KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMACIJOS SISTEMŲ KATEDRA

MOKAMŲJŲ KORTELIŲ AUTORIZAVIMO (REALIAME LAIKE) INTERNETINĖSE ELEKTRONINĖSE PARDUOTUVĖSE SPRENDIMAS

Magistro darbas

Vadovas:……

doc. Bronius Paradauskas

2005 12 17

Autorius:……

IFM-3 gr. studentas

Vigantas Nikontas

2005 12 17

KAUNAS

2006

1. ANOTACIJA

Šiuolaikinių informacinių technologijų vystymosi tendencijos skatina nuolatinį informacinių sistemų sudėtingumo laipsnio kilimą, tad sėkmingam projekto realizavimui projektavimo objektas (informacinė sistema) turi būti adekvačiai aprašytas.

Šiame darbe apžvelgiamos internetinių taikomųjų programų UML (*Unified Modelling Language*) projektavimo priemonės, programavimo technologijų ir reliacinių duomenų bazių valdymo sistemų RDBVS (*Relational Database Management System*) sprendžiamam uždaviniui pasirinkimas.

Praktiniame darbe pateikiamas alternatyvus Lietuvos rinkoje saugios elektroninės komercijos sprendimas, pasižymintis itin aukštu perduodamos informacijos saugumo lygiu. Tai užtikrina naudojamas RSA šifravimo algoritmas, bei skaitmeninis sertifikatas (X509 algoritmas).

Ši informacinė sistema suprojektuota panaudojant CASE (*Computer Aided- Software Engineering*) įrankius. Naudojant objektiškai orientuotą projektavimo metodą bei remiantis greita aplikacijų paruošimo metodologija RAD (*Rapid Application Development*) buvo realizuotas sistemos prototipas panaudojant Microsoft.NET technologijas.

ANNOTATION

The tendencies of the development of the modern information technologies encourage continuous growth of the complexity degree of the information systems, therefore the object of the design (information system), has to be adequately described for the project to be successfully implemented.

The present paper surveys the design tools of the Internet application programming (UML) as well as the selection of programming technologies and relational database management system (RDBVS) for the solution of the given task.

The practical work presents the alternative solution, which is characterized by high safety level of passing information for the save electronic commerce in the Lithuanian market. This was guaranteed by both the RSA cryptographic algorithm and digital certificate (X509 algorithm) used.

The present information system has been designed using CASE tools. The system prototype was realized by using the object oriented design method and following the rapid application development (RAD) while applying Microsoft.NET technologies.

TURINYS

2. TERMINŲ IR SANTRUMPŲ ŽODYNAS

SEIP – saugi elektroninė-internetinė prekyba

IS – informacinė sistema

VAM – *Virtual Authorize Module* – virtualus autorizacijos modulis realizuotas per VB IS

VB – Vilniaus Bankas

CASE – *Computer Aided- Software Engineering* – programinė įranga arba programų paketai, skirti supaprastinti programų sistemų kūrimą ir palaikymą

HTTP – *Hypertext Transfer Protocol* – protokolas, skirtas duomenų perdavimui pasauliniame tinkle

HTTPS – *Secured Hypertext Transfer Protocol* – protokolas, skirtas duomenų perdavimui šifruotu (128 bitų) kanalu

PA – *Use Case* - panaudojimo atvejis (sistemos elgsenos vienetas – vartotojo ir sistemos sąveikų seka, kuri duoda vartotojui reikšmingą rezultatą)

OP – objektinis projektavimas – dalykinės srities konceptų identifikavimas ir jų sistemos organizavimas, po kurio seka realizavimas programavimo kalba

Konceptas –idėja ar sąvoka, kuri taikoma suprantamiems daiktams ar objektams

Objektas – tai, kam tinka konceptas – koncepto egzempliorius

Dalykinė sritis – apribota analizės erdvė, kurioje yra mus dominantis objektų rinkinys

Klasė – konceptas, nusakantis objektų aibę. Tai yra esybės abstrakcija

Komponentas – fiziškai pakeičiama sistemos dalis, kuri suderinama su vienais interfeisais ir realizuoja kitus interfeisus

Interfeisas – operacijų rinkinys, aprašantis paslaugas, kurias teikia klasė arba komponentas

Paketas – tai modelio elementų grupė. Elementai gali būti klasės, paketai ir kiti komponentai

DB – duomenų bazė

DBVS – duomenų bazių valdymo sistema

SQL – *Structured Query Language* – struktūrizuotų užklausų kalba

TCP/IP – *Transmission Control Protocol/Internet Protocol* – standartinis interneto protokolas duomenų perdavimui

UML – *Unified Modelling Language* – veiklos ir programų sistemų projektavimo rezultatų vaizdavimo, specifikavimo, konstravimo ir dokumentavimo kalba

IE – Interneto naršyklė Internet Explorer

OS – operacinė sistema

RSA – šifravimo algoritmas

RAD - *Rapid Application Development* – greito aplikacijų paruošimo metodologija

3. ĮVADAS

Problema

Analizuojant saugios elektroninės internetinės prekybos (*SEIP*) paplitimą Lietuvos rinkoje nustatyta, jog daug kur nenaudojama tikroji elektroninė komercija. Dažnai prekių ir paslaugų tiekėjai apsiriboja tik prekių krepšelių sudarymu. Tuo tarpu, klientui tenka susimokėti grynaisiais pinigais atvykus į kompanijos būstinę ar įgaliotą vietą. Taipogi daugelis organizacijų nurodo savo banko sąskaitų rekvizitus, t.y. klientams pasiūlomas kitas alternatyvus apmokėjimo už paslaugas būdas – pinigų pavedimas per banką. Beabejo, įmonė turi įsitikinti, jog klientas sumokėjo už paslaugas ar prekes, todėl visas apmokėjimo procesas gali užtrukti net iki kelių parų ir tik gavusi išrašą iš banko prekes perduoda klientui.

Prieš metus atsirado naujas spartesnis atsiskaitymo už paslaugas būdas – apmokėjimas SMS žinute. Bet šis apmokėjimo būdas tinkamas tik nedidelėms sumoms apmokėti, ir dažniausiai fiksuoto dydžio.

Paminėti apmokėjimo būdai apsaugo nuo nesankcionuotų duomenų nutekėjimo tretiesiems asmenims – bet tai dar ne viskas ką gali pasiūlyti šiuolaikinės informacinės technologijos.

Darbo tikslas

Išanalizuoti ir pateikti naują ar alternatyvų Lietuvos rinkoje saugios elektroninės komercijos sprendimą. Suprojektuoti šią informacinę sistemą panaudojant CASE įrankius. Sukurti IS prototipą, kuris pademostruotų saugios elektroninės komercijos principus, bei vykdomus procesus.

Projektas yra susijęs su didesniu projektu – neatsiejama apmokėjimo už prekes ar paslaugas bankinėmis mokėjimo kortelėmis dalimi – autorizacijos centro informacine sistema.

Darbo aktualumas

Šiuolaikinių informacinių technologijų vystymosi tendencijos skatina nuolatinį informacinių sistemų sudėtingumo laipsnio kilimą, tad sėkmingam projekto realizavimui

projektavimo objektas (informacinė sistema) turi būti adekvačiai aprašytas. Tam panaudota modeliavimo ir specifikacijų kūrimo kalba (UML).

SEIP IS skirta vykdyti saugius pardavimus pasauliniame interneto tinkle. Tai įgalina naujų nišų atsiradimą pardavinėjant prekes ar paslaugas tiek Lietuvos, tiek užsienio rinkose.

Verslo atžvilgiu panašios informacinės sistemos įdiegimas panašaus profilio įmonėse yra labai aktualus.

Kam skirtas – vartotojai

IS skirta visų pirma įmonėms besiverčiančioms komercija - norinčioms pardavinėti prekes ar paslaugas pasauliniame interneto tinkle. Taipogi pirkėjams – norintiems saugiai, neišeinant iš namų ar darbo vietos, užsisakyti ir apmokėti už paslaugas ar įsigytus pirkinius bankinėmis mokėjimo kortelėmis nesibaiminant, jog informacija nutekės tretiesiems asmenims.

Palyginimas su analogiškais sprendimais, metodais, sistemomis

Užsienyje yra panašių sprendimų tokių kaip PayPal, BIBIT Global Payment Service, WorldPay ir kitų teikiančių *online* autorizacijų paslaugas. Lietuvoje iki šiol nebuvo plėtojama ši paslauga dėl kelių priežasčių – tai bankų nerimas dėl nesankcionuotų įsilaužimų, informacijos nutekėjimo tretiesiems asmenims, dėl nesaugaus interneto tinklo, kuriame įmanoma "perskaityti" perduodamus paketus, ko pasekoje tokia slapta informacija kaip kortelės numeriai, slaptažodžiai ir pan., gali pakliūti tretiesiems asmenims ir kuria pasinaudojant įmanoma atsiskaityti kito asmens sąskaita.

Lietuvoje vieni iš pirmųjų *online* atsiskaitymų iniciatoriai buvo projektas www.klase.lt – kuri pasirinko WorldPay teikiamas paslaugas. Šiame tinklalapyje vartotojai apmokėdavo už forumo paslaugų paketo pratęsimo laiką.

Projektas www.books.lt- tai pirmasis atsiskaitymo už pirkinius banko mokamosiomis kortelėmis sprendimas Lietuvoje. Jame buvo realizuotas virtualus autorizavimo modulis (VAM) su Vilniaus banku. Vertinant šį sprendimą šiandien – galiu pasakyti iš asmeninės praktinės patirties, jog šis sprendimas nėra saugus.

Rinkoje pasirodė jungtinės mokėjimo – lojalumo kortelės, tad artimiausioje ateityje kils poreikis autorizuoti ir atsiskaitymus lojalumo taškais. Lietuvos rinkoje dalinai pirmasis

žengė šį žingsnį projektas www.naminukas.lt, kuriame galima atsiskaityti ne tik mokėjimo kortele, bet ir įmonės lojalumo taškais – šlamučiais.

Savotiškai įdomūs yra www.tiketa.lt ir www.bilietai.lt projektai, kurie teikia bilietų platinimo paslaugas, bet čia tik galima rezervuoti bilietus – pirkėjams visvien tenka nuvykti iki paskirtos vietos ir apsimokėti grynaisiais. Panašias paslaugas teikia ir lošimo bendrovės: www.orakulas.lt.

Taigi galima sakyti, kad Lietuvos rinka pribrendo sekančiam elektroninės komercijos žingsniui.

Pasirinktas realizavimo metodas ir įrankiai

Projektavimui naudojami CASE įrankiai ir unifikuota modeliavimo kalba UML. Grafinės UML diagramos vizualiai atspindi sistemą ir jos architektūrą.

IS programinei realizacijai pasitelkiamas internetas ir jo dinaminės technologijos, galinčios realiu laiku pateikti duomenis įvairiais vartotojui priimtinais formatais- tai Microsoft.NET technologija. Centralizuotai įstaigos informacijos bazei saugoti naudojama reliacinė duomenų bazių valdymo sistema MS SQL 2000.

Informacijos apsikeitimui naudojami SOAP, bei HTTPS protokolai. Elektroninės parduotuvės identifikavimui banko ar autorizacijos centro informacinėje sistemoje naudojami skaitmeniniai sertifikatai (X509 algoritmas).

Dėl aukštų banko saugumo reikalavimų naudojama kriptografija. Duomenų užšifravimui ir iššifravimui naudojamas RSA algoritmas (pasiūlytas 1978 metais Ron Rivest'o, Adi Shamir'o, Leonard Adleman'o).

Gautas rezultatas ir išskirtinės savybės

Ši objektine technologija pagrįsta sistema yra orientuota į verslą. Versle klientų aptarnavimo kokybė ir operatyvumas, taip pat saugus informacijos pateikimas bei gavimas laiku yra vieni svarbiausių veiksnių.

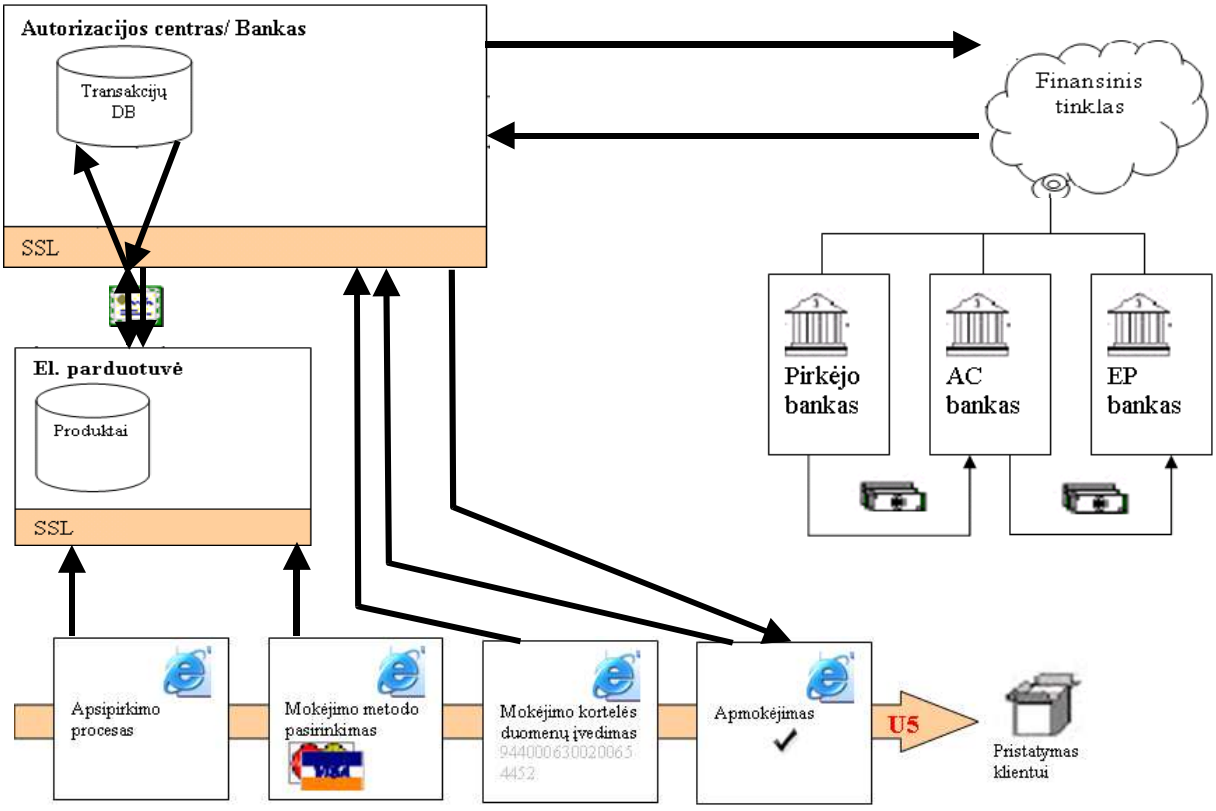
Šis kuriamos sistemos modelis padeda didinti apyvartines lėšas, bei pateikti vartotojams platesnį atsiskaitymų ratą. Pirkėjas bet kuriuo laiku ir bet kurioje vietoje, turėdamas interneto ryšį, gali naudotis šia sistema. Jos pagalba gali užsakyti prekes ar paslaugas būdamas išvykęs.

Kitas svarbus šios informacinės sistemos kriterijus yra perduodamų duomenų saugumas. Internetinės prekybos paslaugas teikianti bendrovė neturi galimybės perskaityti ar apdoroti konfidencialios informacijos, kuri yra žinoma tik klientui ir bankui. Tai įgalina dar labiau didinti šio verslo plėtrą Lietuvoje, užtikrinant, jog konfidenciali informacija nepateks tretiesiems asmenims.

4. ANALIZĖS DALIS

4.1. Analizės tikslas

Išanalizuoti *SEIP* IS vykdomus procesus, klientų ir pardavėjų poreikius, duomenų valdymo galimybes ir šios analizės pagrindu išskirti kompiuterizuojamus procesus, parinkti realizavimo metodus bei priemones.



Atsiskaitymo mokėjimo kortele veiksmų schema

4.2. Problemų ir galimybių analizė

Pagrindinis sprendžiamos problemos uždavinys yra sudėtingas, saugus informacijos perdavimo būdas tarp kliento ir autorizacijos centro ar banko. Sudėtinga komunikacija tarp

pirkėjų ir pardavėjų, o taip pat tarp pirkėjų ir banko priverčia bankus pristabdyti tikrosios elektroninės komercijos plėtrą Lietuvoje.

Galimybė sprendžiant šias problemas yra vienintelė – t.y. naudotis bankų ar autorizacijos centrų teikiama programine įranga, algoritmais. Telieka įmonei pasirinkti per kokią banko IS ar autorizacijos centrą bus autorizuojami mokėjimai ir užsakyti šią paslaugą.

Kuriant šią informacinę sistemą pasirinktos išimtinai Microsoft technologijos. Sistema tokiu atveju bus lankstesnė, patikimesnė. Kuriamą sistemą realizuojant būtina pasitelkti interneto technologijas leidžiančias realiu laiku dinamiškai perduoti informaciją.

Duomenims saugoti būtinai reikalinga DBVS, kuri galėtų kontroliuoti atliekamas operacijas transakcijų lygyje.

4.3. UML projektavimo priemonės

Net ir labai paprastos programinės įrangos realizavimas be projektavimo priemonių gali tapti gana sudėtinga užduotimi. O didelių sistemų įgyvendinimas be jų sunkiai įsivaizduojamas. Modeliavimas padeda lengviau suprasti sistemos detales, susidaryti tikslų jos vaizdą . Modeliavimo pasirinkimas turi didžiulę įtaką sprendžiamos problemos supratimui ir paties sprendimo formai. Internetinės taikomosios programos, kaip ir kitos programinės įrangos sistemos, yra pavaizduojamos pasitelkiant modelius: panaudojimo atvejų *(use cas*e), realizacijos *(implementation mode*l), išdėstymo *(deployment mode*l) ir kt.

Išskirtinis internetinių programų modelis yra tinklalapio žemėlapis, kurį sudaro abstraktus puslapių bei navigacijos kelių, galimų esamoje sistemoje, rinkinys.

Sprendžiant kaip ir ką modeliuoti, labai svarbu nustatyti abstrakcijos ir detalizavimo lygmenį, rasti optimalų variantą sukuriant rezultatą, kuriuo bus patenkinti modelio naudotojai. Paprastai geriausia ir naudingiausia modeliuoti sistemos artefaktus, t.y. tas realaus gyvenimo esybes, kurios bus sukurtos ir kuriomis bus manipuliuojama siekiant realizuoti galutinį produktą – išbaigtą sistemą . Programų tarnybinės stoties vidaus komponentų modeliavimas ar interneto naršyklės detalių modeliavimas neduos laukiamų rezultatų ir nepalengvins sistemos architektų bei projektuotojų darbo. Tinklalapių modeliavimas, nuorodų tarp jų išskyrimas, dinaminio turinio, pateikiamo juos sudarant ir perduodamo klientui analizė yra labai svarbi. Tinklalapiai, nuorodos tarp jų, dinaminio turinio elementai serverio bei kliento pusėje – tai ir yra tie sistemos artefaktai, kuriuos privalu modeliuoti ir analizuoti.

Norint atlikti modeliavimą, reikia sistemos elementus susieti su modelių elementais. Lengviausiai tai padaroma su nuorodomis, kurios automatiškai virsta ryšiais tarp modelio

elementų . Tinklalapiai (interneto puslapiai) gali būti suprantami kaip klasės loginiame modelio pjūvyje. Tinklalapius apibrėžus kaip klases, visos kodo funkcijos, naudojamos juose automatiškai virsta klasių operacijomis. Tinklalapio lygmenyje naudojami programavimo kodo kintamieji yra paverčiami klasės atributais. Problema iškyla kuomet tinklalapyje yra naudojama kodo operacijų aibė, kuri veikia tarnybinės stoties pusėje, formuojant dinaminį tinklalapio turinį, bei visiškai skirtingi kliento pusės kodo elementai (pvz., *JavaScrip*t), valdantys sąsajos elementus ir atliekantys pirminę įeinančių duomenų kontrolę arba dalinį tinklo navigacijos sistemos valdymą. Tai sukelia sumaišti, kadangi vienu atveju turime dar nesuformuotą internetinį puslapį tarnybinės stoties pusėje, kitu atveju – jau kitą aktyvią dalį, kuri yra pateikiama kliento dalyje. Taigi paprastas interneto puslapio susiejimas su UML klase nepadės projektuotojams geriau suprasti kuriamos sistemos.

UML kūrėjai numatė atvejį, kada jų pateikta išbaigta UML semantika nesugebės padengti specifinės nagrinėjamų sistemų objektų bei komponentų aibės. Tam tikslui yra naudojamas formalus UML išplėtimo mechanizmas kuris yra naudojamas standartinės UML semantikos išplėtimui. Šis mechanizmas leidžia nustatyti stereotipus, žymėtą šias reikšmes bei apribojimus, kurie gali būti taikomi modelio elementams.

Stereotipas yra elementas suteikiantis galimybę nustatyti naują semantinę reikšmę modelio elementui. Žymėtosios reikšmės yra pagrindinės reikšmių poros kurios gali būti susietos su modelio elementu taip priskiriant reikšmę šiam elementui. Apribojimus sudaro taisyklės, kurios garantuoja kuriamo modelio teisingumą. Šios taisyklės gali būti išreikštos kaip laisvos formos tekstas arba formalizuotos panaudojant objektų apribojimų kalbą *(OCL – Object Constraint Languag*e).

