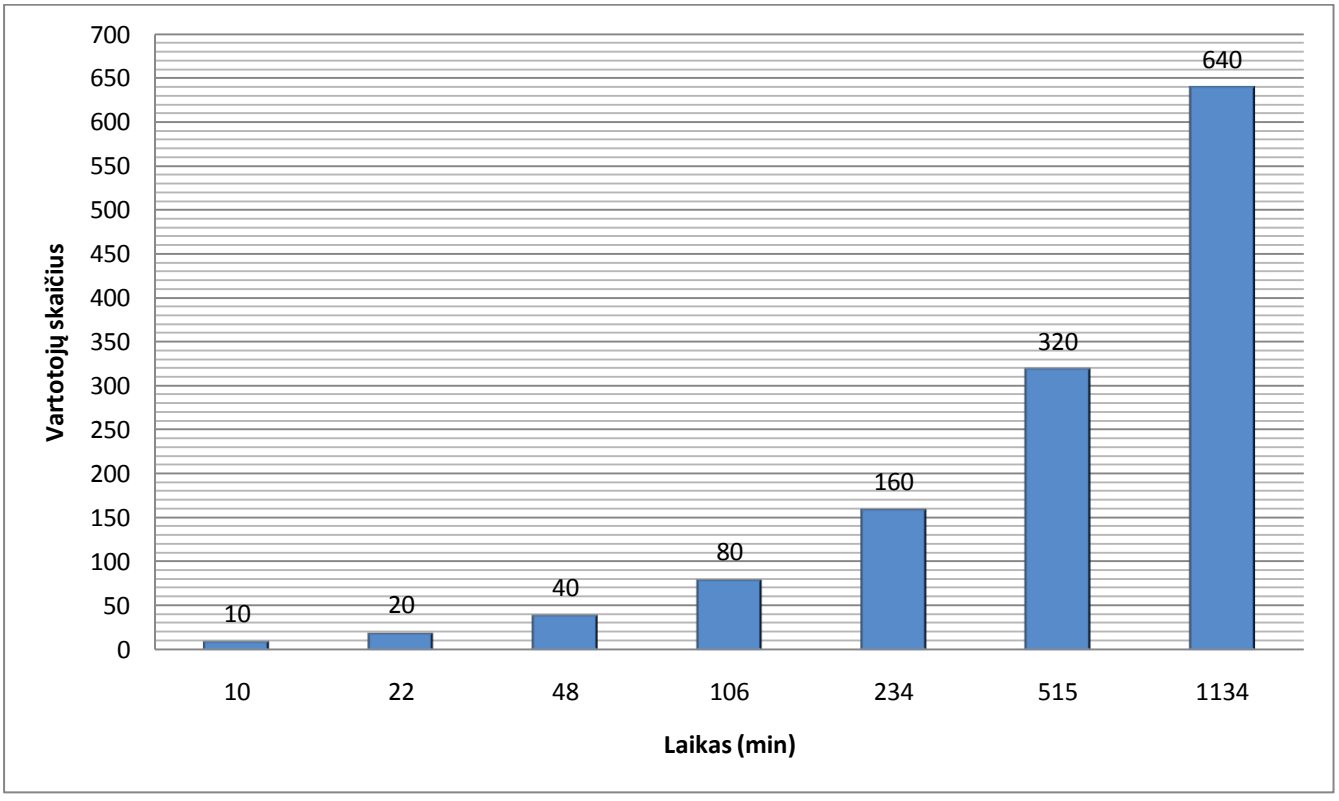
4.3 Tyrimo išvados

- Atlikus tyrimą buvo nustatyta, kad sistemos kritinė dalis, kurią reikia tobulinti yra sistemos teisių paskirstymo mechanizmas;
- Nustatyta, kad tinkamiausias sistemai prieigos teisių valdymo modelis yra RBAC (rolėmis paremtas prieigos teisių valdymo modelis);
- Nustatyta, kad tobulinant sistemą reikia kiekvienam vartotojui priskirti tam tikra pasitikėjimo indeksą, pagal kurį būtų galima priskirti tam tikrą rolę;
- Vartotojų teisės dinamiškai turi kisti priklausomai nuo pasitikėjimo indekso reikšmės, tokiu būdu taupomas brangus administravimo laikas ir kaštai;
- Vartotojai pasitikėjimo indeksą gali pasikelti skelbdami daugiau informacijos, raportuodami apie netinkamos informacijos atsiradimą, prisidėdami prie portalo vystymo;
- RBAC prieigos teisių valdymo modelio įgyvendinimui siūlomas Zend programavimo karkaso komponentas Zend_Acl, kuris padėtų lengvai įgyvendinti modelio keliamus reikalavimus.

