KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Vytas Sinkevičius

Saugos priemonių panaudojimo tyrimas ir pritaikymas Framework programinėms konstrukcijoms

Magistro darbas

Darbo vadovas
doc. dr. A. Venčkauskas

Kaunas, 2010

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
KOMPIUTERIŲ KATEDRA

Vytas Sinkevičius

Saugos priemonių panaudojimo tyrimas ir pritaikymas Framework programinėms konstrukcijoms

Magistro darbas

Recenzentas

doc. dr. A. Lopata

2010-05-24

Vadovas

doc. dr. A. Venčkauskas

2010-05-24

Atliko

IFN-8/3 gr. stud.
Vytas Sinkevičius
2010-05-24

Kaunas, 2010

Turinys:

Research of security tools implementation in practice and its application for Framework programmable constructions

Summary

In a modern world, there is a big demand for various programmable systems. Highly important aspect in system development is time. For effective (fast and qualitatively) system development there are widely used system frames, widely known as Framework. For system developers (programmers) Framework gives the ability to develop various business systems very fast and effectively, depending on Framework purposes.

After exploring the usage of Framework‘s, when developing systems for small and middle business, it was observed that in such systems the attention to the security is noticeably low. For that reason was suggested a complex tool, which is oriented for such Framework‘s modification. Main purpose of this modification – to solve Framework systems security problems, defined in this work.

.NET platform and MS SQL Server DBMS was selected to perform work. By experiment accomplished analysis was appreciated affect of security tools to the system performance, convenience of secure Framework system and comparison between improved Framework system and usual Framework system.

Santrauka

Šiuolaikiniame pasaulyje, esant dideliam įvairių programinių sistemų poreikiui, labai svarbus aspektas sistemų kūrime – laikas. Tam, kad efektyviai, t.y. greitai ir kokybiškai kurti sistemas, plačiai naudojami sistemų karkasai, labiausiai žinomi terminu Framework. Sistemų kūrėjus (programuotojus) Framework karkasai įgalina greitai ir efektyviai sukurti įvairius programinius verslo sprendimus, priklausomai nuo Framework paskirties.

Išnagrinėjus Framework panaudojimą, kuriant mažo bei vidutinio dydžio organizacijoms skirtas sistemas, pastebėta, kad jose mažai dėmesio skiriama sistemos saugumui. Dėl šios priežasties pasiūlytas priemonių kompleksas, orientuotas į tokių sistemų modifikavimą, kurio tikslas – išspręsti apibrėžtas darbe Framework sistemų saugumo problemas.

Darbui atlikti pasirinkta .NET platforma ir MS SQL Server DBVS. Eksperimento metu atlikti tyrimai, įvertinantys panaudotų saugos priemonių įtaką greitaveikai, saugios Framework sistemos patogumo galimybes bei palyginantys įprastą Framework sistemą su patobulinta Framework sistema.

1. ĮVADAS

Verslo procesus įgyvendinančios sistemos, paremtos objektiškai orientuotomis kūrimo technologijomis ir reliacinėmis duomenų bazėmis, tapo labai populiarios. Didelį verslo efektyvumą organizacijoms suteikia būtent tokios sistemos ir plačios jų pritaikymo individualiems poreikiams galimybės, o sistemų funkcionalumas dažnai pakeičia keleto žmonių darbą. Tačiau gilinantis į sistemų funkcionalumą ir naudą, dažnai pamirštamas sistemos saugumo faktorius.

Šiame darbe analizuojamos Framework tipo sistemos, kurios dažniausiai tampa pagrindu verslo sistemoms kurti. Framework sistemos paremtos universaliais metodais ir funkcijomis, kurie turi perdengimo galimybę. Panaudojant Framework sistemos galimybes, galima sukurti daugelį verslo valdymo sistemų.

Šio darbo paskirtis – pasiūlyti Framework sistemos sprendimą, kuris būtų ne tik plačiai panaudojamas, bet ir tuo pačiu saugus. Vartotojų ir jų grupių politika, teisių valdymas, apsauga nuo SQL injekcijų, duomenų ir konfigūracinių failų apsauga – tai pagrindiniai aspektai, kuriais analizuojama Framework sistema.

Atlikus saugumo grėsmių ir galimų sprendimo būdų analizę, yra parenkamos priemonės ir technikos, kurios geriausiai tinka mūsų Framework sistemai. Remiantis tuo sukuriamas Framework sistemos patobulinimas, kurio paskirtis – išspręsti apibrėžtas saugumo spragas, nenutolstant nuo pagrindinių Framework sistemų kūrimo principų.

Tyrimo metu, naudojant atsitiktinius duomenų rinkinius, nustatoma saugumo priemonių (šifravimo algoritmo, šifravimo rakto ilgio, SPROC technikos, šifruoto konfigūracinio failo) įtaka sistemos greitaveikai bei patobulintos Framework sistemos naudojimo patogumas bei palyginimas su įprastine Framework sistema.

Šio darbo metu sukurti sprendimai įdiegti „Ranbaxy“ verslo valdymo sistemoje.

2. FRAMEWORK PAGRINDU SUKURTŲ TAIKOMŲJŲ PROGRAMŲ SAUGOS PROBLEMŲ ANALIZĖ

2.1. Analizės tikslas

Analizės tikslas – ištirti saugumo grėsmes, kurios aktualios Windows taikomosioms programoms, įgyvendinančioms vidutinių ir didelių įmonių verslo logiką ir paremtoms duomenų saugojimu SQL Server duomenų bazėse. Taip pat ištirti būdus ir priemones skirtas apsaugoti duomenų bazes, jose saugomą informaciją, užtikrinti informacijos vientisumą ir slaptumą (konfidencialumą).

2.2. Tyrimo sritis, objektas ir problema

Taupydamos laiko ir kitų išteklių sąnaudas, verslo įmonės ir organizacijos perkelia savo verslo logiką į specifines, dažniausiai pagal užsakymą sukurtas taikomąsias programas, paremtas Framework konstrukcijomis, kurios įgalina paspartinti įvairių duomenų įvedimą, užsakymų ir pasiūlymų formavimą, automatinį ataskaitų generavimą, duomenų importą į sistemą iš tiekėjo pateiktų failų ir taip toliau. Kadangi tokio tipo logikoje figūruoja dideli duomenų kiekiai, jiems saugoti dažniausiai naudojamos SQL Server duomenų bazių valdymo sistemos (DBVS), tokios kaip SQL Server 2005 Express, SQL Server 2005 Business, SQL Server 2008 ir taip toliau. Kadangi organizacija dažniausiai būna išskaidyta į padalinius keliuose miestuose, ji priversta naudotis sistema, kurios duomenų bazė būna patalpinta nutolusiame serveryje, aptarnaujančiame visus padalinius. Organizacijos duomenys, tai ko gero pats didžiausias jos turtas. [18]

Viena didžiausių problemų šiuo atveju yra organizacijos sistemos saugumas, kuris apima pačios taikomosios programos (tuo pačiu pačio Framework‘o, kurio pagrindu ji sukurta), o ypač duomenų bazės saugumą.

Įvairios Framework konstrukcijos kuriamos ir pritaikomos tam tikriems uždaviniams spręsti. Mūsų atveju Framework konstrukcija kuriama spręsti santykių su klientais valdymo (angl. Customer Relationship Management) ir produktų gyvavimo ciklo (angl. Product Lifeline Management) tipo uždaviniams mažo ir vidutinio dydžio įmonėms ir organizacijoms. Taigi mūsų taikymo sritis – mažų ir vidutinio dydžio organizacijų sistemos, kurių verslo logikos sistemos funkcionuoja taikomųjų programų (angl. application), naudojančių nutolusią

SQL duomenų bazę, pagrindu. Tokių organizacijų sistemų poreikiams pakanka SQL Server 2005(2008) Express duomenų bazės, todėl ją pasirenkame sistemos kūrimui.

2.3. Uždavinių klasės sprendimų apžvalga

Minėtiems uždaviniams spręsti, egzistuoja nemažai įvairių sprendimų. Pavyzdžiui jei kalbame apie CRM, tai šioje srityje pirmauja stambių korporacijų produktai, tokie kaip „Microsoft Dynamic“ arba „Oracle Siebel“, bei dar keletas kitų, kurių apžvalgos pateikiamos internete. [4]

Jei kalbame apie finansų valdymo ir produktų gyvavimo ciklo valdymo programas – čia neabejotinas etalonas būtų „SAP“ korporacijos programinė įranga.

Nagrinėjant saugumo įgyvendinimo prasme, tiek vieno, tiek kito tipo produktų gamintojai pateikia tik minimalią orientacinę informaciją šiuo klausimu. Bendruoju atveju dažniausiai minimos tokios galimybės:[12]

- saugumo modelis paremtas privilegijų principu;
- vaidmenimis (angl. role) paremtas saugumo parametrų administravimas;
- vartotojų skirstymas į kategorijas ir privilegijų skyrimas grupės lygiu;
- tiesioginė prieiga prie duomenų, turinčių SQL duomenų bazės rodinio (angl. View) pobūdį ir taip toliau.

Pagrindinė problema žiūrint iš mūsų pozicijos – minėtos sistemos nors ir gana saugios, bet jos yra kuriamos didelėms įmonėms ar organizacijoms, todėl lygiuotis į jas negalime. Mūsų atveju tai gali pasitarnauti tik kaip tam tikras geras pavyzdys projektuojant sistemą saugumo prasme. Analizuojant sprendimus mažoms organizacijoms tendencinga tai, kad labai dažnai taupoma saugumo sąskaita. Kaip žinia mažų įmonių sistemų saugumas būna labai abejotinas, todėl turime iškelti reikalavimus šio uždavinio sprendimui taip, kad sistema būtų kartu ne tik saugi, bet ir priimtina kitais aspektais, tokiais kaip greitaveika, naudojimosi patogumas, kaina ir t.t.

2.4. Programavimo technologijų analizė

Kaip žinia šiuo metu sukurta daugybė programavimo technologijų ir programavimo kalbų. Kuriant sistemą mūsų atveju reikalinga programavimo technologija, turinti/palaikanti aibę priemonių, skirtų saugumui įgyvendinti. Šiuo požiūriu dvi didžiausios technologijos –

konkurentės yra .NET ir Java. Tiriamuosiuose darbuose [2] ir [14] analizuojami šių platformų skirtumai.

Java ir .NET turi panašius saugumo tikslus ir mechanizmus. .NET sistemos dizainas panaudojo nemažai privalumų iš ankstesnės Java patirties. Naujojo „švaresnio“ .NET dizaino pavyzdžiai – tai MSIL(angl. Microsoft Intermediate Language) instrukcijų rinkinys, kodo prieigos teisių aiškumas, saugumo politikos konfigūracija. .NET naujoji architektūra turi galimybę apsaugoti sistemos kūrėją nuo pamatinio kompleksiškumo kūrimo procese.

Ten kur Java tik išvystė pradinės savo platformos su ribotomis saugumo galimybėmis technikas, .NET įtraukė daugiau saugumo galimybių tiesiai į originalųjį platformos dizainą. Su amžiumi ir naujomis galimybėmis, dauguma Java kodo vis dar palaiko senesnes platformos versijas (suderinamumą su senesnėmis versijomis) ir tuo pačiu „Null SecurityManager“ ir absoliutų pasitikėjimą klasėmis „bootclasspath“ sektoriuje. Taigi atskirose srityse .NET dėl savo paprastesnio ir „švaresnio“ dizaino turi savo saugumo privalumų lyginant su Java.

Nepaisant to, kad ir .NET dizainas nėra tobulas, jis teikia pasitikėjimo vertą pagrindą. Sistemų kūrėjai tokiu būdu gali kurti labiau saugias sistemas. [9]

Atsižvelgdami į programavimo technologijų palyginimą, kuriamai sistemai pasirenkame .NET programavimo technologiją.

2.5. SQL injekcijos

Naudodami pažangias technologijas, pvz. ASP.NET, .NET ir duomenų bazių variklius SQL Server 2005(2008), kūrėjai turi galimybę sukurti dinamiškus duomenų bazėmis paremtus sprendimus – tiek taikomąsias programas, tiek ir interneto puslapius su aukšto lygio patogumais. Bet .NET ir SQL galingumas gali labai lengvai būti panaudotas prieš mus pačius – programišiai naudoja šiurkščias atakas – SQL injekcijas.

Egzistuoja nemažai įvairių grėsmių tipų, vienos iš jų būna nukreiptos prieš pačią vykdomąja programą, kitos tiesiogiai prieš duomenų bazes. Kadangi mes labiau orientuojamės į duomenų bazių saugumą – detaliau išnagrinėsime pagrindinę ir labiausiai grėsmingą atakos tipą – SQL injekcijas.

Tiek interneto puslapiai, tiek ir verslo logikos taikomosios programos suteikia galimybę jų vartotojams atlikti veiksmus, susijusius su duomenų apdorojimu (įvedimu, modifikavimu, trynimu). Informacijos įvedimas/redagavimas dažniausiai atliekamas iš redagavimo laukų (angl. text box). Šie laukai yra viena iš galimybių programišiams įvesti kenkėjiškai suformuotą užklausą, kuri gali pakeisti užklausos natūrą taip, kad užklausa suteiks

galimybę įsilaužti, pakeisti arba išvis sugadinti duomenų bazę. Išnagrinėkime žinomas pagrindines tokių atakų technikas.

Dauguma .NET taikomųjų programų vartotojų autentifikavimui naudoja formas vartotojo vardui ir slaptažodžiui įvesti. Kai vartotojas suveda reikiamus duomenis ir paspaudžia „prisijungti“ mygtuką, vykdomas metodas, kuris bando autentifikuoti vartotoją vykdydamas užklausą, kuri skaičiuoja įrašų skaičių lentelėje „vartotojai“. Ieškoma įrašų, kuriuose būtų atitinkamos laukų „vartotojo vardas“ ir „slaptažodis“ reikšmės, įvestos minėtuose laukeliuose vartotojo sąsajos formoje.[11]

Daugumoje atvejų forma veikia taip, kaip numatyta. Vartotojas įveda vartotojo vardą ir slaptažodį, atitinkančius vieną iš lentelės „vartotojai“ įrašų. Atitinkančių įrašų skaičiui išgauti naudojama dinamiškai sugeneruota SQL užklausa. Vėliau vartotojas autentifikuojamas ir nukreipimas į prašomą langą. Vartotojai, kurie įveda neteisingus duomenis nėra autentifikuojami. Nepaisant to, šioje vietoje yra galimybė programišiui įvesti tariamai nekenksmingą tekstą į vartotojo vardo laukelį, tam kad išgauti priėjimą prie sistemos nežinant nei vieno tinkamo vartotojo vardo ir slaptažodžio.

Programišius įsilaužia į sistemą įterpdamas suformuotą SQL į užklausą. Taip atsitinka todėl, kad vykdoma užklausa suformuojama naudojant fiksuotų ir vartotojo įvedamų reikšmių apjungimą. Pavyzdžiui:

```
string strQry = "SELECT Count(*) FROM Users WHERE UserName='" +
   txtUser.Text + "' AND Password='" + txtPassword.Text + "'";
```

Jei vartotojas tarkim įveda teisingas savo prisijungimo reikšmes, užklausa tampa tokia:

```
SELECT Count(*) FROM Users WHERE UserName='Paulius' AND Password='plspsw'
```

Bet jeigu programišius vietoj to įvestų:

```
' Or 1=1 --
```

Užklausa tokiu atveju taptų tokia:

```
SELECT Count(*) FROM Users WHERE UserName='' Or 1=1 --' AND Password=''
```

Kadangi užklausos gale panaudoti du brūkšneliai, užklausos pabaiga išvis ignoruojama, nes SQL sintaksėje šie du vienas po kito einantys simboliai reiškia komentavimo pradžią, todėl toje eilutėje visi už jų esantys žodžiai yra ignoruojami. Rezultate gaunama labai parasta užklausa:

```
SELECT Count(*) FROM Users WHERE UserName='' Or 1=1
```

Ši išraiška 1=1 visada yra teisinga – kiekvienai visos lentelės eilutei, tokiu būdu visada grąžinamas daugiau nei viena lentelės eilutė – programišius gauna prieigą prie sistemos.

Ne visos SQL injekcijų atakos apima autentifikavimo formas. Taikomosios programos su dinamiškai sukonstruotu SQL ir nepatikimomis įvedimo galimybėmis taip pat yra nesaugios. Patyręs programišius, turintis SQL konfigūracijos ir kalbos žinių tai pat gali padaryti daug žalos. Tokios galimybės, kai sistemos vartotojui leidžiama pačiam atlikti dinaminę SQL paiešką pagal įvestą užklausą – tiesiog rojus programišiui. Pasinaudodamas tokiais redagavimo laukais jis gali ne tik išgauti reikiamą informaciją, bet ir pakeisti duomenų bazės duomenis, pažeisti įrašus ir net sukurti naujus duomenų bazės vartotojus.

Dauguma su SQL suderinamų duomenų bazių, įskaitant SQL Server, saugo metaduomenis tam tikroje sisteminių lentelių serijoje, turinčioje pavadinimus „sysobjects“, „syscolumns“, „sysindexes“ ir taip toliau. Tai reiškia, kad programišius gali panaudoti šias sistemines lenteles tam, kad nustatyti scheminę duomenų bazės informaciją, kad ją galėtų vėliau panaudoti kitai duomenų bazės kompromitacijai. Štai kad ir tokia užklausa, įvesta į redagavimo laukelį, gali būti panaudota išgauti duomenų bazės lentelių pavadinimus:

```
' UNION SELECT id, name, '', 0 FROM sysobjects WHERE xtype ='U' --
```

UNION sakinys ypatingas tuo, kad jis įgalina suregzti vienos užklausos rezultatą į kitą užklausą. Šiuo atveju programišius ištraukia vartotojų lentelės duomenis, pasinaudodamas visai kitos lentelės filtravimui skirtu redaguojamuoju lauku. Vienintelis triukas, kurį reikia atlikti – parašyti užklausą taip, kad būtų atitikimas tarp stulpelių laukų tipų. Tai reiškia, kad ir vienur ir kitur turi sutapti laukų tipai, pavyzdžiui „int“, „ntext()“ arba „nvarchar()“. Tokiu būdu galima apkeliauti labai daug duomenų bazės lentelių.

SQL injekcijų atakos taip pat gali būti naudojamos pakeisti duomenis arba sugadinti duomenų bazę. Pavyzdžiui galima kokių nors produktų ar kitų objektų lentelėje pakeisti kainas ir pasinaudojus tuo iškart įsigyti tą gaminį toje parduotuvėje internete. Štai vienas iš tokių užklausų pavyzdžių:

```
'; UPDATE Products SET UnitPrice = 0.01 WHERE ProductId = 1--
```

Tokia ataka suveikia dėl vienos labai paprastos priežasties – SQL serveris leidžia atlikti vienu metu keletą sakinių, atskirtų vieną nuo kito kabliataškiais arba tarpo simboliu. Šiuo atveju duomenų atvaizdavimo komponentas neparodys nieko, bet pati užklausa sėkmingai suveiks. Tokia pati technika gali būti taikoma ir kitokioms komandoms, tarkim „drop table“, arba tam, kad įvykdyti vieną iš SQL serveryje esančių procedūrų („stored procedure“), kuri tarkim sukuria naują vartotoją arba jį priskiria „sysadmin“ vaidmeniui.

Svarbu priminti tai, kad SQL injekcijų tipo atakos pavojingos ne vien SQL Server duomenų bazėms. Kitos duomenų bazės, įskaitant „Oracle“, „MySQL“, „DB2“, „Sybase“ ir kitos taip pat yra jautrios šiam atakos tipui. Galimybės SQL atakoms kyla iš to, kad SQL kalba turi daugybę galimybių, kurios šia kalbą padaro galinga ir lanksčia, tai yra:

- galimybė įterpti komentarus į SQL sakinį naudojant brūkšnelių porą (--);
- galimybė nurodyti iškart keletą sakinių vienoje eilutėje ir juos įvykdyti krūvoje;
- galimybė naudoti SQL užklausas tam, kad išgauti metaduomenis iš standartinio sisteminių lentelių paketo.

Bendruoju atveju, kuo galingesnis yra SQL dialektas duomenų bazėje, tuo tokia duomenų bazė turi didesnę galimybę būti atakuojama. Todėl nieko nėra keisto, kad SQL duomenų bazės yra gana populiarus atakuotojų SQL injekcijomis taikinys. Svarbu pabrėžti tai, kad SQL injekcijų atakos neapsiriboja vien tik ASP.NET taikomosiomis programomis. Klasikinės ASP, Java JSP ir PHP taikomosios programos taip pat turi tokias grėsmes. [10]

2.5.1. Apsisaugojimo nuo SQL injekcijų būdai ir priemonės

Apsisaugojimui nuo SQL injekcijų ir kitų susijusių grėsmių principai ir įgyvendinimo būdai pateikti 1 lentelėje.

1 lentelė. Apsisaugojimo nuo SQL injekcijų principai

Principas	Įgyvendinimas
Nepasitikėti vartotojo įvestimi	Panaudojant ratifikavimo (angl validation) kontrolę, reguliarias išraiškas ir kitas priemones, tikrinti visų redaguojamų laukų įvestį
Nenaudoti dinaminio SQL	Naudoti parametrizuotą SQL arba „stored“ procedūras
Nesijungti prie duomenų bazės naudojant administratoriaus teisės priėjimą	Naudoti apribotą prieigą prisijungimams prie duomenų bazės
Nesaugoti slaptų duomenų ir konfigūracijos atviru tekstu	Šifruoti arba skaičiuoti maišos reikšmę (angl. hash) slaptažodžiams ir kitai jautriai informacijai, pavyzdžiui jungties su duomenų baze parametrui ("connection string")
Klaidų pranešimai turi atskleisti tik minimalią informaciją	Neatskleisti per daug informacijos klaidų pranešimuose; naudoti bendrąsias numatytas klaidas ("customErrors") pateikiant klaidų pranešimus; nustatyti neigiamą „debug“ parametrą sistemoje

Vienas iš svarbiausių principų – vartotojo įvesties kontrolė. Tai reiškia, kad bet kokia vartotojo įvestis gali būti kenksminga. Negalima naudoti nepatikrintos vartotojo įvesties, tuo labiau naudoti tokį įvestį dinaminėje SQL užklausoje. Paprasčiausias būdas – naudoti ASP.NET patikros kontrolę ("RegularExpressionValidator")- gana gera priemonė, tačiau ji nėra vaistas nuo visų ligų. Kur kas geriau yra sukurti savus validatorius su numatytomis patikros galimybėmis.

Egzistuoja dvi validavimo traktuotės: neleisti naudoti tam tikrų simbolių arba kita traktuotė – leisti naudoti tik mažą reikalingų simbolių poaibį. Pirmasis variantas yra neblogas, bet čia egzistuoja pavojus pamiršti vieną ar kelis svarbius simbolius, o tuo atakuotojas sugebėtų pasinaudoti. Kur kas geresnis yra antras variantas - identifikuoti leidžiamus simbolius ir tik juos leisti naudoti. Šis sprendimas reikalauja daugiau darbo, bet tuo pačiu garantuoja kur kas patikimesnę kontrolę vartotojo įvestyje ("input"). Verta nepamiršti, kad įvestyje reikia uždėti galimų įvesti simbolių skaitliuką, nes kai kurioms atakoms įgyvendinti reikia gana didelio įvedamų simbolių kiekio.

Dinaminio SQL išvengimas - dauguma aptartų SQL injekcijų atakų remiasi dinaminiu SQL – t.y. daug ką lemia vartotojo įvedama informacija redagavimo laukeliuose. Naudojant parametrizuotą SQL, gerokai sumažėja SQL injekcijos galimybė ir pavojingumas.