Bendroji internetinių taikomųjų programų architektūra yra sudaroma analizuojant naršyklių, tinklo bei programų tarnybinės stoties galimybes. Naršyklės „pareikalauja“ tinklalapių iš tarnybinės stoties.

Kiekvieną tokį tinklalapį galima apibūdinti kaip turinio bei jo suformavimo taisyklių rinkinį, išreikštą pasinaudojant HTML. Kai kurie tinklalapiai naudoja kliento pusės programinius kodus, kurie yra interpretuojami naršyklės. Šie kodo elementai suteikia papildomą dinamiką tinklalapio elgsenai bei jo pavaizdavimui, kurį vykdo naršyklė. Dažnai tokie metodai yra naudojami dinamiškai keisti turinio vaizdavimą, papildomas valdyklės (apletus ir kt.) naudojamus internetiniame puslapyje. *Web* aplikacijos naudotojas stebi pateikiamą tinklalapio informaciją ir sąveikauja su sistema pasinaudodamas jam suteikiamais valdymo elementais – informacijos laukais arba nuorodomis į kitus sistemos tinklalapius. Bet kurie vartotojo atlikti veiksmai įtakoja sistemos darbą ir keičia jos veiklos būseną .

Žiūrint į *Web* puslapį iš vartotojo pusės galima teigti, jog tai visuomet bus HTML pagrindu suformuotas dokumentas. Serverio pusėje tas pats puslapis gali keisti savo turinį ir būseną pagal įvairius scenarijus, kurie yra sukuriami pasinaudojant scenarijų rašymo metodologijomis bei kalbomis.

Modeliuojant *Web* tinklalapius kiekvienas interneto puslapis yra susiejamas su UML komponentais.

Komponentu yra laikoma fizinė ir keičiama kuriamos sistemos dalis. Modelio realizacijos pjūvis (komponentų pjūvis) atvaizduoja sistemos komponentus ir ryšius tarp jų. Internetinių taikomųjų programų atveju, šiame pjūvyje yra atvaizduoti visi sistemą sudarantys tinklalapiai ir ryšiai tarp jų (šie ryšiai sudaromi panaudojant nuorodas). Viename lygmenyje *Web* sistemos komponentų diagrama atrodo kaip tinklalapio žemėlapis.

Tinklalapio komponentai atvaizduoja tik fizines sąsajos dalis, jie nėra tinkami vaizduoti bendradarbiavimą bei sąveiką tinklalapio viduje. Toks abstrakcijos lygmuo yra labai svarbus sistemos kūrėjams ir programuotojams ir turi būti modelio dalimi. Kiekvienas *Web* puslapis yra klasė atvaizduojama UML dizaino pjūvyje (loginiame pjūvyje). Atvaizduojant tinklalapyje esančius metodus kaip klasės operacijas, o jų naudojamus kintamuosius kaip klasės atributus yra susiduriama su serverio bei kliento pusės metodų nesuderinamumu. Vienas iš šios problemos sprendimo būdų yra panaudoti stereotipus taip kiekvienam klasės atributui arba operacijai nurodant atitinkamą priklausomybę . Tačiau tuomet modeliavimas iš paprasto ir lengvai suprantamo tampa komplikuotu ir sunkiai suvaldomu.

Daug geresnis šios problemos sprendimas yra „interesų išskaidymas“. Mąstant logiškai, *Web* puslapio elgesys serverio ir kliento pusėje yra visiškai skirtingas. Kai tinklalapis yra apdorojamas serverio pusėje, jam suteikiama prieiga prie serverio resursų (failų sistemos, duomenų bazių ir kt.). Tas pats puslapis sudarytas iš srauto HTML komandų kliento pusėje yra visiškai kitoks, jei nagrinėsime jo galimus veiksmus bei resursų panaudojimą. Kliento pusėje tinklalapis gali naudoti naršyklės resursus ir elementus, pasinaudojant dokumento objektų modelių *(DOM – Document Object Mode*l) ir kiekvienu jame specifikuotų įskiepiu, Java apletų arba ActiveX komponento valdykle. Taipogi galimi ryšiai su kitais tuo
metu aktyviais puslapiais, esančiais kitoje tinklalapio karkaso dalyje arba naršyklės esybėje.

Išskiriant tokius interesus, galima modeliuoti serverio pusės aspektus kaip vieną klasę, o kliento pusės – kaip kitą. Toks išskyrimas į dvi dalis atliekamas pasinaudojant UML stereotipais: serverio tinklalapių ir kliento tinklalapių.

Serverio tinklalapis vaizduoja *Web* puslapį, kuriame yra programinio kodo vykdomo serverio pusėje. Šis programinis kodas sąveikauja su serverio resursais (duomenų bazėmis,

failais ir kt.). Šio stereotipo objektai yra programiniame kode aprašytos funkcijos, o atributai – tinklalapio lygmens globalūs kintamieji.

Kliento tinklapio esybė yra HTML pagrindu suformuotas *Web* tinklalapis. Kaip ir bet kuris kitas tinklalapis, jis yra duomenų, įvykių apdorojimo mechanizmo ir vaizdavimo elementų rinkinys. Šio stereotipo metodai yra kliento dalies funkcijos, išreikštos tinklalapyje, o atributai – tinklalapio lygmens kintamieji. Kliento tinklalapis gali turėti asociacijų tiek su kliento, tiek ir serverio tinklapiais.

Taip sukuriama nauja UML semantika modeliuojamiems elementams. Klasė su nurodytu stereotipu gali būti išskirta panaudojant ikonas.

Build ryšys tarp UML interneto stereotipų

Interneto puslapių atveju, stereotipas paženklina klasę kaip loginės tinklalapio elgsenos abstrakciją serverio arba kliento pusėje. Abi šios klasės yra susiejamos kryptiniu ryšiu, nukreiptu iš serverio pusės tinklalapio į kliento pusės tinklalapį ir suteikiant ryšiui stereotipą <<*build*>>, kadangi serverio dalies tinklalapis suformuoja kliento dalies tinklalapį. Kiekvienas dinaminis tinklalapis (toks tinklalapis, kurio turinys yra suformuojamas serverio pusėje vykdomų aplikacijų) yra konstruojamas kartu su serverio puslapiu. Dažniausiai vienas serverio puslapis suformuoja vieną kliento pusės internetinį tinklalapį, tačiau yra sistemų kuriuose vienas serverio tinklalapis formuoja net kelis kliento dalies puslapius.

Paprasčiausias ryšys tarp tinklalapių yra nuoroda. Nuoroda *Web* aplikacijoje atlieka navigacijos kelio sistemoje vaidmenį. Toks ryšys tarp tinklalapių modeliuose yra vaizduojamas pasinaudojant stereotipu <<*link*>>. Tokio stereotipo asociacija visuomet prasideda kliento dalies tinklalapyje ir rodo į serverio arba kliento dalies internetinį puslapį.

Link ryšys tarp UML interneto stereotipų

Nuorodos yra įdiegiamos į sistemą kaip tinklalapio paklausą, o *Web* puslapiai yra modeliuojami komponentai, kuomet visi jie vaizduojami realizacijos pjūvyje *(Implementation Vie*w). Nuorodos pagrindu sudaromas ryšys tarp serverio ir kliento klasių (su nurodytu stereotipu <<*buil*d>>) ir paprasta nuoroda sudarytas ryšys (su stereotipu <<*lin*k>>) yra ekvivalentai, kadangi tiek vienu, tiek ir antru atveju šie ryšiai yra tinklalapio pareikalavimas.

Žymių reikšmės *(tagged value*s) yra naudojamos aprašyti parametrams, kurie yra perduodami per nuorodas užklausiant internetinio tinklalapio. Prie nuorodos asociacijos pridėta žymių reikšmė yra sąrašas parametrų vardų, kurie yra naudojami formuojant užklaustą internetinį tinklalapį.

Stereotipų panaudojimas labai palengvina internetinių tinklalapių programinių kodų bei jų naudojamų ryšių modeliavimą. Serverio puslapių stereotipo klasės operacijos tampa atitinkamo tinklalapio metodais, o atributai – to tinklalapio lygmens globaliais kintamaisiais. Kliento puslapių operacijos tampa kliento pusėje galimais metodais, o atributai – atitinkamo lygmens kintamaisiais. Taip serverio pusės tinklalapiams yra modeliuojami santykiai su serverio pusės resursais, o kliento pusės – su naršyklės elementais ir Java apletais.

Vienas iš didžiausių privalumų, pastebimų naudojant klasių stereotipus modeliuojant loginę tinklalapio elgseną yra tas, kad jo sąveika ir bendradarbiavimas su serverio pusės komponentais yra išreiškiama taip pat kaip bet kuri kita tik serverio dalies komponentų sąveika. Serverio puslapio stereotipu pažymėta klasė yra tiesiog kita klasė dalyvaujanti sistemos veiklos logikos modeliavime. Daugiau konceptualiame lygmenyje šios serverio lygmens klasės dažniausiai nustatinėja sistemos būsenos valdiklius, taip aktyvuodamos įvairius sistemos veiklos objektus norint sukurti tam tikrą, nuo sistemos einamos būsenos priklausantį, veiklos rezultatą, kurio naršyklės pagalba pareikalavo sistemos naudotojas. Kliento pusėje vykstančių sąveikų ir bendradarbiavimo modeliavimas gali būti šiek tiek komplikuotas. Tai įtakoja plati aibė technologijų, kurios gali būti panaudotos šioje dalyje. Paprasčiausias kliento pusės tinklalapis yra HTML dokumentas, kuris sudarytas iš valdančiojo navigacinio kodo bei turinio informacijos. Naršyklė naudotojui atvaizduoja tinklalapį remdamasi HTML formatavimo instrukcijomis, pateikiamomis jame su kartais panaudojamais vaizdavimo stiliais. Vaizdavimo stilių stereotipas gali būti išskirtas ir suformuota atskira klasė, su priklausomybės ryšiu nuo kliento pusės tinklalapiu. Tačiau šio stereotipo klasės yra išskiriamos gana retai.

Pagrindinis duomenų įvedimo mechanizmas internetiniuose puslapiuose yra forma. Formos yra apibrėžiamos HTML tinklalapiuose panaudojant <*for*m> žymę. Kiekvienai formai yra nurodomas tinklalapis, į kurį yra siunčiami surinkti duomenys. Formą sudaro aibė

informacijos įvesties elementų, visi jie yra išreiškiami HTML žymėmis. Dažniausiai naudojamos žymės yra <*input*>, <*select*> ir <*textarea*>.

HTML žymė <*input*> gali būti daugiareikšmė įvesties elementų formos atžvilgiu, nes ją galima vaizduoti kaip teksto įvedimo lauką, daugybinės arba vienetinės pasirinkties laukas, paslėptas laukas ir kt.

Modeliuojant formas sukuriamas dar vienas klasių stereotipas <<*form*>>. Formos stereotipo klasė neturi operacijų, kadangi bet kokia operacija aprašyta formos žymių viduje yra tinklalapio, o ne formos lygmens.

Formos duomenų įvedimo elementai yra formos stereotipo klasės atributai. Visiems jiems, atsižvelgiant į įvesties tipą, yra priskiriamas atitinkamas stereotipas. Formos stereotipo klasė gali turėti ryšių su apletais, jei šie yra naudojami kaip informacijos įvesties elementai. Kiekviena forma taipogi turi ryšį su serverio dalies tinklalapiu, kuriame yra apdorojami formos surinkti duomenys ir kuris yra nurodytas kaip formos duomenų siuntimo tikslas. Šis ryšys yra parašomas kaip <<*submit*>>.

Submit ryšys tarp UML interneto stereotipų

Viena ar kelios JavaScript bibliotekos gali būti aprašytos kaip <<*JavaScript*>> stereotipo klasės. Jų metodai tampa operacijomis, o globalūs kintamieji – atributais.

Interneto aplikacijose vis dažniau yra imami naudoti karkasai *(frame*s), kurie leidžia vienu metu naudotojui matyti kelis tinklalapius ir šie tinklalapiai gali bendradarbiauti ir sąveikauti tarpusavyje keisdamiesi duomenimis bei aktyvuodami įvairius objektus. Naujausios naršyklių versijos leidžia sąveikauti net skirtinguose naršyklės vienetuose esantiems tinklalapiams. Tai yra vykdoma naudojant dinaminio HTML programinį kodą. Ar naudoti karkasą internetinėje aplikacijoje turi nuspręsti sistemos architektai bei programuotojai.

Jei karkasai yra naudojami, jie turi būti išreikšti ADM *(Application Domain Modelin*g). Modeliuojant karkasų naudojimą atsiranda dar dvi stereotipinės klasės – <<*frameset*>> ir <<*target*>> – bei asociacijos stereotipas <<*target lin*k>>. Karkasų aibės klasė yra tiesiogiai susieta su HTML <*frameset*> žyme. Joje yra nurodomi kliento dalies tinklalapiai. Tikslo klasė ir yra tinklalapis, kuris yra talpinamas karkaso viduje. Jei šie tinklalapiai bendradarbiauja tarpusavyje, tai turi būti sukurtas tikslinės nuorodos ryšys. Kaip

karkaso dalis yra vaizduojama ne tik vienos naršyklės vaizduojami tinklalapiai, bet ir kelių naršyklių tinklalapiai, kurie tarpusavyje yra susiję tam tikrais sąveikos mechanizmais. Prie ryšių tarp karkaso ir jo elementų gali būti nurodomos žymimos reikšmės, kurios nusako eilutę bei stulpelį, į kuriuos yra nukreipiamas vaizduojamasis elementas.

Pasinaudojus visais šiais klasių stereotipais bei asociacijomis tarp jų galima labai detaliai ir išsamiai atlikti internetinių taikomųjų programų modeliavimą, neatsižvelgiant į jų struktūros sudėtingumą ir lankstumą. Toks UML profilis interneto svetainėms projektuoti gali būti naudojamas bet kokio tipo internetinėms aplikacijoms.

4.4. Technologijų analizė

Renkantis kūrimo platformą, svarstyta keletas alternatyvų: PHP, .NET ir Java technologijos. Išstudijavus literatūrą buvo apibendrinti technologijų privalumai ir trūkumai.

	PHP	.NET	Java
Palaikomos kalbos	PHP	Palaiko apie 25 skirtingas programavimo kalbas (*JScript*, *VBScript* ir t. t.)	Java
Programavimo principai	Objektinis arba procedūrinis programavimas, taip pat dažnas abiejų principų mišinys	Priklausomai nuo programavimo kalbos, taikomas ir objektinis, ir procedūrinis programavimo principai	Objektinis programavimas
Priklausomybė nuo platformos	Pritaikyta beveik kiekvienai platformai	Geriausiai pritaikyta „Microsoft“ operacinėms sistemoms	Nepriklausoma
Funkcionalumas	Pasiekiamas per atskirus modulius (grafikos įrankius, susijungimo su DB tvarkykles ir t. t.)	Gaunamas sudėtingo kodo pagalba	Jį sąlygoja pakartotinio panaudojimo komponentai ir žymos
Atviras kodas	✓	Dauguma paketų priklauso „Microsoft“ korporacijai	✓
Patikimumas	Turi nežymių trūkumų, tačiau dėl atviro kodo aptiktos klaidos greitai ir efektyviai pataisomos	Kadangi tai ne atviro kodo produktas, programų paketo klaidų negali pataisyti nepriklausomi programuotojai	Patikima
Sparta	Interpretuojama programavimo kalba, sparta nusileidžia daugumai kompiliuojamųjų	Dauguma paketo programavimo kalbų kompiliuojamos	Tik iš dalies kompiliuojama (programos kodas iš pradžių kompiliuojamas į baitinį kodą, o paskui virtuali mašina tą kodą interpretuoja)
Palaikomumas		Galimybė naudoti daug programavimo kalbų programų palaikymą daro sudėtingu	Jį sąlygoja pakartotinio panaudojimo komponentai

Programavimo technologijų charakteristikos

Galutinis pasirinkimas buvo .NET technologijos dėl spartos, bei artimiausią programavimo technologija pritaikyta Microsoft operacinei sistemai bei MS SQL Server duomenų bazių valdymo sistemai. PHP nusileido dėl silpnesnio objektinio programavimo koncepcijos, didesnio primityvumo.

Kadangi vienas iš reikalavimų yra DBVS kuri atliekamas operacijas galėtų kontroliuoti transakcijų lygyje – tad pasirinkimas susiaurėjo iki dviejų duomenų bazių

valdymo sistemų – tai MS SQL Server, bei Oracle. Šiais metais, taipogi rinkoje, atsirado naujoji MySQL versija kuri palaiko transakcijų kontroliavimą, taigi, galbūt, ateityje bus pereita prie šios DBVS, kaip pigesnė alternatyva DBVS milžinėms (MSSQL, Oracle). Bet šioje pozicijoje Microsoft kompanija žengė žingsnį į priekį pasiūlydama MS SQL Express 2005 Edition už simbolinę 250 dolerių kainą. Tad tenka lyginti kiekvienos iš šių DBVS SQL dialektus.

<table>
<tr><th>Savybės</th><th>T-SQL</th><th>MySQL dialect</th><th>PL/SQL</th></tr>
<tr><td>Požiūriai (Views)</td><td rowspan="3">General Views,
Indexed Views,
Distributed Partitioned Views

AFTER Triggers,
INSTEAD OF Triggers

T-SQL sakiniai</td><td>Nepalaiko</td><td></td></tr>
<tr><td>Trikdžiai (Triggers)</td><td>Nepalaiko</td><td>BEFORE Triggers,
AFTER Triggers,
INSTEAD OF Triggers,
Database Event Triggers</td></tr>
<tr><td>Įsimintos procedūros (Stored Procedures)</td><td>Nepalaiko</td><td rowspan="2">PL/SQL statements,
Java methods,
Third-generation language (3GL) routines</td></tr>
<tr><td>Funkcijos</td><td>Scalar funkcijos
Inline table-valued funkcijos
Multistatement table-valued funkcijos</td><td>C, C++ external libraries</td></tr>
<tr><td>Indeksai</td><td rowspan="2">B-Tree indeksai</td><td></td><td>B-Tree indeksai
Bitmap indeksai
Partitioned indeksai
Function-based indeksai
Domain indeksai</td></tr>
<tr><td>Lentelės</td><td>Relational lentelės
Temporary lentelės</td><td></td><td>Relational lentelės
Object lentelės
Temporary lentelės
Partitioned lentelės
External lentelės
Index organized lentelės</td></tr>
<tr><td>Išorės raktai (Foreign Keys)</td><td>Palaiko</td><td>Palaiko tik InnoDB lentelės</td><td></td></tr>
<tr><td>Žymekliai</td><td>Palaiko</td><td>Nepalaiko</td><td></td></tr>
<tr><td>Masyvai</td><td>Nepalaiko</td><td>Palaiko</td><td>Palaiko</td></tr>
</table>

SQL dialektų charakteristikos

Iš lentelės galime daryti išvadą, jog PL/SQL yra žymiai galingesnis SQL dialektas nei T-SQL, o tuo tarpu MySQL dialektas silpnesnis nei T-SQL.

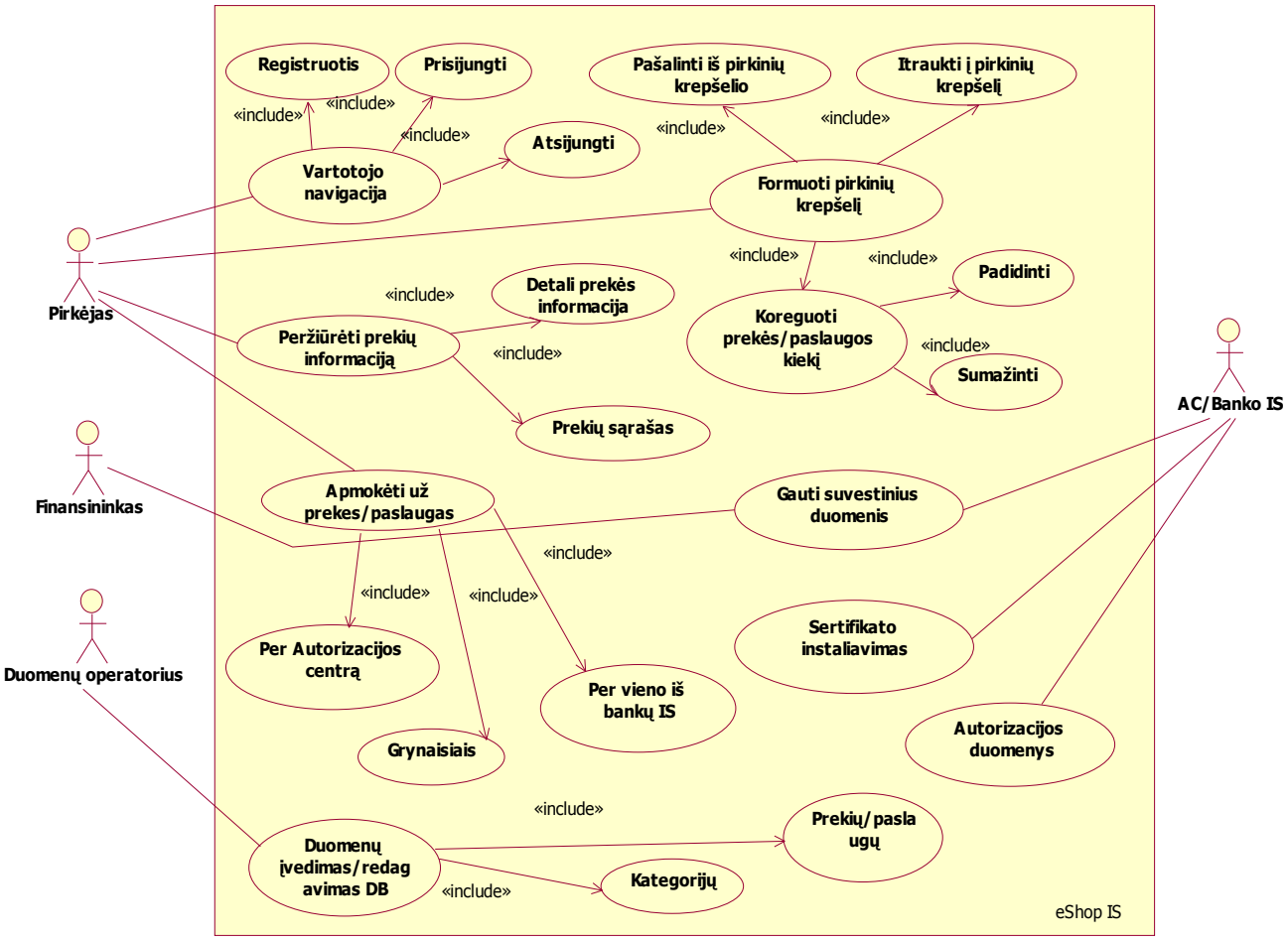
Kadangi kuriamoje sistemoje vyrauja Microsoft technologijos bei MS SQL pigesnis nei Oracle 9, bei MS SQL pirmauja pagal TPC-C testus pasirinkta pastaroji – MS SQL Server 2000 DBVS.