5. Eksperimentinė dalis

Eksperimentinėje dalyje bandoma apskaičiuoti, kiek laiko sutaupo naujasis, dinamiškas RBAC prieigos teisių valdymo modelis. Laikas apskaičiuojamas vertinant praktiškai, kiekvieno registruoto vartotojo prieigos teisių kontrolei administratoriaus skiriamą laiką. Taip pat bandoma prognozuoti, koks bus sistemos rolių procentinis pasiskirstymas.

Praktiškai buvo apskaičiuota, kad sistemoje neįdiegus dinamiško RBAC modelio, kas dieną administratorius turėtų sugaišti po 10 minučių kiekvieniem 10 registruotų vartotojų. Į šias 10 minučių įskaičiuojama vartotojų įvestos informacijos peržiūra, nusiskundimų peržiūra ir prieigos teisių pakeitimų atlikimas. Įvertinant nuolatinį vartotojų skaičiaus augimą, buvo nustatyta, kad laikas skiriamas administruoti vartotojus auga ne vienodu tempu, o padaugėjus vartotojų dvigubai, laiko skiriamo administravimui reikia 2,2 karto daugiau. Kitaip tariant laikas skiriamas administravimui tiesiogiai nepriklauso nuo vartotojų skaičiaus augimo. Tai galima matyti pateiktame grafike 29. paveikslėlyje.