Viena iš dažnai pasitaikančių saugumo grėsmių – „sa“ vartotojo naudojimas „Web.Config“ arba „App.config“ faile. Priminsime, kad „sa“ vartotojas – tai numatytasis SQL Server duomenų bazių vartotojas, turintis administratoriaus teises ir galintis tiek kurti naujas lenteles, tiek prisijungimus tiek ir visa kita. Tai labai bloga programavimo praktika, vietoj to turėtų būti naudojami ribotos prieigos vartotojai.[7]

Vienas iš svarbiausių failų, kurį taip pat reikia apsaugoti, tai „.config“ failas. Jame nurodoma informacija apie tai, prie kokios duomenų bazės jungiamasi ir kokiu vartotoju ir slaptažodžiu. Pagal nutylėjimą ši informacija šiame faile saugoma atvirojo teksto formatu. Tam, kad sistema būtų apsaugota, ši informacija mažų mažiausiai turi būti šifruoto formato. Vienas iš saugesnių būdų – naudoti maišos funkcijos skaičiavimą su „druskos“ (angl. salt) priedėliu.

Informatyviai vykdomo klaidų apdorojimas – tai dar vienas būdas, kuriuo gali pasinaudoti programišiai. Iš vienos pusės svarbu įtraukti klaidų apdorojimo mechanizmus į projektą, iš kitos pusės apdorotos ir neapdorotos klaidos turi pateikti minimalų informacijos kiekį, be jokių tikslių klasių ar metodų pavadinimų ir kitos aktualios informacijos.

Papildomai reikia suderinti parametrų „debug“ ir „customErrors mode“ panaudojimą. Pagal numatytąjį režimą, šie parametrai yra sukonfigūruoti priešingai, nei reiktų saugumo prasme. [2]

2.6. Duomenų bazių valdymo sistemų palyginimas

Kaip jau minėjome, mes orientuojamės į SQL Server duomenų bazių valdymo sistemas (DBVS). Šiuo metu prieinamos „Express“ ir „Standard“ SQL Server DBVS sistemos. Taigi palyginkime jų galimybes.

Kaip žinia, mūsų uždavinių klasė apima mažų ir vidutinio dydžio organizacijų CRM ir PLM sistemas. Tokių sistemų duomenims saugoti dažniausiai pasirenkama „Express“, o ne „Standart“ versijos SQL Server DBVS, kadangi ji yra nemokama. Panagrinėkime, ką gi prarandame pasirinkdami ne įprastinę, o nemokamą versiją. Mums aktualiausias saugumo faktorius. 2 lentelėje pateiktas „Standart“ ir „Express“ versijų palyginimas. [17]

2 lentelė. DBVS palyginimas saugumo prasme

Galimybė	„Express“	„Standard“	Komentarai
Pažangus auditas, autentifikacija ir autorizacija	✔	✔	
Duomenų šifravimas ir raktų valdymas	✔	✔	Integruotas duomenų šifravimas didesniam duomenų saugumui.
Integracija su „MS Baseline Security Analyzer“	✔	✔	Skenuoja sistemą ir tikrina įprastus pažeidžiamumus.
Integracija su MS atnaujinimais	✔	✔	

Kaip matome, saugumo prasme abi versijos turi tas pačias galimybes. Nepaisant to, vertėtų atkreipti dėmesį ir į kitus aspektus. Vienas iš svarbiausių – greitaveika ir išplečiamumas (angl. scalability)(3 lentelė). Tam, kad būtų dar vaizdesnis palyginimas, greta palyginame ir „Enterprise“ versiją, kuri įprastai naudojama didelių organizacijų sistemose. [8]

3 lentelė. DBVS palyginimas greitaveikos ir išplečiamumo prasme

Galimybė	**„Express“**	**„Standard“**	**„Enterprise“**
Palaikomų CPU kiekis	1	4	Maksimalus OS palaikomas kiekis
Operatyvioji atmintis (RAM)	1 GB	OS maksimumas	OS maksimumas
64-bit‘ų palaikymas	Windows on Windows (WOW)	✔	✔
Maksimali talpa	4 GB	Neribota	Neribota
Skirstymas į dalis (angl. Partitioning)			✔
Paralelinės indeksavimo operacijos			✔
Indeksuoti vazdiniai (angl. indexed views)			✔

Iš pirmo žvilgsnio gali pasirodyti, kad saugumo ir greitaveikos bei išplečiamumo sąveika(sąryšis) nereikšmingas. Tačiau iš tikrųjų šie faktoriai glaudžiai susiję. Duomenų šifravimas, iššifravimas, raktų valdymas ir kitos operacijos, garantuojančios didesnį saugumą bendrąja prasme, reikalauja kur kas daugiau resursų nei įprastos nesaugios operacijos. Galutiniame rezultate, kalbant apie mūsų tematika aktualius DBVS skirtumus, gauname, kad:

- Abi DBVS turi vienodas galimybes saugumo sityje;
- „Express“ DBVS turi žymiai mažesnes galimybes greitaveikos, talpos ir išplečiamumo prasme.

Kadangi projektuojamą sistemą kuriame „SQL Server 2005 Express“ DBVS pagrindu, susiduriame su greitaveikos ir saugumo priešprieša. Taigi mūsų sistemoje kuriami saugumo mechanizmai turėtų būti naudojami tik ten kur jie neabejotinai būtini. Kituose sistemos taškuose, turėtume numatyti dalinius arba minimalius ir pakankamus saugumo mechanizmus. Tokiu būdu išspręstume minėtą dilemą ir patenkintume numatytus reikalavimus.

2.7. Šifravimo duomenų bazėse ir kriptografinių sistemų apžvalga

Kasdien auga programų kompleksiškumas, sudėtingumas, didėja apdorojamų duomenų skaičius. Tuo pačiu vystosi ir programišiai – jie tampa vis gudresni. Dėl šių priežasčių šifravimas tampa viena iš aktualiausių dalykų duomenų bazių valdymo sistemų (DBVS) saugume.

Kaip jau minėjome programų sudėtingumas ir apdorojamų duomenų kiekis pastoviai auga, tuo pačiu auga ir tikimybė, kad programose bus palikta saugumo spragų. Tokiu būdu šifravimas tampa paskutiniu kertiniu saugumo DBVS elementu. Iš vienos pusės saugumo reikalus tvarko DB administratorius – jis įdiegia naujausius saugumo atnaujinimus, apsaugo duomenų bazę įvesdamas „stiprius“ slaptažodžius ir t.t. Bet to tikrai nepakanka, nes atakuotojai išnaudoja pačias įvairiausias galimybes ir įrankius, gali rasti tam tikras konfigūravimo spragas, galų gale gali aptikti atsitiktinai paliktą atvirą jungtį su duomenų baze.

Yra daug faktorių, kurie lemia ar organizacija turėtų šifruoti duomenų bazės duomenis. Vienas iš jų gali būti prisiderinti valstybinių ar tarptautinių reglamentų. Egzistuoja nesuskaitoma daugybė reglamentų, apibrėžiančių asmenų identifikacijos informacijos saugojimą duomenų bazėse, pavyzdžiui Europos Sąjungos duomenų apsaugos direktyva, Sarbanes-Oxley aktas (2002 m.) ir kiti.

2.7.1. Kriptografijos naudojimas SQL kalboje

Prieš leidžiant konkrečiam vartotojui užšifruoti ar dešifruoti duomenis, generuoti arba naudoti raktą, arba/ir generuoti arba naudoti sertifikatą, DBVS turėtų autentifikuoti vartotoją ir tuomet užtikrinti, kad tas vartotojas turi teisę atlikti minėtus veiksmus. Be to turi būti atliktas patikrinimas, kad konkretus vienetas turi teisę peržiūrėti šifruotus duomenis (šifruotą tekstą).Yra dvi pagrindinės duomenų, kurie gali būti šifruojami kategorijos, ir organizacija turi apsispręsti, ar ji nori apsaugoti abu ar tik vieną iš jų. Šie tipai, tai [15]:

- „Data-in-motion“ (Judantys duomenys)
- „Data-in-rest“ (Nejudantys duomenys)

„Data-in-motion“ apibrėžia duomenis, kurie siunčiami iš ir į duomenų bazę (pavyzdžiui į/iš klientinės taikomosios programos naudojantis internetu arba lokaliu intranetu). DBVS tam naudoja SSL, TLS ir/arba IPSec protokolus, tam kad minėti duomenys būtų apsaugoti siuntimo metu nuo įvairių atakų.

„Data-in-rest“ apibrėžia duomenis, kurie egzistuoja pačioje duomenų bazėje. Kadangi šio tipo duomenys čia esti ilgą laiko periodą, būtent prieš juos ir nukreipiamos kur kas didesnės piktavalių pastangos.

Jei organizacija nusprendžia apsaugoti „Data-in-rest“ tipo duomenis, reikia nuspręsti, kokiu lygiu duomenys bus šifruojami. Sąrašas galimų būdų su jų privalumais ir trūkumais, pateiktas žemiau:

- Duomenų bazės lygis – čia užšifruojamas vienas arba keletas failų. Dažniausiai jie būna gana dideli ir sudaro visą duomenų bazės specifiką, pavyzdžiui SQL Server duomenų bazėse naudojami du failai – vienas su prefiksu „mdf“, kitas su prefiksu „log“. Duomenų bazės šifravimo lygio privalumas tas, kad čia šifruojami absoliučiai visi duomenys. Tuo pačiu tai ir trūkumas – kiekvieną kartą kai duomenys nuskaitomi ar rašomi, jie yra apdorojami šifravimo procesu, dėl ko labai apkraunamas duomenų bazės serverio procesorius.
- Eilutės arba stulpelio lygis – tai tik pasirinktų eilučių arba stulpelių šifravimas lentelės viduje. Toks šifravimas gerokai sumažina apdorojimo sąnaudas. Svarbu paminėti tai, kad taip šifruotiems stulpeliams negalėsime taikyti „WHERE“ sakinio ir priskirti pirminio rakto atributo.

2.8. Sistemos vartotojų ir jų teisių analizė

Kadangi kalbama apie verslo logikos sistemas, verta atlikti ir vartotojų analizę. Verslo logikos sistemos ypatingos tuo, kad jomis naudojasi daug asmenų. Kaip pavyzdį galime paimti tarkime įmonę, užsiimančią vaistų importu ir jų platinimu šalies teritorijoje. Tokiai įmonei reikalinga sistema, kurios pagalba būtų valdomi visi produktai, pradedant užsakymo faze ir baigiant pardavimo dokumentų fiksavimu. Iš to seka, kad šia sistema naudosi daugybė įvairias pareigas atliekančių žmonių: vadybininkai, buhalteriai, sandėlio personalas, įvairių skyrių vadovai, administratoriai ir kiti. Svarbu, kad vartotojai būtų suskirstyti į kategorijas. Kategorijos reikalingos tam, kad vartotojai galėtų būti priskirti kuriai nors iš jų ir kad naudojantis vartotojų kategorijomis būtų galima priskirti atitinkamas teises. Teisės šiuo atveju reikalingos, kad būtų galima kontroliuoti prieigą prie tam tikrų langų, pavyzdžiui sandėlio personalui, suvedančiam gautas prekes, tikrai nereikalinga ir nebūtina prieiga prie nustatymų lango arba vartotojų administravimo ir teisių jiems suteikimo lango.

Kalbant apie verslo logikos sistemas, teisių valdymo galimybės dažnai būna pamirštamos. Tokiu būdu sistemos tampa mažiau saugios, labiau pažeidžiamos tiek dėl atsitiktinių, tiek ir tyčinių veiksmų. Modernios ir tinkamai suprojektuotos Framework programinės konstrukcijos turėtų numatyti vartotojų grupių ir atskirų vartotojų egzistavimą, galimybę tvarkyti tiek vartotojų, tiek ir vartotojų grupių teises. Sistema laikoma pakankamai lanksčia, jei teises vartotojams galima priskirti langų lygyje, t.y. tam tikras vartotojas turi arba neturi teisės prieiti prie tam tikro lango. Jei vartotojas turi tokią teisę, ši teisė papildomai gali apibrėžti, kokius veiksmus vartotojas gali atlikti tame lange (sukurti naują įrašą, redaguoti įrašą, trinti įrašą, generuoti įrašų ataskaitą ir panašiai). [1]

2.9. Technologijų analizė

Sprendžiant sistemos saugumo klausimus dėl SQL injekcijų pavojaus, dažnai susiduriama su apsaugos priemonių (technologijų) pasirinkimo problema. Jei vienos technologijos įgyvendinamos lengviau šiek tiek pakeičiant tam tikrus sistemos elementus, tai kitos gali reikalauti perdaryti sistemą iš pagrindų. Projektuojant sistemą, saugumo veiksnys dažniausiai nebūna prioritetinis, tačiau vėliau pasekmės gali priversti į tai pažvelgti rimčiau, o galbūt net perdaryti tam tikrą sistemos dalį ar visą sistemą. Bendruoju atveju pasirinkimas platus, tad susiauriname apžvalgą orientuodamiesi į .NET technologijas.

Apžvelkime žinomas technologijas, skirtas SQL injekcijų pavojų, problemų sprendimui.

- **Kompleksinė technologija (SPROC)**

Ši technologija grindžiama tam tikromis taisyklėmis, kurių laikymasis labai sumažina SQL injekcijų riziką. Pagrindą sudaro šios taisyklės:

a) Nenaudoti dinaminio SQL, visada apibrėžti parametrus ir jų tipą (*string, integer, date* ir tt), naudoti SP (angl. stored procedures);

b) Visada atlikti saugumo patikrą prieš pateikiant sistemą naudojimui;

c) Nesaugoti slaptažodžių atviro teksto formatu. Rekomenduojama naudoti vienkryptes šifravimo technologijas.

d) Kūrimo procese naudoti automatizuotus testavimo įrankius ir rinkinius, tokius kaip nUnit.

e) Minimizuoti SQL prieigos teises vartotojams, suteikiant tik būtinas veikimui privilegijas. Nenaudoti „sa“ (trumpinys iš anglų kalbos „system acount“) vartotojo, turinčio aukščiausius įgaliojimus.

❖ **ORM technologija**

ORM (angl. Object Relational Mapping) technologija valdo .NET objektų abipusį sąryšį su reliacinėmis DB, automatiškai generuoja SQL duomenų užkrovimo ir išsaugojimo veiksmus, naudoja žymėjimo („mapping“) kalbą SQL sąryšiams su programos atributais nurodyti. Kitaip tariant sistemoje sukuriamas tam tikras objektas, kuriame apibrėžiamos jo charakteristikos (pvz.: Objektas yra vartotojas, o jo charakteristikos: Vardas, Pavardė, Prisijungimo vardas, Slaptažodis ir tt). Taip pat sukuriamas „mapping“ failas, kuriame nurodomos objekto ir jį atitinkančio SQL objekto sąryšis. Taip pat reikalingas dar vienas bendras arba kiekvienam objektui atskiras konfigūracinis failas, apbrėžiantis prisijungimo prie DB parametrus. Toliau nebereikia kurti jokių įterpimo, redagavimo, duomenų išrinkimo ar kitų procedūrų, nes ORM automatiškai generuoja SQL procedūras ir jų parametrus vykdymo metu ir tokiu būdu šis procesas dėl aukšto objektiškumo lygio (.NET (CLR) objektai ir į objektus orientuota idioma) yra santykinai saugus ir SQL injekcijų prasme, nes veiksmai (nurodymai) kaip operuojama su duomenimis yra rašomi abstraktesniais sakiniais, labiau paprastam žmogui suprantama kalba, kurią vėliau interpretuoja ORM bibliotekos ir paverčia žemesnio lygio SQL kalba, kuri ir perduodama į DB vykdymui. Minėtos ORM bibliotekos pasižymi gana stipriu saugos ir tvarkos lygiu ir komplikuoja SQL injekcijų galimybę.[8, 9]
Naudojimo proceso pavyzdys (etapai):

Iš principo ORM panaudojimas atliekamas tokiais žingsniais

- Jei dar neturime DB lentelės(-ių), sukuriame jas. Šios lentelės vėliau bus susiejamos su .NET klasėmis, pasinaudojant „mapping“ failais.
- Sukuriame .NET klases, kurios savyje apibrėžia savo duomenis kaip .NET tam tikro tipo (integer, string, date ar pan.) objektus, turinčius „get“ ir „set“ metodus (angl. „read-write property“).
- Sukuriame XML formato susiejimo („mapping“) failus tam, kad ORM įgyvendinanti technologija (pvz. NHibernate) žinotų, kurie ir kokios DB lentelės laukai bus susiejami su kuriais .NET klasės objektais. Dažniausiai kiekvienai klasei sukuriamas atskiras „mapping“ failas.
- Sukuriamas XML formato konfigūracinis failas, kuriame apibrėžiami prisijungimo prie DB duomenys, panašiai kaip standartiniame .NET programos konfigūraciniame faile.
- Naudojamės ORM technologija, kuri mums palengvina komunikavimo su DB užduotis – duomenų užkrovimą iš DB, duomenų išsaugojimą, užklausų atlikimą ir t.t. ORM technologija leidžia interpretuoti objektais aukšto lygio kalba, kurios dėka nereikia rašyti sudėtingų kodo fragmentų ar SQL procedūrų – visa tai generuoja ORM technologijos bibliotekos. Pavyzdžiui, objekto „user“ išsaugojimas atliekamas tokia paprasta komanda: „session.Save(user)“. Tuo tarpu įprastiniu atveju, kai nenaudojamas ORM, tam pačiam rezultatui pasiekti reikia iš eilės atlikti tokius veiksmus: atidaryti SQL DB jungtį; sukurti SqlCommand kintamąjį; parašyti SQL procedūrą ir jai priskirti parametrus; SqlCommand kintamąjam priskirti minėtą procedūrą ir SQL DB jungtį; įvykdyti komandą; uždaryti jungtį.

❖ **Regex komponentų technologija**

Regular Expression klasė, sutrumpintai apibrėžiama kaip Regex sudaro galimybė tikrinti įvedamų duomenų atitikimą apibrėžtiems šablonams. Kaip rodo praktika, ypač dažnai naudojama esamų sistemų patobulinimui, kai ratifikuojami vartotojo įvedami duomenys, t.y. tikrinam ar jie atitinka/neatitinka tam tikras taisykles. Nors šios klasės pagrindinis tikslas nėra saugumas, tačiau ji naudojama ratifikavimui ir dėl saugumo klausimų. Paprastas panaudojimo pavyzdys:

```
public static bool IsInjection(string inputText)
```

```
        {
            if (inputText == "") // Įvestis tuščia
            {
                return false;
            }

            string regexForUnion = @"^.*((%27)|'|(\-\-
))\s*(u|%75|%55)(n|%6E|%4E)(i|%69|%49)(o|%6F|%4F)(n|%6E|%4E).*$";

            // Regex taisyklė:
            // ^ - pradžia
            // .* - bet koks (tame tarpe ir nulis) bet kokių simbolių
kiekis
            // (%27)|' - kabutės simbolis arba jo šešioliktainis atitikmuo
            // (\-\-)) - "--" simboliai, reiškiantys komentaro pradžią SQL
kalboje
            // (u|%75|%55) - u raidė arba jos šešioliktainiai atitikmenys
(didžioji ir mažoji raidė)
            // (n|%6E|%4E) - n ...
            // (i|%69|%49) - i ...
            // (o|%6F|%4F) - o ...
            // (n|%6E|%4E) - n ...
            // .* - bet koks (tame tarpe ir nulis) bet kokių simbolių
kiekis
            // $ - pabaiga

            Regex reUn = new Regex(regexForUnion); // Regex objekto
sukūrimas
            if (reUn.IsMatch(inputText)) // Ar įvestis atitinka taisyklę
                return true; // Atitinka - galimas SQL inj. pavojus

            return false; // Neatitinka - grąžinti neigiamą požymį

        }
```

Pavyzdyje pateikta taisyklė, ieškanti įvestyje „[Bet kokie simboliai]--union [bet kokie simboliai]“ taisyklę atitinkančio teksto. Union – SQL kalboje naudojamas vykdomasis žodis, įgyvendinantis duomenų išrinkimą iš skirtingų objektų ir atliekantis bendro rezultato apjungimą.

Pagal šį analogą galime sukurti aibę taisyklių ir pasinaudojant jomis atlikti įvesties ratifikaciją.

Pateikiame lentelę, kurioje palyginame apžvelgtas technologijas su įprastinėmis technologijomis.

4 lentelė. Technologijų tarpusavio palyginimas

Technologija	Kompleksinė (SPROC)	ORM (Object Relational Mapping)	Regular Expresion (Regex)	Įprastinės priemonės (palyginimui)
Veikimo principas	„Tvarkingas“ programavimas, laikantis tam tikrų taisyklių: nenaudoti tiesioginių SQL išraiškų kode bei dinaminių užklausų; naudoti tik SP(stored procedure); būtinai parametrizuoti visas užklausas	Valdo .NET objektų abipusį sąryšį su reliacinėmis DB, automatiškai generuoja SQL duomenų užkrovimo ir išsaugojimo veiksmus, naudoja žymėjimo (mapping) kalbą SQL sąryšiams su programos atributais nurodyti.	Įvesties atitikimo SQL injekcijos šablonui tikrinimas	Įprasta programos ir SQL serverio komunikacija įprastinėmis priemonėmis
Privalumai	Paprastumas, suprantamumas.	Sudengimas (suliejimas); nereikia įgyvendinti interfeiso su DB įprastiniame lygmenyje. Galima projektuoti sistemas pagal	Didelė komponento greitaveika, platus pritaikymo spektras, galimybė sukurti didelius šablonų rinkinius.	Didžiausias paprastumas, suprantamumas ir pavyzdžių gausa

		grynai objektiškai orientuotą idiomą.		
Greitaveika	Didelė – nurodomas tik SP pavadinimas ir perduodami parametrai	Mažesnė – kiekvieną kartą iš naujo automatiškai formuojama SQL užklausa ir jos parametrai	Mažesnė – prieš perduodant vykdymui į serverį kiekvieną kartą vyksta tikrinimas dėl injekcijų, kuo daugiau taisyklių – tuo tikrinimas ilgesnis.	Didelė
Pavojai mūsų sistemos atžvilgiu	Sistemos kūrėjų darbo tvarkos ir principų nesilaikymas, neišprusimas	Nepavykus realizuoti tam tikros vietos visada galima eiti lengvesniu keliu rašant įprastą, ne ORM modelių pagrįstą kodą.	Komponento pagrindinė paskirtis nėra atakų aptikimas. Taisyklių formavimas – nelengvas uždavinys net ir patyrusiam programuotojui.	Nesilaikant taisyklių ir technologinių principų, kuriamai sistemai iškyla saugumo grėsmė.
Bendri technologijos pavojai	Taisyklių nesilaikymas, saugumo apžvalgos ir testavimo užduočių neatlikimas prieš atiduodant produktą vartojimui.	Galimybė apeiti ORM logiką kūrimo procese; visada reikia apgalvoti, ar tikrai užklausa sudaroma saugiau būdu, nepaliekant spragų.	Blogai suformuluotos Regex taisyklės; Regex taisyklių komplikuota kūrimo technologija	Mažas saugumo lygis, informacijos gausa internete, technologijos populiarumas.