4.5. Panaudojimo atvejų modelis

Panaudojimo atvejų bei sudaryto dalykinės srities klasių modelis yra glaudžiai susiję tarpusavyje ir turi iteracinę prigimtį . Iš esmės visada pradedama nuo veiklos panaudojimo atvejų diagramos, susijusios su išoriniais veiklos procesais.

Išskiriamus panaudojimo atvejus pravartu ne tik registruoti, bet ir juos dokumentuoti, t. y. trumpai aprašyti, sudaryti kiekvieno jų specifikacijas. Nors naudojant UML įrankius tai padaryti galima daugeliu būdų, praktika rodo, kad efektyviausiai panaudojimo atvejų specifikacijos nustatomos ir surenkamos interviu su srities ekspertais (subjektais) metu, sistemos reikalavimų nustatymo etape. Be to, srities ekspertai gali prisidėti ir patys kurdami panaudojimo atvejų modelius. Pagrindinis šio etapo tikslas – surinkti informaciją apie tai, ko reikia išoriniams vykdytojams iš įmonės arba ką įmonė jiems turi padaryti ir kaip veiklos darbuotojai turi bendradarbiauti, kad pasiektų šį tikslą. Kiekvieno tokių pirminių poreikių rinkinio arba išorinio vykdytojo tikslas diagramoje virsta panaudojimo atveju.

Siekiant išvengti neefektyvių sprendimų ateityje, siūloma atlikti panaudojimo atvejų analizę pasikartojančių scenarijų atžvilgiu ir gauti lanksčią, pokyčių atžvilgiu efektyvią architektūrą, kuri leidžia sumažinti projektavimo, kūrimo bei sistemos priežiūros darbų apimtis.



eShopIS panaudojimo atvejų diagrama

4.6. Veiklos procesų diagramos

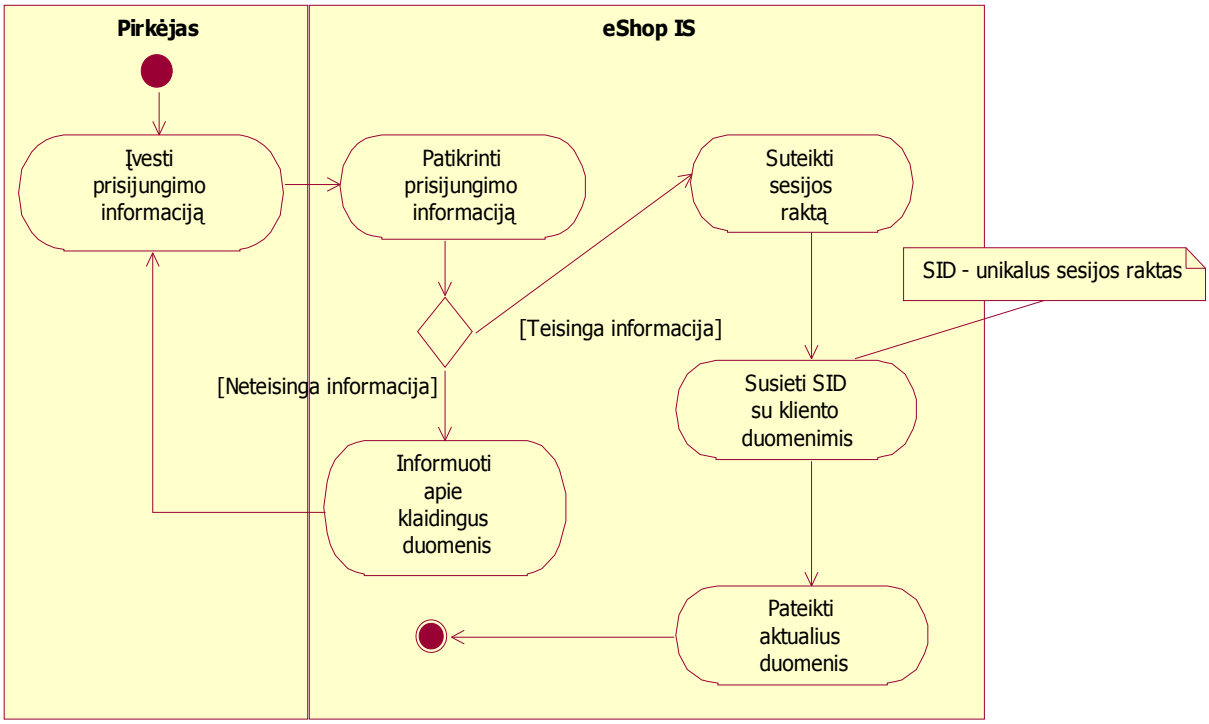
Veiklos diagrama detalizuoja pasirinktą panaudojimo atvejį įvairiais scenarijais. Ji sudaroma iš objektų, veiklų ir įvykių tarpusavio ryšių, kas apima sąveikavimą tarp sistemos ir išorinių vykdytojų .

Svarbu pažymėti, jog veiklos diagramos nėra duomenų srautų diagramos – jos parodo įvykių, o ne duomenų srautą. Šio tipo diagramose kiekvienas takelis atspindi skirtingą išorinį vykdytoją (aktorių) ar veiklos darbuotoją (vidinį aktorių).

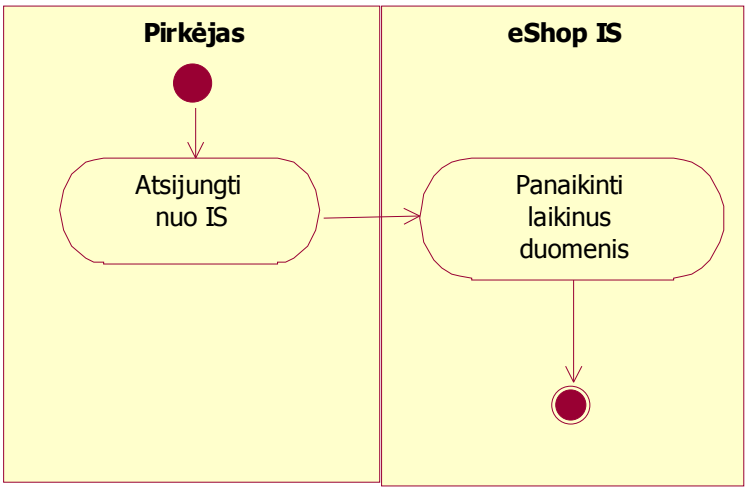
Kaip jau minėta anksčiau, dalykinės srities ir veiklos modeliai yra glaudžiai tarpusavyje susiję.

Todėl tolesniuose projektavimo etapuose papildyti juos taip pat reikia lygiagrečiai, priklausomai nuo informacijos, esančios antrame iš jų. Pvz., veiklos diagramoje esantys veiklos procesai gali būti papildyti elementais, įtakojančiais tuos procesus (pvz., diagrama gali būti papildyta elementais, nurodančiais kur ir kada reikia papildomų dokumentų tam tikrai

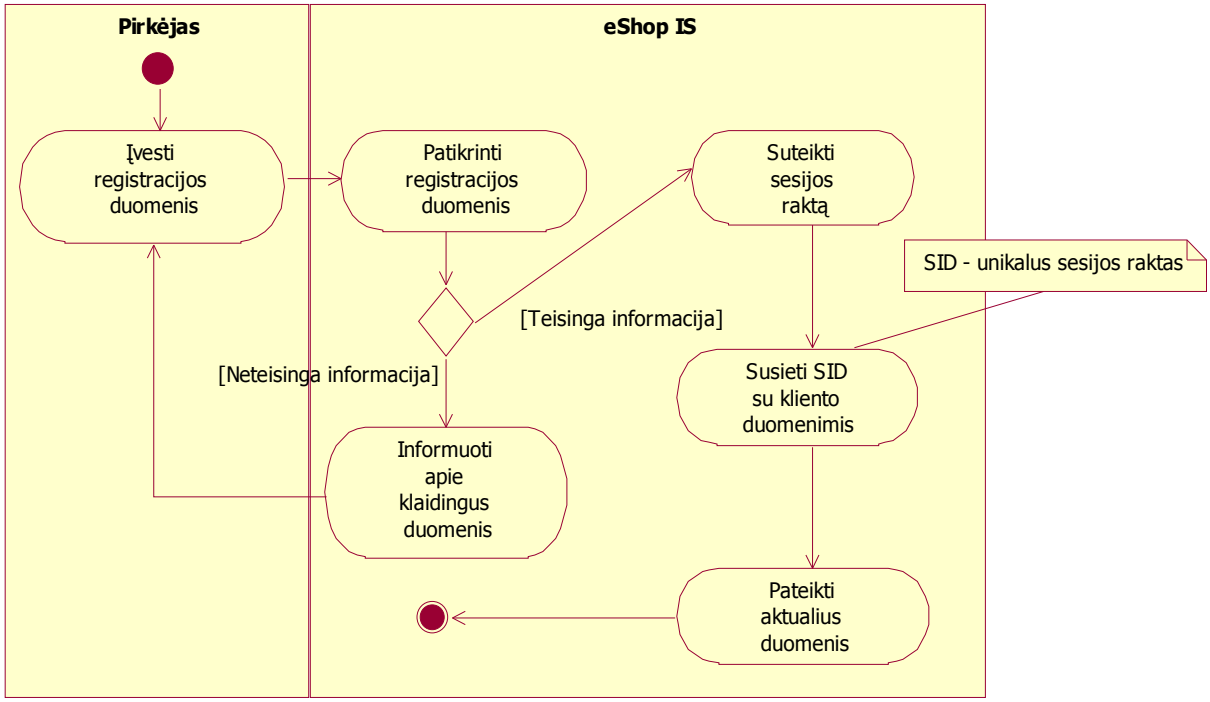
veiklai įvykdyti). Tuomet dalykinės srities klasių modelis taip pat turi būti papildytas atitinkamomis klasėmis (jei jame dar nėra tą objektą aprašančios klasės).



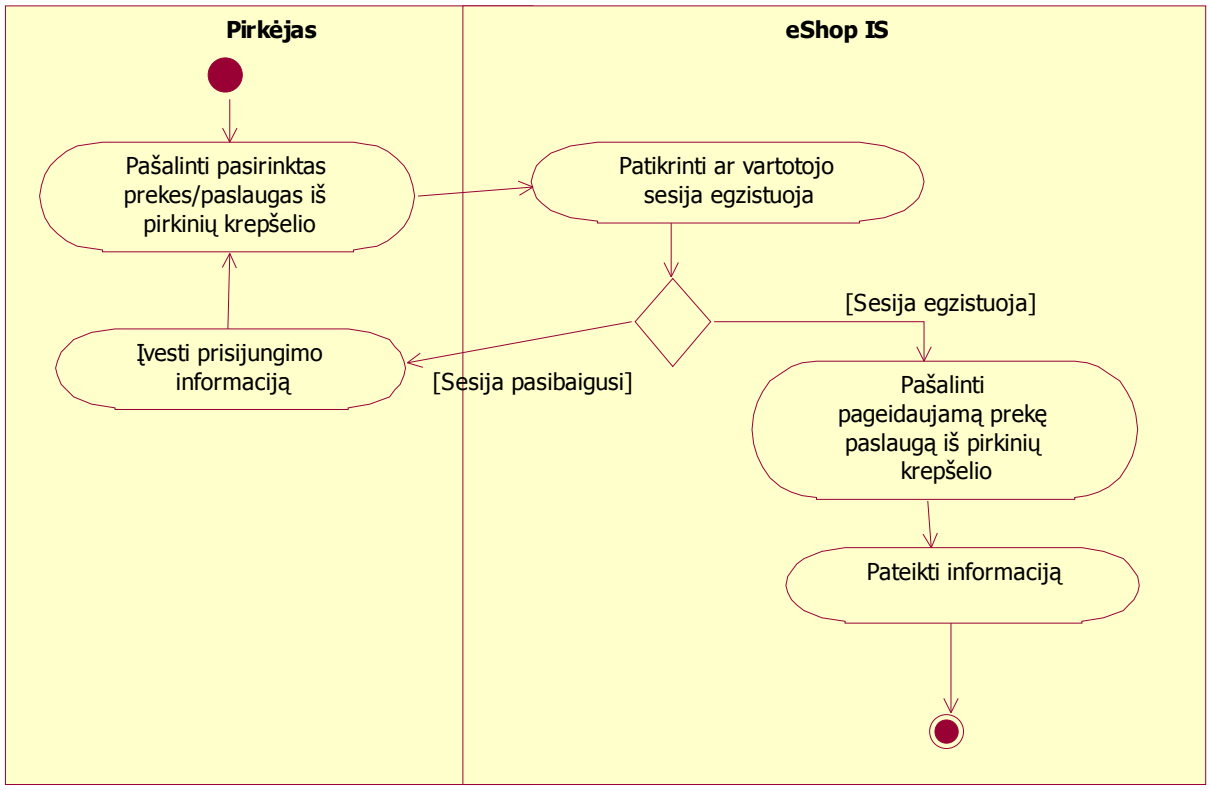
Prisijungimo veiklos diagrama



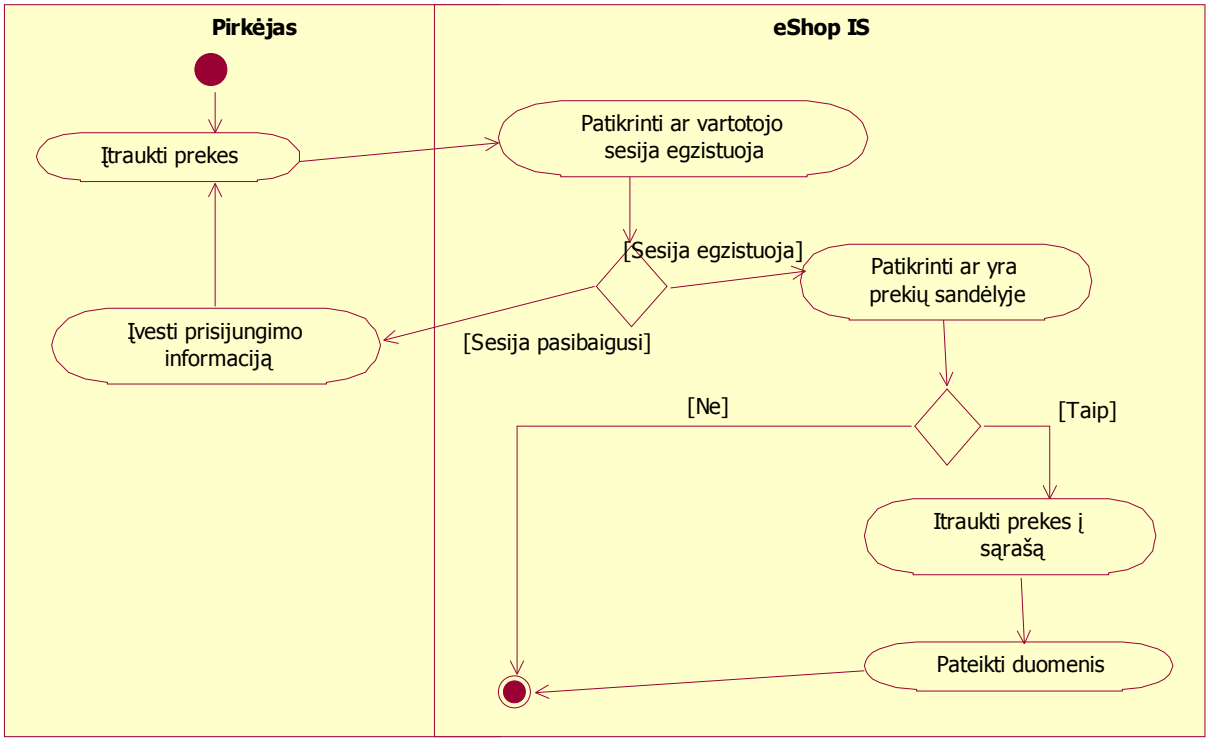
Atsijungimo veiklos diagrama



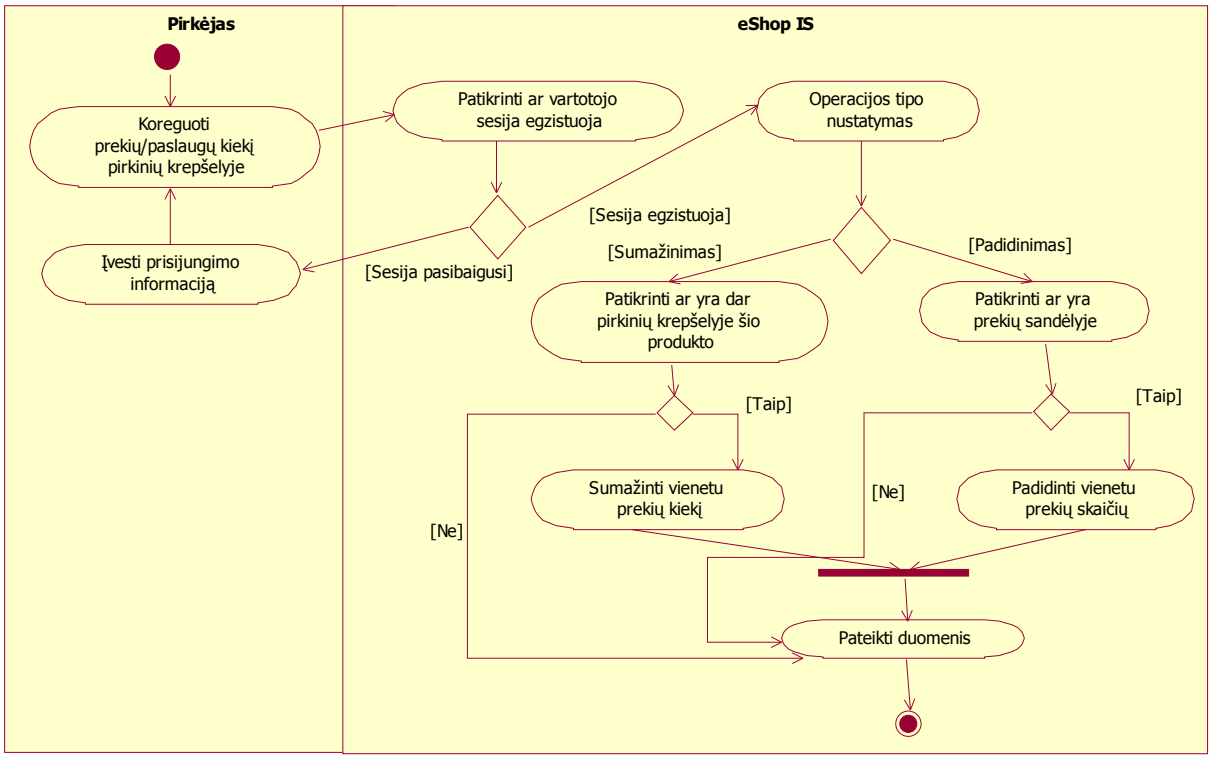
Registravimosi veiklos diagrama



Pašalinti pasirinktas prekes/paslaugas iš pirkinių krepšelio veiklos diagrama



Įtraukti prekes/paslaugas į pirkinių krepšelį veiklos diagrama

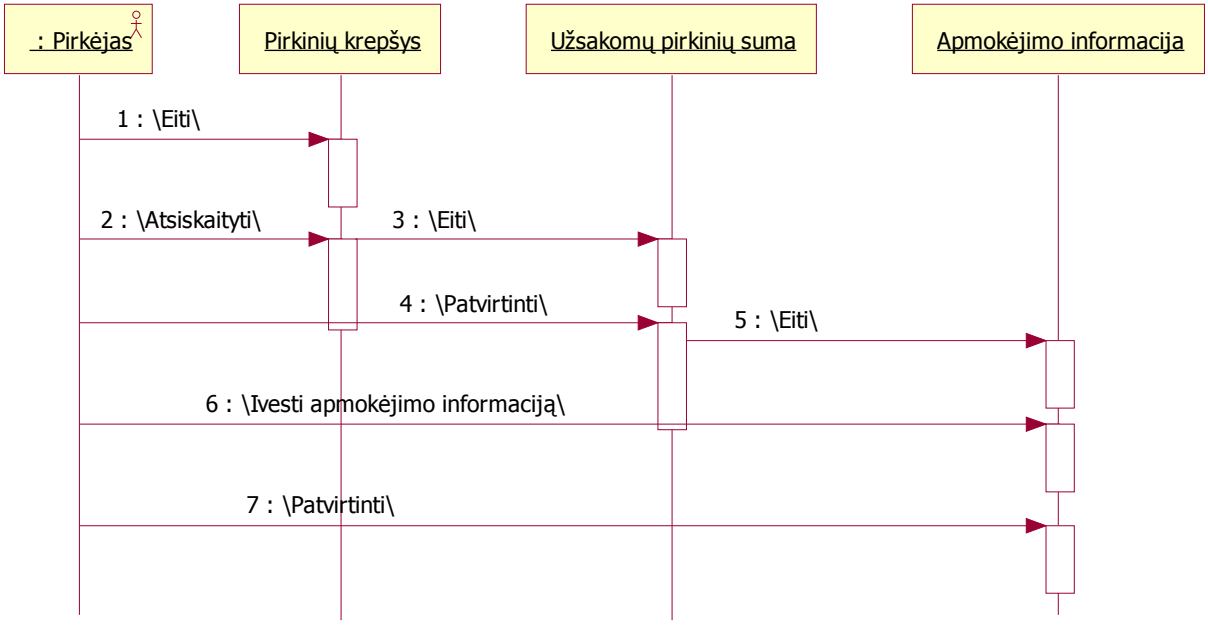


Koreguoti prekes/paslaugas pirkinių krepšelyje veiklos diagrama

4.7. Sąveikos diagramos

Sistemų projektavimo metu sąveikos diagramos gali būti naudojamos daugelyje modelių: veiklos, reikalavimų, analizės, projekto. Jų pagalba aprašomos sąveikos tarp ne tik tarp klasių objektų, bet ir sistemų, posistemių ar programinių komponentų.