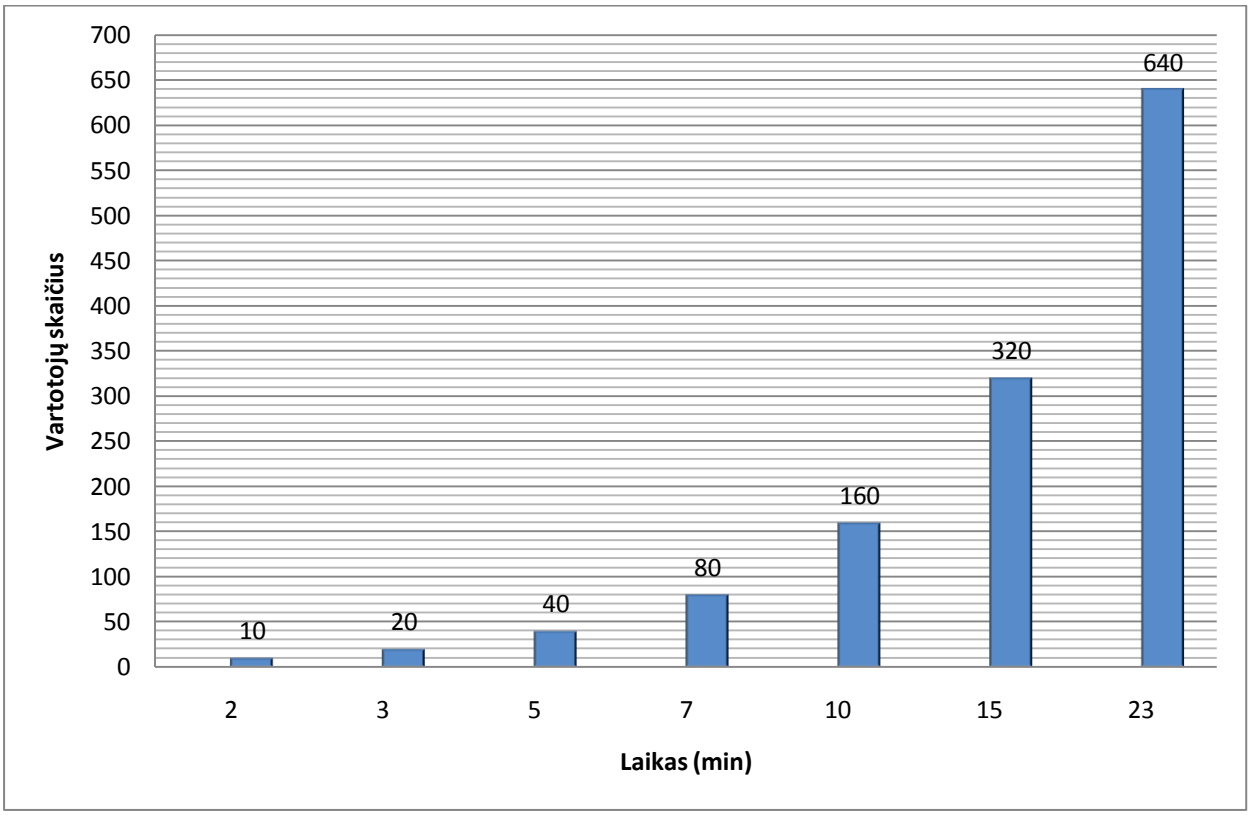


29. pav. Sistemos administravimo laikas, priklausomai nuo vartotojų skaičiaus

Vartotojų skaičiui padidėjus dvigubai administraivimo laikas padidėja daugiau nei dvigubai dėl keleto preižasčių. Viena iš jų yra ta, kad kuo daugiau sistemoje vartotojų, tuo daugiau tarp jų pačių kyla konfliktų, tuo dažniau kyla nusiskundimų ir tuo sunkiau juos išsiaiškinti.

Iš šio 29. pav grafiko galima aiškiai matyti, kad daugėjant vartotojų, sistemos administravimas taptų labai sudėtingas. Sistema būtų sunkiai kontroliuojama jei administratoriaus darbą atliktų vienas žmogus. Šiai problemai spręsti reikėtų arba samdyti daugiau žmonių padėsiančių administruoti sistemą arba reikėtų keisti prieigos teisių valdymo sistemą.

Pakeitus prieigos teisų valdymo sistemą ir įdiegus tyrimo metu patobulintą dinaminį RBAC modelį, sutaupoma nemažai administratoriaus laiko ir tokią sistemą gali prižiūrėti vienas administratorius, kadangi šiuo atveju daugėjant sistemos vartotojų dvigubai, laiko skiriamo administravimui poreikis išaugtų tik 1,5 karto. Tai galima matyti iš pateikto grafiko 30. paveikslėlyje.



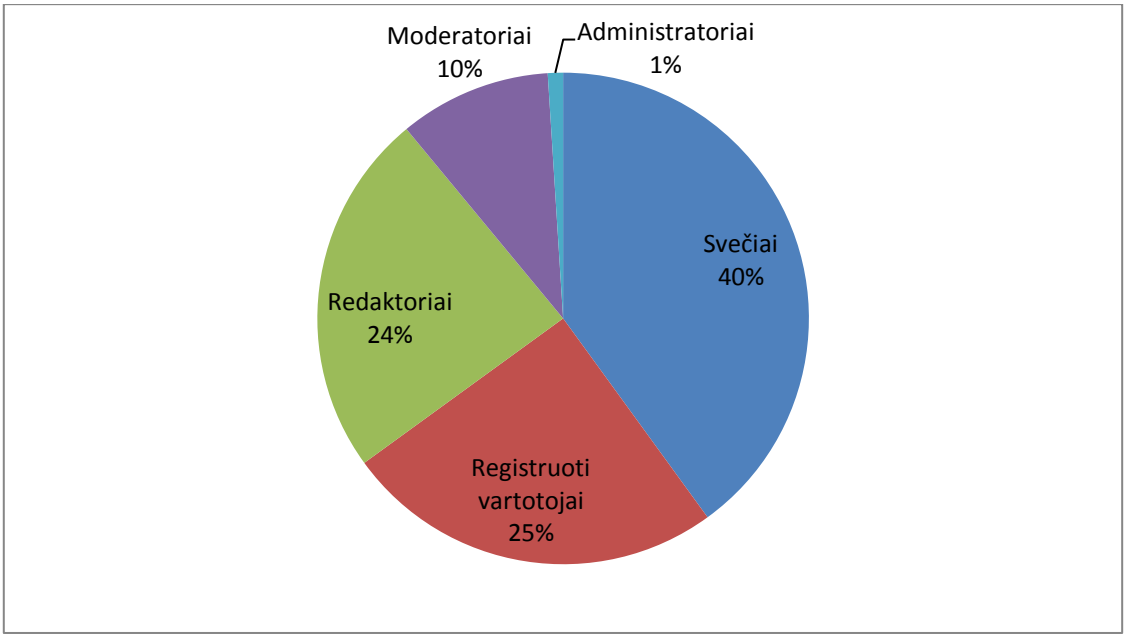
30. pav. Vartotojų administravimo laikas naudojant dinaminį RBAC modelį