Trūkumai	SQL objektų(funkcijų, procedūrų) kūrimas ir testavimas reikalauja daugiau laiko	Sudėtinga ir daug laiko užimanti klaidų paieška, komplikuotas derinimo (Debug) procesas. Sąryšiui su DB naudojamas XML konfigūracinis failas, kuris nėra šifruotas.	Problematiška parašyti universalią tam tikros atakos aptikimo taisyklę dėl komponento specifikos.	Mažas saugumo lygis

Iš apibendrinančios lentelės matome, kad kiekviena technologija (apsisaugojimo technika) turi savo privalumų ir tuo pačiu trūkumų. Įvertinant kuri technologija labiausiai tinka mūsų atvejui, reikia atsižvelgti į konkrečią sistemą ir jos specifiką, architektūrą, panaudojimą. Lentelėje palyginimui įdėta standartinė technologija (kai naudojamasi standartinėmis kūrimo priemonėmis), nesilaikant jokių technologinių principų ar sistemos kūrimo modelių. Be abejo, kurti sistemą tokiu būdu yra labai neprofesionalu ir nesaugu. Mūsų sistemai būdinga tai, kad ji bus naudojama daugelio sistemų kūrimui, prie jos dirbs daug asmenų, iš kurių tam tikra dalis gali turėti nedidelę darbo patirtimį saugumo srityje. Dėl šios priežasties labai svarbu, kad saugumo elementų komponavimas sistemoje nereikalautų drastiškų sistemos modulių keitimo arba sudėtingų programinių priemonių, kurios galėtų būti komplikuotas uždavinys mažiau patyrusiems asmenims ir kad šių veiksnių pasekoje neatsirastų rimtų saugumo spragų. Atsižvelgiant į šiuos kriterijus, nusprendžiame, kad:

1. Mūsų sistemos atveju labiausiai priimtina SPROC technologija. Svarbiausias pasirinkimo kriterijus yra parankus veikimo principas, susidedantis iš keleto paprastų priemonių, kurių kompleksas sudaro santykinai aukštą saugumo lygį. Antrasis svarbus kriterijus – didelė greitaveika, kuri palyginus su alternatyviomis technologijomis duoda geriausią rezultatą. Trečias svarbus kriterijus – suprantamumas, t.y. nesudėtingas panaudojimas.

2. Regex technologija nėra tinkama dėl to, kad jos pagrindinė paskirtis nėra saugumo įgyvendinimas ir dėl to, kad taisyklių rašymas yra labai komplikuotas ir sudėtingas; labai sunku parašyti tokią taisyklę, kurios nebūtų galima apeiti; norint apsisaugoti nuo įvairių SQL injekcijų, vienos universalios taisyklės jokiu būdu sukurti nepavyks – reikės daug taisyklių, ko pasekoje bus ir daug tikrinimo žingsnių ir gali stipriai sumažėti greitaveika.

3. ORM technologijos atsisakyta dėl komplikuoto derinimo (angl. debug) režimo ir dėl to, kad šios technologijos naudojimas negarantuoja didelio saugumo laipsnio, reikalauja sudaryti daug konfigūracinių „mapping“ failų, kurie nėra koduojami.

2.10. Analizės išvados

Organizacijos sistemos saugumas turi būti įgyvendinamas užtikrinant taikomosios programos ir naudojamos duomenų bazės saugumą. Pagrindinės priežastys, dėl kurių saugumo lygis šiuolaikinėse verslo sistemose ir jas įgyvendinančiose programinėse Framework konstrukcijose yra žemas:

- atsaini sistemos vartotojų skirstymo į grupes ir grupių teisių politika arba jos nebuvimas;
- šifravimo mechanizmų trūkumas arba nebuvimas saugant asmenų identifikacinę informaciją, slaptažodžius ar kitą slaptą informaciją;
- sistemų neatsparumas SQL injekcijoms, atsaini SQL vartotojų (angl. SQL login) politika, svarbių duomenų saugojimas duomenų bazėje atviro teksto formatu;
- ribotos galimybės greitaveikos, talpos, išplečiamumo ir saugumo prasme SQL Server Express duomenų bazių valdymo sistemose, lyginant su komerciniais gaminiais, skirtais didelėms įmonėms.

Analizės metu pasirinktos sistemos kūrimo technologijos, įvertinant pasirinktą uždavinių klasę ir siekiamus tikslus. Taip pat peržvelgti galimi būdai, kurių dėka galima žymiai padidinti sistemos saugumo lygį. Sukaupus į vieną visumą šiuos apsaugos principus, papildant esamų priemonių funkcionalumą, galimybes ir pritaikius juos Framework programinėms konstrukcijoms, galima žymiai padidinti kuriamų sistemų saugumą.

3. SAUGIOS FRAMEWORK SISTEMOS PROJEKTAS

3.1. Reikalavimai Framework sistemai, įgyvendinančiai CRM ir PLM uždavinius

3.1.1. Pagrindiniai sistemos reikalavimai

Remiantis analizės metu nustatytų esamų Framework konstrukcijų trūkumais ir literatūroje [13, 14] nurodomais pagrindiniais .NET saugos kūrimo principais, projektuojamai sistemai galime iškelti tokius pagrindinius reikalavimus:

- Sistema turi įgyvendinti apsaugą nuo SQL injekcijų.
- Sistema turi įgyvendinti vartotojų ir jų grupių politiką.
- Sistema turi autentifikuoti ir autorizuoti vartotoją.
- Sistemos vartotojas turi būti unikalus tarp kitų sistemos vartotojų, t.y. turėti unikalų identifikatorių (ID) sistemoje.
- Darbo su sistema procesas turi turėti tokią seką: prisijungimas, darbo atlikimas, atsijungimas.
- Sistema turi įgyvendinti slaptažodžių politiką, konfidencialių duomenų šifravimą.

3.1.2. Detalizuoti sistemos reikalavimai

- Prieš pradėdamas naudotis sistema, vartotojas privalo prisijungti.
- Baigęs darbą, vartotojas privalo atsijungti.
- Sistema privalo leisti naudoti tik saugius slaptažodžius.
- Sistema privalo automatiškai atjungti neaktyvų tam tikrą laiką vartotoją.
- Duomenų paieškos ir filtravimo bei prisijungimo languose, sistema turi riboti vedamų simbolių kiekį.
- Sukompiliuotoje sistemoje turi būti išjungtas „Debug“ parametras.
- Prieigą prie vartotojų ir klientų slaptos informacijos gali turėti tik administratoriaus tipo vartotojai.
- Sistema privalo apdoroti klaidas taip, kad jose talpinama informacija neatskleistų programos sandaros ypatybių, SQL duomenų bazės lentelių, lentelių laukų ir jų tipų pavadinimų bei kitos svarbios informacijos.

- Sistema turi turėti vartotojų klasifikavimo į grupes logiką.
- Sistema turi turėti vartotojų ir grupių teisių suteikimo logiką.
- Sistemos vartotojas turi grupės teises, bet jos gali būti koreguojamos papildomai, panaikinant tam tikras teises.
- Sistemoje grupės teisės yra aukštesnio prioriteto ir vartotojas, priklausantis tam tikrai grupei, negali turėti teisių, aukštesnių nei grupės teisės.
- Sistemos teisės nustatomos langų lygiu.
- Sistemos teisės gali būti nustatomos lango komponento lygiu, jei lango komponentas yra įgyvendintas kaip atskiras vienetas ir turi nuosavą teisę sistemoje.
- Turi būti numatyta pasirinkimo galimybė šifruoti/nešifruoti tam tikrų svarbių lentelių informaciją, tokiu būdu įgyvendinant funkciją, kad vieni tos pačios lentelės duomenys yra šifruoti, o kiti ne.
- Turi būti įgyvendintas raktų generavimo mechanizmas.
- Sistema turi bendrauti su duomenų baze naudodama saugų protokolą, pavyzdžiui SSL.
- Slaptažodžių saugojimui duomenų bazėje turi būti naudojamas šifravimas su maišos funkcija (angl. Hash).
- Maišos funkcija turi būti realizuojama naudojant efektyvius algoritmus, tokius kaip SHA1 arba MD5.

3.1.3. Reikalavimai duomenų bazei

- Sistemoje turi būti šifruojamas prisijungimo prie DB parametrų ir programos nustatymų failas („*.config“ failas).
- Sistemoje naudojant maišos funkciją turi būti šifruojami vartotojų slaptažodžiai.
- Duomenų paieškos ir filtravimo bei prisijungimo languose, sistema turi naudoti patikros priemones, draudžiančius šias SQL kalbos galimybes:
    - Komentarų galimybę;
    - Keleto sakinių galimybę;
    - Metaduomenų išgavimo iš sisteminių lentelių galimybę.
- Sistema turi naudoti tik parametrizuotas SQL užklausas arba „stored“ procedūras (SP).

- Sistema turi turėti apribotą prieigą prisijungimams prie duomenų bazės.
- Sistema turi drausti naudoti „sa“ tipo vartotojui prisijungti prie DB.

3.1.4. Reikalavimai greitaveikai

- Sistemos greitaveikai užtikrinti, turi būti numatyta pasirinkimo galimybė naudoti(nenaudoti) įvairias apsaugos priemones tam tikruose languose.
- Nereikšmingų duomenų klasifikatorių languose turėtų būti naudojami tik būtini saugos elementai.

3.1.5. Reikalavimai sistemai palaikyti

- Kompiuteryje turi būti įdiegta .NET aplinka.
- Kompiuteryje, kuriame bus įdiegta sistemos duomenų bazė, turi būti SQL Server 2005 arba aukštesnė duomenų bazių valdymo sistema (toliau DBVS). Taip pat tinka ir nemokama SQL Server 2005 Express arba aukštesnė Express šeimos DBVS.

3.1.6. Reikalavimai vartotojo sąsajai

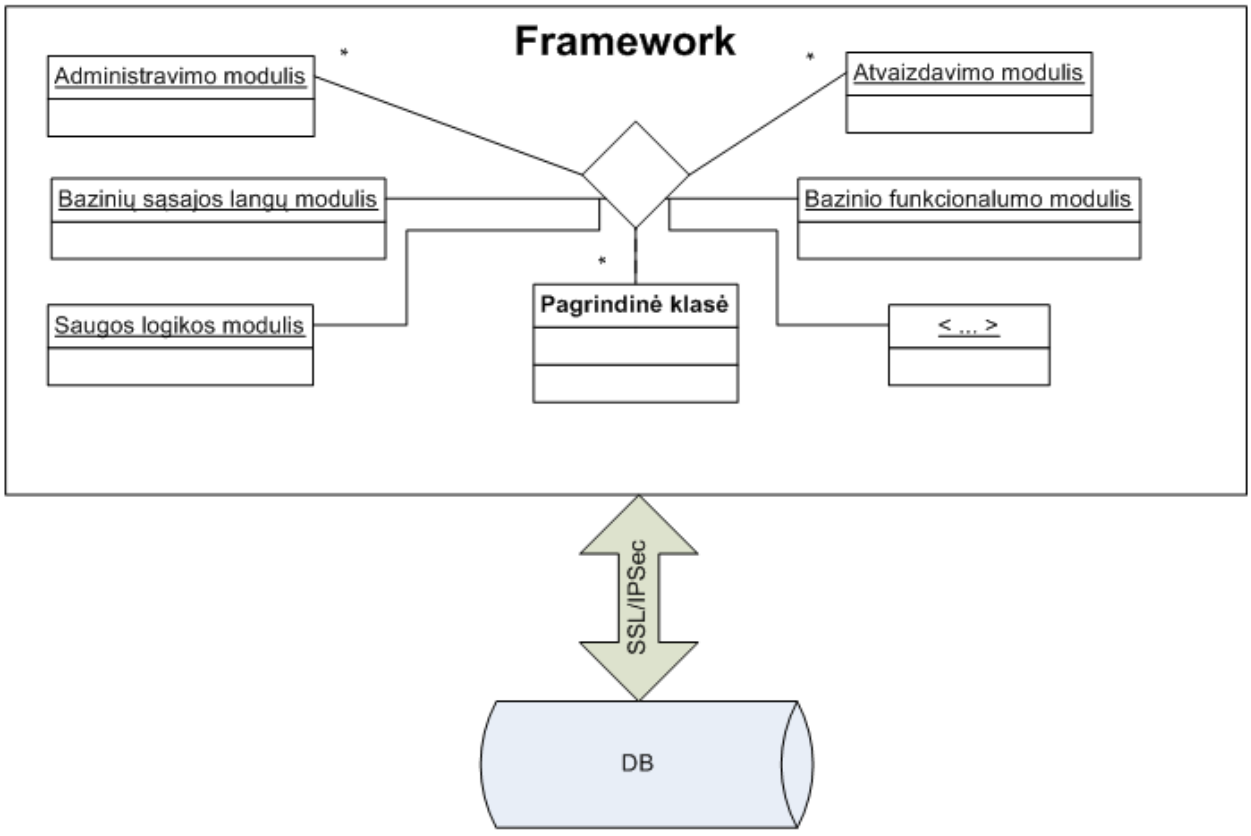
- Vartotojo sąsajos elementų išdėstymas turėtų būti nuoseklus ir išdėstytas tokiu principu, kad dažniausiai naudojami komponentai ar atskiro meniu punkto pasirinkimai būtų aukščiau, o mažiau naudojami – žemiau.
- Meniu punktų elementai turėtų būti suskirstyti į logiškas kategorijas, kad vartotojui būtų lengviau juos rasti, skirtingų kategorijų objektai taip pat turėtų būti atskirti skiriamaisiais elementais (angl. Split line).
- Meniu gylis neturėtų viršyti trijų pakopų.
- Saugumo objektų pasiekimui turėtų būti sukurtas atskiras meniu punktas, kuris būtų aktyvus tik vartotojui, turinčiam administratoriaus teises. Visiems kitiems vartotojams jis turėtų būti neaktyvus arba apskritai nematomas.
- Kiekvienas langas, iškviečiamas per meniu punktą turi turėti jam priskirtą teisę, kuri apspręstu jo prieigos lygį tam tikrai vartotojų grupei ir priklausomai nuo to jis turėtų būti aktyvus(matomas) arba neaktyvus (nematomas).

3.2. Projektuojama sistema

Tikslumo dėlei pirmiausiai reikėtų apibrėžti, ką reiškia žodis „sistema“ mūsų nagrinėjamu atveju. Tai būtų suprantama, kaip rėminė programinė konstrukcija, dažnai vadinama terminu Framework. Ši konstrukcija taip pat dažnai dar vadinama terminu variklis, kuris ypač populiarus žaidimų kūrėjų rinkoje. Žinomas ne vienas pavyzdys, kai geras variklis (grafikos variklis) būna panaudojamas sukurti keletui to paties žanro žaidimų. Mūsų atveju universalumas taip pat yra vienas iš svarbiausių Framework‘o aspektų. Turint gerą Framework‘ą, jį galima pritaikyti praktiškai bet kokios verslo šakos sistemai kurti. [3]

Dar vienas kertinis mūsų sistemos komponentas yra duomenų bazė. Nepritaikyta jokiems verslo procesams ar specifiniams poreikiams, tokia duomenų bazė jau turi turėti pradinę lentelių struktūrą. Šios lentelės reikalingos vartotojams, teisėms, grupėms, nustatymams ir kitiems aktualiems dalykams, kurie reikalingi bet kuriai sistemai, saugoti.

Apibendrinta sistemos koncepcinė schema pateikta 1 pav.



1 pav. Saugios Framework sistemos koncepcinė schema

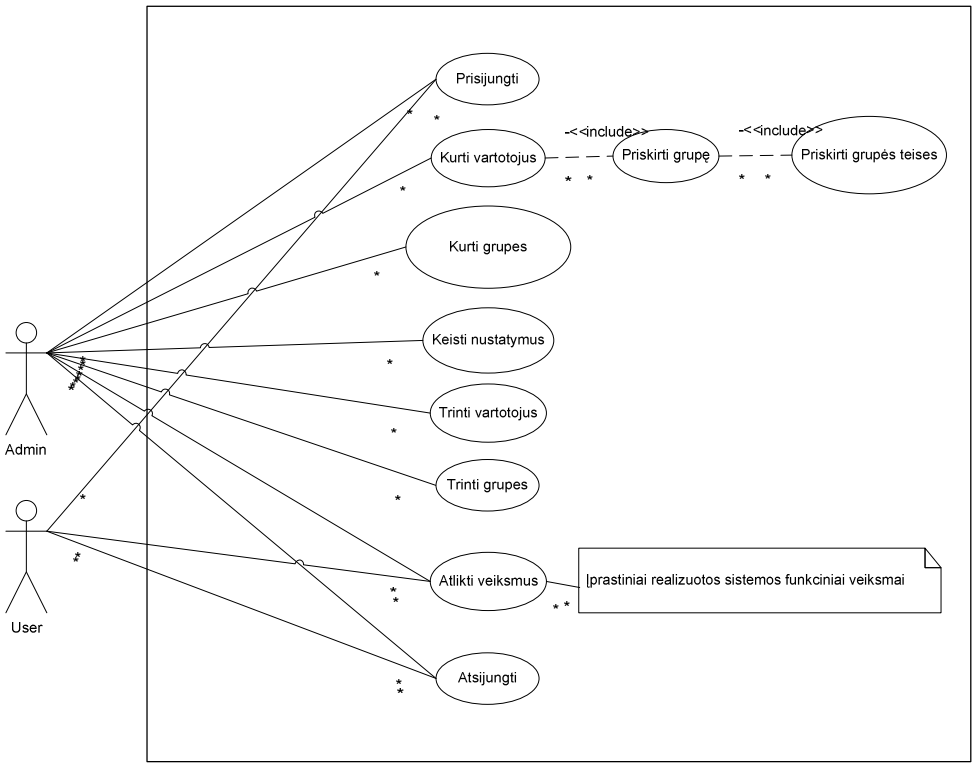
3.2.1. Sistemos panaudojimo atvejų modelis

Sistemos kūrėją galime interpretuoti kaip vieną iš pagrindinių ir svarbiausių sistemos „aktorių“, nes mūsų kuriama sistema orientuota į sistemų kūrėjus. Pateikime galimų panaudojimo atvejų sistemos kūrėjo atžvilgiu aibę (2 pav.).

2 pav. Panaudojimo atvejų diagrama sistemos kūrėjo atžvilgiu

Kaip matome sistemos kūrėjas atlieka nemažai sistemos pradinių konfigūravimo veiksmų, pradedant pirminio vartotojo sukūrimu, vartotojų grupių kūrimu, teisių kūrimu ir baigiant teisių priskyrimu sistemos komponentams. Teisės priskyrimas komponentui reškia, kad minėtas komponentas bus interpretuojamas kaip atskiras sistemos objektas, o jo prieigos ir galimų veiksmų aibė konkrečiam vartotojui priklausys nuo priskirtų teisių rinkinio šio komponento atžvilgiu. Teisių rinkinio priskyrimas taip pat išskirtas kaip atskiras panaudojimo atvejis.

3 pav. pavaizduota sistemos panaudojimo atvejų (angl. Use case) diagrama:



3 pav. Sistemos panaudojimo atvejų diagrama

Kaip matome sistemoje egzistuoja dviejų tipų vaidmenys, t.y. administratorius ir įprastas vartotojas. Administratorius ypatingas tuo, kad jis gali kurti kitus vartotojus i koreguoti jų teises, įtraukti vartotojus į tam tikrą grupę, pašalinti vartotojus, prieiti prie bet kokių programos modulių ir panašiai.[16]

Įprastas vartotojas – tai asmuo, kuris naudosis sistema savo darbui atlikti. Sistemose, apimančiose mūsų uždavinių klasę, dažniausiai figūruoja įvairios vartotojų grupės, turinčios savo vartotojų aibę. Bet kuris vartotojas, priklausantis grupei, turi tos grupės prieigos teises. Priklausomai nuo jų, vartotojas gali atlikti tam tikrą didesnę ar mažesnę galimų operacijų dalį.

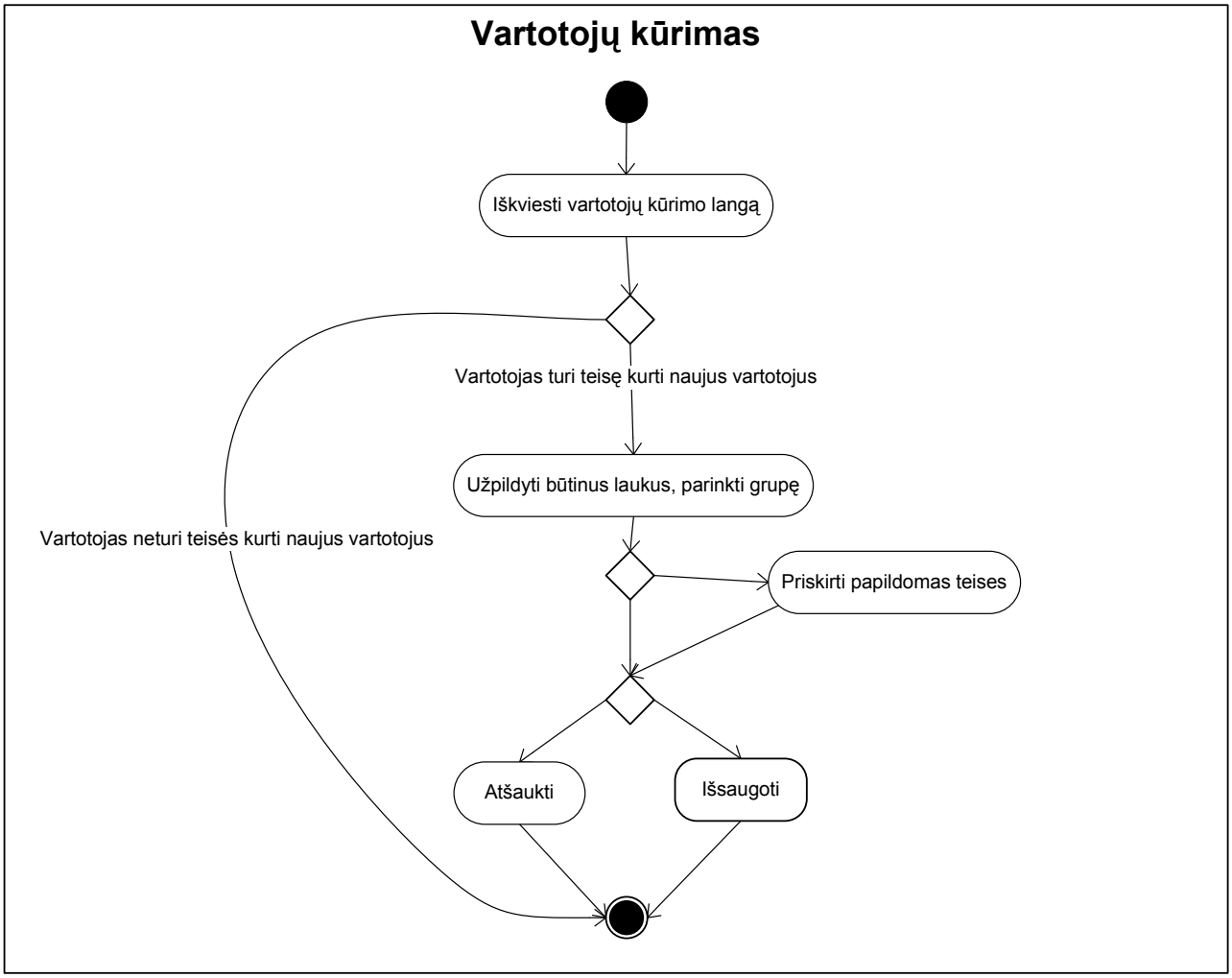
Kadangi sistemos, kuriamos Framework pagrindu gali būti skirtos atlikti įvairius veiksmus, jie diagramoje apibendrinti į vieną panaudojimo atvejį „Atlikti veiksmus“. Šie veiksmai mums nėra tokie aktualūs, kadangi mes labiau orientuojamės į saugumą, o be to likę panaudojimo atvejai yra gana stipriai su tuo susiję.

Sistemos komponentams gali būti priskiriamas administratoriaus tipo tikrinimas. Tai reiškia, kad toks komponentas bus prieinamas tik tokiems vartotojams, kurie sistemoje egzistuoja administratoriaus statusu.