Sekų diagramoje galima sukurti naujus, dar neklasifikuotus, objektus arba įtraukti jau egzistuojančių klasių egzempliorius.



Apmokėjimo už prekes sekų diagrama

4.8. Analizės išvados

- ✓ Analizės dalyje išanalizuoti SEIP realizavimo metodai, taikomos programinės priemonės, reikalavimai techninei įrangai, funkcionavimo sričiai. Įvertinti nefunkciniai reikalavimai.

- ✓ Išanalizuota ir specifikuota SEIP dalykinė sritis ir veiklos tikslai. Išnagrinėti ir specifikuoti vartotojų poreikiai.

- ✓ Apžvelgtos programavimo technologijų ir DBVS pagrindinės charakteristikos.

- ✓ Suformuluota techninė užduotis projektavimui. Atlikta galimybių analizė įvertinant sistemos pagrindinius veiksnius: kainą, lankstumą, patikimumą, paprastumą.

5. PROJEKTO DALIS

5.1. Projekto tikslas

Suprojektuoti analizės dalyje išnagrinėtą saugios elektroninės internetinės prekybos informacinę sistemą, taikant CASE priemones ir realizuoti šią sistemą dinaminėmis interneto technologijomis.

5.2. Sistemai keliamų funkcinių ir nefunkcinių reikalavimų modelis

1.1.1. Kompiuterizuojamų panaudojimo atvejų specifikacijos

3.5 skyriuje specifikuotiems pagrindiniams kompiuterizuojamiems panaudojimo atvejams sudarome specifikacijas.

Panaudojimo atvejai	Prisijungti prie sistemos
Aktorius	Pirkėjas
Sistema	SEIP
Prieš sąlyga	Vartotojas turi savo prisijungimo vardą ir slaptažodį
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas įveda sistemos adresą naršyklėje 2. Vartotojas įveda prisijungimo vardą ir slaptažodį	1.1 Sistema pateikia langą prisijungimo vardui ir slaptažodžiui įvesti 1.1 Sistema tikrina vartotojo įvestus duomenis ir gražina teisingą atsakymą 1.2 Sistema atidaro pagrindinį SEIP langą bei išveda duomenis apie vartotoją, bei aktyvuoja pirkinių krepšelio vizualizaciją
Po sąlyga	Vartotojas gali naudotis SEIP
Alternatyvos (nesėkmės atvejai)	2.1 a Blogas vartotojo vardas sistema grįžta į 1.1 2.1 b Blogas slaptažodis sistema grįžta į 1.1
Veiklos taisyklės	Vartotojas turi teisingai įvesti duomenis.
Sudarė	Vigantas Nikontas
Sudarymo data	2006 01 04

Panaudojimo atvejai	Registruotis sistemoje
Aktorius	Pirkėjas
Sistema	SEIP
Prieš sąlyga	Vartotojas neturi būti registruotas šioje sistemoje
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas įveda sistemos adresą naršyklėje 2. Vartotojas užpildo registravimosi laukus	1.1 Sistema pateikia langą registracijos įvedimui 2.1 Sistema tikrina vartotojo įvestus duomenis ir gražina teisingą atsakymą 2.2 Sistema atidaro pagrindinį SEIP langą bei išveda duomenis apie vartotoją, bei aktyvuoja pirkinių krepšelio vizualizaciją
Po sąlyga	Vartotojas gali naudotis SEIP
Alternatyvos (nesėkmės atvejai)	2.1 a Įvestas vartotojo vardas jau egzistuoja - sistema grįžta į 1.1 2.1 b Blogai pakartotas slaptažodis - sistema grįžta į 1.1 2.1 c Nevisi laukai suvesti formoje – sistema grįžta į 1.1
Veiklos taisyklės	Vartotojas turi teisingai įvesti duomenis.
Sudarė	Vigantas Nikontas
Sudarymo data	2006 01 04

Panaudojimo atvejai	Atsijungti nuo sistemos
Aktorius	Pirkėjas
Sistema	SEIP
Prieš sąlyga	Vartotojas turi būti prisijungęs prie sistemos
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas aktyvuoja atsijungimo komandą	1.1 Sistema tikrina ar vartotojo sesija vis dar egzistuoja 1.2 Sistema atidaro pagrindinį SEIP langą bei ištrina visus laikinus objektus iš IS
Po sąlyga	Vartotojas gali saugiai uždaryti SEIP vartotojo sąsajos langą
Alternatyvos (nesėkmės atvejai)	1.1 a Jei vartotojo sesijos galiojimo laikas pasibaigęs - sistema grįžta į 1.2
Veiklos taisyklės	
Sudarė	Vigantas Nikontas
Sudarymo data	2006 01 04

Panaudojimo atvejai	Įtraukti prekes į pirkinių krepšelį
Aktorius	Pirkėjas
Sistema	SEIP
Prieš sąlyga	Vartotojas prisijungė prie sistemos, bei norimų prekių yra sandėlyje
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas prekių sąraše išsirenka prekę 2. Vartotojas spaudžia "Įtraukti prekes į pirkinių krepšelį"	1.1 Sistema atidaro prekės peržiūros langą 2.1 Sistema įtraukia prekę į pirkinių krepšelį
Po sąlyga	Prekių krepšelyje įtraukiama prekė.
Alternatyvos (nesėkmės atvejai)	
Veiklos taisyklės	
Sudarė	Vigantas Nikontas
Sudarymo data	2006 01 04

Panaudojimo atvejai	Pašalinti prekę iš pirkinių krepšelio
Aktorius	Pirkėjas
Sistema	SEIP
Prieš sąlyga	Vartotojas prisijungė prie sistemos, bei pirkinių krepšelyje yra pasirinktų prekių
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas prekių krepšelio sąraše ištrina nepageidaujamus pirkinius	1.1 Sistema tikrina ar yra pažymėtų ir ištrina jas 1.2 Sistema atnaujina prekių krepšelio langą
Po sąlyga	Pirkinių krepšelyje pašalinamos nepageidaujamos prekės.
Alternatyvos (nesėkmės atvejai)	1.1 a nėra pažymėtų prekių - sistema grįžta į 1.1
Veiklos taisyklės	
Sudarė	Vigantas Nikontas
Sudarymo data	2006-01-04

Panaudojimo atvejai	Padidinti prekės skaičių vienetu pirkinių krepšelyje
Aktorius	Pirkėjas
Sistema	SEIP
Prieš sąlyga	
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Aktyvuoja pirkinių krepšelio sąrašo meniu 2. Vartotojas nuspaudžia mygtuką "dešinė rodyklė"	1.1 Sistema perjungia langą į pirkinių krepšelio prekių sąrašo langą 1.1 Jei prekių yra sandėlyje – sistema padidina vienetu šios prekės pirkinių krepšelyje. 1.2 Sistema atnaujina pirkinių krepšelio sarašą
Po sąlyga	Sistemos vartotojas sumažina prekės skaičių pirkinių krepšelyje
Alternatyvos (nesėkmės atvejai)	2.1 a Sistema palieka nepakitųsį prekių kiekį
Veiklos taisyklės	Vartotojas turi būti prieš tai pasirinkęs nors vieną prekę
Sudarė	Vigantas Nikontas
Sudarymo data	2006-01-04

Panaudojimo atvejai	Sumažinti prekės skaičių vienetu pirkinių krepšelyje
Aktorius	Pirkėjas
Sistema	SEIP
Prieš sąlyga	
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Aktyvuoja pirkinių krepšelio sąrašo meniu 2. Vartotojas nuspaudžia mygtuką "kairė rodyklė"	1.1 Sistema perjungia langą į pirkinių krepšelio prekių sąrašo langą 2.1 Jei pasirinktos prekės kiekis ya lygus vienam – sistema pašalina šią prekę iš pirkinių krepšelio. 2.2 Sistema atnaujina pirkinių krepšelio sarašą
Po sąlyga	Sistemos vartotojas sumažina prekės skaičių pirkinių krepšelyje
Alternatyvos (nesėkmės atvejai)	

Veiklos taisyklės	Vartotojas turi būti prieš tai pasirinkęs nors vieną prekę
Sudarė	Vigantas Nikontas
Sudarymo data	2006-01-04

Panaudojimo atvejai	Peržiūrėti detalią prekės informaciją
Aktorius	Pirkėjas
Sistema	SEIP
Prieš sąlyga	Vartotojas prisijungė prie sistemos
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas pasirenka prekių sąraše norimą prekę	1.1 Sistema atverčia prekės detalios informacijos peržiūros langą
Po sąlyga	Vartotojas peržiūri detalią pasirinktos prekės informaciją.
Alternatyvos (nesėkmės atvejai)	
Veiklos taisyklės	
Sudarė	Vigantas Nikontas
Sudarymo data	2006-01-04

Panaudojimo atvejai	Peržiūrėti prekių sąrašą
Aktorius	Pirkėjas
Sistema	SEIP
Prieš sąlyga	Vartotojas prisijungė prie sistemos
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas pasirenka grupių sąraše norimą prekių grupę	1.1 Sistema atverčia prekių sarašo langą priklausančioms pasirinktąjai kategorijai
Po sąlyga	Vartotojas peržiūri prekių sąrašą.
Alternatyvos (nesėkmės atvejai)	
Veiklos taisyklės	
Sudarė	Vigantas Nikontas
Sudarymo data	2006-01-04

Panaudojimo atvejai	Apmokėti už prekes/paslaugas grynaisiais
Aktorius	Pirkėjas
Sistema	SEIP

Prieš sąlyga	Vartotojas prisijungė prie sistemos ir turi suformavęs pirkinių krepšelį
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas pasirenka pirkinių krepšelyje punktą „apmokėti“ 2. Vartotojas pasirenka apmokėjimo grynaisiais sprendimą 3. Vartotojas užpildo visus duomenis bei patvirtina apmokėjimą	1.1 Sistema atverčia į apmokėjimo pasirinkimo būdo formą 2.1 Sistema atverčia apmokėjimo grynaisiais formą 3.1 Sistema patikrina duomenų korektiškumą ir įvykdo užsakymą
Po sąlyga	Vartotojas apmoka už prekes grynaisiais.
Alternatyvos (nesėkmės atvejai)	3.1 a Jei duomenys neteisingi sistema grąžina į 2.1 punktą
Veiklos taisyklės	
Sudarė	Vigantas Nikontas
Sudarymo data	2006-01-04

Panaudojimo atvejai	Apmokėti už prekes/paslaugas per AC/Banko IS
Aktorius	Pirkėjas
Sistema	SEIP
Prieš sąlyga	Vartotojas prisijungė prie sistemos ir turi suformavęs pirkinių krepšelį
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas pasirenka pirkinių krepšelyje punktą „apmokėti“ 2. Vartotojas pasirenka apmokėjimo per AC/Banko IS 3. Vartotojas užpildo visus duomenis bei patvirtina apmokėjimą	1.2 Sistema atverčia į apmokėjimo pasirinkimo būdo formą 2.1 Sistema persiunčia užklausą AC/Banko IS 2.2 Sistema perjungia vartotoją į AC/Banko IS 3.1 AC/Banko IS patikrina duomenų korektiškumą ir įvykdo užsakymą 3.2 AC/Banko IS perjungia vartotoją į SEIP pagrindinį puslapį
Po sąlyga	Vartotojas apmoka už prekes mokėjimo kortele.

Alternatyvos (nesėkmės atvejai)	3.1 a Jei duomenys neteisingi sistema grąžina į 2.2 punktą
Veiklos taisyklės	
Sudarė	Vigantas Nikontas
Sudarymo data	2006-01-04

Panaudojimo atvejai	Sertifikato instaliavimas
Aktorius	AC/Banko IS
Sistema	SEIP
Prieš sąlyga	Vartotojas užsakęs apmokėjimo mokėjimo kortele paslaugą
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas sugeneruoja unikalų sertifikatą 2. Vartotojas suinstaliuoja SEIP IS skaitmeninį sertifikatą be galimybės jį eksportuoti	2.1 Sistema operuoja daugelį operacijų su šio sertifikato privačiu ir viešuoju raktais
Po sąlyga	Sistema turi kriptografijai reikalingus duomenis, bei autorizacijos teises AC/Banko IS
Alternatyvos (nesėkmės atvejai)	
Veiklos taisyklės	
Sudarė	Vigantas Nikontas
Sudarymo data	2006-01-04

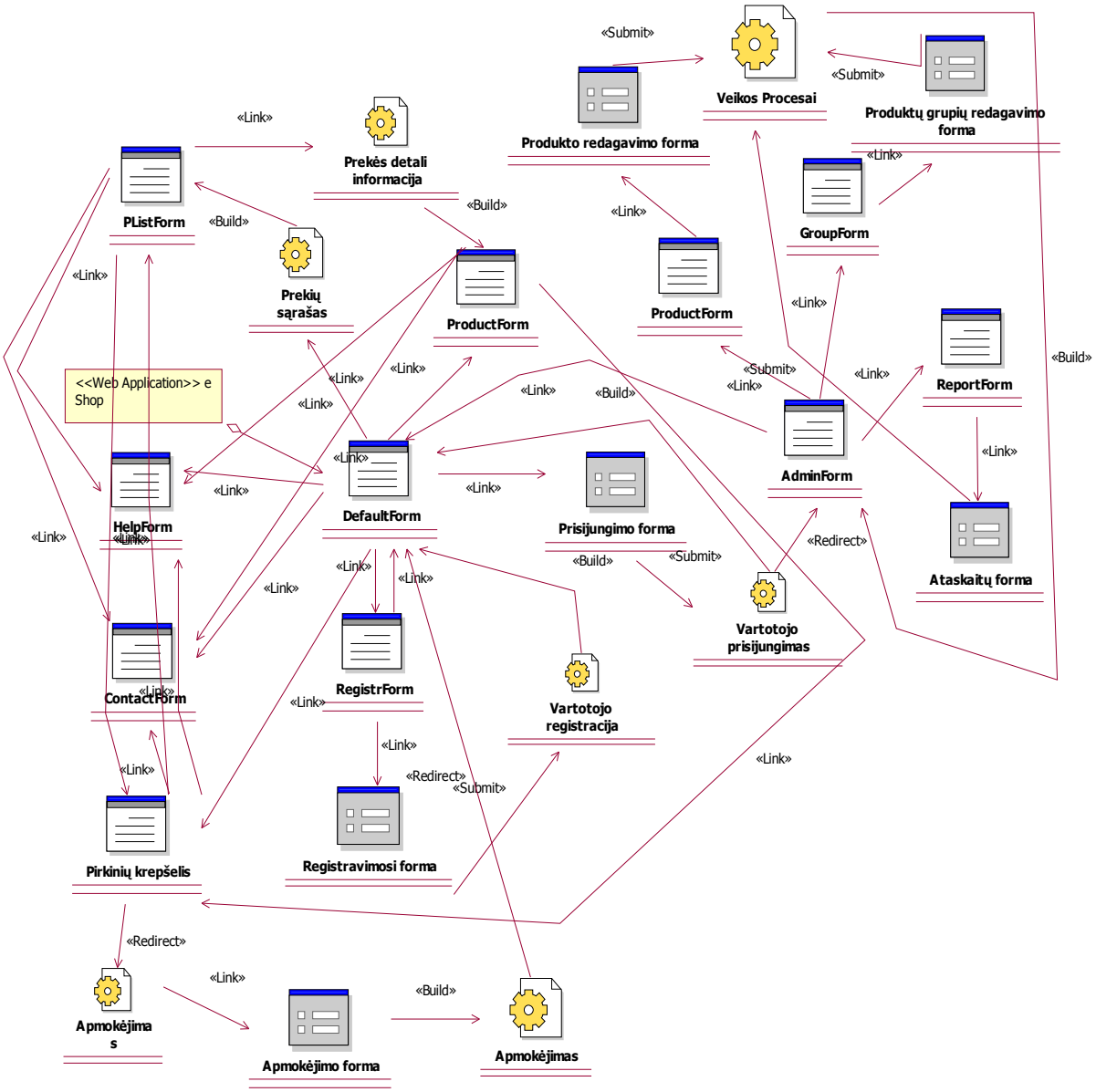
Panaudojimo atvejai	Duomenų įvedimas redagavimas DB
Aktorius	Duomenų operatorius
Sistema	SEIP
Prieš sąlyga	Vartotojas prisijungė prie sistemos
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**

1. Vartotojas pasirenką norimo objekto redagavimo/įtraukimo režimą 2. Vartotojas įveda/redaguoja objektus	1.1 Sistema atverčia pasirinkto objekto langą 2.1 Sistema patikrina duomenų korektiškumą ir įvykdo pakitimus
Po sąlyga	Vartotojas redaguoja duomenis DB.
Alternatyvos (nesėkmės atvejai)	2.1 a Jei duomenys neteisingi sistema grąžina į 1.1 punktą
Veiklos taisyklės	
Sudarė	Vigantas Nikontas
Sudarymo data	2006-01-04

Panaudojimo atvejai	Gauti suvestinius duomenis
Aktorius	Finansininkas, AC/Banko IS
Sistema	SEIP
Prieš sąlyga	Vartotojas prisijungė prie sistemos
Pagrindinis įvykių srautas	**Sistemos reakcija ir sprendimai**
1. Vartotojas pasirenka ataskaitos periodą	1.1. Sistema pasiūlo išsaugoti sugeneruotus duomenis vartotojo lokaliame kompiuteryje.
Po sąlyga	Vartotojas gauna suvestinius duomenis
Alternatyvos (nesėkmės atvejai)	
Veiklos taisyklės	
Sudarė	Vigantas Nikontas
Sudarymo data	2006-01-04

1.1.2. Vartotojų interfeiso modelis – navigavimo planas

Toliau trumpai aprašomi pagrindiniai kuriamos informacinės sistemos sąsajų komponentai – *Web* puslapiai ir formos. Šios svetainės schema pateikta žemiau:



Internetinio puslapio navigavimo planas

1.1.3. Formos, ataskaitos

Dauguma formų turi vienodas dinamines sritis, kuriose tam tikrom sąlygom yra užkraunami duomenys XML srautais – ko pasekoje gaunama ganėtinai dinamiška objektiškai orientuota internetinė svetainė.