Šiuo atveju, laiko administravimui, galima sakyti, skirti iš viso nebereiktų, nes rolės paskirstomos automatiškai panaudojant pasitikėjimo indeksą, bet yra įvertinama tai, kad tarp sistemos vartotojų vis tiek kils konfliktų, kurių pati sistema negalės išspręsti, tokių konfliktų sprendimui ir

buvo nuspręsta skirti šiek tiek laiko. Konfliktai gali būti tokie, pavyzdžiui, kai neteisingai pranešama apie nusižengimus ar netikslios informacijos skelbimą.

Atlikus eksperimentinius administratoriaus darbo laiko skaičiavimus buvo nustatyta, kad įdiegus RBAC modelį administratoriaus darbo laikas nebus ypatingai įtakojamas vartotojų skaičiaus didėjimo, o dar vertinant tai, kad daugėjant sistemos vartotojų, didės ir ją administruojančių žmonių skaičius, galima teigti, kad sistemos administravimui reikalingas laiko tarpas taps minimalus.

Skaičiuojant administravimui skiriamą laiką buvo paskaičiuotos prognozės, koks ir kokių rolių pasiskirstymas bus tarp sistemos vartotojų. Šį pasiskirstymą galima pamatyti 31. paveikslėlyje.



31. pav. Prognozuojamas rolių pasiskirstymas sistemoje

Pastaba. Eksperimento metu laiko trukmės buvo skaičiuojamos simuliuojant sistemos darbą su 80 sistemos vartotojų. Visos laiko trukmės pateiktos esant daugiau nei 80 vartotojų buvo prognozuojamos pagal vartotojų daugėjimo ir laiko sugaišimo tendencijas. Visi skaičiai buvo imami vertinant vidutines darbui atlikti reikalingas laiko trukmes, įvertinant galimus netikėtus nutikimus ir panaudojant asmeninę sistemų administravimo patirtį. Laikas buvo apskaičiuotas laikant, kad sistemą administruoja vienas administratorius, todėl daugėjant sistemos administratorių ir moderatorių, sistemos priežiūros laikas gali skirtis.

Vartotojų rolių pasiskirstymo procentai buvo pateikti pagal sistemos užsakovų ir kūrėjų prognozes.

6. Išvados

Šiuo darbu buvo siekiama optimizuoti socialinio portalo prieigos teisių valdymo sistemą.

Pagrindiniai atlikti darbai ir pasiekimai:

1) Išanalizuotos ir pasirinktos rinkoje esančios technologijos, leidžiančios įgyvendinti užsibrėžtus reikalavimus;
    - Pasirinkta PHP programavimo kalba;
    - MySQL duomenų bazių valdymo sistema;
    - Zend programavimo karkasas;

    Visos pasirinktos technologijos yra nemokamos ir prieinamos kiekvienam vartotojui. Zend programavimo karkasas pasirinktas dėl to, kad jis atitiko visus keliamus reikalavimus, turi pilną dokumentaciją ir plačią programuotojų bendruomenę; Išanalizuotas Zend karkaso Zend_Acl komponentas, kuris skirtas prieigos teisių kontrolės sistemos kūrimui;
2) Išanalizuoti pagrindiniai sistemos prieigos teisių valdymo modeliai;
    - DAC – savarankiško prieigos valdymo modelis;
    - MAC – privalomos prieigos kontrolės modelis;
    - RBAC – rolėmis paremtas prieigos kontrolės modelis;
3) Atliktas sukurto portalo prieigos teisių kontrolės tyrimas, nustatyta, kad tinkamiausias sistemai prieigos teisių valdymo modelis yra RBAC (rolėmis paremtas prieigos valdymo modelis), kurio pagalba sistemoje buvo išskirtos penkios pagrindinės rolės ir šioms rolėms buvo suteikti tam tikri sistemos leidimai. Rolės suskirstytos hierarchiniu principu – kiekviena aukštesnė rolė paveldi prieš tai buvusios rolės leidimus;
4) Pasiūlytas pakoreguotas RBAC modelis. Prieigos teisių paskirstymas vykdomas panaudojant vartotojo pasitikėjimo indeksą, kuris priklauso nuo vartotojo veiklos sistemos ribose, prieigos teisės vartotojams paskirstomos dinamiškai, priklausomai nuo pasitikėjimo indekso reikšmės;
5) Pasiūlytas modelis žymiai sumažina portalo administravimui reikalingo laiko tarpą, tuo pačiu mažindamas ir administravimui reikalingus kaštus.
6) Atlikti eksperimentiniai prieigos teisių administravimui skiriamo laiko skaičiavimai prieš įdiegiant ir įdiegus RBAC modelį. Nustatyta, kad įdiegus patobulintą RBAC modelį žymiai sumažinamas administravimui skiriamas laiko tarpas ir prieigos teisių administravimas tampa efektyvesnis.