Komponentams taip pat gali būti uždėtas SQL injekcijų tikrinimas. Tai reiškia, kad šis komponentas turi kažkokių įvesties laukų ir jais pasinaudojant yra tikimybė atlikti SQL injekciją.

Kai kurie sistemos komponentai ar objektai gali neturėti jokių tikrinimų dėl teisių, SQL injekcijų ar administratoriaus tipo vartotojo. Tokiu atveju komponentas (objektas) bus prieinamas bet kuriam prisijungusiam sistemos vartotojui.

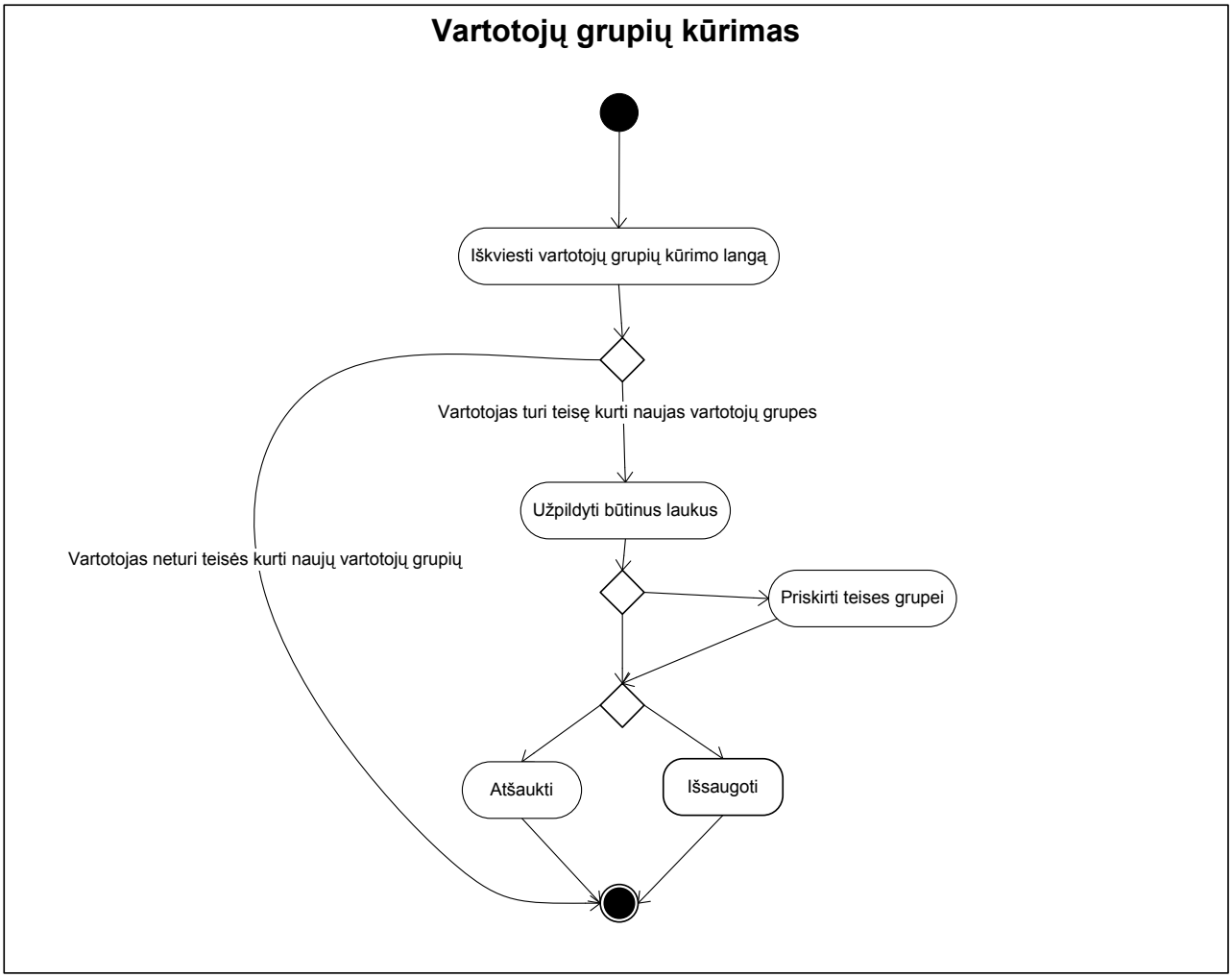
3.2.2. Sistemos veiklų diagramos



4 pav. Vartotojų kūrimo veiklos diagrama

Sistemoje galima kurti naujus vartotojus. Tam pirmiausiai reikia iškviesti naujo vartotojo sukūrimo langą. Tuo atveju, kai iškviečiantysis vartotojas neturi teisės sistemoje

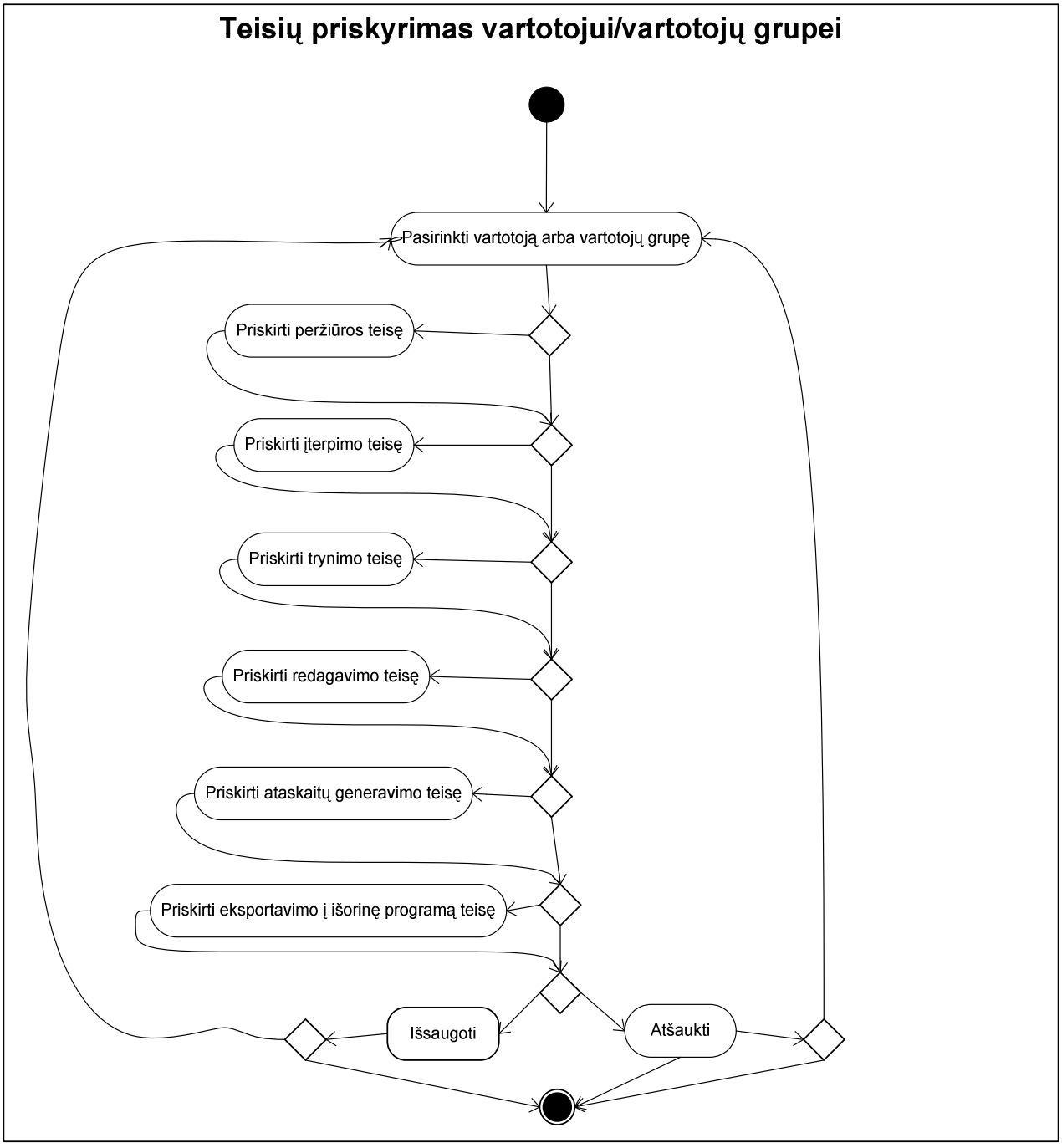
kurti naujus vartotojus, jam parodomas atitinkamas pranešimas ir vartotojų kūrimo langas uždaromas. Kuriant naują sistemos vartotoją, suvedamos privalomų laukų reikšmės, kurios gali varijuoti priklausomai nuo vartotojų logikos sistemoje. Taip pat būtina naujai kuriamam vartotojui parinkti vartotojų grupę. Grupės parinkimas vartotojui reiškia, kad vartotojas įgauna priskirtos grupės įgaliojimus(teises) sistemoje. Jei reikia, papildomai galima šias teises koreguoti, t.y. papildyti naujomis teisėmis. Po šių etapų seka vartotojo išsaugojimas arba veiksmų atšaukimas.



5 pav. Vartotojų grupių kūrimo veiklos diagrama

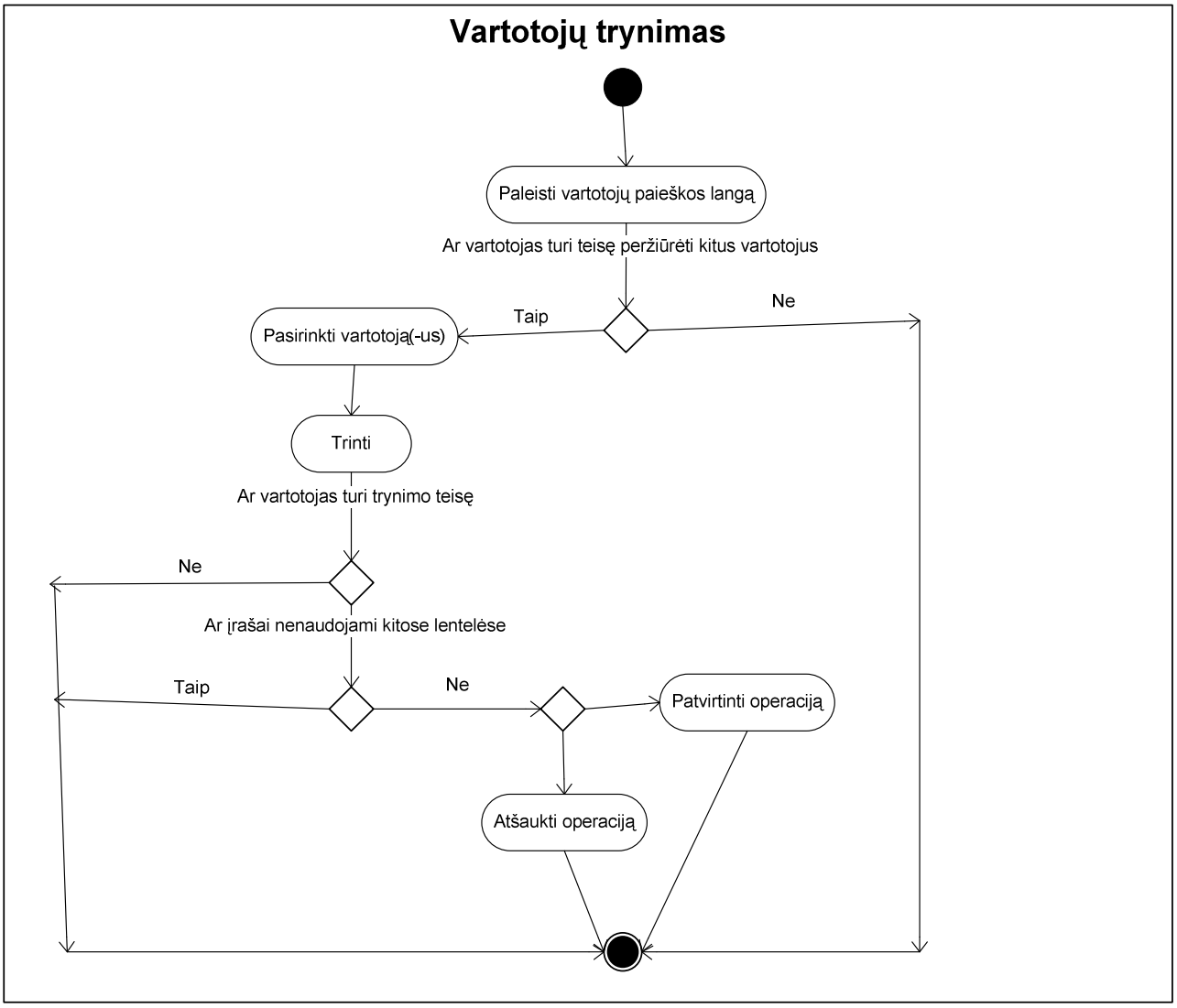
Vartotojų grupių kūrimas iš esmės labai panašus į vartotojų kūrimą. Pirmiausiai vartotojas iškviečia vartotojų grupių kūrimo langą. Toliau sistema patikrina, ar šis vartotojas turi teisę kurti vartotojų grupes. Jei vartotojas turi teisę, jis gali atlikti sekančius veiksmus: užpildyti reikiamus duomenų laukus, priskirti naujai turimai grupei teises sistemoje ir išsaugoti arba atšaukti veiksmus. Kaip matome grupės teisių priskyrimą galima ir praleisti. Tokiu atveju vartotojas, priskirtas šiai grupei galės atlikti tik jam individualiai priskirtų teisių (vartotojų kūrimo/redagavimo procesas) numatomus veiksmus.

Teisių priskyrimas aprašytas sekančioje diagramoje (6 pav.).



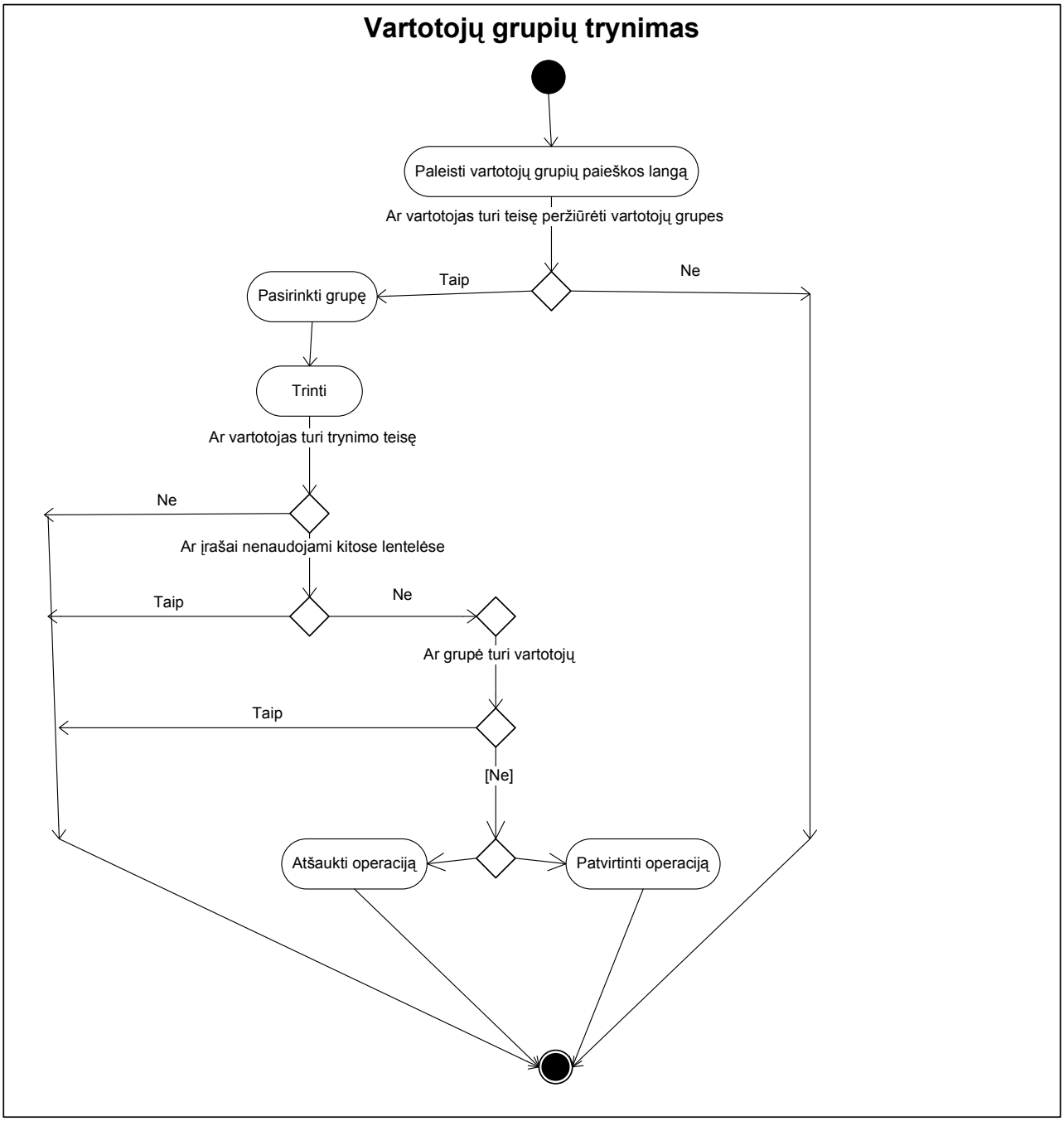
6 pav. Teisių priskyrimo vartotojui veiklos diagrama

Norint priskirti ar kitaip modifikuoti teises sistemos vartotojui ar vartotojų grupei, pirmiausiai reikia patekti į vartotojo ar grupės redagavimo langą, kuris dažniausiai iškviečiamas iš atitinkamo paieškos lango. Toliau atliekamas pasirinktų teisių priskyrimas ir veiksmų išsaugojimas arba atšaukimas. Tokiu pačiu principu gali būti atliekamas ir teisių pašalinimas.



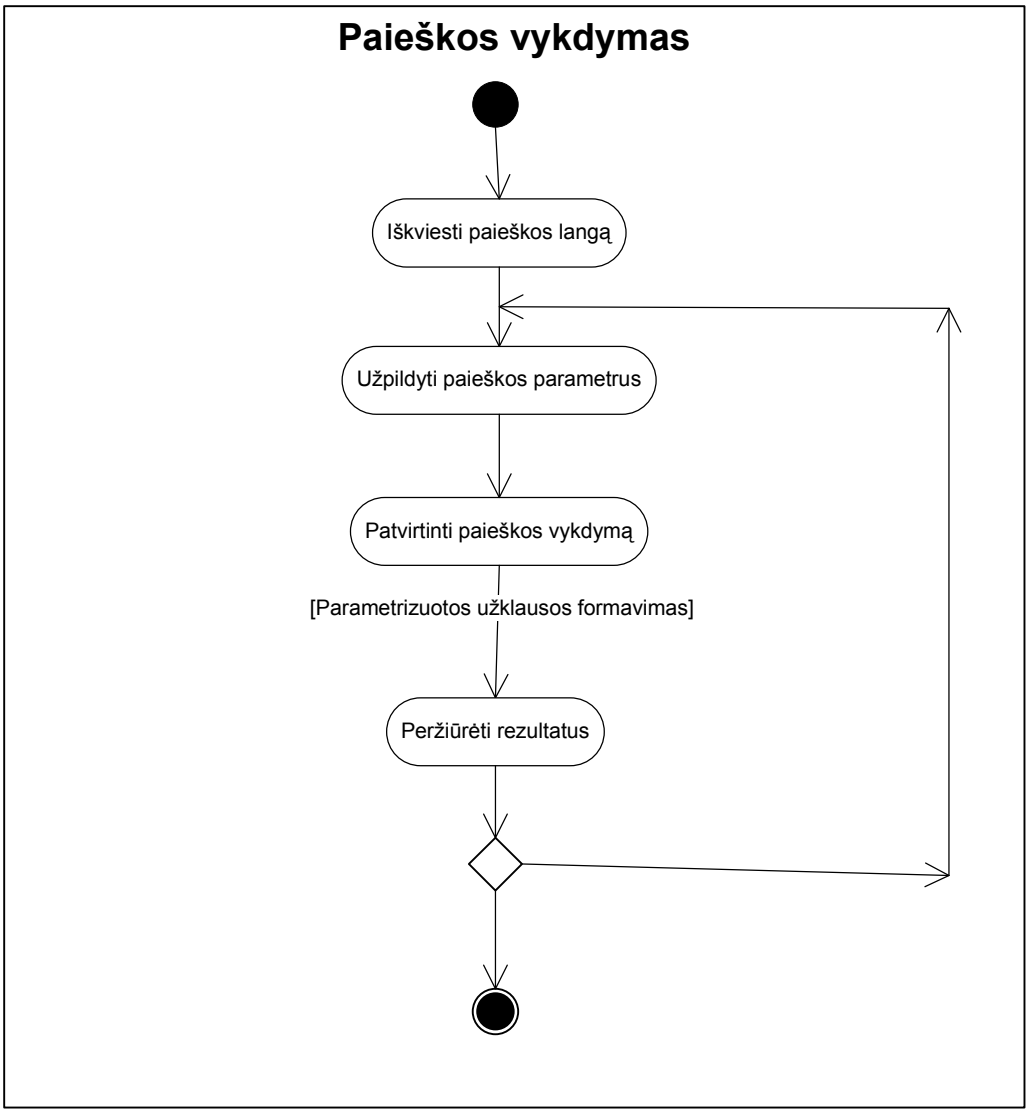
7 pav. Vartotojų trynimo veiklos diagrama

Norint ištrinti sistemos vartotoją, pirmiausiai reikia turėti prieigą prie sistemos vartotojų lango ir teisę šiame lange atlikti trynimą. Toliau sistema tikrina, ar pasirinktas ištrinti vartotojas ir jo duomenys nenaudojami kituose sistemos įrašuose. Tuo atveju, kai vartotojo identifikacinė informacija naudojama kituose objektuose, vartotojas negali būti ištrintas dėl to, kad tai pažeistų veiksmų istorijos principą. T.y. vartotojas negali būti pašalintas, jei jis jau yra sukūręs kokiu nors objektų, kaip užsakymai, pasiūlymai ar pan.



8 pav. Vartotojų grupių trynimo veiklos diagrama

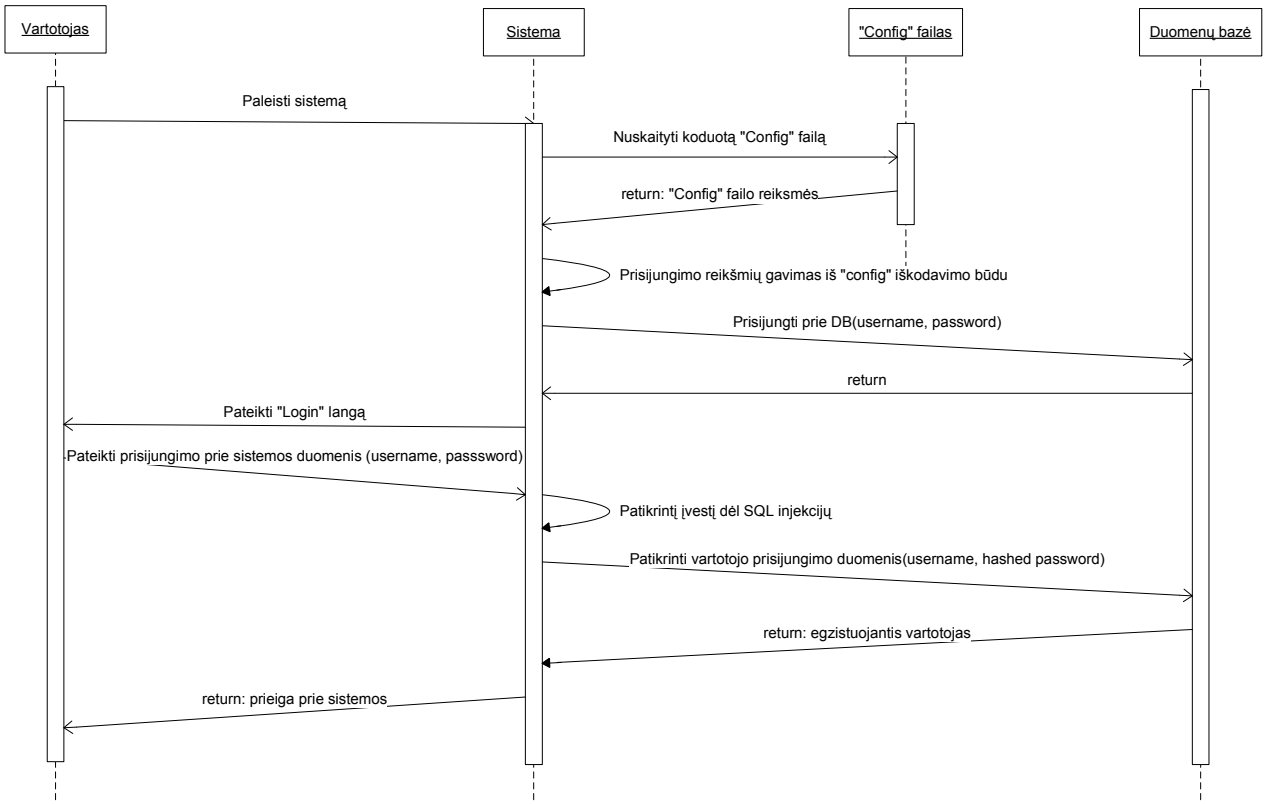
Vartotojų grupių trynimas taip pat gana kompleksiškas procesas. Norint sėkmingai pašalinti grupę, reikia, kad grupė neturėtų vartotojų, t.y. jie turėtų būti priskirti kitoms grupėms.



9 pav. Paieškos vykdymo veiklos diagrama

Paieškos vykdymas yra jautrus sistemos procesas, kadangi jo metu gali būti įvykdyta SQL injekcija. Dėl šios priežasties paieškos lange atliekant paiešką, turi būti būtinai naudojama SPROC technika, t.y. „stored“ parametrizuota procedūra. Saugi Framework sistema vykdymo metu turi automatiškai suformuoti tokiai procedūrai parametrus ir perduoti juos vykdymui. Griežtas parametrizavimas panaikina tiesioginio SQL vykdymo galimybę įsilaužėliui.

3.2.3. Sistemos paleidimo ir prisijungimo sekų diagrama



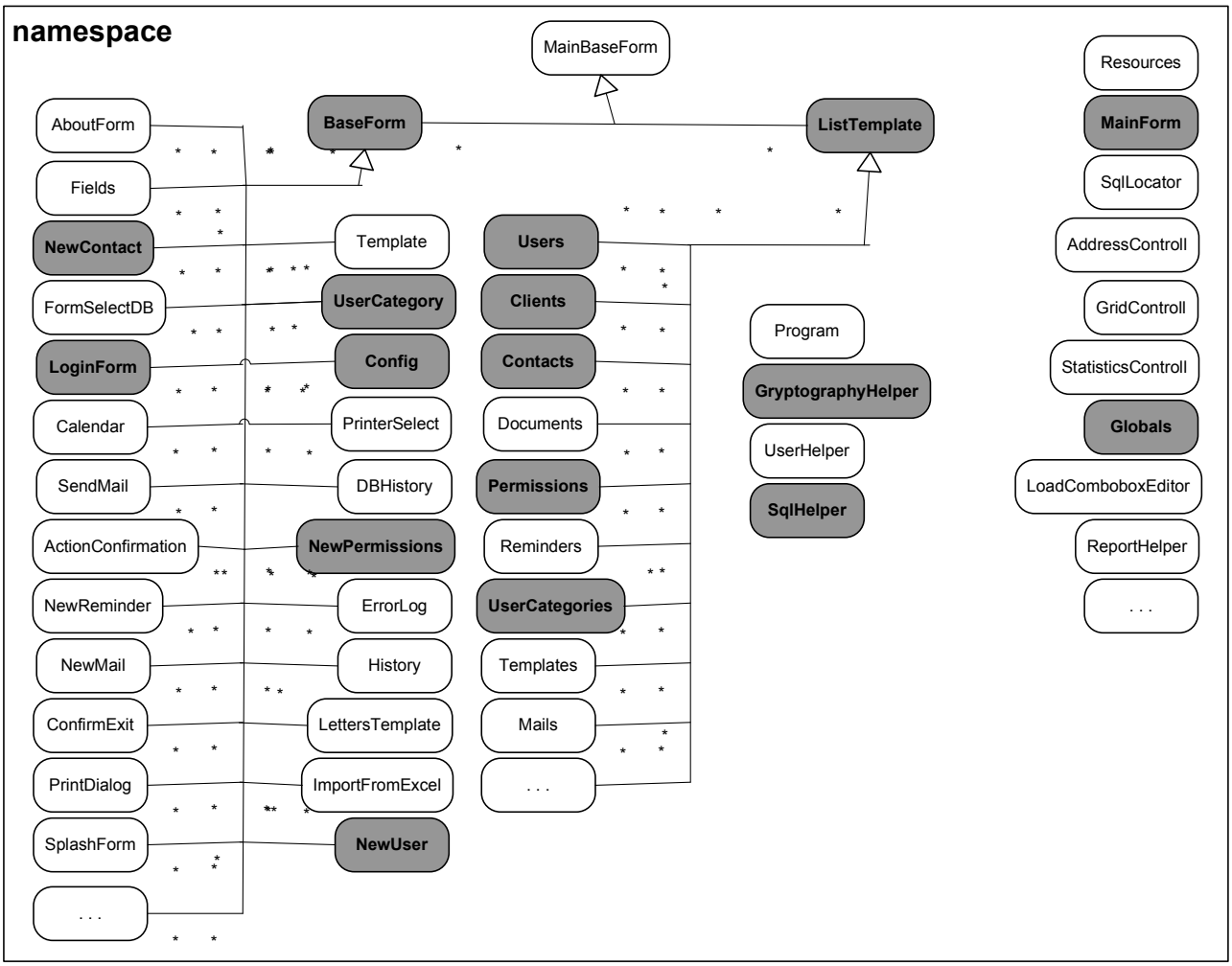
10 pav. Programos paleidimo (prisijungimo prie sistemos) sekų diagrama

Sistemos paleidimas (prisijungimas) svarbus procesas .NET taikomosiose programose, todėl jį pavaizdavome atskiroje sekų diagramoje. Kaip matome iš šios diagramos, inicializavus sistemą, ji kreipiasi į „config“ failą, kuris įprastiniu atveju yra saugomas atviro teksto pavidalu. Iš šio failo nuskaitomi įvairūs konfigūraciniai sistemos parametrai. Vienas iš tokių parametrų yra „conectionString“, apibrėžiantis prisijungimo prie duomenų bazės (DB) parametrus ir prisijungimo duomenis. Mūsų atveju šis failas bus šifruotas, todėl sistema, prieš išgaudama prisijungimo prie DB duomenis, turės jį dešifruoti. Turint reikiamus duomenis, sistema bando jungtis prie DB ir jei pavyksta, vartotojui parodomas prisijungimo („Login“) langas. Šiame lange vartotojas suveda prisijungimo vardą ir slaptažodį. Sistema, prieš patikrindama vartotojo egzistavimą sistemoje, dar patikrina įvesties korektiškumą dėl SQL injekcijų. Vartotojo slaptažodiui, prieš siunčiant tikrinimui į DB, suskaičiuojama maišos funkcija. Duomenų bazėje tikrinama pateikto pagal maišos funkciją suskaičiuoto slaptažodžio ir esančio DB pagal maišos funkciją suskaičiuoto slaptažodžio nurodytam vartotojui

atitikimas. Jei abu slaptažodžiai sutampa, vadinasi minėtas vartotojas egzistuoja sistemoje ir turi galimybę ja naudotis. Šiam vartotojui suteikiama prieiga prie sistemos [5, 6].

3.2.4. Sistemos klasių diagrama

Šioje klasių diagramoje (11 pav.) pateiktos pagrindinės klasės, naudojamos minėto Framework vardų srityje (angl. namespace). Rodyklės parodo klasių ryšius su tėvinėmis klasėmis. Klasės, kurios diagramoje pavaizduotos be rodyklių, pagrinde yra statinės („static“). Daugtaškiu pavaizduotos klasės, kurių buvimas priklauso nuo konkrečios projekto realizacijos, o diagramoje pavaizduotos bazinės klasės, kurios egzistuoja bazinėje sudėtyje. Klasių pavadinimai sudėlioti taip, kad iš jų būtų galima spręsti apie klasės paskyrą projekte.

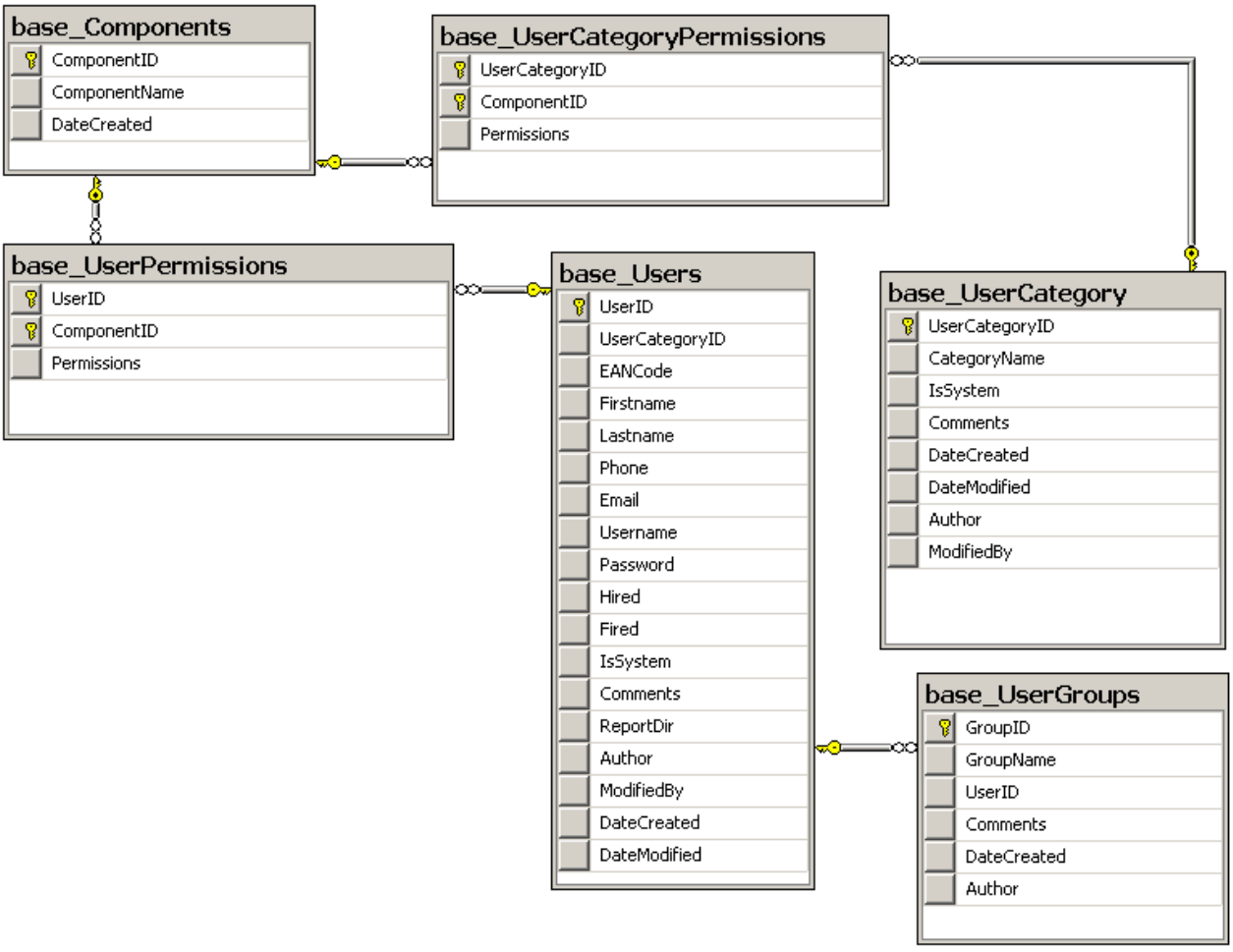


11 pav. Sistemos klasių diagrama

Patamsintai pavaizduotos klasės, kurios sukurtos arba patobulintos šio darbo eigoje ir tobulinamos toliau, t.y. **BaseForm, ListTemplate, NewContact, LoginForm, UserCategoty, NewPermissions, NewUser, Users, Clients, Contacts, Permissions, UserCategories, CryptographyHelper, SqlHelper, MainForm ir Globals**.

3.2.5. Duomenų bazės diagramos

12 pav. pavaizduota diagrama, kurioje pateikta vartotojų, vartotojų grupių, teisių ir komponentų logika, kuria remiantis įgyvendinama teisių politika. Lentelėje „base_Components“ saugomi teisių pavadinimai ir aprašai. Šios teisės apibrėžia anksčiau minėtą langų lygio prieigą.

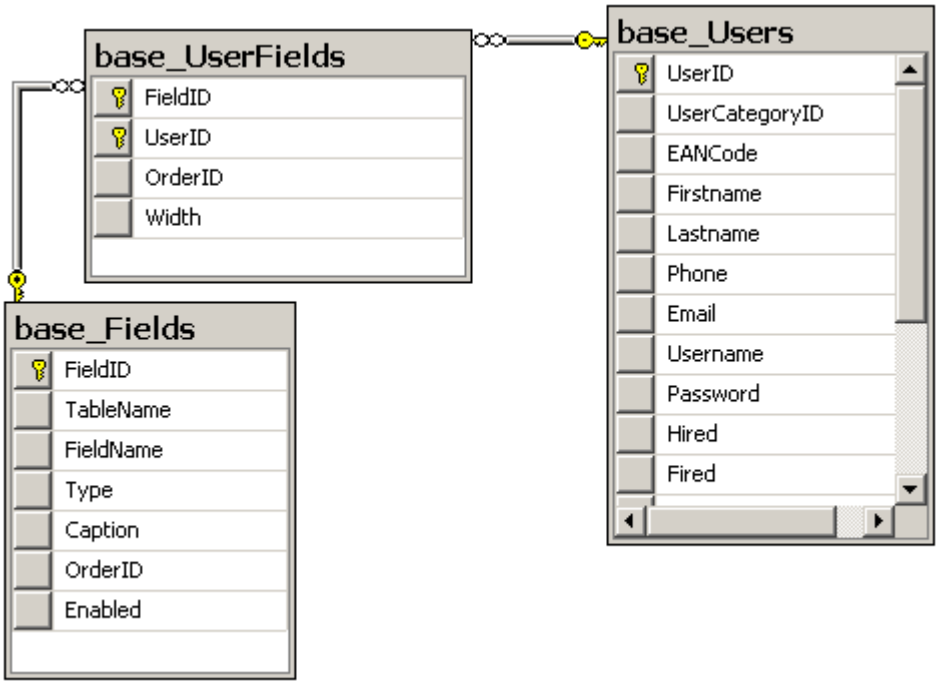


12 pav. Vartotojų ir teisių logikos DB schema

Lentelėje „base_UsercategoryPermissions“ saugomos vartotojų kategorijų teisės, lentelėje „base_Usercategory“ – pačios vartotojų kategorijos.

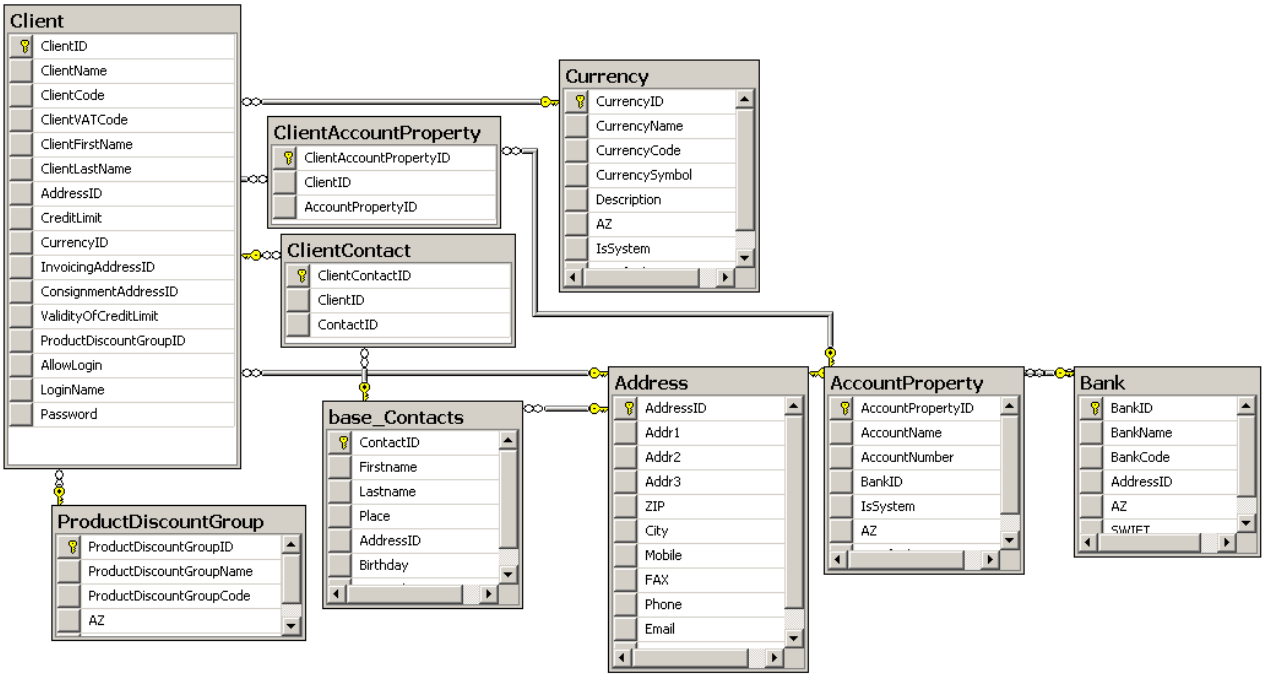
Kitų DB lentelių vaizduoti netikslinga, nes jos dažniausiai priklauso nuo konkrečios realizacijos. Verta paminėti lentelę „base_Config“, kurioje saugomi programos nustatymai, kuriuos galima pakeisti vartotojo sąsajoje. Kiti klasifikatoriai, kurie dažniausiai egzistuoja beveik visose sistemose: adresas, klientas, šalis, miestas, regionas, užsakymas, sąskaita-faktūra, bankas, sąskaita, kontaktas, valiuta, priminimas, dokumentas, laiškas ir taip toliau.

13 pav. vaizduoja trijų lentelių kombinaciją, kurios pagalba būtų galima įgyvendinti vartotojui pateikiamų laukų aibę. T.y. kiekvienam vartotojui galima uždrausti matyti tam tikrą lauką (-us) konkrečiame lange.



13 pav. Atvaizduojamų laukų logika

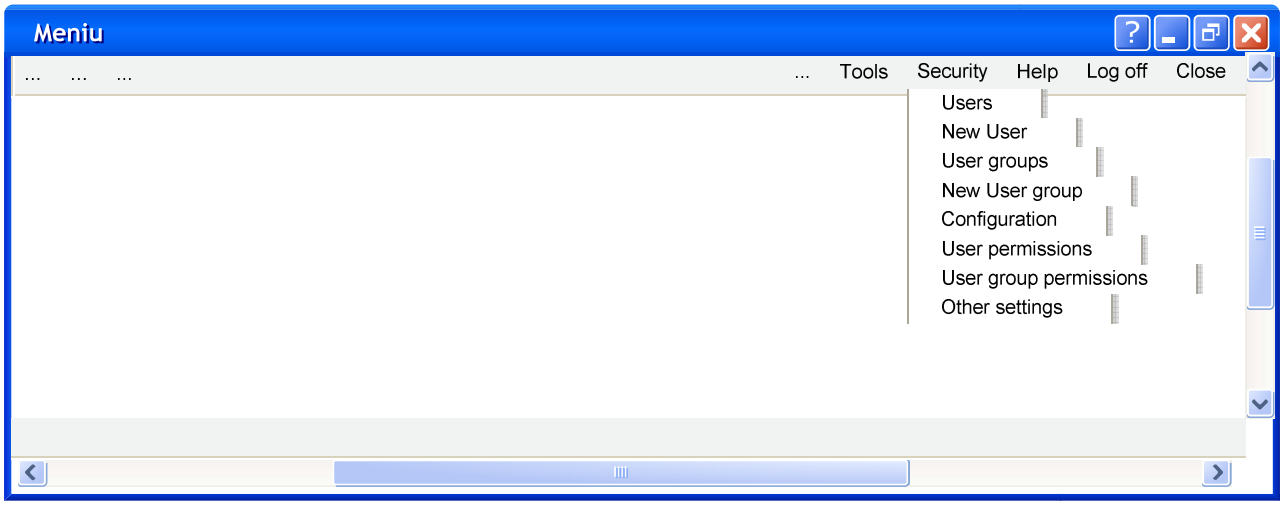
Bazinė klientų, kliento kontaktinių asmenų, adresų, bankų ir kitos informacijos logika duomenų bazėje, kuri gali šiek tiek vyrauti priklausomai nuo realizacijos:



14 pav. Šabloninė klientų duomenų DB logika

3.2.6. Vartotojo sąsajos projektas

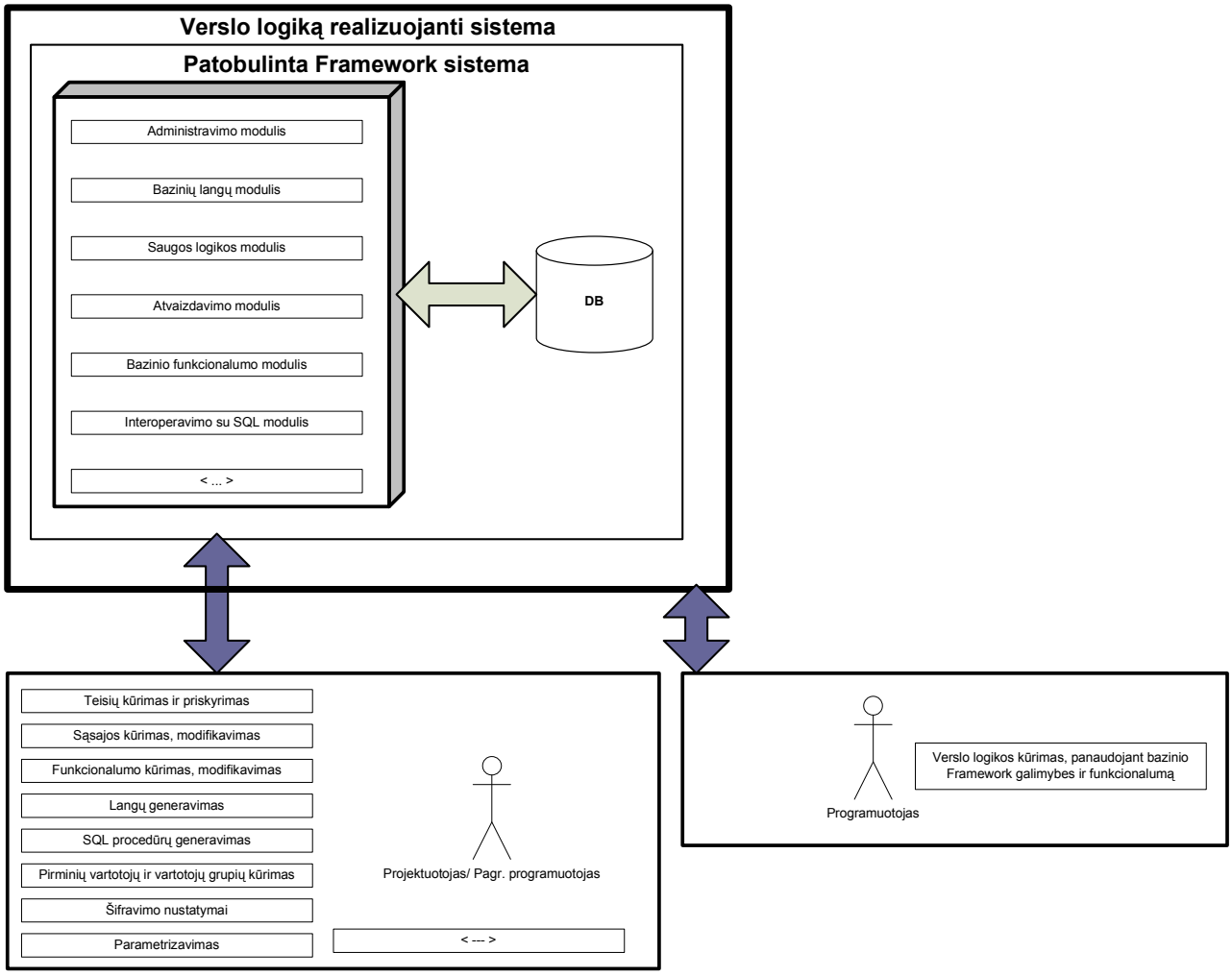
15 pav. pateiktas bazinės vartotojo sąsajos projektas, visuomet turėsiantis standartinius meniu punktus: „Tools“ (Nustatymai); „Security“ (Saugumas); „Help“ (Pagalba); „Log off“ (Atsijungti); „Close“ (Užverti). Likę meniu punktai pažymėti daugtaškiu ir bendruoju atveju jų panaudojimas ir specifika priklausys nuo konkretaus sprendimo specifikos. „Security“ meniu punktas skirtas saugumo ir su tuo susijusių kitų objektų įvedimui, redagavimui, saugojimui ir konfigūravimui.



15 pav. Pagrindinio programos meniu lango schema

4. PATOBULINTOS FRAMEWORK SISTEMOS REALIZACIJA

4.1. Sistema kūrėjo atžvilgiu



16 pav. Bendra sistemos struktūra kūrėjo požiūriu

Patobulinta Framework sistema visų pirma skirta sistemų kūrėjams arba programuotojams, todėl pirmiausiai apžvelgsime ją būtent šiuo aspektu. 16 pav. pavaizduota bendra sistemos struktūra. Kaip jau minėta anksčiau pagrindinis tikslas yra ne kažkokios konkrečios verslo sistemos produktas, o įrankis, skirtas tokioms sistemoms kurti (sistemų pagrindas). Šio įrankio pagrindinis tikslas – universalumas, pritaikomumas įvairiems sprendimams, t.y. mūsų sukurta patobulinta Framework sistema gali būti įvairių verslo logiką realizuojančių sistemų pagrindu. Tam, kad šis įrankis būtų tinkamas ir saugumo prasme, buvo padaryti įvairūs papildymai ir patobulinimai, aprašyti ankstesniuose skyriuose.

Patobulinta Framework sistema susideda iš daugelio modulių (administravimo, bazinių langų, saugumo ir t.t.), kurių junginys sudaro bendrą sistemą, sąveikaujančią su

duomenų baze (DB). Kaip matome paveikslėlyje, projektuotojas arba pagrindinis programuotojas atlieka pirminį sistemos konfigūravimą, t.y. jis modifikuoja patį „karkasą“ (Framework). Toliau atliekamas realios sistemos, paremtos Framework‘u kūrimas, t.y. kūrimas, panaudojant Framework galimybes ir funkcionalumą. Kitaip tariant atliekamas aukšto lygio programinio universalaus kodo panaudojimas. Šis procesas pasižymi tuo, kad jį gali atlikti mažesnės kvalifikacijos sistemų kūrėjai, nes jiems tereikia mokėti panaudoti esamas Framework galimybes. Tai ypač paspartina sistemos kūrimo procesą, o saugiai sukurtas Framework tuo pačiu padidina ir galutinės sistemos saugumo lygį.

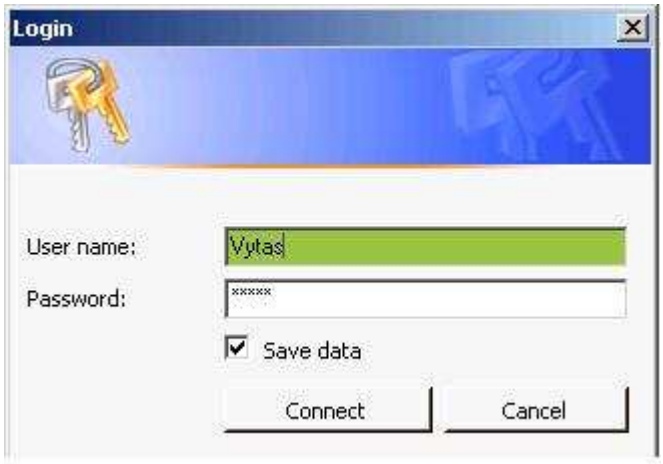
„Sistemų analitikas/Pagr. Programuotojas“ ir „Programuotojas“ veikėjai paveikslėlyje išskirti į atskiras grupes, bet konkrečiu atveju tai gali atlikti ir vienas žmogus arba pareigų pasiskirstymas gali būti kitoks.

Sistemų analitiko/Pagr. Programuotojo atliekamas pradinis sistemos konfigūravimas bendruoju atveju yra gana sudėtingas ir daug aiškinimo reikalaujantis procesas, todėl jo neaprašysime.

4.2. Sistema vartotojo požiūriu

Panagrinėkime sistemos naudojimą vartotojo požiūriu nuosekliai atliekant visus tipinius veiksmus sistemoje. 17 pav. matomas prisijungimo („Login“) langas. Šis langas pasirodo tada, kai paleidžiame sistemą. Sistema galėsime naudoti tik tuo atveju, jeigu įvesime teisingus vartotojo vardą ir slaptažodį. Jeigu pasitikime savo kompiuterio saugumu, galima uždėti opciją „Save data“, kuri reiškia, kad kompiuteris atsimins mūsų prisijungimo duomenis ir jų nereikės suvedinėti kiekvieną kartą iš naujo.

Sistemos paleidimo metu pirmiausiai kreipiamasi į „config“ failą, kuriame saugoma prisijungimo prie DB reikšmė (angl. connection string). Priminsime, kad mūsų atveju, skirtingai nei įprastiniu, šis failas yra šifruotas. Šiame faile nurodyti duomenys iššifrojami ir pagal juos bandoma pasiekti DB. Jei DB pasiekti nepavyksta, parodomas papildomas langas, kuriame galima suvesti kitą prisijungimo prie DB reikšmę. Jei prisijungimas prie DB sėkmingas, pasirodo „Login“ langas, kuriame įvedami prisijungimo duomenys. Įvestam slaptažodžiui skaičiuojama maišos funkcija ir kartu su prisijungimo vardu šie duomenys perduodami į DB, kur patikrinama ar toks vartotojas egzistuoja. Jei egzistuoja – gaunama prieiga bei vartotojo teisių ir matomų laukų rinkinys, pagal kurį sistema pritaiko grafinę sąsają naudojimui.



17 pav. Prisijungimo langas

Sėkmingai prisijungus, patenkame į pagrindinį sistemos langą (18 pav.).

18 pav. Pagrindinis meniu langas

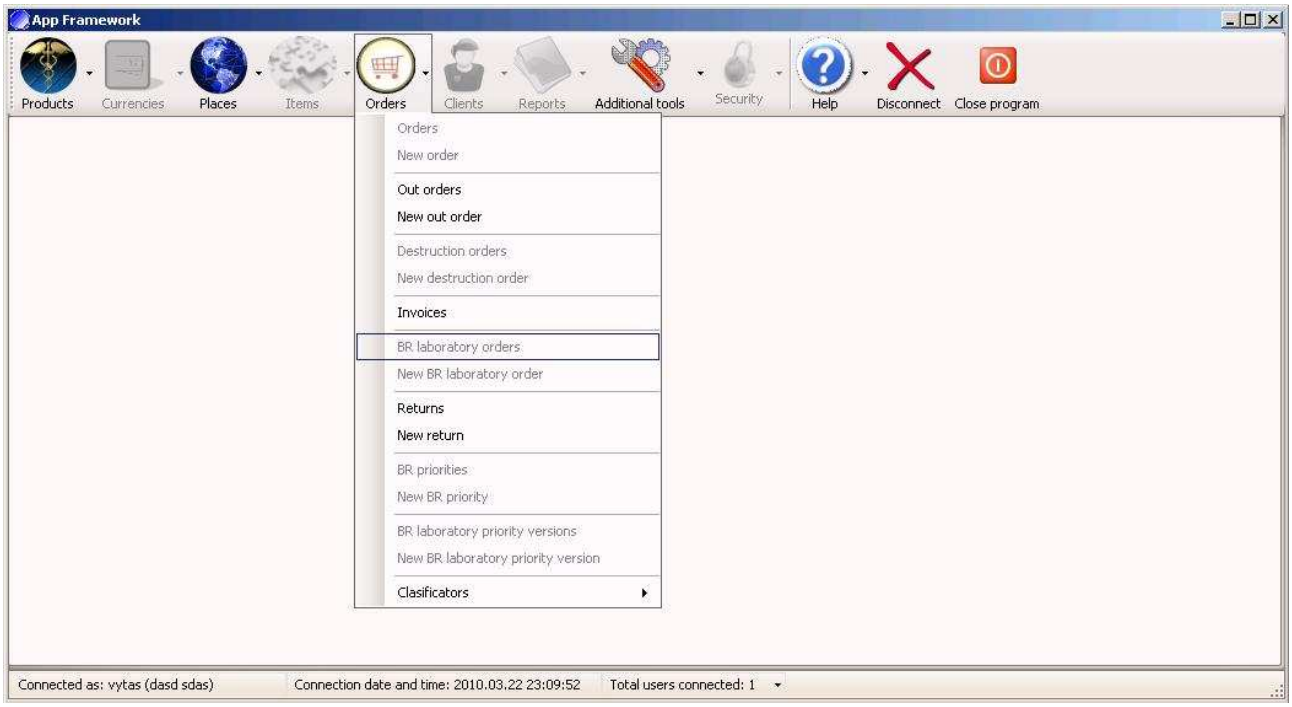
Vartotojo sąsają įgyvendina klasė „MainForm“. Šios klasės tikslas – užtikrinti grafinę vartotojo sąsają su sistema, patogų naudojimąsi ir navigaciją tarp suskirtytų kategorijų, vartotojo sąsajos adaptavimą pagal vartotojo poreikius. Čia matomi ir meniu komponentai, kurie įeina į bazinę sudėtį – **Clients, Reports, Additional tools, Security, Help, Disconect, Close program**. Minėtame lange matome, kad visi meniu skyriai ir poskyriai yra aktyvūs – t.y. prisijungęs vartotojas turi pilnas teises sistemoje (sistemos administratorius).

Jei prisijungęs vartotojas yra sistemos administratorius, apatinėje lango juostoje, matoma ne tik prisijungusių prie sistemos vartotojų kiekis, bet papildomai galima peržiūrėti ir jų vardą, pavardę ir prisijungimo vardą.

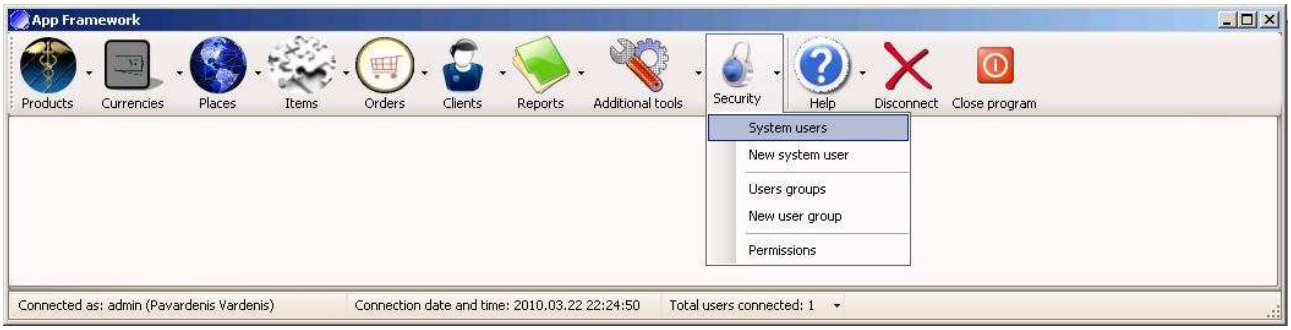
Sekantis paveikslėlis (19 pav.) iliustruoja atvejį, kai prisijungęs vartotojas turi tik tam tikrą dalį teisių sistemoje (prieigos galimybes pagal suteiktas teises). Jei tam tikrame meniu skyriaus viduje, vartotojas neturi teisės prieiti prie visų papunkčių, tai ir pagrindinis meniu punktas bus neaktyvus. Meniu punktų organizavimo taisyklės ir jų pasiekimo galimybė priklausomai nuo suteiktų teisių gali būti organizuojamos įvairiais būdais. Šiuo atveju draudžiami langai yra matomi, bet neaktyvūs. Kitas galimas variantas – draudžiamų langų

nerodymas meniu komponente, t.y. vartotojas mato tik tuos meniu punktus ir tik tuos papunkčius, kurie jam yra leistini pagal jo teises sistemoje.

Manipuliacija teisėmis paremta dvejetainių skaičių principu. Dvejetainis skaičius yra tam tikro fiksuoto ilgio, kur kiekvienas simbolis sekoje atitinka tam tikrą tipinę sistemos operaciją (įterpimas, trynimas, atnaujinimas ir t.t.). Jei operacijos vietą atitinkantis simbolis yra 1, tai reiškia leidimą, o jei 0 – draudimą. Tam, kad teisės atrinkimo greitaveika būtų didesnė, panaudotos postūmio operacijos.

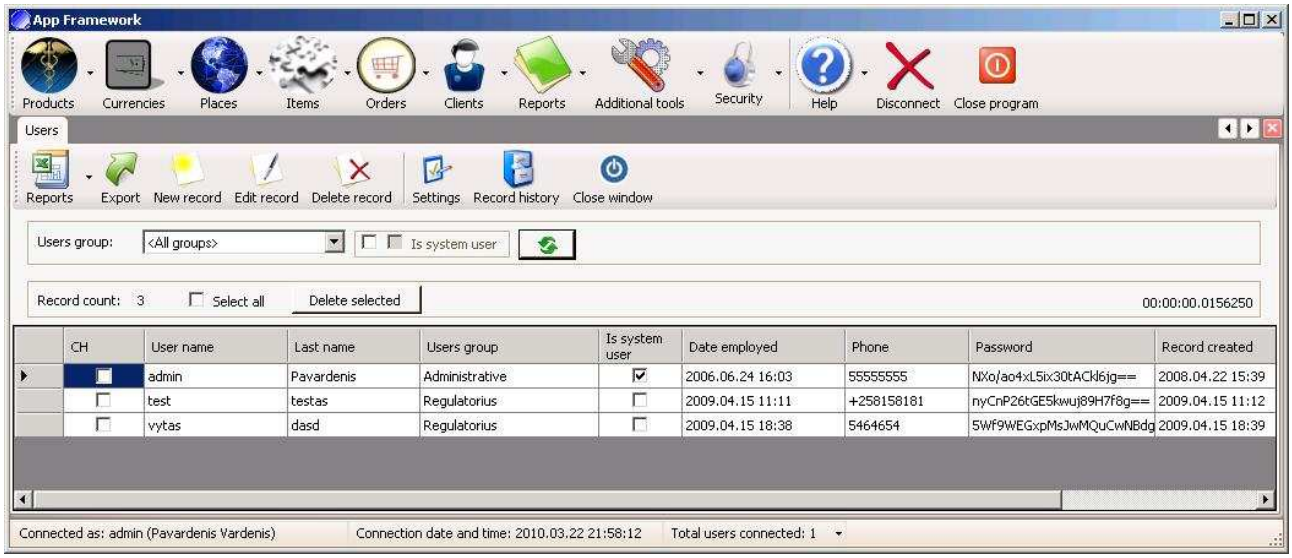


19 pav. Meniu ribotomis teisėmis



20 pav. Saugumo skilties meniu

Meniu srityje „Security“ vykdoma vartotojų ir vartotojų grupių veiksmų aibė. Jei prisijungęs vartotojas neturi administratoriaus įgaliojimų, jis neturi teisės prieiti prie šių duomenų, todėl šis meniu punktas ir visi vidiniai punktai jam būna neaktyvūs. Meniu srityje „Additional tools“ įprastai rasime sistemos konfigūravimo langą bei kitus konfigūravimo langus arba programos klasifikatorius.

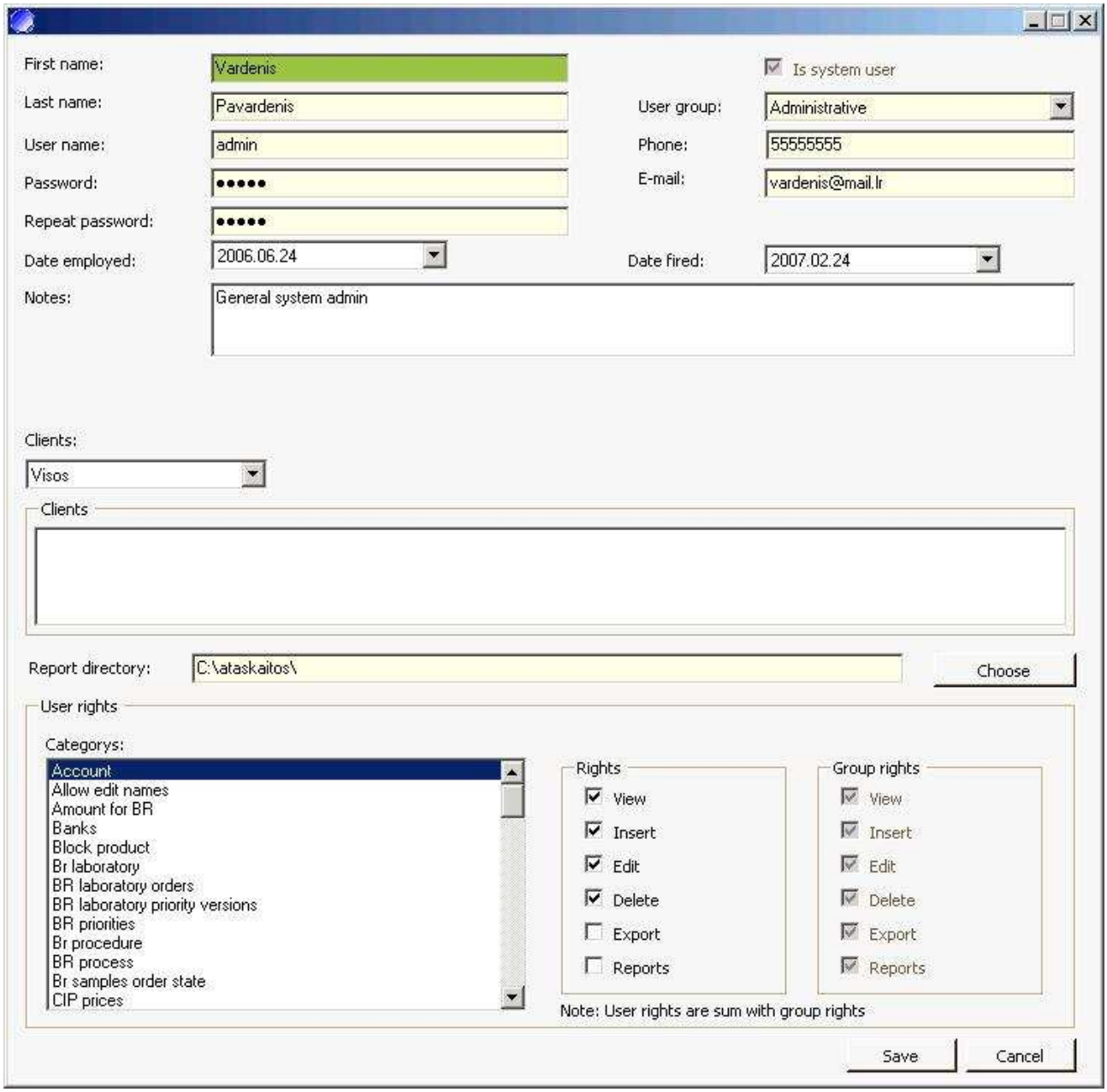


21 pav. Vartotojų paieškos langas

Gilesniame lygyje matome paieškos langą, kuriame galima atlikti konkretaus elemento redagavimą ar kitą veiksmą, atlikti paiešką pagal raktinį žodį ir kitus parametrus. Raktinio žodžio paieška naudojama visuose paieškos languose ir atlieka duomenų filtro pagal tekstinius laukus vaidmenį. Sudėtingesniuose ir daugiau duomenų talpinančiuose languose naudojami papildomi įvairių tipų (pasirenkamasis sąrašas, pažymimasis sąrašas, požymio kriterijus ir t.t.) kriterijai.

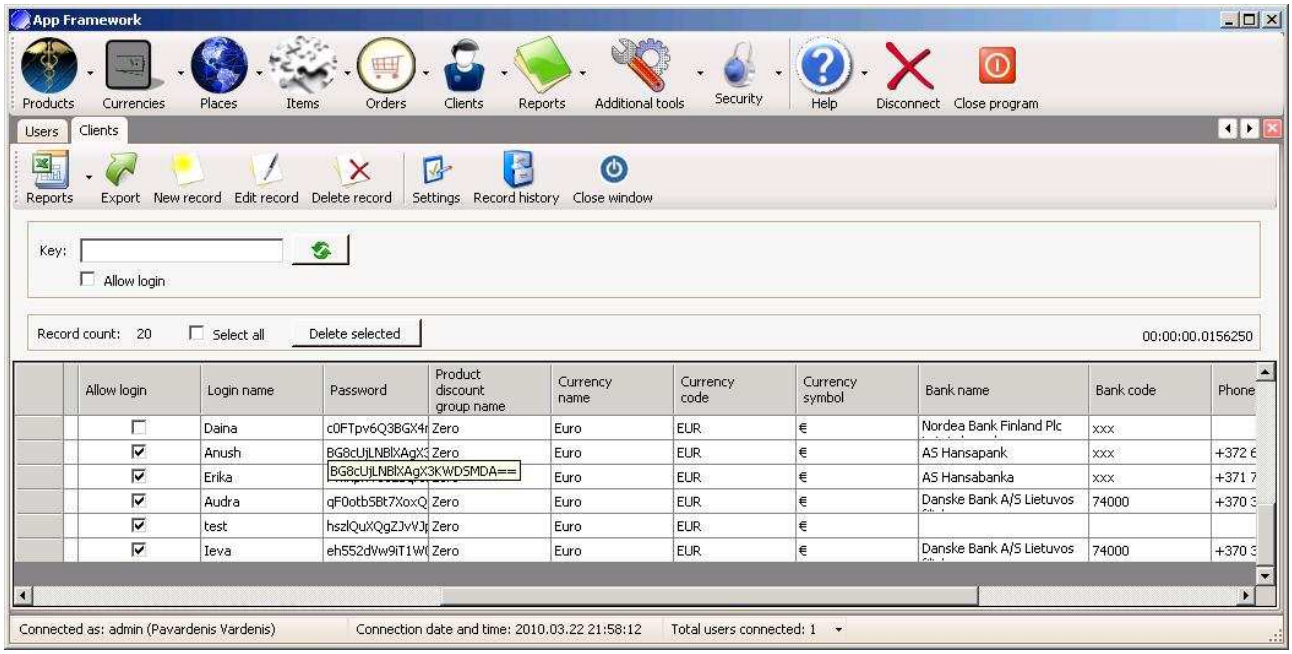
Visuose paieškos languose dukart spustelėjus pele ant konkretaus įrašo, atsidaro to įrašo redagavimo langas.

Jei vartotojas neturi teisės atlikti tokį veiksmą, redagavimo langas neatidaromas ir parodomas pranešimas „Jūs neturite teisės atlikti šiuos veiksmus“.



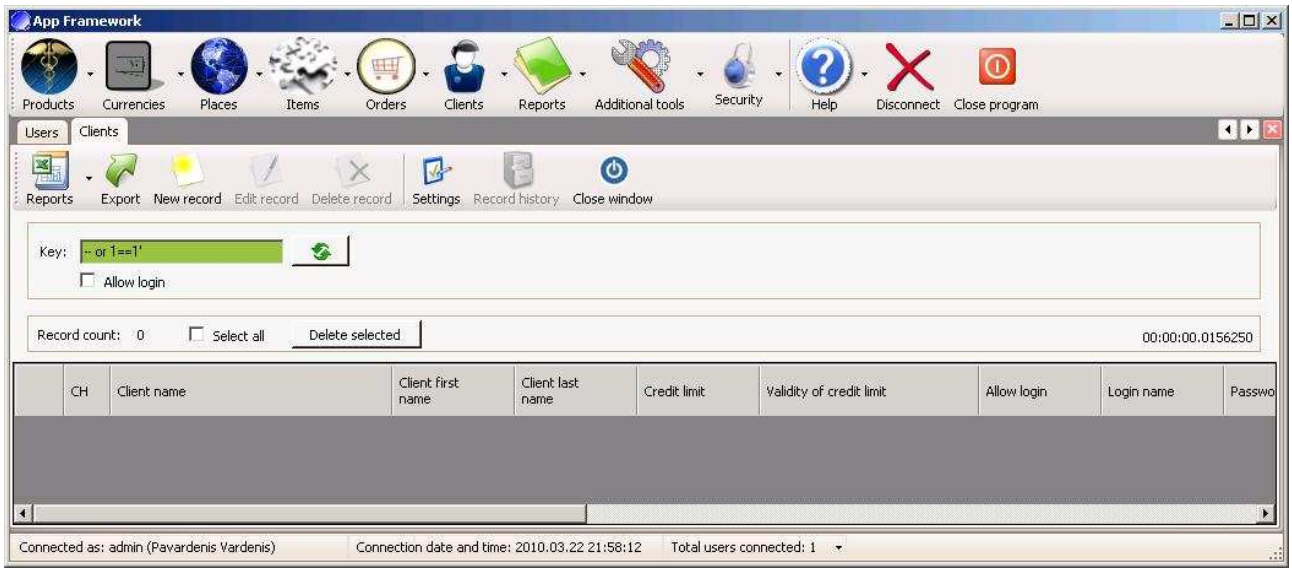
22 pav. Vartotojo įvedimo ir redagavimo langas

Sistemos vartotojo kūrimo/redagavimo langas pavaizduotas 22 paveikslėlyje. Šiame lange be įprastų įvedimo laukų matoma ir teisių sritis. Kaip jau minėta anksčiau, pagal nutylėjimą vartotojui priskiriamos jo grupės teisės, kurias papildomai galima paredaguoti.



23 pav. Klientų paieškos langas

Klientų paieškos langas (23 pav.) – taip pat vienas svarbiausių langų sistemoje. Kaip matome, čia taip pat realizuotas maišos funkcijos skaičiavimas slaptažodžiams šifruoti.



24 pav. SQL injekcijos tikrinimas klientų paieškos lange

Paieškos languose, kur yra laukeliai paieškos raktui nurodyti, galima SQL injekcijos rizika. 24 paveikslėlyje matome, kad bandymas atlikti SQL injekciją nesėkmingas ir jokių rezultatų neduoda.

Bazinėje klasėje „SqlHelper“, per kurią atliekamos visos operacijos su DB, galimas tik parametrizuotų procedūrų iškvietimas. Tai reiškia, kad visi perduodami parametrai yra griežtai tipizuoti ir kad dingsta dinaminės SQL užklausos galimybė. 24 paveikslėlyje pavaizduotame pavyzdyje yra tekstinis įvedimo laukas, kuriame įvesta reikšmė perduodama į

DB kaip „nvarchar(250)“ per parametrą. Tai reiškia, kad SQL kalbos interpretatorius priima įvesties tekstą tik kaip tekstinę eilutę, kuri visada bus patalpinta SQL sakinyje pavidalu N‘x‘, kur x- įvestas tekstas. Tokiu būdu įvestis SQL‘e bus interpretuota kaip tekstas, net jei ir jame įvesta SQL injekcijos išraiška, ir ši įvestis negalės pakeisti normalaus (saugaus) SQL vykdymo.

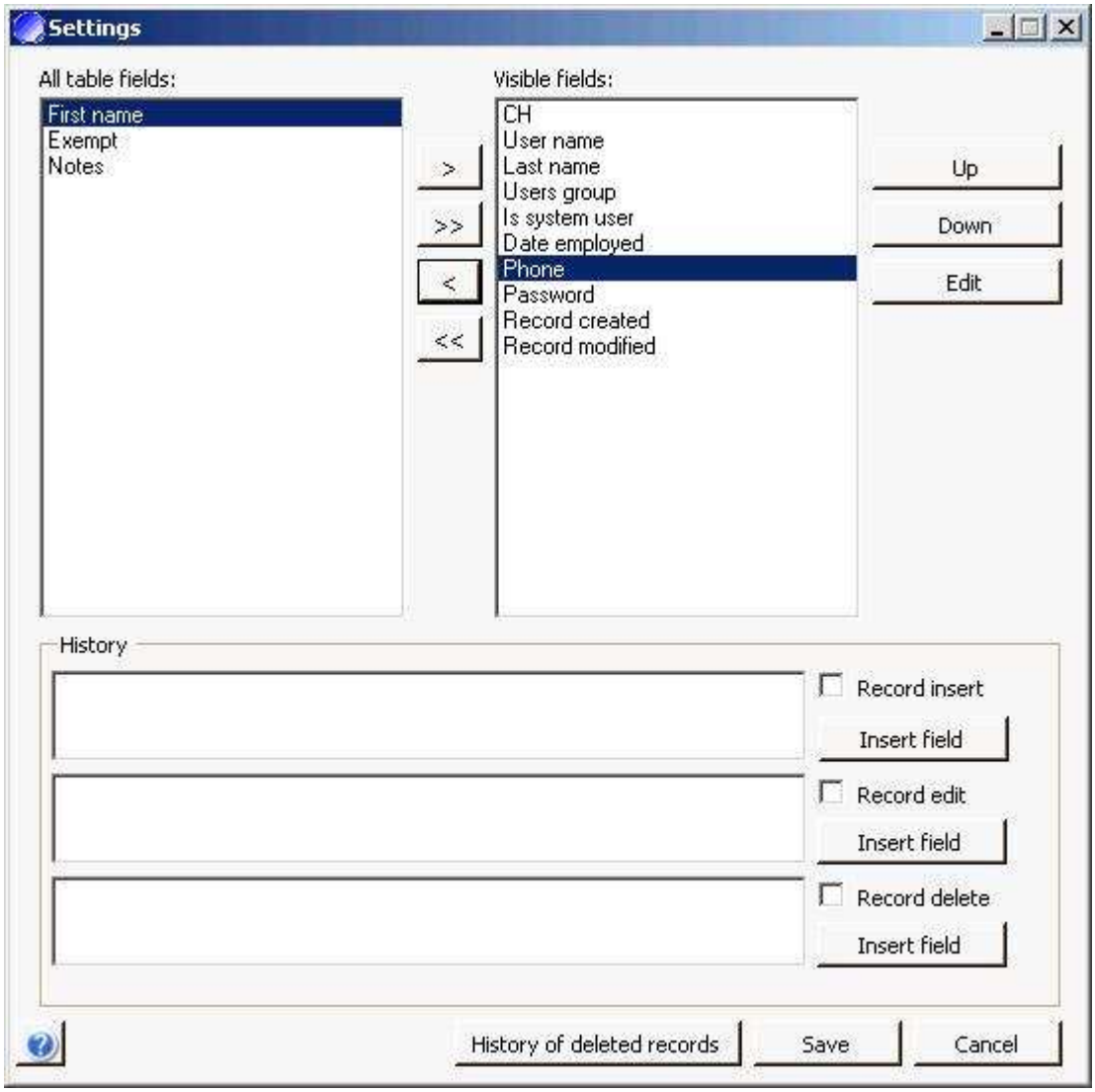
25 pav. Vartotojų grupių paieška ir redagavimas

25 pav. matome vartotojų grupių įvedimo/redagavimo langą. Šio lango dėka kuriamos ir redaguojamos vartotojų grupės ir jų teisės. Vartotojų grupių egzistavimas sistemoje labai palengvina teisių realizaciją. T.t., kad nereiktų keliems vartotojams priskyrinėti tas pačias teises, galima sukurti naują grupę, priskirti jai teises ir po to minėtiems vartotojams tereiks priskirti naująją grupę. Tokiu būdu šie vartotojai įgaus visas jiems priskirtos grupės teises iš karto.

Taip pat rekomenduojama turėti grupę, kuri neturi jokių teisių. Ją naudoti tikslinga tada, kai pavyzdžiui atleidžiamas iš darbo koks nors sistemos vartotojas – kad nereiktu nuiminėti jam daugelio teisių sistemoje, jis paprasčiausiai keletu paspaudimų priskiriamas grupei be teisių ir tokiu būdu pasiekiame tą patį rezultatą per trumpesnį laiką. Iš kitos pusės tai patogu ir dėl to, kad tampa efektyvesnis duomenų atrinkimas – visi nebedirbantys ar neturintys teisės naudotis sistema vartotojai bus randami toje grupėje.

26 pav. Sistemos teisių langas

26 pav. matomas paieškos sisteminis langas, kuriame filtruojama ir išvedama sistemoje esančių teisių pagrindinė informacija, atliekamas naujų teisių įvedimas (nerealizuota) ir priskyrimas sistemos langams ar langų komponentams (nerealizuotas variantas, kai tai atliekama tiesiai per vartotojo sąsają).



27 pav. Laukų matomumo redagavimo langas

27 pav. matomas „Settings“ langas, realizuojantis laukų matomumą, eiliškumą ir atvaizdavimo formą vartotojui. „Settings“ langas iškviečiamas paieškos languose, meniu juostoje paspaudus mygtuką „Settings“. Srityje „Visible fields“ sukelti tie laukai, kurie bus matomi vartotojui. Cia taip pat gali būti koreguojamas ir jų eiliškumas. Srityje „All fields“ matomi likę galimi laukai, kurie nebus matomi vartotojui. „History“ grupėje galima sudaryti šablonus, pagal kuriuos gali būti papildomai saugomi vartotojo veiksmai sistemoje: įterpimas, redagavimas, trynimas.

4.3. Realizacijos išvados

Šioje patobulintoje Framework sistemoje įgyvendinti reikalavimuose apibrėžti elementai: vartotojų autentifikacija ir autorizacija, vartotojų ir jų grupių bei teisių politika, konfigūracinio failo šifravimas, duomenų šifravimas, SPROC technikos panaudojimas interaktyvumui su duomenų baze įgyvendinti ir taip toliau. Kitaip tariant panaudotas gana

nemažas įvairių patobulinimų, technikų, papildomo funkcionalumo paketas, dėl ko gavome plačiai pritaikomą ir saugesnę Framework sistemą.

5. PATOBULINTOS FRAMEWORK SISTEMOS TYRIMAS

5.1. Greitaveikos tyrimas

- Sistemos startavimas

Sistemos paleidimas (startavimas) – tai procesas, kurio rezultatas yra užkrauta naudojimui sistema (jos grafinė sąsaja). Sistemos startavimo metu nuskaitomas „config“ failas, savyje talpinantis sistemos konfigūraciją. Mūsų sistemoje, skirtingai nei įprastiniu atveju, „config“ failas yra šifruotas.

5 lentelė. Koduoto "config" failo įtaka sistemos startavimo procesui

	Įprastinė .NET sistema (nešifruotas „config“ failas)	Patobulinta Framework sistema (šifruotas „config“ failas)
Vidutinė sistemos startavimo trukmė, sek.	1,5	1,75

Išvada: Sistemos startavimas naudojant šifruotą konfigūracijos failą užtrunka maždaug 0,25 sekundės ilgiau, nei įprastiniu atveju, kai šis failas yra atvirojo teksto. Svarbu paminėti tai, kad šis laiko skirtumas priklauso ir nuo konfigūracinio failo apimties, t.y. nuo panaudotų šiame faile konfigūracijos elementų kiekio. Taip pat svarbu pabrėžti, kad tiek šio, tiek ir likusių testų rezultatai gali skirtis, priklausomai nuo naudojamos techninės įrangos; vietinio tinklo ar interneto bei duomenų bazės serverio spartos (tuo atveju, kai DB yra ne tame pačiame kompiuteryje) bei kitų aspektų, įtakojančių greitaveiką.

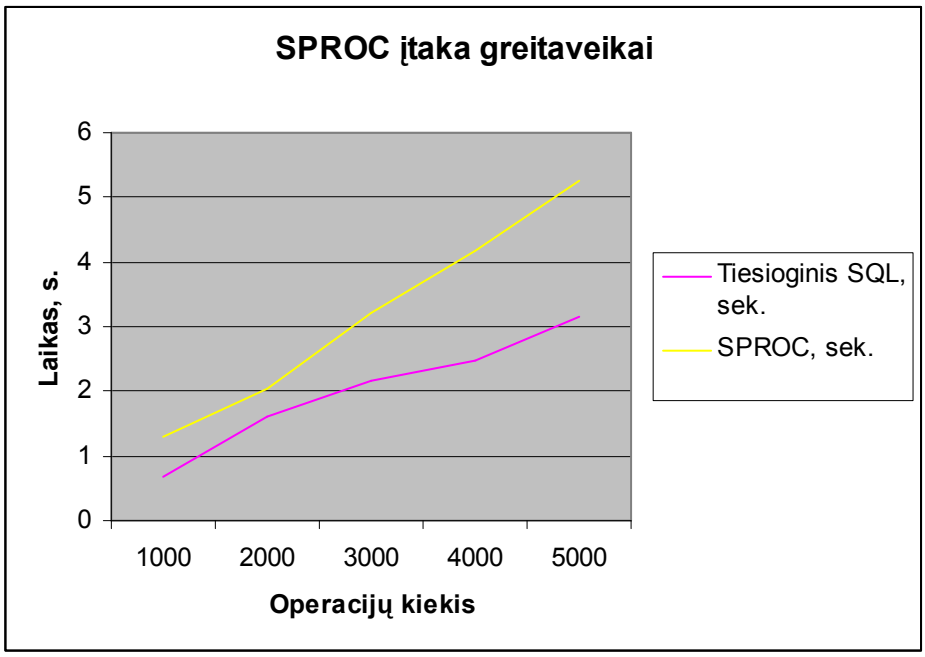
- Apsaugos priemonių nuo SQL injekcijų įtaka Framework‘o greitaveikai (užklausų vykdymo greitaveikos skirtumai, naudojant ir nenaudojant SQL injekcijų tikrinimo priemones);

Tikriname SPROC technologijos ir nesaugaus tiesioginio SQL (angl. direct SQL) greitaveikos skirtumus, atliekant duomenų įterpimo operacijas:

6 lentelė. SPROC įtaka greitaveikai

Operacijų kiekis	Tiesioginis SQL, sek.	SPROC, sek.
1000	0,6875	1,3125

2000	1,5938	2,0469
3000	2,1563	3,2031
4000	2,4844	4,1719
5000	3,1500	5,2656



28 pav. SPROC įtaka greitaveikai

Išvada: Kaip matome iš šio eksperimento rezultatų, SPROC technologija pasižymi mažesne greitaveika, lyginant su tiesioginiu SQL naudojimu. Tai įtakoja SQL užklausų parametrizavimas, bei Framework'e naudojamas automatinis parametrų užpildymas pagal lango duomenis.

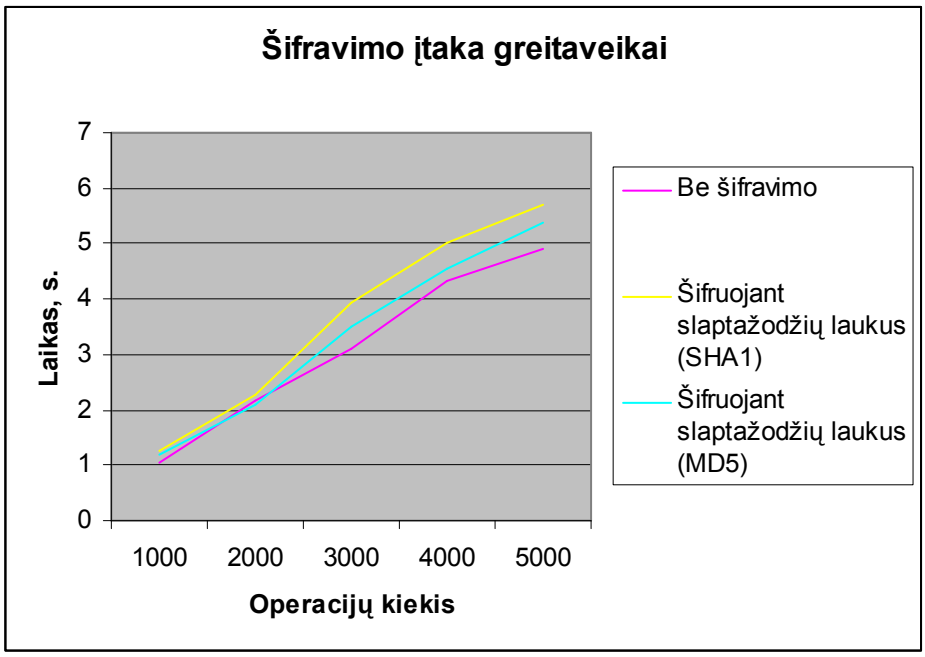
- Šifravimo ir dešifravimo įtaka sistemos (Framework'o ir DB) greitaveikai;

Tiriame šifravimo įtaką greitaveikai. Vienu atveju šifruojame slaptažodžių laukus SHA1 arba MD5 algoritmu, kitu atveju paliekame atvirojo formato. Atliekame duomenų įterpimo operacijas į lentelę, turinčią vieną lauką slaptažodžiui saugoti ir keturiolika paprastų nekoduojamų laukų.

7 lentelė. Šifravimo įtaka greitaveikai

Operacijų kiekis	Be šifravimo	Šifruojant slaptažodžių laukus (SHA1)	Šifruojant slaptažodžių laukus (MD5)
1000	1,0625	1,2656	1,1875

2000	2,1719	2,2656	2,0781
3000	3,0938	3,9219	3,5000
4000	4,3438	5,0313	4,5625
5000	4,9063	5,7188	5,3721



29 pav. Šifravimo įtaka greitaveikai

Išvada: Naudojant duomenų šifravimą, sistema papildomai kreipiasi į „CryptographyHelper“ klasę ir papildomai šifruoja slaptažodžio lauką. To pasekoje sumažėja sistemos greitaveika. Sumažėjimas santykinai nedidelis, kadangi šifruojame tik vieną iš DB lentelės stulpelių. Papildomai pastebime, kad SHA1 algoritmas yra šiek tiek lėtesnis, tačiau skirtumas yra menkas ir žymiau pasireiškia tik prie didelio operacijų kiekio. Žinoma, jog SHA1 algoritmas yra saugesnis negu MD5, todėl rekomenduojame naudoti būtent jį.

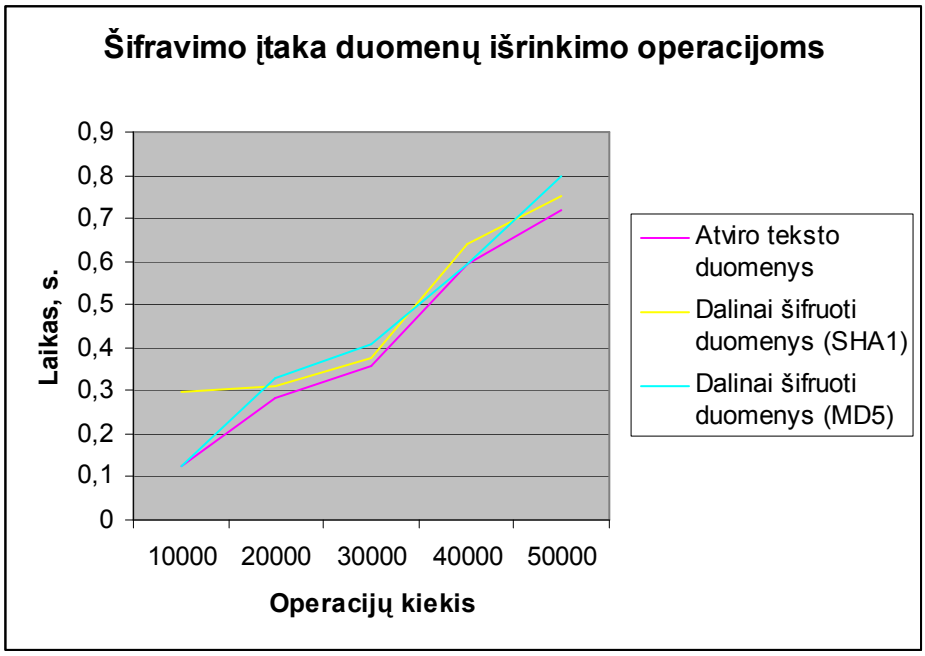
- ❖ Duomenų išrinkimas iš DB

Atliekamas tyrimas, kai vienu atveju duomenys atvirojo teksto, kitu atveju slaptažodžių laukai šifruoti, o likę laukai atvirojo teksto. Kad būtų lengviau pastebėti tendenciją, parinktas didelis operacijų kiekis.

8 lentelė. Šifravimo įtaka duomenų išrinkimo operacijoms

Operacijų kiekis	Atviro teksto duomenys	Dalinai šifruoti duomenys (SHA1)	Dalinai šifruoti duomenys (MD5)

10000	0,1250	0,2969	0,1250
20000	0,2813	0,3125	0,3281
30000	0,3594	0,3750	0,4063
40000	0,5938	0,6404	0,5938
50000	0,7188	0,7500	0,7969



30 pav. Šifravimo įtaka duomenų išrinkimo operacijoms

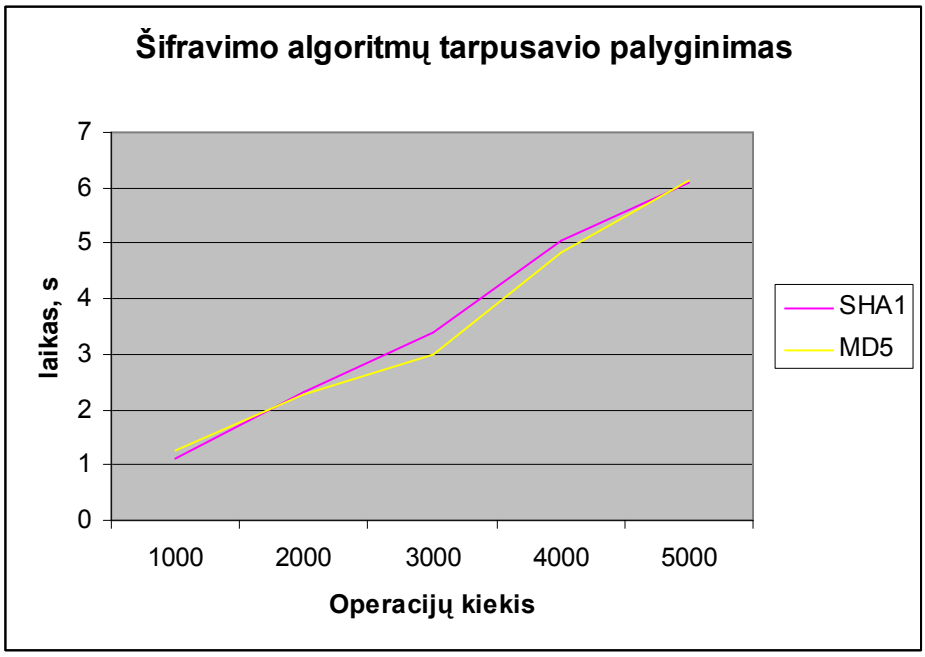
Išvada: Duomenų išrinkimo iš DB eksperimentas gana komplikuotas rezultatų prasme. Kaip matome, prie mažesnio operacijų kiekio, rezultatai nesiskiria. Šiokia tokia tendencija paryškėja tik prie didelio operacijų kiekio. Dalinai šifruotiems duomenims išrinkti reikia šiek tiek daugiau laiko. Tai yra todėl, kad šifruoti duomenys paprastai užima daugiau atminties, nei atvirojo teksto duomenys – čia turima galvoje maišos funkcijos skaičiavimas, kai visi slaptažodžiai paverčiami į fiksuoto ilgio maišos (angl. hash) reikšmę, kuri dažniausiai būna ilgesnė nei pats nešifruotas slaptažodis. Tai priklauso ir nuo to, kaip šifruojame duomenis, t.y. nuo šifravimo parametrų: rakto ilgio, algoritmo, „druskos“ reikšmės (angl. salt-value), nustatyto iteracijų kiekio ir pradinio vektoriaus reikšmės.

- Šifravimo algoritmų palyginimas

Šifravimo algoritmai lyginami atliekant įterpimo operacijas. Lyginame SHA1 ir MD5 metodus.

9 lentelė. Šifravimo algoritmų tarpusavio palyginimas

Operacijų kiekis	SHA1	MD5
1000	1,1250	1,2500
2000	2,3125	2,2656
3000	3,4063	3,0000
4000	5,0625	4,8281
5000	6,1093	6,1406



31 pav. Šifravimo algoritmų tarpusavio palyginimas

Išvada: Alikę šifravimo algoritmų palyginimą, pastebime, kad bendruoju atveju SHA1 algoritmas šiek tiek lėtesnis nei MD5. Norint patikrinti algoritmo įtaką greitaveikai, buvo parinkti gana dideli įrašų kiekiai. Realiame sistemos panaudojime apčiuopiamo skirtumo nepajustume, todėl visgi rekomenduojama naudoti SHA1 algoritmą, kuris yra saugesnis lyginant su MD5.

- ❖ Algoritme naudojamo šifravimo rakto ilgio įtaka

Šiame bandyme tiriame būtent rakto ilgio įtaką, kai naudojamas SHA1 algoritmas, tuo tarpu kai kiti algoritmo parametrai, tokie, kaip iteracijų kiekis ir pradinis vektorius) yra fiksuoti.

10 lentelė. Šifravimo rakto ilgio įtaka greitaveikai

Operacijų kiekis	128 bitai	256 bitai

1000	1,0434	1,1259
2000	2,0313	3,0625
3000	3,7813	3,8906
4000	4,7344	4,7813
5000	6,0625	6,1563

32 pav. Šifravimo rakto ilgio įtaka greitaveikai

Išvada: Rakto ilgis algoritme – vienas iš svarbiausių parametrų. Visada, jei tik yra galimybė, rekomenduojama naudoti ilgesnį raktą ir taip pasiekti stipresnį šifravimo lygį. Bandymo rezultatai rodo, kad padvigubinus rakto ilgį, operacijų greitaveika sumažėja nežymiai, todėl esant galimybei visada geriau naudoti saugesnį (ilgesnio rakto) variantą.

5.2. Konfigūracinių failų palyginimas

Palyginkime įprastinio .NET konfigūracinio failo („app.config“) ir sukurtoje sistemoje panaudoto šifruoto konfigūracinio failo skirtumus:

- Įprastas .NET konfigūracinis failas:

```
<?xml version='1.0' encoding='utf-8'?>
<configuration>
  <configSections>
  </configSections>
```

```
  <connectionStrings>
    <add
name="XSoftArt.AppEngine.Properties.Settings.AppEngineConnectionString"
      connectionString="Data Source=SERVER-RBX;Initial
Catalog=Ranbaxy;Integrated Security=True"
      providerName="System.Data.SqlClient" />
  </connectionStrings>
  <appSettings>
    <add key="ConnectionString" value="Data Source=VYCEK-
D1A6BBED2\SQLEXPRESS;Initial Catalog=Ranbaxy;Integrated Security=SSPI;" />
    <add key="ImageFilter" value="Images
(*.BMP;*.JPG;*.GIF;*.PNG)|*.BMP;*.JPG;*.GIF;*.PNG" />
    <add key="DocFilter" value="Documents (*.DOC;*.XLS)|*.DOC;*.XLS" />
    <add key="DocFilterExtended" value="All documents
(*.JPG;*.GIF;*.PNG;*.JPEG;*.DOC;*.XLS;*.DOCX;*.XLSX;*.RTF;*.PDF;*.TIF)|*.JP
G;*.GIF;*.PNG;*.JPEG;*.DOC;*.XLS;*.DOCX;*.XLSX;*.RTF;*.PDF;*.TIF" />
    <add key="FirstRun" value="2" />
    <add key="ResponsibleForRanbaxyProducts" value="vytas@terrait.net" />
  </appSettings>
</configuration>
```

- Šifruotas konfigūracinis failas:

```
<?xml version='1.0' encoding='utf-8'?>
<configuration>
  <configSections>
  </configSections>
  <appSettings
configProtectionProvider="RsaProtectedConfigurationProvider">
    <EncryptedData Type="http://www.w3.org/2001/04/xmlenc#Element"
      xmlns="http://www.w3.org/2001/04/xmlenc#">
      <EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc" />
      <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
        <EncryptedKey xmlns="http://www.w3.org/2001/04/xmlenc#">
          <EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5" />
          <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
            <KeyName>Rsa Key</KeyName>
          </KeyInfo>
          <CipherData>
```

```
<CipherValue>a7JX2kDUtfPoXYGbJdVj5E8FLFwTPBBoRfxty2HxeD87UU0nyD1Li6giDH6KhH
MnuslCCq+aOLC5kFu56mZ38mhS+5gmzIzlsC85aMV1hMU6yFIMLz/FQBReTTCGDU4gMa7didn37
P67+tFPfSPUdnGgRuJhrLRVbU+AN0Q/mgY=</CipherValue>
          </CipherData>
        </EncryptedKey>
      </KeyInfo>
      <CipherData>

<CipherValue>34WB+AtPsAVfy71hDtBiynQHSJ+UqSjFvj728p5lu0aWsLQ0TI6ujzb2abEy/g
5w2u63165gD/BLqjbXpd+2ep8B77WRan4/kqz2vr3ycaJfw2Fxw27wXKnVJPeIqYiTxMqAmFFg2
Mz42cQFK6HmYoNpEYkyg6O40Ai6MuvVRTHKo3RfeCf/njuoGjCl1oQQ75IyQx7Dj4TChztOXizN
8bAGKhjNkR1ls4glvdwNvzXZxpz/BwpKyq/ABZCuAFLEsmLjDWrgUFbxK8tmGKWCiS9IuzHbDRY
XQMN7LfvIVRF2pkZIWisYbnlsl5Ggpf1p8+6C4G94ocR/NyEyAR2mwVkrrcHqepI268L9de8hnA
yvtXv7RwBrtzYqKS00BhB5TCOC5COcNXZr8cWoeYroLuEhCj7Bce7h7zfYAVMo2RBYZOYYhqKTO
c1oQc+G5S5Okot86uQKyDWWFKiVhxqGbG0mLonSKwTl7v8HjgYouMrALYuly7uaAaS4zgwTPixH
bpnFsOhoqjlEWhYF0xO0nsElGLTEoARPaT+lOenQIXsqCblYAdhxy+ZgpRzQ9i8eLfPx9Q26JDy
cc84rRWgwA62O3Ld/7KlbGheqTl/IP2prgunKtJhU25Pa/a9+661KNKTOCd6XAw8yJQA4WxjP9y
LSvl8/MMW40F8pNLJksY47Z22+d3gkuaNvERxVLHUt11XbR5R9J+SGoE+I8phXeAffXneUlZtrK
+pRQvmH6CBVCJhpMREBTiUatZ9bYFqt</CipherValue>
      </CipherData>
    </EncryptedData>
  </appSettings>
</configuration>
```

Išvada: Įprastinis .NET konfigūracinis failas – didelė grėsmė visai sistemai, ypač duomenų bazei. Visa jame saugoma informacija yra nešifruota, todėl įsilaužėlis gali nuskaityti daugelį svarbių sistemos parametrų. Patobulintoje Framework sistemoje įgyvendintas šifruoto konfigūracinio failo variantas, kuris naudoja viešojo rakto technologiją. Toks konfigūracinis failas yra saugus ir apsaugo sistemą nuo galimų grėsmių, susijusių su sistemos konfigūracijos parametrais [5, 6].

5.3. Saugumo priemonių naudojimo tyrimas

11 lentelė. Saugumo priemonių patogumo palyginimas

Saugumo priemonė	**Įprastinis atvejis**	**Patobulinta Framework sistema**
Sisteminis vartotojas	Naudotis sistema gali bet	Sistemos naudojimui būtinas

	kuris asmuo, turintis prieigą prie kompiuterio, kuriame įdiegta sistema.	bent vienas numatytasis šio tipo vartotojas, turintis prisijungimo vardą, slaptažodį ir pilnas teises sistemoje. Šio vartotojo veiksmų pasekoje gali būti sukuriami kiti sistemos vartotojai ar vartotojų grupės. Sisteminis vartotojas – numatytasis saugios Framework sistemos elementas.
Sisteminė vartotojų grupė	Vartotojų grupių politikos nebuvimas.	Sistemos naudojimui būtina sisteminė vartotojų grupė, turinti bent vieną vartotoją. Patobulinta Framework sistema turi numatytąją grupę, kuriai priklauso sisteminis vartotojas su pilnomis teisėmis. Šio vartotojo veiksmų pasekoje gali būti sukurtos kitos grupės ir atliktas vartotojų priskyrimas joms.
Sistemos vartotojų teisės	Vartotojų teisių nebuvimas – pilna prieiga prie bet kokių sistemos resursų.	Sistemos vartotojų teisės gali būti priskirtos langų lygyje. Tam tikrais atvejais, teisės gali būti priskirtos ir lango komponentų lygyje, jei komponentas numato šią galimybę. Vartotojų teisių sukūrimas yra automatizuota lango sukūrimo dalis. Pagal nutylėjimą sukuriamos „nulinės“ teisės (nieko

		neleidžiančios). Teisių priskyrimas langui ar jo komponentui galimas tik sistemos kūrimo etape. Vėliau galimas tik priskirtos teisės redagavimas, pasinaudojant sukurtos sistemos grafine vartotojo sąsaja.
Turinio pateikimas	Bendra (nekonfigūruojama) sąsaja.	Konfigūruojama sąsaja. Kiekvienas sistemos vartotojas turi (susikuria) sau patogų atvaizduojamų laukų išdėstymą. Teisėmis paremtas (adaptuotas pagal vartotojo teises) meniu langas – vartotojas gali naudotis tik tais meniu punktais, kurių iškviečiamuose languose turi bent skaitymo teisę.
Kriptografija	Informacijos saugojimas atviro teksto formatu.	Integruota kriptografijos užduotims skirta bazinė klasė.
Prisijungusių vartotojų kontrolė	Priklauso nuo konkrečios realizacijos.	Integruotas mechanizmas, administratoriaus teises turinčiam vartotojui įgalinantis matyti kitus einamu momentu prie sistemos prisijungusius vartotojus.
Autentifikacija	Priklauso nuo konkrečios realizacijos.	Integruotas universalus autentifikavimo mechanizmas
Autorizacija	Priklauso nuo konkrečios	Integruotas universalus

	realizacijos.	autorizacijos mechanizmas
Saugus sąveikos su DB mechanizmas	Mišrios technologijos, tiesioginis SQL, neuniversalios funkcijos ir metodai.	Integruotas mechanizmas, užtikrinantis SPROC technologija paremtą universalių funkcijų ir metodų rinkinį, palengvinantį sistemos kūrėjo darbą.

Išvados: Įprastinės sistemos dažniausiai kuriamos vienam konkrečiam atvejui, orientuojantis visų pirma į sistemos funkcionalumą. Saugumo aspektas dažniausiai būna pamirštamas. Tokios sistemos būna „vienkartinės“ ir bet koks kitos sistemos kūrimas prasideda vėl nuo nulio. Patobulinta Framework sistema visų pirma orientuota į saugumą ir universalumą, pritaikomumą daugeliui taikomųjų uždavinių. Bazinių klasių, universalių funkcijų ir metodų panaudojimas įgalina sistemų kūrėjus žymiai sparčiau ir patogiau sukurti galutinę sistemą, o į saugumą orientuota architektūra tuo pačiu padaro sistemą atsparia prieš saugumą nukreiptoms atakoms, tokioms kaip SQL injekcijos. Visu lentelėje aprašytų dedamųjų sumoje gauname, kad sistemos kūrėjui kurti verslo sistemą yra žymiai patogiau ir greičiau, nei įprastiniu atveju.

5.4. Standartinės sistemos palyginimas su mūsų patobulinta Framework programinės konstrukcijos sistema

12 lentelė. Sistemų palyginimas

Standartinė sistema	Patobulinta Framework programinės konstrukcijos pagrindu sukurta sistema
Mišri sistemos realizacija	SPROC metodo pagrindu paremta visos sistemos realizacija
Informacijos saugojimas duomenų bazėje atviro teksto formatu	Informacijos saugojimas duomenų bazėje šifruotu formatu
Įprastas .NET programos konfigūracinis failas (nešifruotas)	Šifruoto .NET konfigūracinio failo realizacija
Atsaini vartotojų ir jų grupių politika arba jos nebuvimas	Sistema, paremta vartotojų ir jų grupių politika, kur tiek grupėms, tiek ir vartotojams suteikiamos tam tikros teisės, kurios kaip ir patys vartotojai ar jų grupės, gali būti

	koreguojamos
Įprastinė įvesties ratifikacija	Perdengtais komponentais ir ratifikacijos tipais paremta automatinė ratifikacija (angl. validation)
„Įsiūta“ atvaizduojama (pateikiama vartotojui) informacija	Matomos informacijos koregavimas, galimybė pritaikyti kiekvienam vartotojui individualiai
Individualus (rankinis) duomenų reikšmių priskyrimas langų komponentams	Automatinis duomenų reikšmių užkrovimas visiems lango komponentams
Įterpimo, redagavimo, duomenų išrinkimo ir kitų procedūrų funkcionalumo kūrimas kiekvienam langui	Automatizuotas įterpimo, redagavimo, duomenų išrinkimo ir kitų procedūrų funkcionalumas, paremtas bazinėmis klasėmis ir metodais (paveldimumas)
Ribotos konfigūracijos grafinė vartotojo sąsaja	Kiekvieno vartotojo individualiai konfigūruojama sąsaja

Išvados: Framework programinės konstrukcijos principu sukurta sistemos pasižymi savo universalumu, aukšto lygio bazinėmis klasėmis ir metodais, leidžiančiais iki maksimumo padidinti sistemos kūrimo efektyvumą, lankstumą ir sumažinti kūrimo laiko sąnaudas. Tam tikrais išskirtiniais atvejais, sistemos funkcionalumas neatitinka jokio veikimo šablono, todėl didelė dalis bazinių metodų yra virtualūs, t.y. esant reikalui juos galima perdengti, pritaikant konkrečiam atvejui.

Saugumo atžvilgiu sistema suprojektuota pagal SPROC modelio taisykles, kurių dėka didėja sistemos saugumo lygis, atsparumas SQL injekcijoms. SPROC taisyklių pritaikymas bazinėse (tėvinėse) klasėse užtikrina, kad ir „vaikų“ klasės, naudojančios „tėvinius“ metodus, bus taip pat aukšto saugumo lygio.

6. IŠVADOS

Šiame darbe analizuotas organizacijų sistemų saugumas, kurio tikslas – užtikrinti taikomosios programos, sukurtos Framework pagrindu, ir duomenų bazės saugą. Analizės metu nustatyta, kad tokio tipo sistemos turi ribotas saugumo galimybes dėl įvairių veiksnių, kuriuos įtakoja Framework konstrukcija ir naudojama duomenų bazės sistema. Taip pat peržvelgti galimi būdai, kurių dėka būtų galima padidinti kuriamų Framework sistemų saugumą.

Remiantis analizės išvadomis iškelti reikalavimai Framework programinei konstrukcijai, sukurtas tokios sistemos projektas, numatantis tiek esamų Framework dalių atnaujinimą ar perdarymą, tiek ir naujas sistemos sudedamąsias dalis.

Realizacijos metu įgyvendintas šifravimas; vartotojų ir jų grupių bei teisių politika; matomų laukų politika ir grafinės vartotojo sąsajos adaptacija pagal teises; priemonės, kurios iki minimumo sumažina SQL injekcijos galimybę. Minėtų priemonių kompleksas, įgyvendintas Framework sistemoje, suteikia plačias galimybes realizuojant saugias verslo sistemas.

Eksperimento būdu atliktas Framework programinės konstrukcijos realizuotų elementų tyrimas, įvertinant poveikį greitaveikai, lyginant įprastinę ir patobulintą Framework sistemą patogumo, bei realizacijos aspektais. Nustatyta, kad įdiegtos saugumo priemonės įprastinio sistemos naudojimo atveju praktiškai neįtakoja sistemos greitaveikos. Skirtumas įžvelgiamas tik simuliuojant operacijas tūkstančių arba dešimties tūkstančių lygmenyje. Sistemos patogumo ir realizacijos aspektų tyrimas parodė, kad Framework pagrindu sukurta patobulinta sistema įgalina pakelti kuriamų sistemų saugumą ir pritaikomumą bei sumažina laiko sąnaudas sistemos kūrimo procese.

Šio darbo metu sukurti sprendimai įdiegti plėtojant „Ranbaxy“ verslo valdymo sistemą.

LITERATŪRA

1. BASS, L.; CLEMENTS, P.; KAZMAN, R. *Software Architecture in Practice, Second Edition*. Addison Wesley, ISBN: 0-321-15495-9.
2. BUCHMANN, D. *Basic Security: Java vs .NET* : Seminaro medžiaga [interaktyvus]. [Žiūrėta 2009-]. Prieiga per internetą: <http://diuf.unifr.ch/softeng/seminars/SE2003/buchmann/htmlpaper/security.html>
3. CHEN, X. *Developing Application Frameworks in .NET*. Apress, 2004, ISBN:1590592883.
4. *CRM Vendors High Level Feature Comparisons* [Interaktyvus]. [Žiūrėta 2009-]. Prieiga per internetą: <http://www.info-sourcing.com/Comparison_CRM_Enterprise_Softwares.htm>
5. *Encrypt ConnectionStrings in App.config* [interaktyvus]. [Žiūrėta 2009-]. Prieiga per internetą: <http://davidhayden.com/blog/dave/archive/2006/03/14/2883.aspx>
6. *Encrypting connection stringa in app.config II* [interaktyvus]. [Žiūrėta 2009-]. Prieiga per internetą: <http://www.nickfessel.com/index.php?post=13>
7. FREEMAN, A.; JONES, A. *Programing .NET security*. O'Reily, 2003, ISBN: 0-596-00442-7.
8. GETHLAND, J. *NHibernate* [interaktyvus]. [Žiūrėta 2009-]. Prieiga per internetą: <http://www.theserverside.net/tt/articles/showarticle.tss?id=NHibernate>
9. *Your first NHibernate based Application* [interaktyvus]. [Žiūrėta 2009-]. Prieiga per internetą: <http://nhforge.org/wikis/howtonh/your-first-nhibernate-based-application.aspx>
10. LITWIN, P. *Stop SQL Injection Attacks Before They Stop You* [interaktyvus]. [Žiūrėta 2009-]. Prieiga per internetą: <http://msdn.microsoft.com/en-us/magazine/cc163917.aspx>
11. MEIER, J.D, et al. *Building Secure ASP.NET Applications: Authentication, Authorization, and Secure Communication* [interaktyvus]. [Žiūrėta 2009-]. Prieiga per internetą: <http://msdn.microsoft.com/en-us/library/aa302415.aspx>
12. *Microsoft Dynamics CRM* [interaktyvus]. [Žiūrėta 2009-]. Prieiga per internetą: <http://crm.dynamics.com/technology/crm-technology.aspx>
13. MISTRY, R.; AMARIS, C.; MINTY, A. *Microsoft® SQL Server 2005 Management and Administration*. SAMS, ISBN-13: 978-0-672-32956-2. 469-482p.
14. PAUL, N.; EVANS, D. *Comparing Java and .NET Security: Lessons Learned and Missed* [Interaktyvus]. Computers & Security, Volume 25, Issue 5, July 2006 [Žiūrėta

2009-]. Prieiga per internetą: <http://www.cs.virginia.edu/~nrp3d/papers/computers_and_security-net-java.pdf>

15. PETERSON, E.; LI, S. *An Overview of Cryptographic Systems and Encrypting Database Data* [interaktyvus]. [Žiūrėta 2009-]. Prieiga per internetą: <http://aspnet.4guysfromrolla.com/articles/021407-1.aspx>
16. ROBINSON, S, et al. *Professional C#, 3rd Edition*. Wiley, 2004, ISBN: 0-7645-5759-9, 415-420 p.
17. *SQL Server 2005 Features Comparison* [interaktyvus]. [Žiūrėta 2009-]. Prieiga per internetą: <http://www.microsoft.com/Sqlserver/2005/en/us/compare-features.aspx>
18. WALTERS, R, et al. *Accelerated SQL Server 2008*. Appress, 2008, ISBN-13 (pbk): 978-1-59059-969-3. 3-9p.

PRIEDAI

1. Priedas. Darbo rezultatų panaudojimo aktas

Akte nurodoma, kad šio darbo rezultatai įdiegti „Ranbaxy“ verslo valdymo sistemoje. Aktas įsegtas darbo gale.

AKTAS

2010-04-30

Kaunas

Vyto Sinkevičiaus magistrinio darbo „Saugos priemonių panaudojimo tyrimas ir pritaikymas Framework programinėms konstrukcijoms“ rezultatai įdiegti plėtojant UAB „Terra IT“ produktą – „Ranbaxy“ verslo valdymo sistemą.

UAB „Terra IT“ direktorius Egidijus Grigas

A.V. (Parašas) (V. Pavardė)