Pagrindinio puslapio lango modelis:

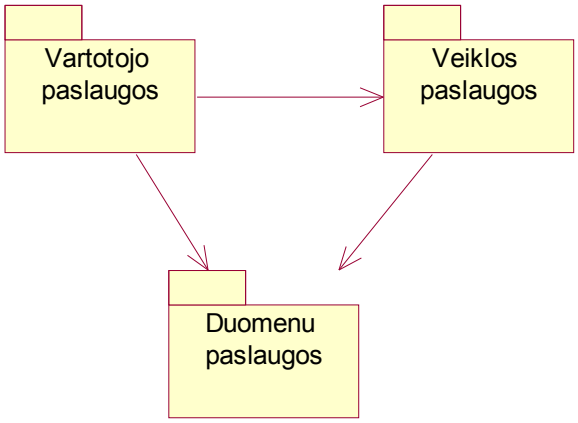
<table>
<tr><td colspan="4">Dinamiškai formuojamas meniu bei prisijungimo forma</td></tr>
<tr><td>Pigiausios prekės sritis</td><td>Perkamiausios prekės sritis</td><td>Naujausios prekės sritis</td><td>Prekių grupių sritis</td></tr>
<tr><td colspan="2">Prekės pasiūlymo sritis</td><td>Prekės naujienų sritis</td><td>Pirkinių krepšelio formavimo sritis</td></tr>
<tr><td colspan="4"></td></tr>
</table>

Produktų sąrašo/ pirkinių krepšelio lango modulis:

<table>
<tr><td colspan="2">Dinamiškai formuojamas meniu bei prisijungimo forma</td></tr>
<tr><td rowspan="2">Prekių sąrašo sritis/
Pirkinių krepšelio sritis</td><td>Prekių grupių sritis</td></tr>
<tr><td>Pirkinių krepšelio formavimo sritis</td></tr>
<tr><td colspan="2">Statinė puslapio informavimo dalis</td></tr>
</table>

5.3. Loginė sistemos architektūra

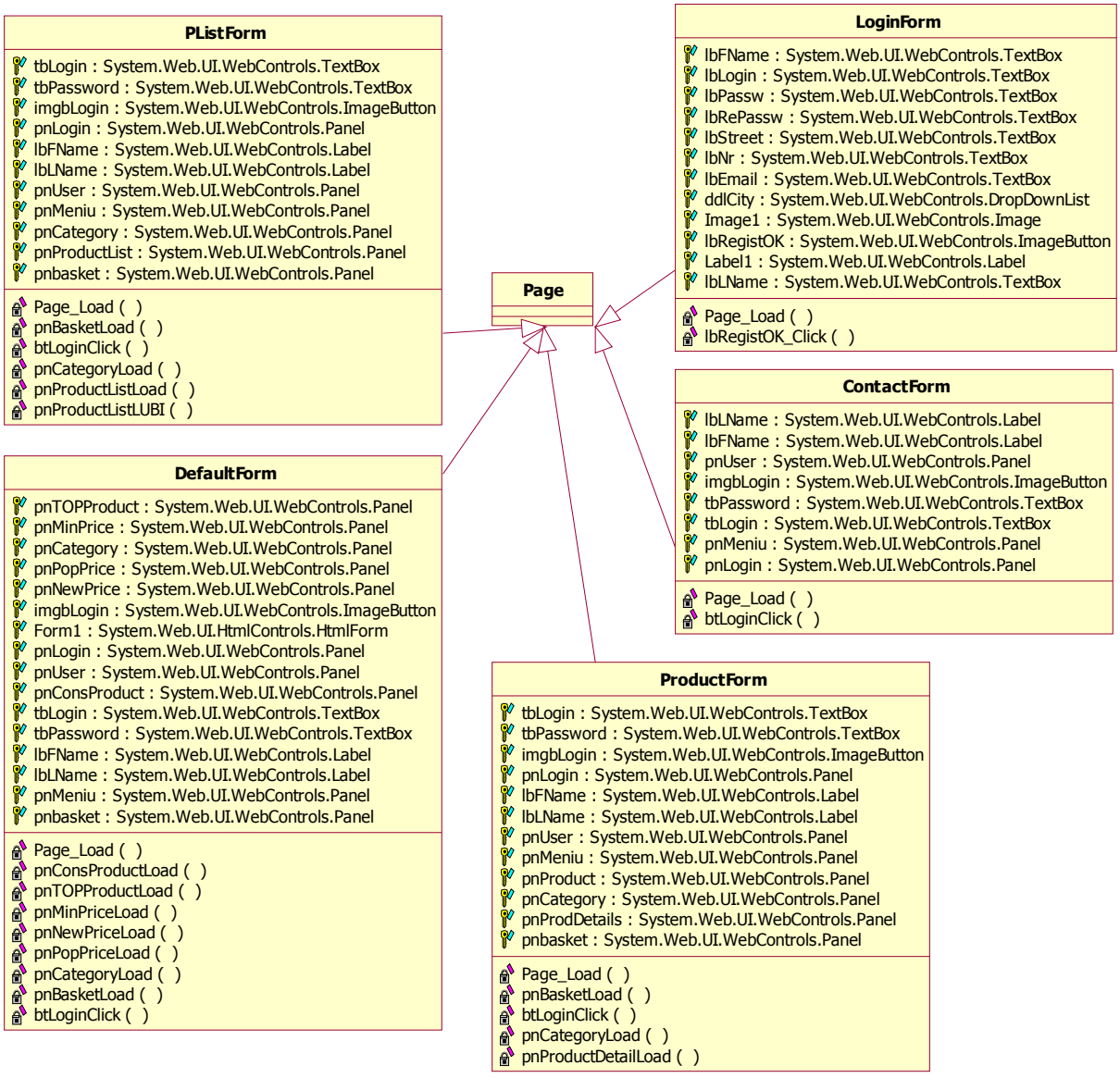
Projektavimui pasirinkome trijų lygių architektūros modelį. Remiantis šiuo modeliu, SEIP sistemą sudarys vartotojo, veiklos ir duomenų paslaugos, sugrupuotos į atitinkamus paketus:



Loginė sistemos architektūra

5.4. Projekto modelis ir specifikacijos

Web aplikacija susideda iš 5 pagrindinių internetinių formų. Kadangi darbas realizuotas remiantis objektiniu programavimu C#.NET, todėl pateiksiu šių formų klasių diagramą:



Internetinių formų objektų klasių diagrama

Beveik visos formos turi dinamines sritis, kur:

- pnLogin: System.Web.UI.WebControls.Panel – prisijungimo informacijos įvedimo sritis;

- pnUser: System.Web.UI.WebControls.Panel – prisijungusio prie IS vartotojo informacijos sritis;

- pnMeniu: System.Web.UI.WebControls.Panel – viršutinė, dinaminė meniu sritis, kurioje pagal veiklos taisykles atvaizduojami atitinkami pasirinkimo punktai;

- pnCategory: System.Web.UI.WebControls.Panel – dešinėje pusėje esanti prekių grupių dinaminė sritis;

- pnBasket: System.Web.UI.WebControls.Panel – pirkinių krepšelio formavimo sritis;

- pnProduct: System.Web.UI.WebControls.Panel – detalios produkto informacijos detalizavimos sritis;

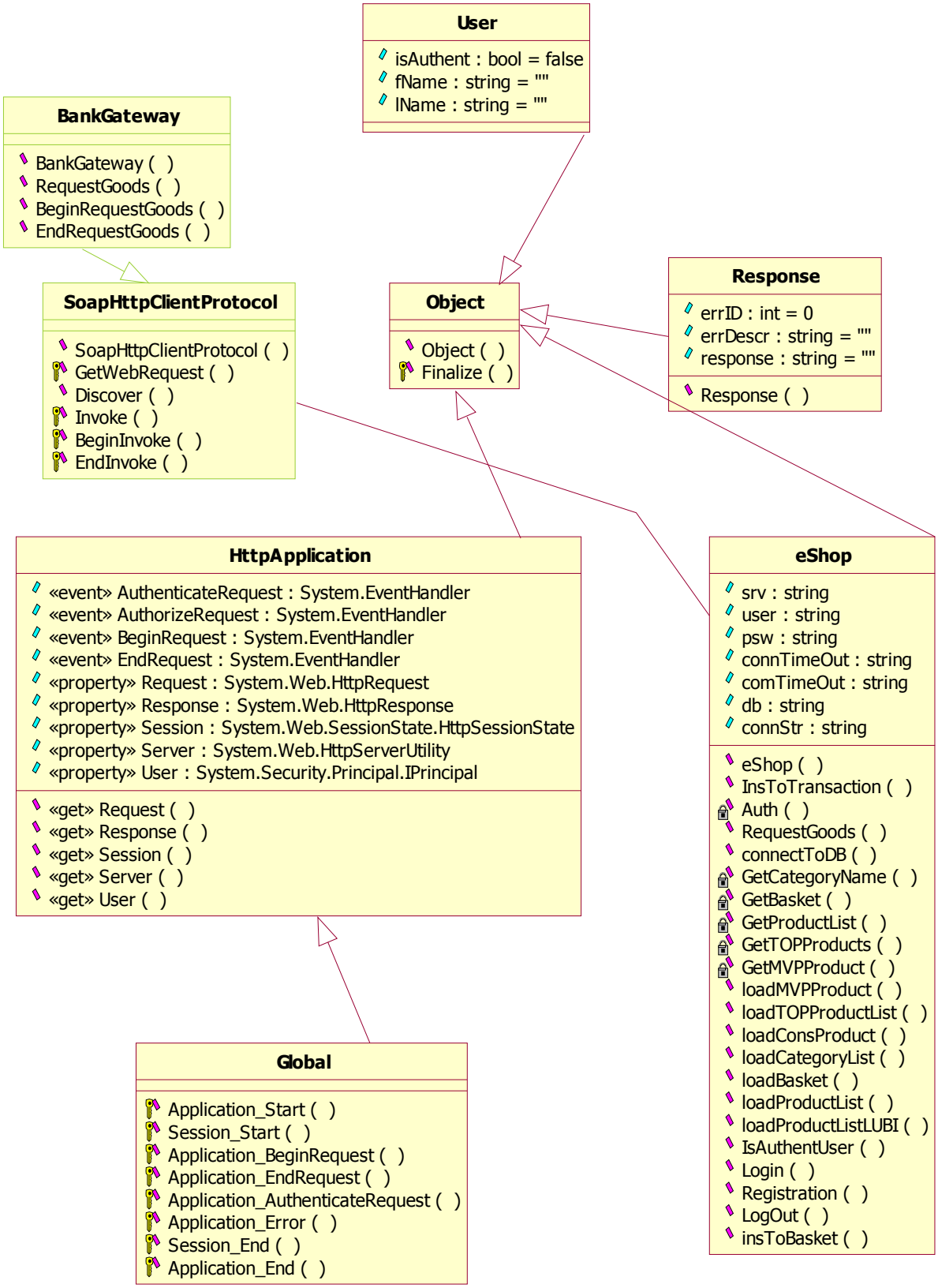
- pnProdList: System.Web.UI.WebControls.Panel – prekių sarašo formavimo sritis.

Taip pat kiekviena forma turi savo metodus. Šie metodai pagrinde pagal veiklos taisykles vizualizuoja internetinio puslapio objektus.

Kaip paminėta aukščiau veiklos taisyklės realizuotos DLL bibliotekoje *eShop*. Pagrindiniai metodai:

- eShop() – konstruktorius;
- insToBasket() – įterpti prekę į pirkinių krepšelio sąrašą;
- LogOut() – baigti vartotojo prisijungimo sesiją šioje IS;
- Login() – pradėti vartotojo užmegztą sesiją šioje IS;
- Registration() – registruoti vartotoją šioje IS;
- IsAuthentUser() – nustato ar vartotojo sesija vis dar aktyvi šioje IS;
- connectToDB() – prisijungia prie IS duomenų bazės;

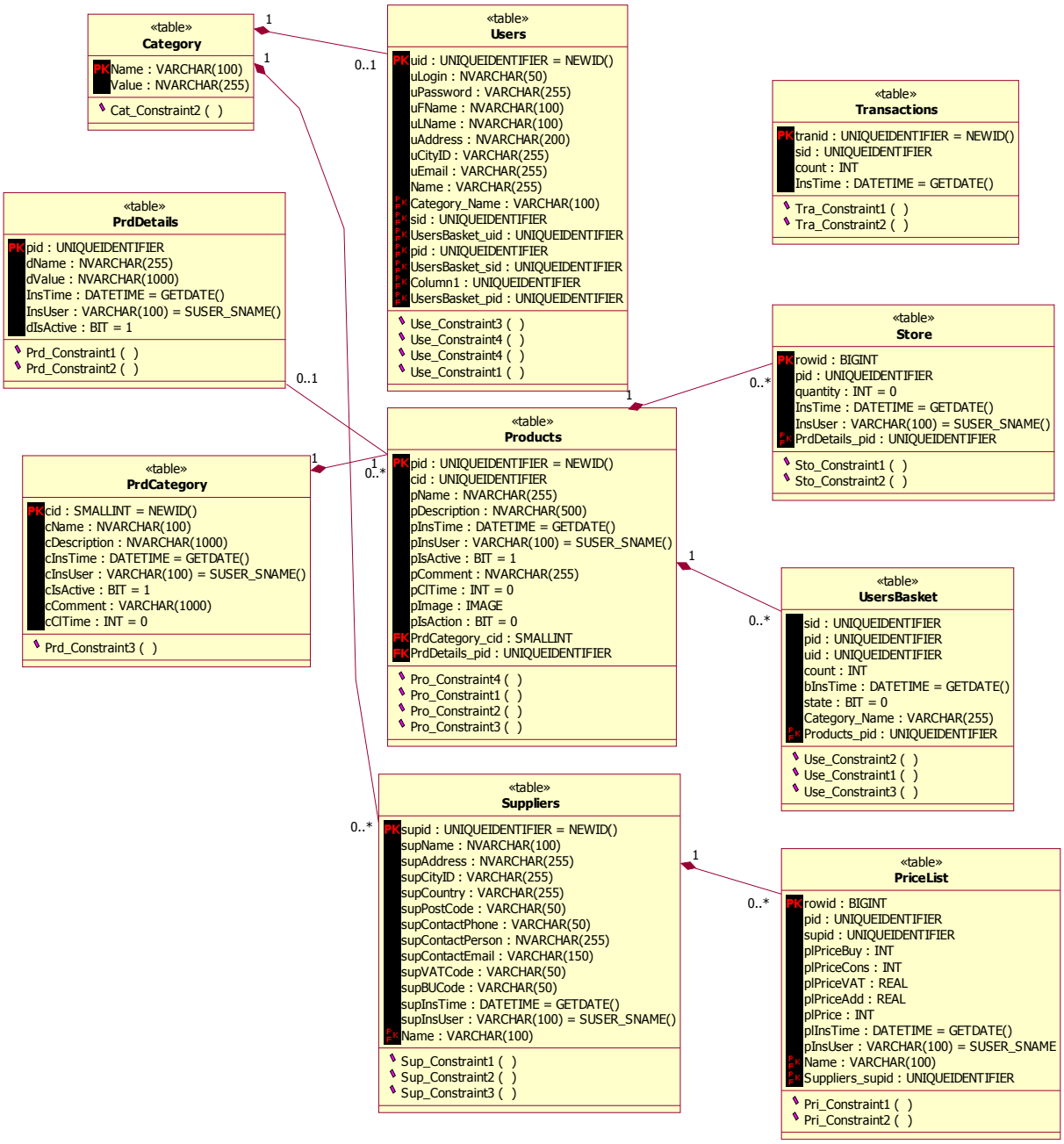
Autorizavimo galimybė yra pasiekiama iš suteikto autorizacijos centro BankGateway.dll failo. Šioje bibliotekoje yra realizuotas visas kriptografijos, tam tikrų taisyklių, bei kitų saugumo sprendimų rinkinys. Dėl konfidencialių įsipareigojimų negaliu viešai deklaruoti šio failo struktūros. Tad darykime prielaidą, jog jame tėra iliustruoti metodai mokamųjų kortelių autorizavimui.



Klasių diagrama

5.5. Duomenų bazės modelis

1.1.4. Duomenų bazės schema



Duomenų bazės schema

1.1.5. Duomenų bazės lentelių aprašas

Lentelė Sessions – skirta vartotojo prisijungimo prie IS sesijoms sekti, manipuliuoti.

Laukas	Aprašymas	Papildoma informacija
sid	Unikalus sesijos raktas	Uniqueidentifier
uid	Vartotojo identifikatorius	Uniqueidentifier
loginTime	Prisijungimo laikas	Datetime, GetDate()

Lentelė PrdCategory – skirta prekių ar paslaugų kategorijų aprašymui

Laukas	Aprašymas	Papildoma informacija
cid	Unikalus prekės kategorijos identifikatorius	Uniqueidentifier, NewID()
cName	Kategorijos pavadinimas	Nvarchar(100)
cDescription	Kategorijos aprašymas	Nvarchar(1000)
cInsTime	Kategorijos įtraukimo į sistemą laikas	Datetime, GetDate()
cInsUser	Kategorijos įtraukimo į sistemą vartotojas	Varchar(100), suser_sname()
cIsActive	Ar kategorija aktyvi	Bit, 1
cComment	Kategorijos komentaras	Varchar(2000)
cClTime	Kiek kartų paspausta ant kategorijos	Int, 0

Lentelė Users – skirta informacijai apie sistemos vartotojus rinkti

Laukas	Aprašymas	Papildoma informacija
uid	Unikalus vartotojo identifikatorius	Uniqueidentifier, NewID()
uLogin	Vartotojo prisijungimo vardas	Nvarchar(50)
uPassword	Vartotojo prisijungimo slaptažodis	Nvarchar(255)
uFName	Vartotojo vardas	Nvarchar(100)

uLName	Vartotojo pavardė	Nvarchar(255)
uAddress	Vartotojo adresas	Nvarchar(200)
uMail	Vartotojo elektroninis paštas	Varchar(255)

Lentelė UsersBasket – skirta vartotojo formuojamiems pirkinių krepšeliams kaupti

Laukas	Aprašymas	Papildoma informacija
sid	Unikalus sesijos raktas	Uniqueidentifier
pid	Unikalus prekės identifikatorius	Uniqueidentifier
uid	Vartotojo identifikatorius	Uniqueidentifier
Count	Prekės kiekis	INT
bInsTime	Įtraukimo laikas	Datetime, GetDate()
State	Būsena	BIT, 0

Lentelė Store – skirta prekių kaupimui, tarnauja kaip sandėlio konceptas

Laukas	Aprašymas	Papildoma informacija
rowid	Unikalus eilutės identifikatorius	Bigint
Pid	Unikalus prekės identifikatorius	Uniqueidentifier
Quantity	Sandėlyje esantis prekių skaičius	Int,0
InsTime	Įtraukimo laikas	Datetime, GetDate()
InsUser	Vartotojas	Varchar(100),suser_sname()

Lentelė PrdDetails – skirta prekių savybių aprašymui

Laukas	Aprašymas	Papildoma informacija
Pid	Unikalus prekės identifikatorius	Uniqueidentifier
dName	Prekės savybės pavadinimas	NVarchar(255)
dValue	Prekės savybės reikšmė	Nvarchar(1000)
insTime	Įterpimo laikas	Datetime,GetDate()
insUser	Vartotojas	Varchar(100), Suser_Sname()

dIsActive	Ar savybė aktyvi?	Bit, 1

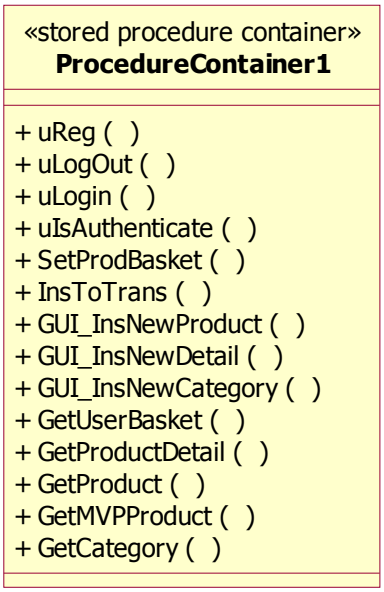
Lentelė PriceList – skirta prekių kainų sudarymui – kainoraščių lentelė

Laukas	Aprašymas	Papildoma informacija
Rowid	Unikalus eilutės įrašo identifikatorius	Bigint
Pid	Unikalus prekės identifikatorius	Uniqueidentifier
Supid	Unikalus tiekėjo identifikatorius	Uniqueidentifier
plPriceBuy	Pirkimo kaina	int
plPriceCons	Nuolaida	Int
plPriceVAT	PVM mokestis	Real
plPriceAdd	Prekės marža	Real
plPrice	Galutinė kaina	Int
pInsTime	Įtraukimo laikas	Datetime, getdate()
pInsUser	Vartotojas	Varchar(100), SUSER_SNAME()

Lentelė Transactions – skirta kaupti transakcijų duomenis, kuriuos siuntė ar gavo iš banko.