7. Literatūra

1. TIM CONVERSE; JOYCE PARK; CLARK MORGAN, *PHP5 and MySQL Bible*. 2004m. ISBN: 0-7645-5746-7
2. ROB ALLEN; NICK LO; STEVEN BROWN, *Zend Framework in Action*. 2008m. ISBN 1-9339-8832-0
3. MICHELE E. DAVIS; JON A. PHILLIPS, *Learning PHP & MySQL (2nd Edition)*. 2007m. ISBN-13: 978-0-596-51401-3
4. HOSSEIN BIDGOLI, *Handbook of information security*. 2006 ISBN 0-4712-2201-1
5. VINCENT C. HU; DAVID F. FERRAIOLO; D. RICK KUHN, *Assessment of Access Control Systems*. 2006 Interagency report 7316. National Institute of Standards and Technology, Gaithersburg USA. [žiūrėta 2009-05-20] Prieiga per internetą: http://www.securitytechnet.com/resource/crypto/standard/fips/NISTIR-7316.pdf
6. LAURI I. W. PESONEN, *A capability-based access control architecture for multi-domain publish/subscribe systems.* Technical report, number 720, University of Cambridge 2008m. [žiūrėta 2009-05-16] Prieiga per internetą: http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-720.pdf
7. KIRAN K. GOLLU; STEFAN SAROIU; ALEC WOLMAN, "A Social Networking-Based Access Control Scheme for Personal Content" 2007 [žiūrėta 2009-05-16] Prieiga per internetą: http://www.cs.toronto.edu/~kkgollu/docs/kkgollu_access.pdf
8. M.HITCHENS; V.VARADHARAJAN, *Design and specification of role based access control policies Software*. IEE proceedings 2000. ISSN: 1462-5970
9. KAR LEONG ONG. *Design and implementation of Wiki services in a multilevel secure environment*. Magistro darbas. Naval Postgraduate School. Monterey, California 2007
10. MATTHEW J. MOYER; MUSTAQUE AHMAD, *Generalizes Role-Based Access Control*. Proceedings of the 21st International Conference on Distributed Computing Systems, (ICDCS) 2001, ISNN: 1063-6927
11. JIN WANG; QIANG LI; DAXING LI, I-RBAC: *An Identity & Role Based Access Control Model*. 2007 IEEE International Conference on Control and Automation , ICCA 2007, ISBN: 978-1-4244-0817-7

12. JONATHAN M. MCCUNE; STEFAN BERGER, RAMON CACERS, TRENT JAEGER, REINER SAILER, *DeuTeRiuM – A system for distributed mandatory access control*. IBM research report 2006 [žiūrėta 2009-05-15] Prieiga per internetą: http://domino.research.ibm.com/library/cyberdig.nsf/papers/EAC6708DB1C54CF38525710F005B7AFF/$File/rc23865.pdf
13. MATUNDA NYANCHAMA; SYLVIA OSBORN, *Access rights administration in role-based security systems*. Proceedings of the IFIP WG11.3 Working Conference on Database Security VII, 1994, ISBN: 0-444-81976-2
14. BINDIGANAVALE V.;JINSONG OUYANG, *Role based Access control in enterprice application – security administration and user management*. 2006 IEEE international conference on information reuse and integration.
15. T. FININ; A. JOSHI; L. KAGAL; J. NIU; R. SANDHU; W. WINSBOROUGH; B. THURAISINGHAM, *Role based Access control and OWL*. Straipsnis [žiūrėta 2009-05-12] Prieiga per internetą http://www.webont.org/owled/2008dc/papers/owled2008dc_paper_21.pdf
16. DEVDATTA KULKARNI; ANAD TRIPATHI, *Context-aware role-based Access control in pervasive computing systems*. Proceedings of he 13th ACM symposium on Access control models and technologies 2008, ISBN:978-1-60558-129-3
17. NINGHUI LI; JI WON NYUN; E. BERNITO, *A critique of the ANSI standard on role-based Access control*, IEEE publikacija, 2007. ISSN: 1540-7993
18. PAOLINA CENTONAZE; GLEB NAUMOVICH; STEPHAN J. FINK; MARCO PISTOIA, *Role-based Access control consistency validation*. Proceeding of the 2006 international symposium on Software testing and analysis. Publikacija ISBN: 1-59593-263-1
19. PHP naudojimo statistika [žiūrėta 20090421], prieiga per internetą: http://www.nexen.net/chiffres_cles/phpversion/18519-php_statistics_for_june_2008.php#sglobal
20. Web serverių naudojimo statistika [žiūrėta 20090421], prieiga per internetą: http://news.netcraft.com/archives/web_server_survey.html
21. Straipsnis: *Access control*, Internetinė enciklopedija Wikipedia, [žiūrėta 2009-05-15], prieiga per internetą http://en.wikipedia.org/wiki/Access_control
22. LARRY ULLMAN, *MySQL (second edition)*. 2006 ISBN0-3213-7573-4
23. KEVIN MCARTHUR, *Pro PHP: Patterns, Frameworks*, Testing and More. 2008 ISBN1-5905-9819-9

24. Zend programavimo karkaso dokumentacija internete. *Zend Framework Documentation* [žiūrėta 2009-05-09] prieiga per internetą: http://framework.zend.com/manual/
25. ELISA BERTINO; PIERANGELA SAMARATI; SUSHIL JAJODIA, *High assurance discretionary access control for object bases*. Proceeding of the 1st ACM conference on Computer and communications security, 1993 ISBN 0-89791-629-8

8. Terminų ir santrumpų žodynas

ACL - (angl. Access Control List) prieigos teisių sąrašas

Apache - internetinis serveris

ASP - (angl. Active Server Pages) programavimo kalba

CGI - (angl. Common Gateway Interface) protokolas apibrėžiantis, kaip turėtų bendrauti WWW serveris ir jo vykdomos programos

DAC - (angl. Discretionary Access Control) savarankiškos prieigos teisių kontrolės modelis

GPL - (angl. General Public License) atviro kodo licencijos tipas

GPS - (angl. Global Positioning System) globalios pozicijos nustatymo sistema

GVS - grafinė vartotojų sąsaja

HTML - (angl. Hypertext Markup Language) internetinių puslapių kūrimo meta programavimo kalba

HTTP - (angl. Hypertext Transfer Protocol) hiperteksto perdavimo protokolas

Java - programavimo kalba

LDAP - (angl. Lightweight Directory Access Protocol) katalogų prieigos protokolas

Linux - operacinė sistema

MAC - (angl. Mandatory Access control) privalomos prieigos kontrolės modelis

MySQL - duomenų bazių valdymo sistema

MVC - (angl. Model View Controller) programų architektūros šablonas

PDF - dokumentų išsaugojimo formatas

Perl - programavimo kalba

PHP - (angl. Personal Home Page) programavimo kalba

PHP MyAdmin - MySQL duomenų bazės valdymo vartotojo sąsaja

Python - programavimo kalba

RBAC - (angl. Role-based Access Control) rolėmis paremtas prieigos kontrolės modelis

Smarty - PHP programavimo karkasas

SQL - (angl. Structured Query Language) struktūruota duomenų bazės užklausų kalba

OS - (angl. Operating System) operacinė sistema

UNIX - operacinė sistema

MS Windows - operacinė sistema

WYSIWYG - (angl. What You See Is What You Get) principo „ką mati tą ir gauni“ angliškas sutrumpinimas

WWW - (angl. World Wide Web) pasaulinis internetinis tinklas

XML - (angl. eXtensible Markup Language) tekstinis duomenų formatas

Zend Framework - PHP programavimo karkasas

Zend_Acl - Zend programavimo karkaso komponentas, ACL sąrašų sudarymui