Laukas	Aprašymas	Papildoma informacija
Tranid	Unikalus transakcijos identifikatorius	Uniqueidentifier
Sid	Unikalus sesijos raktas	Uniqueidentifier
Amount	Transakcijos suma	Int
InsTime	Įterpimo laikas	Datetime,Getdate()

1.1.6. Saugomos procedūros



Procedūra [uReg] skirta vartotojo registracijai SEIP IS. Šios procedūros parametrai pateikti žemiau:

```
CREATE PROCEDURE dbo.uReg
        @login NVARCHAR(50),
        @passw NVARCHAR(255),
        @fname NVARCHAR(100),
        @lname NVARCHAR(100),
        @address NVARCHAR(200),
        @city NVARCHAR(50),
        @mail VARCHAR(255),
        @sid VARCHAR(50),
        @err INT OUTPUT,
        @errDescr VARCHAR(150) OUTPUT
AS
```

Procedūra [uLogOut] skirta vartotojo sesijai nutraukti SEIP IS. Šios procedūros parametrai pateikti žemiau:

```
CREATE PROCEDURE uLogOut
        @sid VARCHAR(50)
AS
```

Procedūra [uLogIn] skirta vartotojo sesijai nutraukti SEIP IS. Šios procedūros parametrai pateikti žemiau:

```
CREATE  PROCEDURE uLogin
            @uLogin NVARCHAR(50),
            @uPassword NVARCHAR(255),
            @sid VARCHAR(50),
            @err INT OUTPUT,
            @errDescr VARCHAR(150) OUTPUT
AS
```

Procedūra [uIsAuthenticate] skirta vartotojo sesijos egzistavimui nustatyti SEIP IS. Šios procedūros parametrai pateikti žemiau:

```
CREATE PROCEDURE uIsAuthenticate
            @sid VARCHAR(50),
            @fName NVARCHAR(50) OUTPUT,
            @lName NVARCHAR(100) OUTPUT,
            @err INT OUTPUT,
            @errDescr VARCHAR(150) OUTPUT
AS
```

Procedūra [SetProdBasket] skirta vartotojo prekės įtraukimui į pirkinių krepšelį SEIP IS. Šios procedūros parametrai pateikti žemiau:

```
CREATE PROCEDURE SetProdBasket
            @pid VARCHAR(50),
            @sid VARCHAR(50),
            @count INT
AS
```

Procedūra [InsToTrans] skirta vartotojo apsipirkimo duomenų įterpimui į transakcijų lentelę SEIP IS. Šios procedūros parametrai pateikti žemiau:

```
CREATE PROCEDURE InsToTrans
            @xml VARCHAR(500),
            @tranid NVARCHAR(50) OUTPUT,
            @amount INT OUTPUT
AS
```

Procedūra [GUI_InsNewProduct] skirta naujų prekių įvedimui SEIP IS. Šios procedūros parametrai pateikti žemiau:

```
CREATE    PROCEDURE GUI_InsNewProduct
          @cid UNIQUEIDENTIFIER,
          @pName NVARCHAR(255),
          @pDescription NVARCHAR(500),
          @pComment VARCHAR(255),
          @pImage VARCHAR(255),
          @quantity INT,
          @supid UNIQUEIDENTIFIER,
          @prBuy INT,
          @prCons INT,
          @prVAT REAL,
          @prADD REAL,
          @err INT OUTPUT,
          @errDescr VARCHAR(255) OUTPUT
AS
```

Procedūra [GUI_InsNewDetail] skirta naujų prekės savybių įvedimui SEIP IS. Šios procedūros parametrai pateikti žemiau:

```
CREATE PROCEDURE GUI_InsNewDetail
          @pid UNIQUEIDENTIFIER,
          @name NVARCHAR(255),
          @value NVARCHAR(1000),
          @err INT OUTPUT,
          @errDescr VARCHAR(255) OUTPUT
AS
```

Procedūra [GUI_InsNewCategory] skirta naujų prekės kategorijų įvedimui SEIP IS. Šios procedūros parametrai pateikti žemiau:

```
CREATE  PROCEDURE GUI_InsNewCategory
          @cName NVARCHAR(100),
          @cDescription NVARCHAR(1000),
          @cComment VARCHAR(2000),
          @err INT OUTPUT,
          @errDescr VARCHAR(255) OUTPUT
AS
```

Procedūra [GetUserBasket] skirta vartotojo prekės sąrašo išvedimui iš pirkinių krepšelio SEIP IS. Šios procedūros parametrai pateikti žemiau:

```
CREATE PROCEDURE GetUserBasket
          @xml VARCHAR(1000)
AS
```

Procedūra [GetProductDetail] skirta vartotojo prekės detalios informacijos išvedimui SEIP IS. Šios procedūros parametrai pateikti žemiau:

```
CREATE PROCEDURE GetProductDetail
        @xml NVARCHAR(1000),
        @img VARCHAR(255) =null OUTPUT,
        @name VARCHAR(255) = null OUTPUT,
        @descr VARCHAR(1000) =null OUTPUT,
        @price MONEY = NULL OUTPUT
AS
```

Procedūra [GetProduct] skirta vartotojo prekės informacijos išvedimui SEIP IS. Šios procedūros parametrai pateikti žemiau:

```
CREATE    PROCEDURE GetProduct
        @xml NVARCHAR(1000)
AS
```

Procedūra [GetMVPProduct] skirta vartotojo prekės informacijos išvedimui SEIP IS. Šios procedūros parametrai pateikti žemiau:

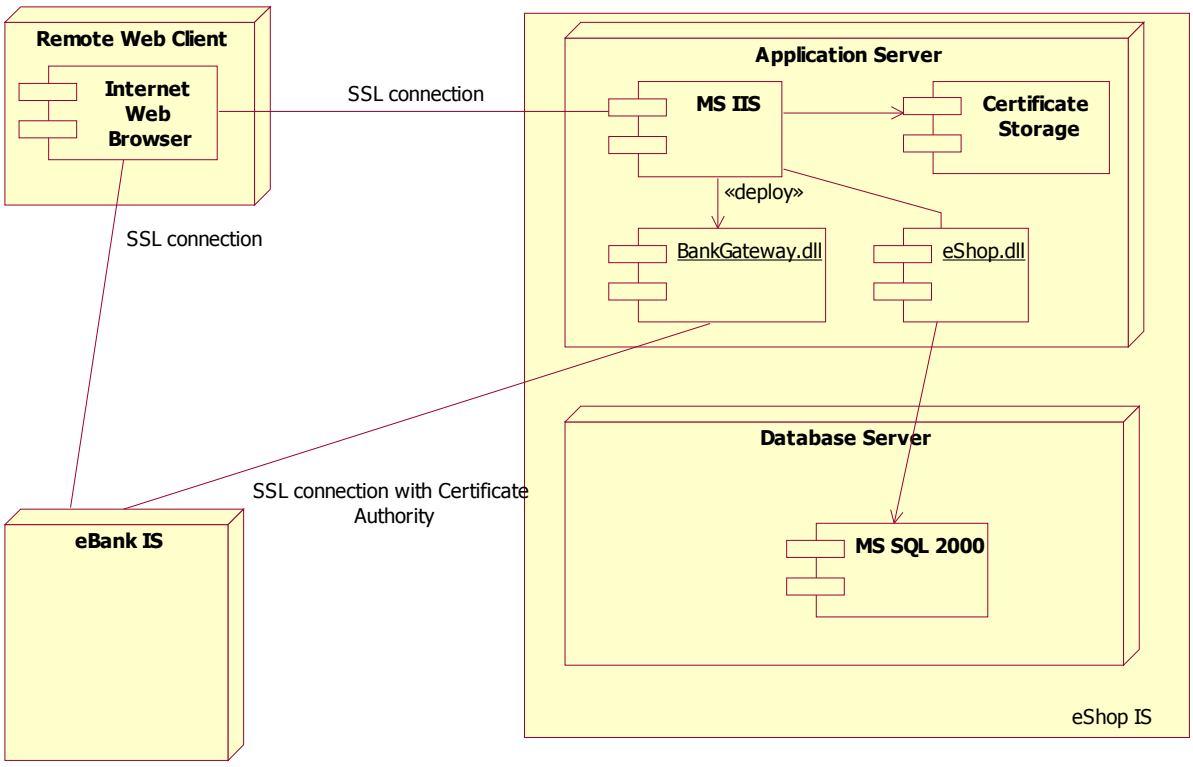
```
CREATE  PROCEDURE GetMVPProduct
        @xml VARCHAR(1000)
AS
```

Procedūra [GetCategory] skirta vartotojo kategorijų sarašo išvedimui SEIP IS. Šios procedūros parametrai pateikti žemiau:

```
CREATE PROCEDURE GetCategory
        @xml NVARCHAR(1000)
AS
```

5.6. Realizacijos modelis

Fizinį modelio vaizdą projektavome komponentų diagrama.



Sistemos komponentai bei jų tarpusavio priklausomybes

Informacinė sistema realizuota kliento-serverio technologija. Serverio platforma pasirinkta Microsoft operacinė sistema. Naudojamas MS IIS interneto serveris ir dinaminių puslapių generavimo programavimo kalba C#.NET. Duomenys saugomi MSSQL DBVS duomenų bazėje.

Pageidautina, jog kliento dalis susidėtų iš Windows operacinės sistemos ir naršyklės Internet Explorer, kurios pagalba galės dirbti su sistema.

5.7. Testavimo modelis

5.7.1. Testavimo metodika

Pirmajame etape programuotojai testuoja savo modulius patys atskirai vienas nuo kito. Kai kiekvienas modulis veikia korektiškai, jie jungiami į bendrą sistemą ir testuojamas bendras veikimas. Paskutiniame etape atliekamas integravimo testavimas, testuojama sujungus visą sistemą su pilna vartotojo sąsają ir valdymu per Internetą. Apie rastas klaidas pasižymime. Ištaisius visas klaidas sistema pateikiama užsakovui.

Naudojamas statinis ir dinaminis testavimas:

- ✓ *Programinės įrangos kodo peržiūra.* Programuotojas peržiūri savo kodą ir taip aptinka klaidas.
- ✓ *Programinės įrangos testavimas.* Testuojama su iš anksto paruoštais duomenimis, pateikiant juos sukompiliuotai programai.

Šiame projekte ir programuotojas ir testuotojas buvo vienas žmogus.

5.7.2. Modulių testavimas

Buvo pasirinkta testuoti tik tuos PĮ sistemos modulius, kurių veikimas nėra elementarus.

Didžiausias dėmesys nukreiptas į bendrąjį programos testavimą, kuris atliekamas modulį integravus į sistemą ir tuomet atlikus bendrą sistemos funkcijų patikrinimą.

5.7.3. Vartotojo sąsajos testavimas

Kadangi vartotojo sąsaja buvo kuriama prototipu pagrindu, tai kiekvienas prototipas buvo testuojamas atskirai. Su prototipais buvo testuojamas tik interfeiso patogumas ir interaktyvumas – bandant įvesti duomenų rinkinį ir lyginant su kitų prototipų analogiškais testais. Žiūrėta kurio prototipo sąsaja leidžia tai padaryti lengviau ir paprasčiau.

Patvirtinus galutinę vartotojo sąsają testavimas vyko etapais:

- ✓ Atskirų komponentų testavimas – būsenų mygtukų, iškrentančių menių, sąrašų. Testuojama buvo stresinio testavimo metodu, bandant įvairiais būdais sutrikdyti komponentų darbą.
- ✓ Komponentų tarpusavio testavimas. Buvo testuojama kaip vieno elemento būsenos pakeitimas atsiliepia kitų elementų būsenoms. Ar taisyklingai atnaujinamas komponentų vaizdavimas. Testuojama buvo stresinio testavimo metodu, bandant įvairiais būdais sutrikdyti komponentų darbą, bandant įvairias būsenų kombinacijas. Papildomas testavimas vyko pat vedant kontrolines duomenų sekas.
- ✓ Pilnas sąsajos testavimas. Buvo vykdomas vedant kontroliniais pradinių duomenų sekas ir žiūrint kaip sistema supranta ir priima tuos duomenis, bei kaip atvaizduoja gautus rezultatus. Vertintas interaktyvumas.

6. IŠVADOS

1. Apžvelgtos internetinių taikomųjų programų UML (*Unified Modelling Language*) projektavimo priemonės.
2. Išanalizuotos programavimo technologijų ir reliacinių duomenų bazių valdymo sistemų RDBVS (*Relational Database Management System*) pagrindinės charakteristikos.
3. Pateikta Lietuvos rinkoje panašių sprendimų analizė.
4. Pateiktas alternatyvus Lietuvos rinkoje saugios elektrininės komercijos sprendimas, pasižymintis itin aukštu perduodamos informacijos lygiu.
5. Šis spendimas suprojektuotas panaudojant CASE (*Computer Aided- Software Engineering*) įrankius.
6. Iliustruoti pagrindiniai saugios elektroninės komercijos procesai.
7. Naudojant objektiškai orientuotą projektavimo metodą bei remiantis greita aplikacijų paruošimo metadologija RAD (*Rapid Application Development*) buvo realizuotas sistemos prototipas panaudojant Microsoft.NET technologijas.

7. LITERATŪRA

[1] Nemuraitė L., Kavaliauskienė L. Informacinių sistemų projektavimo metodų tobulinimas ir automatizavimas taikant UML. Kaunas, 2002

[2] I.Jacobson, G.Booch, J.Rumbaugh. The Unified Software Development Process, Addison-Wesley, 2000

[3] MS SQL Server Books Online

[4] UML Notation Guide. Version 1.4. - OMG, 2001. Prieiga per internetą: <www.omg.org>.

[5] J.Martin, J.J.Odell. Object Oriented Methods: A Foundation. - Prentice Hall, 1996. - pp. 408.

[6] Alexander Chigrik, SQL Server 2000 vs Oracle. Prieiga per internetą: <http://www.mssqlcity.com/Articles/Compare/sql_server_vs_oracle.htm#part_3_1>

[7] Peter A. Bromberg, Ph.D., Using Web Services Enhancements (WSE) for X.509 Certificate Authentication and Digital Signature. Prieiga per internetą: <http://www.eggheadcafe.com/articles/20021231.asp>

[8] Gowri S Paramasivam, Cryptography in Microsoft.NET Part III: Digital Certificates. Prieiga per internetą: <http://www.c-sharpcorner.com/Code/2003/Jan/DigitalCertIII.asp>

[9] Cryptography. Prieiga per internetą: <http://www.csharpfriends.com/Articles/getArticle.aspx?articleID=107>

[10] Nitin Pande, XML Serialization Using C#. Prieiga per internetą: http://www.dotnetjohn.com/articles.aspx?articleid=173

[11] Accessing the XML DOM, prieiga per internetą: <http://www.w3schools.com/dom/dom_access.asp>

[12] eProcessingNetwork, Payment processing service: integration and development. Prieiga per internetą: <http://www.eprocessingnetwork.com/Utilities.html>

[13] Microsoft Visual C# Developer Center. Prieiga per internetą: <http://msdn.microsoft.com/vcsharp/>

[14] Bibit Inc., About Payment Service. Prieiga per internetą: <http://www.bibit.com/payment.shtml#works>

.

8. PRIEDAI

8.1. Duomenų bazės scenarijus:

```
/****** Object:  Database eShop    Script Date: 2006.01.05 22:50:22 ******/
IF EXISTS (SELECT name FROM master.dbo.sysdatabases WHERE name = N'eShop')
	DROP DATABASE [eShop]
GO

CREATE DATABASE [eShop]  ON (NAME = N'eShop_Data', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL\Data\eShop_Data.MDF' , SIZE = 2, FILEGROWTH = 10%) LOG ON (NAME = N'eShop_Log', FILENAME = N'C:\Program Files\Microsoft SQL Server\MSSQL\Data\eShop_Log.LDF' , SIZE = 1, FILEGROWTH = 10%)
 COLLATE SQL_Latin1_General_CP1_CI_AS
GO

exec sp_dboption N'eShop', N'autoclose', N'true'
GO

exec sp_dboption N'eShop', N'bulkcopy', N'false'
GO

exec sp_dboption N'eShop', N'trunc. log', N'true'
GO

exec sp_dboption N'eShop', N'torn page detection', N'true'
GO

exec sp_dboption N'eShop', N'read only', N'false'
GO

exec sp_dboption N'eShop', N'dbo use', N'false'
GO

exec sp_dboption N'eShop', N'single', N'false'
GO

exec sp_dboption N'eShop', N'autoshrink', N'true'
GO

exec sp_dboption N'eShop', N'ANSI null default', N'false'
GO

exec sp_dboption N'eShop', N'recursive triggers', N'false'
GO

exec sp_dboption N'eShop', N'ANSI nulls', N'false'
GO

exec sp_dboption N'eShop', N'concat null yields null', N'false'
GO

exec sp_dboption N'eShop', N'cursor close on commit', N'false'
GO

exec sp_dboption N'eShop', N'default to local cursor', N'false'
GO

exec sp_dboption N'eShop', N'quoted identifier', N'false'
GO

exec sp_dboption N'eShop', N'ANSI warnings', N'false'
GO

exec sp_dboption N'eShop', N'auto create statistics', N'true'
GO

exec sp_dboption N'eShop', N'auto update statistics', N'true'
GO

if( ( (@@microsoftversion / power(2, 24) = 8) and (@@microsoftversion & 0xffff >= 724) ) or ( (@@microsoftversion / power(2, 24) = 7) and (@@microsoftversion & 0xffff >= 1082) ) )
		exec sp_dboption N'eShop', N'db chaining', N'false'
```

```
GO

use [eShop]
GO

/****** Object:  Stored Procedure dbo.GUI_InsNewCategory    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GUI_InsNewCategory]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GUI_InsNewCategory]
GO

/****** Object:  Stored Procedure dbo.GUI_InsNewDetail    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GUI_InsNewDetail]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GUI_InsNewDetail]
GO

/****** Object:  Stored Procedure dbo.GUI_InsNewProduct    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GUI_InsNewProduct]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GUI_InsNewProduct]
GO

/****** Object:  Stored Procedure dbo.GetCategory    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetCategory]') and OBJECTPROPERTY(id, N'IsProcedure')
= 1)
drop procedure [dbo].[GetCategory]
GO

/****** Object:  Stored Procedure dbo.GetMVPProduct    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetMVPProduct]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetMVPProduct]
GO

/****** Object:  Stored Procedure dbo.GetProduct    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetProduct]') and OBJECTPROPERTY(id, N'IsProcedure')
= 1)
drop procedure [dbo].[GetProduct]
GO

/****** Object:  Stored Procedure dbo.GetProductDetail    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetProductDetail]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetProductDetail]
GO

/****** Object:  Stored Procedure dbo.GetUserBasket    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[GetUserBasket]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[GetUserBasket]
GO

/****** Object:  Stored Procedure dbo.InsToTrans    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[InsToTrans]') and OBJECTPROPERTY(id, N'IsProcedure')
= 1)
drop procedure [dbo].[InsToTrans]
GO

/****** Object:  Stored Procedure dbo.SetProdBasket    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[SetProdBasket]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[SetProdBasket]
GO

/****** Object:  Stored Procedure dbo.uReg    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[uReg]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[uReg]
GO

/****** Object:  Stored Procedure dbo.uIsAuthenticate    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[uIsAuthenticate]') and OBJECTPROPERTY(id,
N'IsProcedure') = 1)
drop procedure [dbo].[uIsAuthenticate]
GO

/****** Object:  Stored Procedure dbo.uLogOut    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[uLogOut]') and OBJECTPROPERTY(id, N'IsProcedure') =
1)
drop procedure [dbo].[uLogOut]
```

```
GO

/****** Object:  Stored Procedure dbo.uLogin    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[uLogin]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)
drop procedure [dbo].[uLogin]
GO

/****** Object:  Table [dbo].[PrdCategory]    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[PrdCategory]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[PrdCategory]
GO

/****** Object:  Table [dbo].[PrdDetails]    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[PrdDetails]') and OBJECTPROPERTY(id, N'IsUserTable')
= 1)
drop table [dbo].[PrdDetails]
GO

/****** Object:  Table [dbo].[PriceList]    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[PriceList]') and OBJECTPROPERTY(id, N'IsUserTable') =
1)
drop table [dbo].[PriceList]
GO

/****** Object:  Table [dbo].[Products]    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Products]') and OBJECTPROPERTY(id, N'IsUserTable') =
1)
drop table [dbo].[Products]
GO

/****** Object:  Table [dbo].[Sessions]    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Sessions]') and OBJECTPROPERTY(id, N'IsUserTable') =
1)
drop table [dbo].[Sessions]
GO

/****** Object:  Table [dbo].[Store]    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Store]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[Store]
GO

/****** Object:  Table [dbo].[Suppliers]    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Suppliers]') and OBJECTPROPERTY(id, N'IsUserTable') =
1)
drop table [dbo].[Suppliers]
GO

/****** Object:  Table [dbo].[Transactions]    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Transactions]') and OBJECTPROPERTY(id,
N'IsUserTable') = 1)
drop table [dbo].[Transactions]
GO

/****** Object:  Table [dbo].[Users]    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[Users]') and OBJECTPROPERTY(id, N'IsUserTable') = 1)
drop table [dbo].[Users]
GO

/****** Object:  Table [dbo].[UsersBasket]    Script Date: 2006.01.05 22:50:24 ******/
if exists (select * from dbo.sysobjects where id = object_id(N'[dbo].[UsersBasket]') and OBJECTPROPERTY(id, N'IsUserTable')
= 1)
drop table [dbo].[UsersBasket]
GO

/****** Object:  User dbo    Script Date: 2006.01.05 22:50:22 ******/
/****** Object:  User eShop    Script Date: 2006.01.05 22:50:22 ******/
if not exists (select * from dbo.sysusers where name = N'eShop' and uid < 16382)
        EXEC sp_grantdbaccess N'eShop', N'eShop'
GO

/****** Object:  User eShop    Script Date: 2006.01.05 22:50:22 ******/
exec sp_addrolemember N'db_owner', N'eShop'
GO

/****** Object:  Table [dbo].[PrdCategory]    Script Date: 2006.01.05 22:50:24 ******/
CREATE TABLE [dbo].[PrdCategory] (
        [cid] [uniqueidentifier] NOT NULL ,
        [cName] [nvarchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
        [cDescription] [nvarchar] (1000) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
```

```
	[cInsTime] [datetime] NOT NULL ,
	[cInsUser] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[cIsActive] [bit] NOT NULL ,
	[cComment] [varchar] (2000) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[cClTime] [int] NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[PrdDetails]    Script Date: 2006.01.05 22:50:25 ******/
CREATE TABLE [dbo].[PrdDetails] (
	[pid] [uniqueidentifier] NOT NULL ,
	[dName] [nvarchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[dValue] [nvarchar] (1000) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[insTime] [datetime] NOT NULL ,
	[insUser] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[dIsActive] [bit] NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[PriceList]    Script Date: 2006.01.05 22:50:25 ******/
CREATE TABLE [dbo].[PriceList] (
	[rowid] [bigint] IDENTITY (1, 1) NOT NULL ,
	[pid] [uniqueidentifier] NOT NULL ,
	[supid] [uniqueidentifier] NOT NULL ,
	[plPriceBuy] [int] NOT NULL ,
	[plPriceCons] [int] NOT NULL ,
	[plPriceVAT] [real] NOT NULL ,
	[plPriceAdd] [real] NOT NULL ,
	[plPrice] [int] NOT NULL ,
	[plInsTime] [datetime] NOT NULL ,
	[plInsUser] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[Products]    Script Date: 2006.01.05 22:50:25 ******/
CREATE TABLE [dbo].[Products] (
	[pid] [uniqueidentifier] NOT NULL ,
	[cid] [uniqueidentifier] NOT NULL ,
	[pName] [nvarchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[pDescription] [nvarchar] (500) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[pInsTime] [datetime] NOT NULL ,
	[pInsUser] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[pIsActive] [bit] NOT NULL ,
	[pComment] [nvarchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[pClTime] [int] NOT NULL ,
	[pImage] [varchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[pIsAction] [bit] NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[Sessions]    Script Date: 2006.01.05 22:50:25 ******/
CREATE TABLE [dbo].[Sessions] (
	[sid] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[uid] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[loginTime] [datetime] NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[Store]    Script Date: 2006.01.05 22:50:25 ******/
CREATE TABLE [dbo].[Store] (
	[rowid] [bigint] IDENTITY (1, 1) NOT NULL ,
	[pid] [uniqueidentifier] NOT NULL ,
	[quantity] [int] NOT NULL ,
	[remainder] [int] NOT NULL ,
	[InsTime] [datetime] NOT NULL ,
	[InsUser] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[Suppliers]    Script Date: 2006.01.05 22:50:25 ******/
CREATE TABLE [dbo].[Suppliers] (
	[supid] [uniqueidentifier] NOT NULL ,
	[supName] [nvarchar] (150) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[supAddress] [nvarchar] (250) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[supCity] [nvarchar] (150) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[supCountry] [nvarchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[supPostCode] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[supContactPhone] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[supContactPerson] [nvarchar] (250) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[supContactEmail] [varchar] (150) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
```

```
	[supVATcode] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[supBUcode] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[supInsTime] [datetime] NOT NULL ,
	[supInsUser] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[Transactions]    Script Date: 2006.01.05 22:50:25 ******/
CREATE TABLE [dbo].[Transactions] (
	[tranid] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[sid] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[amount] [int] NOT NULL ,
	[insTime] [datetime] NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[Users]    Script Date: 2006.01.05 22:50:25 ******/
CREATE TABLE [dbo].[Users] (
	[uid] [uniqueidentifier] NOT NULL ,
	[uLogin] [nvarchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[uPassword] [nvarchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[uFName] [nvarchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[uLName] [nvarchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[uAddress] [nvarchar] (200) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[uCity] [nvarchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[uMail] [varchar] (255) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL
) ON [PRIMARY]
GO

/****** Object:  Table [dbo].[UsersBasket]    Script Date: 2006.01.05 22:50:25 ******/
CREATE TABLE [dbo].[UsersBasket] (
	[sid] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[pid] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[uid] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[pname] [varchar] (150) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[count] [int] NOT NULL ,
	[price] [int] NOT NULL ,
	[amount] [int] NOT NULL ,
	[bInsTime] [datetime] NOT NULL ,
	[state] [int] NOT NULL
) ON [PRIMARY]
GO

ALTER TABLE [dbo].[PrdCategory] WITH NOCHECK ADD
	CONSTRAINT [PK_Category] PRIMARY KEY  CLUSTERED
	(
		[cid]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[PriceList] WITH NOCHECK ADD
	CONSTRAINT [PK_PriceList] PRIMARY KEY  CLUSTERED
	(
		[rowid]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[Products] WITH NOCHECK ADD
	CONSTRAINT [PK_Products] PRIMARY KEY  CLUSTERED
	(
		[pid]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[Store] WITH NOCHECK ADD
	CONSTRAINT [PK_Store] PRIMARY KEY  CLUSTERED
	(
		[rowid]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[Suppliers] WITH NOCHECK ADD
	CONSTRAINT [PK_Suppliers] PRIMARY KEY  CLUSTERED
	(
		[supid]
	)  ON [PRIMARY]
GO

ALTER TABLE [dbo].[Transactions] WITH NOCHECK ADD
	CONSTRAINT [PK_transactions] PRIMARY KEY  CLUSTERED
```

```
	(
		[tranid]
	) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Users] WITH NOCHECK ADD
	CONSTRAINT [PK_Users] PRIMARY KEY CLUSTERED
	(
		[uid]
	) ON [PRIMARY]
GO

ALTER TABLE [dbo].[PrdCategory] ADD
	CONSTRAINT [DF_Category_cid] DEFAULT (newid()) FOR [cid],
	CONSTRAINT [DF_Category_cInsTime] DEFAULT (getdate()) FOR [cInsTime],
	CONSTRAINT [DF_Category_cInsUser] DEFAULT (suser_sname()) FOR [cInsUser],
	CONSTRAINT [DF_Category_cIsActive] DEFAULT (1) FOR [cIsActive],
	CONSTRAINT [DF_Category_cClTime] DEFAULT (0) FOR [cClTime]
GO

ALTER TABLE [dbo].[PrdDetails] ADD
	CONSTRAINT [DF_PrdDetails_insTime] DEFAULT (getdate()) FOR [insTime],
	CONSTRAINT [DF_PrdDetails_insUser] DEFAULT (suser_sname()) FOR [insUser],
	CONSTRAINT [DF_PrdDetails_dIsActive] DEFAULT (1) FOR [dIsActive]
GO

 CREATE  INDEX [IX_PrdDetails] ON [dbo].[PrdDetails]([pid]) ON [PRIMARY]
GO

ALTER TABLE [dbo].[PriceList] ADD
	CONSTRAINT [DF_PriceList_plInsTime] DEFAULT (getdate()) FOR [plInsTime],
	CONSTRAINT [DF_PriceList_plInsUser] DEFAULT (suser_sname()) FOR [plInsUser]
GO

 CREATE  INDEX [IX_PriceList] ON [dbo].[PriceList]([pid], [supid]) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Products] ADD
	CONSTRAINT [DF_Products_pid] DEFAULT (newid()) FOR [pid],
	CONSTRAINT [DF_Products_pInsTime] DEFAULT (getdate()) FOR [pInsTime],
	CONSTRAINT [DF_Products_pInsUser] DEFAULT (suser_sname()) FOR [pInsUser],
	CONSTRAINT [DF_Products_pIsActive] DEFAULT (1) FOR [pIsActive],
	CONSTRAINT [DF_Products_pClTime] DEFAULT (0) FOR [pClTime],
	CONSTRAINT [DF_Products_pImage] DEFAULT ('blank.jpg') FOR [pImage],
	CONSTRAINT [DF_Products_pIsAction] DEFAULT (0) FOR [pIsAction]
GO

ALTER TABLE [dbo].[Sessions] ADD
	CONSTRAINT [DF_Sessions_loginTime] DEFAULT (getdate()) FOR [loginTime]
GO

 CREATE  INDEX [IX_Sessions] ON [dbo].[Sessions]([sid], [uid]) ON [PRIMARY]
GO

ALTER TABLE [dbo].[Store] ADD
	CONSTRAINT [DF_Store_quantity] DEFAULT (0) FOR [quantity],
	CONSTRAINT [DF_Store_InsTime] DEFAULT (getdate()) FOR [InsTime],
	CONSTRAINT [DF_Store_InsUser] DEFAULT (suser_sname()) FOR [InsUser]
GO

ALTER TABLE [dbo].[Suppliers] ADD
	CONSTRAINT [DF_Suppliers_supInsTime] DEFAULT (getdate()) FOR [supInsTime],
	CONSTRAINT [DF_Suppliers_supInsUser] DEFAULT (suser_sname()) FOR [supInsUser]
GO

ALTER TABLE [dbo].[Transactions] ADD
	CONSTRAINT [DF_transactions_insTime] DEFAULT (getdate()) FOR [insTime]
GO

ALTER TABLE [dbo].[Users] ADD
	CONSTRAINT [DF_Users_uid] DEFAULT (newid()) FOR [uid]
GO

ALTER TABLE [dbo].[UsersBasket] ADD
	CONSTRAINT [DF_UsersBasket_bInsTime] DEFAULT (getdate()) FOR [bInsTime],
	CONSTRAINT [DF_UsersBasket_state] DEFAULT (0) FOR [state]
GO

 CREATE  INDEX [IX_UsersBasket] ON [dbo].[UsersBasket]([sid], [pid], [uid]) ON [PRIMARY]
GO
```

```
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.uIsAuthenticate    Script Date: 2006.01.05 22:50:25 ******/
CREATE PROCEDURE uIsAuthenticate
	@sid VARCHAR(50),
	@fName NVARCHAR(50) OUTPUT,
	@lName NVARCHAR(100) OUTPUT,
	@err INT OUTPUT,
	@errDescr VARCHAR(150) OUTPUT
AS
IF EXISTS (
	SELECT	1
	FROM	[Sessions]
	WHERE	[sid] = @sid AND
		[loginTime] < DATEADD(mi,-15,GETDATE()))
BEGIN
	SET @err = 1
	SET @errDescr = 'Session is timeout'
END
ELSE
IF EXISTS (
	SELECT	1
	FROM	[Sessions]
	WHERE	[sid] = @sid AND
		[loginTime] >= DATEADD(mi,-15,GETDATE()))
BEGIN
	SET @err = 0
	SET @errDescr = ''
	SELECT	@fName = [u].[uFName],
		@lName = [u].[uLName]
	FROM	[Sessions] AS [s]
	INNER JOIN
		[Users] AS [u]
	ON	[s].[uid] = [u].[uid]
	WHERE	[s].[sid] = @sid AND
		[s].[loginTime] >= DATEADD(mi,-15,GETDATE())
END
ELSE
BEGIN
	SET @err = 2
	SET @errDescr = 'Not started session'
END
RETURN

GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.uLogOut    Script Date: 2006.01.05 22:50:25 ******/
CREATE PROCEDURE uLogOut
@sid VARCHAR(50)
AS
DELETE FROM
	[Sessions]
WHERE	[sid] = @sid

GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
```

```
/****** Object:  Stored Procedure dbo.uLogin    Script Date: 2006.01.05 22:50:25 ******/

CREATE  PROCEDURE uLogin
        @uLogin NVARCHAR(50),
        @uPassword NVARCHAR(255),
        @sid VARCHAR(50),
        @err INT OUTPUT,
        @errDescr VARCHAR(150) OUTPUT
AS
DECLARE @uid UNIQUEIDENTIFIER
IF NOT EXISTS (
        SELECT  1
        FROM    [Users]
        WHERE   [uLogin] = @uLogin AND
                [uPassword] = @uPassword
        )
BEGIN
        SET @err = 1
        SET @errDescr = 'User not found'
END
ELSE
BEGIN
        SELECT  @uid = [uid]
        FROM    [Users]
        WHERE   [uLogin] = @uLogin AND
                [uPassword] = @uPassword
        INSERT INTO
                [Sessions]
                ([sid],
                [uid])
        SELECT  @sid,
                @uid
        IF @@ERROR <> 0
        BEGIN
                SET @err = 2
                SET @errDescr = 'Error with sessions'
        END
        ELSE
        BEGIN
                SET @err = 0
                SET @errDescr = ''
        END
END
RETURN


GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.GUI_InsNewCategory    Script Date: 2006.01.05 22:50:25 ******/
CREATE  PROCEDURE GUI_InsNewCategory
        @cName NVARCHAR(100),
        @cDescription NVARCHAR(1000),
        @cComment VARCHAR(2000),
        @err INT OUTPUT,
        @errDescr VARCHAR(255) OUTPUT
AS
BEGIN TRANSACTION
INSERT INTO
        [PrdCategory]
        ([cName],
        [cDescription],
        [cComment])
SELECT  @cName,
        @cDescription,
        @cComment
IF @@ERROR <> 0 --OR @@ROWCOUNT <> 0
BEGIN
        ROLLBACK TRAN
        SET @err = 1
        SET @errDescr = 'Not inserted category'
```

```
END
ELSE
BEGIN
	COMMIT TRAN
	SET @err = 0
	SET @errDescr = ''
END
RETURN		@err

GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.GUI_InsNewDetail    Script Date: 2006.01.05 22:50:25 ******/
CREATE PROCEDURE GUI_InsNewDetail
	@pid UNIQUEIDENTIFIER,
	@name NVARCHAR(255),
	@value NVARCHAR(1000),
	@err INT OUTPUT,
	@errDescr VARCHAR(255) OUTPUT
AS
BEGIN TRANSACTION
INSERT INTO
	[PrdDetails]
	([pid],
	[dName],
	[dValue])
SELECT	@pid,
	@name,
	@value
IF @@ERROR <> 0
BEGIN
	ROLLBACK TRAN
	SET @err = 1
	SET @errDescr = 'Not inserted to PrdDetails'
END
ELSE
BEGIN
	SET @err = 0
	SET @errDescr = ''
	COMMIT TRAN
END
RETURN @err

GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.GUI_InsNewProduct    Script Date: 2006.01.05 22:50:25 ******/
CREATE	PROCEDURE GUI_InsNewProduct
	@cid UNIQUEIDENTIFIER,
	@pName NVARCHAR(255),
	@pDescription NVARCHAR(500),
	@pComment VARCHAR(255),
	@pImage VARCHAR(255),
	@quantity INT,
	@supid UNIQUEIDENTIFIER,
	@prBuy INT,
	@prCons INT,
	@prVAT REAL,
	@prADD REAL,
	@err INT OUTPUT,
	@errDescr VARCHAR(255) OUTPUT
AS
```

```
DECLARE		@pid UNIQUEIDENTIFIER
SET @pid = NEWID()
BEGIN TRANSACTION
INSERT INTO
		[Products]
		([pid],
		[cid],
		[pName],
		[pDescription],
		[pComment],
		[pImage])
SELECT	@pid,
		@cid,
		@pName,
		@pDescription,
		@pComment,
		@pImage

IF @@ERROR <> 0
BEGIN
		ROLLBACK TRAN
		SET @err = 1
		SET @errDescr = 'Not insert product'
END

INSERT INTO
		[Store]
		([pid],
		[quantity],
		[remainder])
SELECT	@pid,
		@quantity,
		@quantity
IF @@ERROR <> 0
BEGIN
		ROLLBACK TRAN
		SET @err = 2
		SET @errDescr = 'Not insert to Store'
END

INSERT INTO
		[priceList]
		([pid],
		[supid],
		[plPriceBuy],
		[plPriceCons],
		[plPriceVAT],
		[plPriceAdd],
		[plPrice])
SELECT	@pid,
		@supid,
		@prBuy,
		@prCons,
		@prVAT,
		@prADD,
		CAST((@prBuy*(1+@prADD/100)-@prCons)*(1+@prVAT/100) AS INT)
IF @@ERROR <> 0
BEGIN
		ROLLBACK TRAN
		SET @err = 3
		SET @errDescr = 'Not insert to priceList'
END

SET @err = 0
SET @errDescr = ''
COMMIT TRAN
RETURN		@err
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.GetCategory    Script Date: 2006.01.05 22:50:25 ******/
```

```
CREATE PROCEDURE GetCategory
	@xml NVARCHAR(1000)
AS
IF @xml = ''
	SELECT	TOP 15
		[cid],
		[cName],
		[cDescription]
	FROM	[PrdCategory]
	WHERE	[cIsActive] = 1
	ORDER BY
		[cName] ASC
ELSE
	SELECT	[cid],
		[cName],
		[cDescription]
	FROM	[PrdCategory]
	WHERE	[cName] = @xml
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.GetMVPProduct    Script Date: 2006.01.05 22:50:25 ******/
CREATE  PROCEDURE GetMVPProduct
	@xml VARCHAR(1000)
AS
IF @xml = 'min'
SELECT	TOP 1
	[p].[pid],
	[s].[remainder] [likutis],
	CAST([pl].[plPriceCons] AS MONEY)/100 + 0.0001 [nuolaida],
	CAST([pl].[plPrice] AS MONEY)/100 + 0.0001 [kaina],
	[p].[pName],
	[p].[pDescription],
	[p].[pImage],
	[c].[cName]
FROM	[Store] AS [s]
INNER JOIN
	[PriceList] AS [pl]
ON	[s].[pid] = [pl].[pid]
INNER JOIN
	[Products] AS [p]
ON	[s].[pid] = [p].[pid]
INNER JOIN
	[PrdCategory] AS [c]
ON	[p].[cid] = [c].[cid]
WHERE	[p].[pIsActive] = 1 AND
	[c].[cIsActive] = 1 AND
	[s].[remainder] > 0
ORDER BY
	[pl].[plPrice] ASC

IF @xml = 'pop'
SELECT	TOP 1
	[p].[pid],
	[s].[remainder] [likutis],
	CAST([pl].[plPriceCons] AS MONEY)/100 + 0.0001 [nuolaida],
	CAST([pl].[plPrice] AS MONEY)/100 + 0.0001 [kaina],
	[p].[pName],
	[p].[pDescription],
	[p].[pImage],
	[c].[cName]
FROM	[Store] AS [s]
INNER JOIN
	[PriceList] AS [pl]
ON	[s].[pid] = [pl].[pid]
INNER JOIN
	[Products] AS [p]
ON	[s].[pid] = [p].[pid]
INNER JOIN
	[PrdCategory] AS [c]
ON	[p].[cid] = [c].[cid]
```

```
WHERE   [p].[pIsActive] = 1 AND
        [c].[cIsActive] = 1 AND
        [s].[remainder] > 0
ORDER BY
        [p].[pClTime] DESC

IF @xml = 'new'
SELECT  TOP 1
        [p].[pid],
        [s].[remainder] [likutis],
        CAST([pl].[plPriceCons] AS MONEY)/100 + 0.0001 [nuolaida],
        CAST([pl].[plPrice] AS MONEY)/100 + 0.0001 [kaina],
        [p].[pName],
        [p].[pDescription],
        [p].[pImage],
        [c].[cName]
FROM    [Store] AS [s]
INNER JOIN
        [PriceList] AS [pl]
ON      [s].[pid] = [pl].[pid]
INNER JOIN
        [Products] AS [p]
ON      [s].[pid] = [p].[pid]
INNER JOIN
        [PrdCategory] AS [c]
ON      [p].[cid] = [c].[cid]
WHERE   [p].[pIsActive] = 1 AND
        [c].[cIsActive] = 1 AND
        [s].[remainder] > 0
ORDER BY
        [p].[pInsTime] DESC

IF @xml = 'act'
SELECT  TOP 1
        [p].[pid],
        [s].[remainder] [likutis],
        CAST([pl].[plPriceCons] AS MONEY)/100 + 0.0001 [nuolaida],
        CAST([pl].[plPrice] AS MONEY)/100 + 0.0001 [kaina],
        [p].[pName],
        [p].[pDescription],
        [p].[pImage],
        [c].[cName],
        [p].[pComment]
FROM    [Store] AS [s]
INNER JOIN
        [PriceList] AS [pl]
ON      [s].[pid] = [pl].[pid]
INNER JOIN
        [Products] AS [p]
ON      [s].[pid] = [p].[pid]
INNER JOIN
        [PrdCategory] AS [c]
ON      [p].[cid] = [c].[cid]
WHERE   [p].[pIsActive] = 1 AND
        [c].[cIsActive] = 1 AND
        [s].[remainder] > 0 AND
        [p].[pIsAction] = 1
ORDER BY
        [p].[pInsTime] DESC
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.GetProduct    Script Date: 2006.01.05 22:50:25 ******/
CREATE  PROCEDURE GetProduct
        @xml NVARCHAR(1000)
AS
IF @xml = ''
        SELECT  TOP 5
                [p].[pid],
                [c].[cName],
                [p].[pName],
```

```
                [p].[pDescription],
                [p].[pImage],
                [s].[quantity],
                CAST([pl].[nuolaida]/[pl].[kiekis] AS MONEY)/100,
                CAST([pl].[kaina]/[pl].[kiekis] AS MONEY)/100
        FROM    [Products] AS [p]
        INNER JOIN
        (SELECT [pid],
                SUM([quantity]) AS [quantity]
        FROM    [Store]
        GROUP BY
                [pid]) AS [s]
        ON      [p].[pid] = [s].[pid]
        INNER JOIN
        (SELECT [pid],
                COUNT(1) AS [kiekis],
                SUM([plPriceCons]) AS [nuolaida],
                SUM([plPrice]) AS [kaina]
        FROM    [PriceList]
        GROUP BY
                [pid]) AS [pl]
        ON      [p].[pid] = [pl].[pid]
        INNER JOIN
                [PrdCategory] AS [c]
        ON      [p].[cid] = [c].[cid]
        WHERE   [p].[pIsActive] = 1 AND
                [c].[cIsActive] = 1
        ORDER BY
                [p].[pInsTime] DESC

ELSE
        SELECT
                [p].[pid],
                [c].[cName],
                [p].[pName],
                [p].[pDescription],
                [p].[pImage],
                [s].[quantity],
                CAST([pl].[nuolaida]/[pl].[kiekis] AS MONEY)/100,
                CAST([pl].[kaina]/[pl].[kiekis] AS MONEY)/100
        FROM    [Products] AS [p]
        INNER JOIN
        (SELECT [pid],
                SUM([quantity]) AS [quantity]
        FROM    [Store]
        GROUP BY
                [pid]) AS [s]
        ON      [p].[pid] = [s].[pid]
        INNER JOIN
        (SELECT [pid],
                COUNT(1) AS [kiekis],
                SUM([plPriceCons]) AS [nuolaida],
                SUM([plPrice]) AS [kaina]
        FROM    [PriceList]
        GROUP BY
                [pid]) AS [pl]
        ON      [p].[pid] = [pl].[pid]
        INNER JOIN
                [PrdCategory] AS [c]
        ON      [p].[cid] = [c].[cid]
        WHERE   [c].[cid] = @xml AND
                [p].[pIsActive] = 1 AND
                [c].[cIsActive] = 1
        ORDER BY
                [p].[pInsTime] DESC
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.GetProductDetail    Script Date: 2006.01.05 22:50:26 ******/
CREATE PROCEDURE GetProductDetail
@xml NVARCHAR(1000),
```

```
@img VARCHAR(255) =null OUTPUT,
@name VARCHAR(255) = null OUTPUT,
@descr VARCHAR(1000) =null OUTPUT,
@price MONEY = NULL OUTPUT
AS
SELECT @img = [pImage],
        @name = [pName],
        @descr = [pDescription]
FROM    [Products]
WHERE   [pid] = @xml

SELECT @price =  CAST(AVG([plPrice]) AS MONEY)/100
FROM    [PriceList]
WHERE   [pid] = @xml

SELECT  [dName],
        [dValue]
FROM    [PrdDetails]
WHERE   [pid] = @xml
ORDER BY
        [insTime] ASC
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.GetUserBasket    Script Date: 2006.01.05 22:50:26 ******/
CREATE PROCEDURE GetUserBasket
        @xml VARCHAR(1000)
AS
SELECT  ub.[pid],
        ub.[pname],
        ub.[count],
        CAST(ub.[price] AS MONEY)/100,
        CAST(ub.[amount] AS MONEY)/100,
        p.pimage
FROM    [UsersBasket] AS ub
INNER JOIN
        [Products] AS p
ON      ub.pid=p.pid
WHERE   ub.[sid] = @xml AND
        ub.[count] > 0 AND ub.state = 0
ORDER BY
        ub.[bInsTime] DESC
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS OFF
GO

/****** Object:  Stored Procedure dbo.InsToTrans    Script Date: 2006.01.05 22:50:26 ******/
CREATE PROCEDURE InsToTrans
@xml VARCHAR(50),
@tranid NVARCHAR(50) OUTPUT,
@amount INT OUTPUT
AS

SELECT @amount = SUM(amount)
FROM    UsersBasket
WHERE   sid = @xml and state = 0

IF EXISTS (
        SELECT 1
        FROM    [transactions]
        WHERE sid = @xml)
BEGIN
        SET @tranid = -1
```

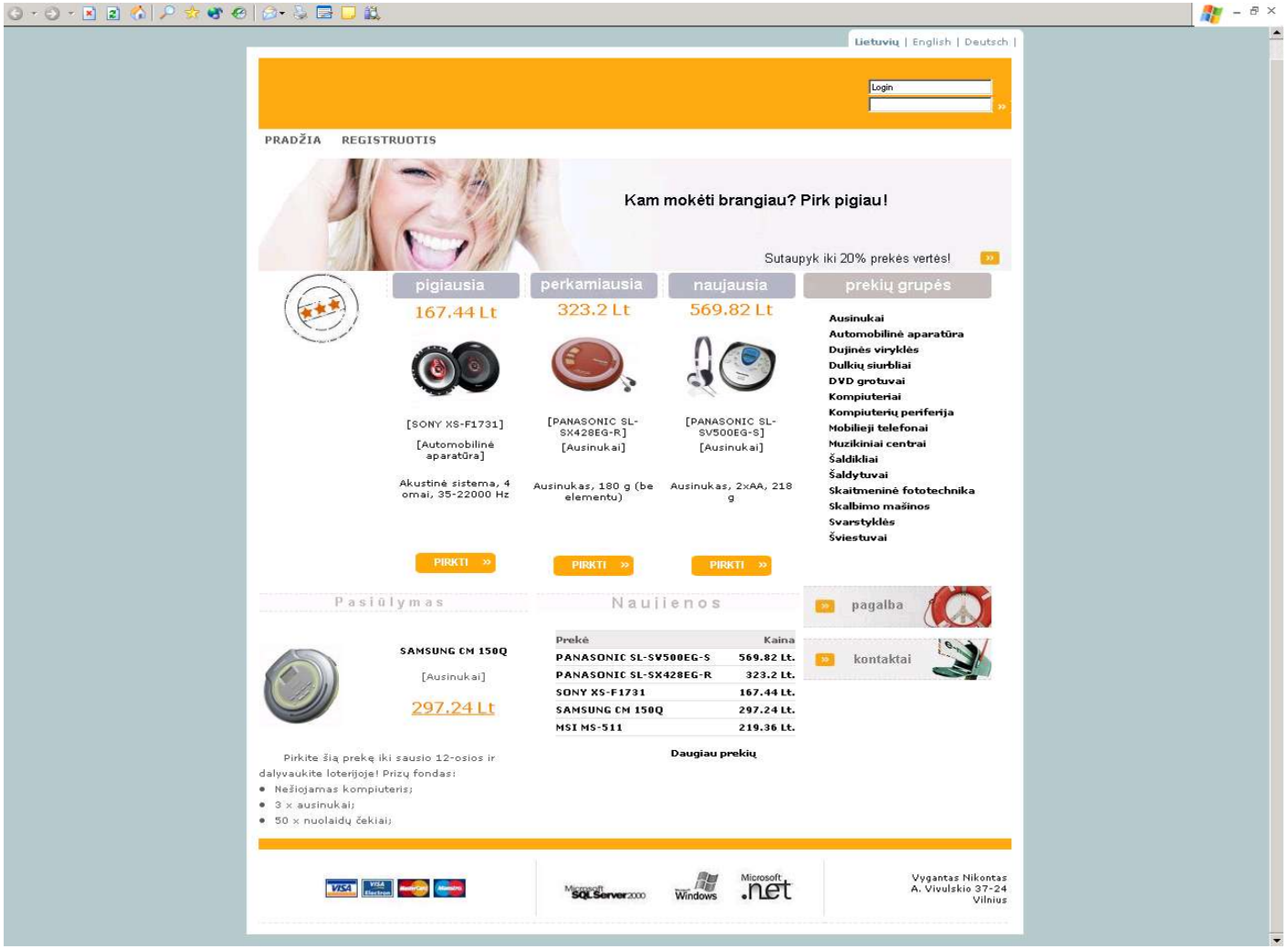
```
END
ELSE
BEGIN
SET @tranid = NEWID()

INSERT INTO
        [Transactions]
        ([tranid],
        [sid],
        [amount])
SELECT  @tranid,
                @xml,
                @amount

UPDATE Products SET pclTime = pclTime + 1
FROM UsersBasket as u
WHERE   Products.pid = u.pid AND
        sid = @xml


UPDATE UsersBasket set state = 1 where sid = @xml
END
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.SetProdBasket    Script Date: 2006.01.05 22:50:26 ******/
CREATE PROCEDURE SetProdBasket
@pid VARCHAR(50),
@sid VARCHAR(50),
@count INT
AS
IF NOT EXISTS
        (SELECT  1
        FROM    [UsersBasket]
        WHERE   [sid] = @sid AND
                [pid] = @pid)
BEGIN
        INSERT INTO
                [UsersBasket]
                ([sid],
                [pid],
                [uid],
                [pname],
                [count],
                [price],
                [amount])
        SELECT  @sid,
                @pid,
                s.[uid],
                p.[pname],
                @count,
                pl.plPrice,
                @count * pl.plPrice
        FROM    [Sessions] AS s
        INNER JOIN
                [Products] AS p
        ON      1 = 1
        INNER JOIN
                [PriceList] AS pl
        ON      p.pid = pl.pid
        WHERE   s.sid = @sid AND
                p.pid = @pid
END
ELSE
BEGIN
        UPDATE  [UsersBasket]
        SET     [count] = [count] + @count,
                [amount] = [price] * ([count] + @count)
        WHERE   [sid] = @sid AND
                [pid] = @pid
END
```

```
GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO

SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO

/****** Object:  Stored Procedure dbo.uReg    Script Date: 2006.01.05 22:50:26 ******/
CREATE PROCEDURE dbo.uReg
	@login NVARCHAR(50),
	@passw NVARCHAR(255),
	@fname NVARCHAR(100),
	@lname NVARCHAR(100),
	@address NVARCHAR(200),
	@city NVARCHAR(50),
	@mail VARCHAR(255),
	@sid VARCHAR(50),
	@err INT OUTPUT,
	@errDescr VARCHAR(150) OUTPUT
AS
IF	(LEN(@fname) < 3 OR
	LEN(@lname) < 3 OR
	LEN(@login) < 3 OR
	LEN(@passw) < 3 OR
	LEN(@address) < 3 OR
	LEN(@city) < 3 OR
	LEN(@mail) < 6)
BEGIN
	SET @err = 1
	SET @errDescr = 'Not all data inserted to textboxes'
	RETURN 1
END
IF NOT EXISTS (
	SELECT	1
	FROM	[Users]
	WHERE	[uLogin] = @login
)
BEGIN
	BEGIN TRAN
	INSERT INTO
		[Users]
		([uLogin],
		[uPassword],
		[uFName],
		[uLName],
		[uaddress],
		[ucity],
		[umail])
	SELECT	@login,
		@passw,
		@fname,
		@lname,
		@address,
		@city,
		@mail
	IF @@ERROR <> 0 OR @@ROWCOUNT = 0
	BEGIN
		ROLLBACK TRAN
		SET @err = 2
		SET @errDescr = 'Not inserted to Users'
		RETURN 2
	END
	ELSE
	BEGIN
		DECLARE @RC int
		DECLARE @uLogin nvarchar(50)
		DECLARE @uPassword nvarchar(255)
		EXEC @RC = [dbo].[uLogin]
			@uLogin = @login,
			@uPassword = @passw,
			@sid = @sid,
			@err = @err OUTPUT ,
			@errDescr = @errDescr OUTPUT
		IF @@ERROR <> 0 OR @RC <> 0 OR @err <> 0
```

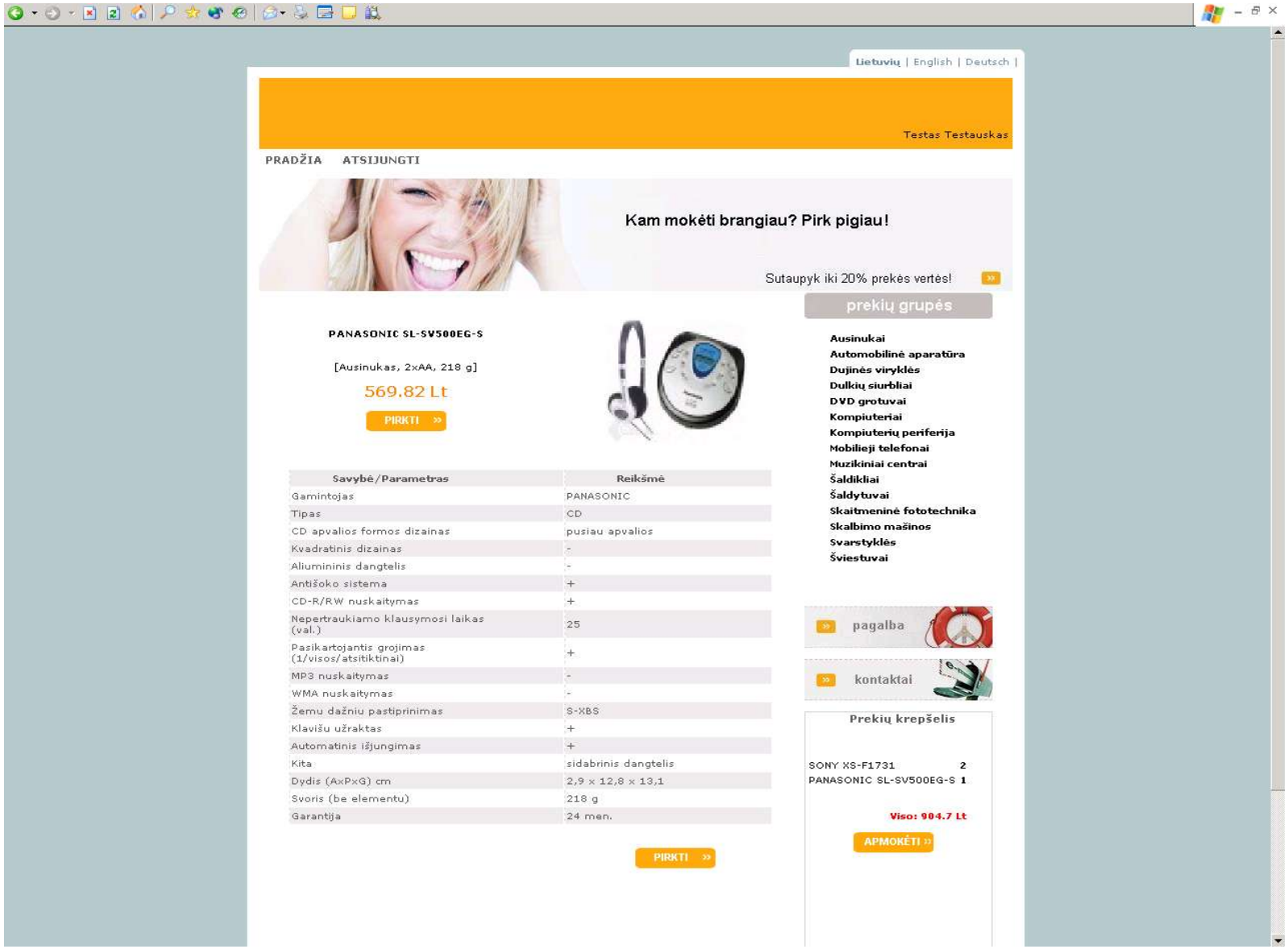
```
                        BEGIN
                                ROLLBACK TRAN
                                SET @err = 3
                                SET @errDescr = 'Not login'
                                RETURN 3
                        END
        END
END
ELSE
BEGIN
        SET @err = 4
        SET @errDescr = 'User exists'
        RETURN 4
END
COMMIT TRAN
SET @err = 0
SET @errDescr = ''
RETURN 0

GO

SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

8.2. Vartotojų ekranų formos ir pavyzdžiai

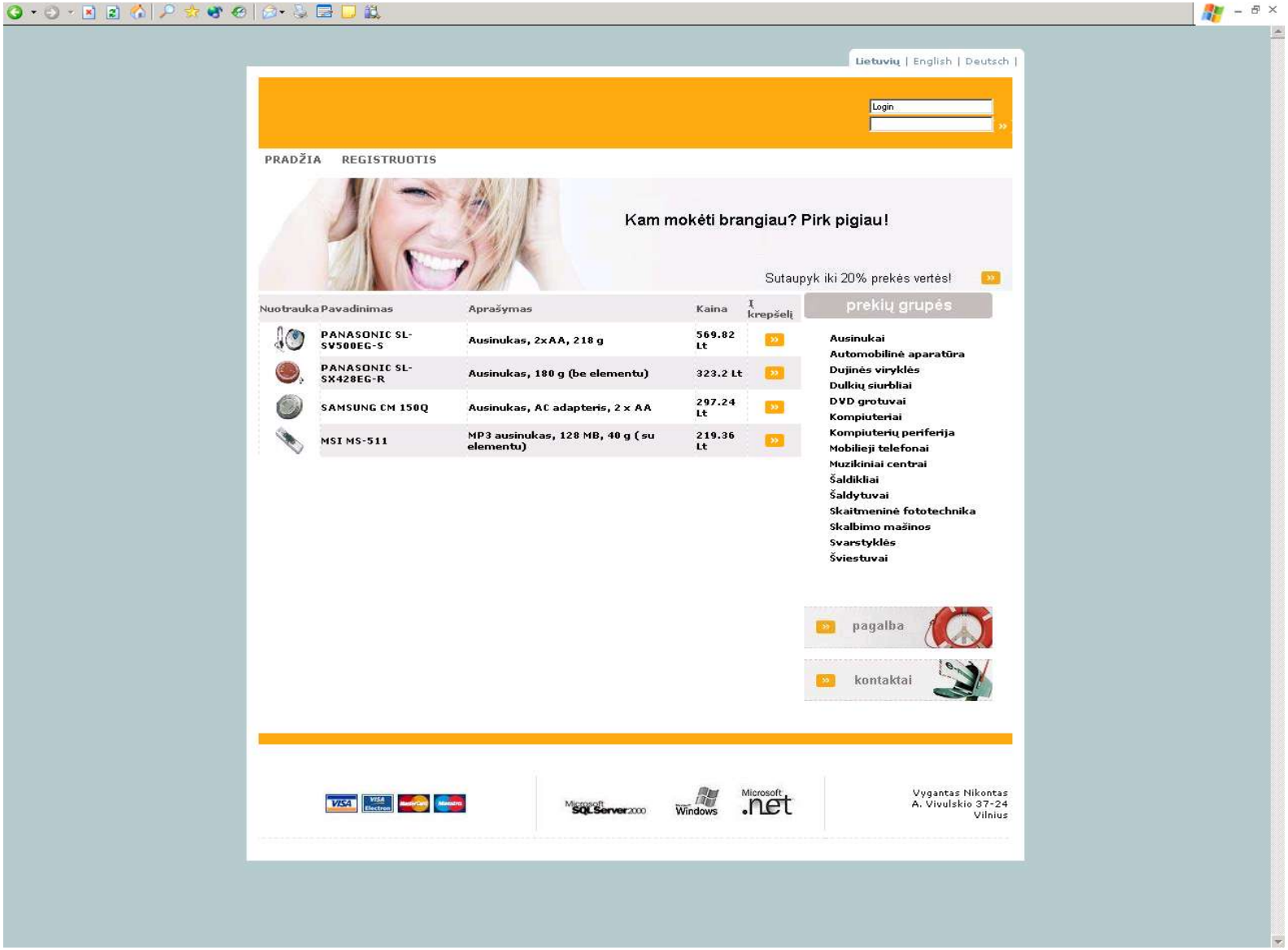
Pagrindinis langas:



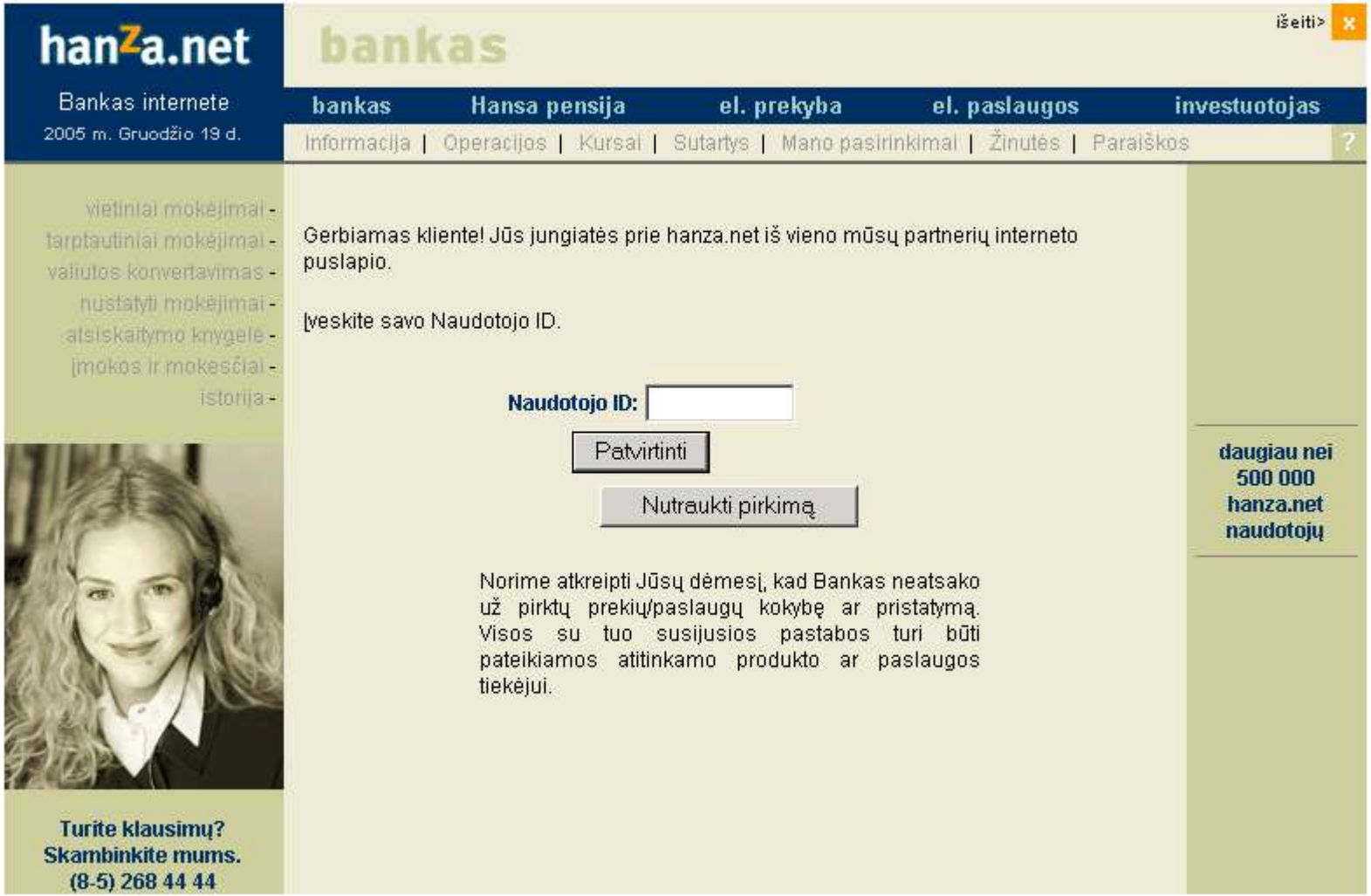
Prekių sąrašo langas:



Detalus prekės informacijos puslapis:



Banko IS prisijungimo forma:



Vieno iš AC duomenų užpildymo forma: