KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACINIŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

VOLDEMARAS ŽITKUS

INTERNETINIŲ SISTEMŲ PRITAIKYMO IŠMANIESIEMS ĮRENGINIAMS GALIMYBIŲ TYRIMAS

Magistro darbas

Darbo vadovas
prof.R. Butleris

KAUNAS, 2013

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACINIŲ SISTEMŲ INŽINERIJOS STUDIJŲ PROGRAMA

VOLDEMARAS ŽITKUS

INTERNETINIŲ SISTEMŲ PRITAIKYMO IŠMANIESIEMS ĮRENGINIAMS GALIMYBIŲ TYRIMAS

Magistro darbas

Darbo vadovas:
prof.R. Butleris
2013-05-24

Konsultantas:
lekt. T. Danikauskas
2013-05-24

Recenzentas:
doc. dr. S. Drąsutis
2013-05-24

Atliko:
IFM-1/4 gr. Studentas
Voldemaras Žitkus
2013-05-24

KAUNAS, 2013

AUTORIŲ GARANTINIS RAŠTAS
DĖL PATEIKIAMO KŪRINIO

20.. - ..... - ..... d.
Kaunas

Autoriai,_____
(vardas, pavardė)

_____,

patvirtina, kad Kauno technologijos universitetui pateiktas baigiamasis bakalauro (magistro) darbas (toliau vadinama – Kūrinys)_____
(kūrinio pavadinimas)
_____

pagal Lietuvos Respublikos autorių ir gretutinių teisių įstatymą yra originalus ir užtikrina, kad

1) jį sukūrė ir parašė Kūrinyje įvardyti autoriai;
2) Kūrinys nėra ir nebus įteiktas kitoms institucijoms (universitetams) (tiek lietuvių, tiek užsienio kalba);
3) Kūrinyje nėra teiginių, neatitinkančių tikrovės, ar medžiagos, kuri galėtų pažeisti kito fizinio ar juridinio asmens intelektinės nuosavybės teises, leidėjų bei finansuotojų reikalavimus ir sąlygas;
4) visi Kūrinyje naudojami šaltiniai yra cituojami (su nuoroda į pirminį šaltinį ir autorių);
5) neprieštarauja dėl Kūrinio platinimo visomis oficialiomis sklaidos priemonėmis.
6) atlygins Kauno technologijos universitetui ir tretiesiems asmenims žalą ir nuostolius, atsiradusius dėl pažeidimų, susijusių su aukščiau išvardintų Autorių garantijų nesilaikymu;
7) Autoriai už šiame rašte pateiktos informacijos teisingumą atsako Lietuvos Respublikos įstatymų nustatyta tvarka.

Autoriai

_____ _____
(vardas, pavardė) (parašas)

_____ _____
(vardas, pavardė) (parašas)

Turinys

Terminų ir santrumpų žodynas

IS– informacinė sistema.

OS– operacinė sistema.

PHP(*angl. Hypertext preprocessor*) –programavimo kalba skirta

Java– plataus panaudojimo programavimo kalba. Naudojama *„Android“* operacinėje sistemoje ir Eclipse įrankio įskiepių kūrimui.

Javascript–interpretuojama programavimo kalba, kuripaprastai veikia vartotojo naudojamoje interneto naršyklėje.

Flash – multimedijai skirta platforma.

HTML(*angl. Hypertext Markup Language*) – žymėjimo kalba skirta internetinių sistemų turinio pateikimui.

CSS (*angl. Cascading Style Sheets*) – kalba skirta nurodyti kitos kalbos (dažniausiai HTML) atvaizdavimo stiliu.

XML – duomenų struktūrų ir jų turinio aprašomoji kalba.

DBVS – duomenų bazių valdymo sistema.

NoSQL duomenų bazė– nereliacinė duomenų bazė.

CouchDB – nereliacinė duomenų bazė pasižyminti geru sinchronizacijos metodu.

MongoDB– nereliacinė duomenų bazė pasižymti sparčiu veikimu.

SQL duomenų bazė – reliacinė duomenų bazė.

SQLite– reliacinė duomenų bazė naudojama išmaniuosiuose įrenginiuose.

Eclipse – programinės įrangos kūrimo platforma.

Išmanusis įrenginys, išmanusis telefonas (*angl. Smartphone*) – mobilusis telefonas pasižymiantis geresnėmis techninemis charakteristikomis negu standartinis mobilusis telefonas.

Android –išmaniųjų įrenginių operacinė sistema.

Zend Framework – programavimo karkasas skirtas kurti internetines sistemas naudojant PHP programavimo kalbą.

Karkasas Android– karkasas skirtas kurti taikomasias programas operaciniai sistemai Android.

Programinis karkasas – programinės įrangos karkasas (struktūra) pateikiantis standartini funkcionalumą ir apibrėžiantis sistemos struktūrą.

MVC – programinės įrangos architektūrinis modelis atskiriantis informacijos pateikimą nuo vartotojo sąveiku su informacija.

Lentelių sąrašas

Paveikslų sąrašas

Įvadas

Kasdien augantis išmaniųjų įrenginių vartotojų skaičius ženkliai plečia ir siūlomų paslaugų, skirtų šiems įrenginiams skaičių. Šiuo metu ryškėja tendencija, kad kiekvieną naują, internetinę informacinę sistemą lydi ir išmaniesiems įrenginiams skirtas sprendimas. Galima būtų teigti, kad naujausių išmaniųjų įrenginių interneto naršyklės gana gerai susidoroja su esamos internetinės sistemos turinio vaizdavimu ir funkcijų palaikymu arba galima internetinių sistemų vartotojo sąsają pritaikyti, pasinaudojant vienu iš galimų sprendimų [1] ir taip išspręsti panaudojamumo problemas. Tačiau reikia nepamiršti, kad didžioji dauguma šiuolaikinių išmaniųjų įrenginių pasižymi įvairiomis funkcinėmis savybėmis, tokiomis kaip giroskopinis sensorius, globalios pozicijos nustatymo sistema, liečiamas ekranas, kurios suteikia šiems įrenginiams panaudojamumo pranašumą, palyginti su asmeniniais kompiuteriais. Todėl kalbant apie esamų internetinių sistemų pritaikymą išmaniesiems įrenginiams reikia įvertinti ir išnaudoti vartotojo sąsajos kūrimo bei funkcines galimybes. Pagrindinės problemos pritaikant internetines sistemas išmaniesiems įrenginiams kyla dėl skirtingų programinių platformų ir programinio kodo palaikymo sudėtingumo.

Šio darbo tikslas –pateikti sprendimą, kuris padėtų automatizuoti internetinių sistemų pritaikymo išmaniųjų įrenginių platformoms procesą, taip paspartinant programuotojų darbą bei sumažinat klaidų galimybę..

Siekiant įgyventinti užsibrėžtą tikslą buvo pasirinkti keli darbo konteksto apribojimai: pritaikoma internetinė sistema turi būti sukurta programinio karkaso *„Zend Framework“* pagrindu, o sistema pritaikomaišmaniesiems įrenginiams, kurių pagrindas yra operacinė sistema *„Android“*. Sukurto sprendimo eksperimentinis tyrimas buvo atliktas naudojant kolekcionieriams skirtą internetinę informacinę sistemą ir keletą kitų internetinių sistemų.

Summary

Feasibilty study of internet based systems adaptation for smartphones.

Currently smartphones are one of the fastest growing technologies. As with each popular technology there is noticeable demand for systems working on smartphones. But a lack of well defined methods on how to adapt existing internet systems for this new platform can be noticed.

First chapter overviews smartphone technology, ranging from usability to technical capabilities in rendering modern internet systems. Comparison is also made between programming frameworks of different platforms.

Second chapter covers proposed method for adapting internet systems to these new platforms.

Third and Fourth chapters cover an "Eclipse" plugin that is developed based on the proposed method in order to automate parts of the mechanical work required for adapting systems to new platforms. For experiment an internet website that focuses on collectors was used.

1. Tyrimo analizė

1.1. Tyrimo sritis ir objektas

Tyrimo sritis yra esamos informacinės sistemos dalinio funkcionalumo pritaikymo, mobiliesiems įrenginiams, galimybių tyrimas ir metodinių sprendimų pasiūlimas. Pagrindinis dėmesys skiriamas programinio kodo struktūros generavimui naujai platformai, pasinaudojant egzistuojančios sistemos kodo baze.

Tyrimo objektas yra kolekcionierių informacinė sistema, kuri bus pritaikoma išmaniesiems įrenginiamsremiantis jos programiniu kodu.

1.2. Tyrimo problema ir numatomas sprendimas

Šiame darbe susiduriama su keliomis problemomis:

- Sąlyginai ribotas interneto ryšys. Jeigu sistemoje saugojami asmeniniai duomenys, vartotojas gali norėti jais pasinaudoti ir tuomet, kai interneto ryšis jam yra nepasiekiamas. Dėl šios priežasties būtina pritaikant sistemą paruošti ir *„offline“* rėžimą, kuris leistų vartotojui naudotis sistema ir esant problemoms su ryšiu.
- Išmaniųjų įrenginių platformos palaiko ne visas duomenų bazes, kurios yra palaikomas stacionariųjų kompiuterių aplinkoje. Gali prireikti atlikti duomenų sinchronizavimą tarp skirtingų duomenų bazių, norint išlaikyti duomenų validumą.
- Išmanieji įrenginiai turi aibę daviklių ir išskirtinio funkcionalumo, todėl pritaikant sistemą neišvengiamai reikia atsižvelgti į šias naujas galimybes ir pritaikyti jas. Bendro sprendimo šiai problemai negalima pateikti, kadangi kiekvienai sistemai gali būti naudingos skirtingos funkcijos.
- Išmaniųjų įrenginių vartotojo sąsajos valdymas ir sąsajos elementai turi specifinius sprendimus, kurie nebūdingi stacionariems kompiuteriams. Todėl netikslingavartotojo sąsają tiesiogiai perkelti iš stacionarios aplinkos į išmaniojo įrenginio aplinką.

1.3. Tyrimo tikslas ir uždaviniai

Šio tyrimo tikslas yra pateikti sprendimą, kuris padėtų automatizuoti internetinių sistemų pritaikymo išmaniųjų įrenginių platformoms procesą, taip paspartinant programuotojų darbą bei sumažinat klaidų galimybę.

Darbo metu bus sprendžiami šie uždaviniai:

- Parengti internetinės informacinės sistemos pritaikymo išmaniesiems įrenginiams metodiką.
- Ištirti galimybędalinai automatizuoti reikiamų komponentų sukūrimą programiniam karkasui*„Android“* remiantis egzistuojančios sistemos komponentais.
- Remiantis parengta koncepcija sukurti programavimo aplinkose *„Eclipse“*įskiepį, kuris sugeneruotų dalį naujos sistemos programinio kodo struktūros.

1.4. Analizės tikslas ir metodai

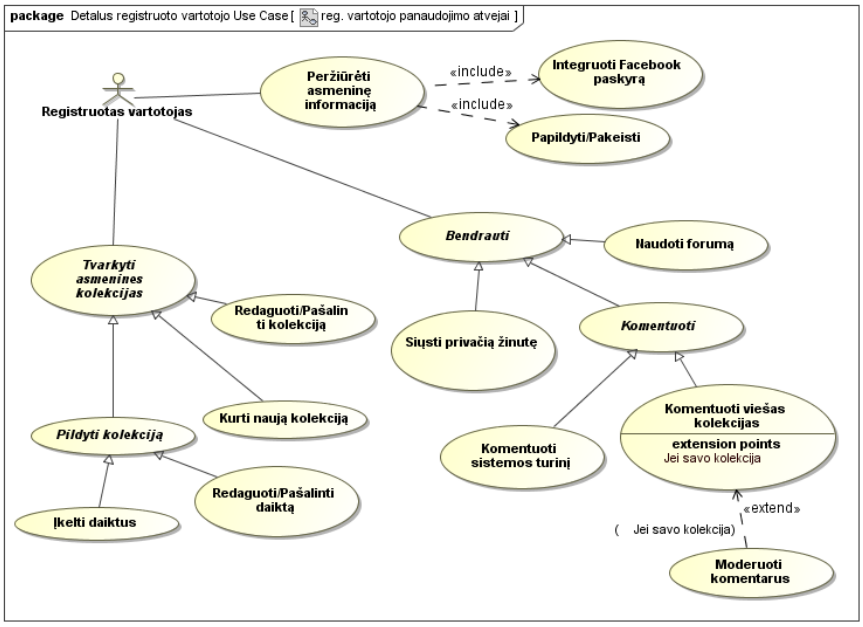
Šios analizės tikslas yra išanalizuoti išmaniuosius įrenginius, jų specifiką ir operacinę sistemą *„Android“*. Analizuojami programavimo karkasai – *„Zend Framework“* (kuriuo sukurta eksperimentui naudojama sistema) ir *„Android“*, nustatomi jų panašumai. Taip pat analizuojami vartotojų tipai į kuriuos taikomas sistemos pritaikymas išmaniųjų įrenginių platformai.

Analizės metu bus atliekama literatūros šaltinių, kuriuose sprendžiami panašaus pobūdžio uždaviniai, bei atliekama esamų sprendimų lyginamoji analizė.

1.5. Tyrimo eksperimentinė sistema

Tyrimo metu ekperimentui naudojama internetinė kolekcionierių informacinė sistema (toliau IS)sukurta programinio karkaso *„Zend Framework“* pagrindu.

Internetinė kolekcionerių IS iš dalies pasižymi standartiniu internetinių informacinių sistemų funkcionalumu. Pagrindinio, registruoto vartotojo, panaudojimo atvejų modelis pavaizduotas 1 paveikslėlyje.



1pav. Ekperimentinės informacinės sistemos panaudojimo atvejų diagrama

Išskirtinis panaudojimo atvejis šioje IS yra „Tvarkyti asmenines kolekcijas“. Daugumos kolekcionierių sistemų pagrindinis apribojimas yra ribotos kolekcijų kūrimo ir meta informacijos kaupimo galimybės– kitu atveju reikėtų naudoti sudėtingą duomenų bazės struktūrą ir daug neefektyvių „*SQL“* užklausų. Analizuojamoje kolekcionierių sistemoje ši bėda išsprendžiama naudojant dokumentinę*„NoSQL“* duomenų bazę *„MongoDB“*. Ji pasižymi laisvesne struktūra negu reliacinės duomenų bazės, kas leidžia vartotojui duoti daugiau laisvės kolekcijų kūrime ir meta informacijos kaupime.

Detali sistemos reikalavimų specifikacija pateikiama pirmame priede.

1.6. Techninė analizė

1.6.1. Išmanieji įrenginiai

Sąvoka išmanusis telefonas neturi aiškaus apibrėžimo, tai daugiau marketingo terminas apibūdinantis technologiškai naujausius ir pajėgiausius mobiliuosius įrenginius.Kuriant, ar pritaikant, IS šiems įrenginiams svarbu atlikti dviejų lygių analizę– išmaniųjų telefonų technines galimybės, ir jų vartotojo sąsajos privalumus, trūkumus bei unikalias savybes, kurių kiti įrenginiai neturi.

1.6.1.1. Techninės charakteristikos

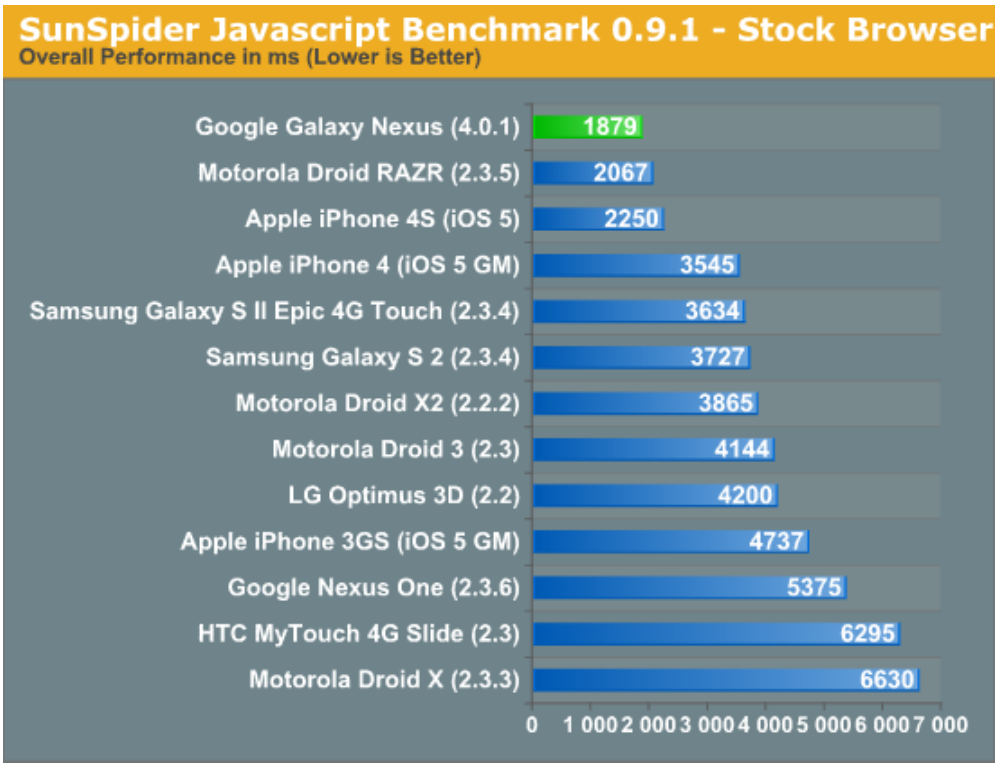
Išmanieji įrenginiai techniškai tobulėja kiekvieną dieną, tadtampa vis patrauklesnivartotojams, beto, vyksta ir pačių įrenginių inovacijų progresas. Bet vis dar galima teigti, jog vidutinis išmanusis įrenginys gerokai nusileidžia pajėgumu vidutinam stacionariam kompiuteriui. Šiuo metu asmeniniai kompiuteriai lengvai

apdoroja internetinių informacinių sistemų pateikiamą turinį, o tuo tarpu, nemažai rinkoje esančių išmaniųjų įrenginių negali nepriekaištingai susidoroti su pažangių internetinių sistemų pateikiamu turiniu. Esamas internetines IS aktualu pritaikyti ir optimizuoti išmaniesiems įrenginiams, sukuriant taikomąsias programėles, ir taip siekiant išlaikyti sistemos patrauklumą, beiišlikti tarp išmaniųjų įrenginių naudotojų.

Pritaikant internetinę sistemą išmaniesiems įrenginiams svarbiausia techninė charakteristika yra išmaniųjų telefonų našumas internetinių puslapių apdorojime. Šiuo metu, didelę naršyklės apkrovos dalį sudaro kliento pusėje vykdomas kodas–*„Flash“*, *„JavaScript“* ir kitos panašios technologijos.

SunSpider testas[2]yra vienas iš labiausiai paplitusių testų *„JavaScript“* apdorojimo greičio matavimui. Jo rezultatai pateikiami 2 paveikslėlyje. Naujesnių išmaniųjų telefonų matavimai svyruoja nuo 1400 iki 2300 ms. Kaip palyginimui,naujausių stacionarių kompiuterių ir jų naršyklių matavimai svyruoja nuo 300 iki 500 ms, oplačiai kritikuojama ir pasenusi naršyklė *„Internet Explorer 8“*nuo 3000 iki 3500 ms. Šie matavimai rodo, jog vidutiniai išmanieji telefonai neapdoroja interneto puslapių taip greitai kaip stacionarūs (ar nešiojamieji) kompiuteriai. Galima tikėtis, jog ateinančiais metais išmaniųjų telefonų technologijos tobulės ir testų rezultatai gerės, tačiau rizikinga kurti informacinę sistemą „rytojui“.

„Flash“ technologija šiuo metu nėra plačiai naudojamainternetinėse IS dėl lėto veikimo ir nedidelio pridėtinės vertės sukūrimo. Deja, priklausomai nuo sistemos specifikos ir alternatyvų trūkumo, jos panaudojimas dažnai neišvengiamas. Ne visos išmaniųjų įrenginių naršyklės (ir patys įrenginiai) pilnai palaiko šią technologiją, o palaikančios negali pasigirti didele sparta. Dėl šių priežaščių reikia, kiek įmanoma, minimizuoti šios technologijos naudojimą, pritaikant sistemas išmaniesiems įrenginiams.



2 pav. SunSpider testo rezultatai [2].

Vienas iš galimų būdų pagreitinti internetinio puslapio krovimą išmaniajam telefonui yra turinio suskirstymas į blokus. Kadangi išmaniojo telefono ekranas yra nedidelis, vartotojas vienu metu matys nedidelę puslapio dalį. Pirmiausia, reikėtų užkrauti tik dalį puslapio, kuri iškarto būtų matoma vartotojui. Kol vartotojas peržiūri pateiktą turinį,įrenginys siunčiasi sekantį bloką, kurį parsiuntus pirmojo bloko gale atsiranda nuoroda – ją paspaudus pridedamas naujai parsiųstas blokas ir siunčiamas sekantis. Šį metodą naudoja *„Facebook“*(ir kiti socialiniai tinklai ar sistemos turinčios didelį kiekį turinio) vartotojo „sienoje“ rodydama *„View older posts“* mygtuką.

1.6.1.2. Vartotojo sąsajos analizė

Patogi vartotojo sąsaja dažnai laikoma viena iš svarbiausių IS pasisekimo priežasčių. Deja, nėra aiškios formulės parodančios kas yra patogi ir efektyvi vartotojo sąsaja internetiniuose puslapiuose, o sukurti sprendimai greitai pakeičiami naujais. Ne išimtis ir išmaniųjų telefonų vartotojo sąsajos turinčios savo unikalių privalumų bei trūkumų, lyginant su stacionarių kompiuterių vartotojo sąsajomis.

Dažniausiai išskiriamos šešios[3] pagrindinės problemos, su kuriomis susiduriama projektuojant vartotojo sąsają išmaniesiems įrenginiams:

- **Mažas ekranas**

Pagrindinis išmaniųjų telefonų privalumas ir trūkumas yra jų dydis. Dėl mažo dydžio juos patogu visur nešiotis, tačiau įrenginio ekranas yra gana mažas, ir ergonomikos ekspertai teigia, jog toliau didinant ekrano dydį būtų neigiamai atsiliepiama dėl įrenginio patogumo. Žmonės pripratę naudoti telefonus viena ranka einant gatve, vairuojant automobilį ar stovint troleibuse. Didinant ekrano dydį didėja aktyvus ekrano plotas, kurį reikia valdyti. Pasiekus keturių colių įstrižainę darosi sudėtinga pasiekti aktyvias, priešingoje pusėje esančias ekrano vietas, tai pavaizduota 3 paveikslėlyje. Taip pat, didėjant ekranui neišvengiamai didėja ir pats telefono dydis, darosi nepatogu jį nešiotis kišenėje. Pasiekus tam tikrą dydį planšetiniai kompiuteriai pradėtų konkuruoti su jais dėl patogumo ir efektyvumo. Išimčių visada bus, tačiau dėl šių priežaščių galima manyti, jog išmaniųjų telefonų įstrižainė smarkiai neviršys 5 colių.Šis apribojimas įtakoja informacijos kiekį, kurį galima pateikti išmaniojo telefono ekrane,į tai reikia atsižvelgti projektuojant išmaniojo įrenginio IS vartotojo sąsają.

3pav. Viena ranka pasiekiamas ekrano plotas

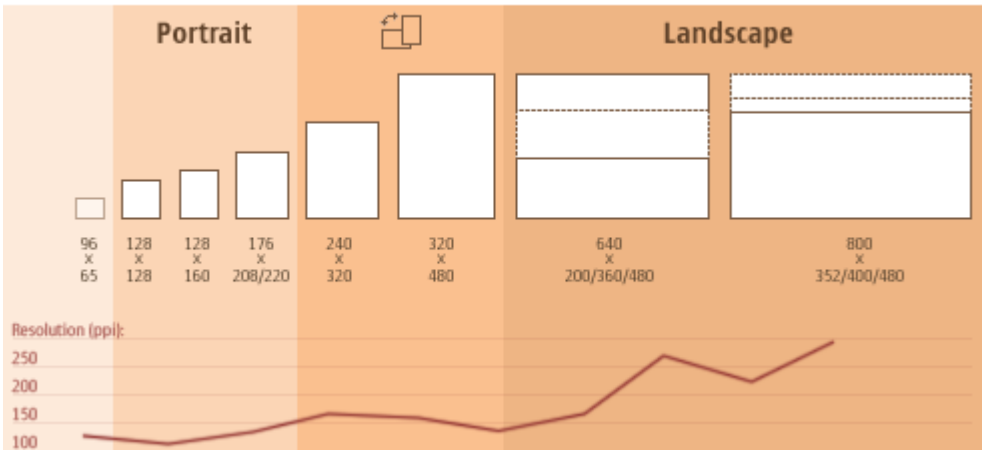
Dėl ekrano dydžio apribojimų, pritaikant informacinę sistemą,ne visas turinys tilps viename išmaniojo įrenginio ekrane. Vartotojai yra labiau pratę slankioti puslapių turinį vertikaliai, o ne horizontaliai. Todėl siūloma naudoti *„Vertical stack“* modelį, kurio pagrindasyra viso turinio išdėstymas vertikaliai.

- **Įvairus ekrano plotis**

Naudojant *„Vertical stack“* modelį nesusidaro problemų dėl ekrano aukščio – netelpančią ekrane informaciją vartotojas gali pasiekti slankiodamas turinį vertikaliai, tačiau dėl įvairaus ekrano pločio kyla problemų dėl panaudojamumo. Jeigu vartotojo sąsaja ir dizainas pritaikytas didelio pločio ekranui, tai peržiūrint jį su mažesnį ekraną turinčiu įrenginiu gali kilti problemų– sumažinti sąsajos elementai yra per maži ar pilnai netelpa ekrane. Iš kitos pusės, mažesniam ekranui pritaikyta sąsaja bus „ištempta“ naudojantis didesnį ekraną turinčiu įrenginiu, gali atsirasti dideli ir nereikalingi tarpai tarp vartotojo sąsajos elementų – nepilnai išnaudojamas ekrano plotas, vartotojai gauna per mažai informacijos viename ekrane.

Vienas iš galimų sprendimų yra kurti skirtingas vartotojo sąsajas įrenginiams, turintiems skirtingus ekranų išmatavimus, tačiau tai nėra efektyvus ir realistiškas sprendimas. 2008 metais atliktas tyrimas[4] parodė (4 paveikslėlis) telefonų ekranų įvairovę.Per pastaruosius metus buvo išleista nemažai naujų įrenginių tad nepanašu, jog bus sustota ties vienu (ar keliais) ekrano dydžiu.

Kita problema, jog vartotojas gali pakeisti ekrano poziciją savo išmaniajame telefone, ir vaizdas jam bus pateikiamas ne portreto (*„Portrait“*), o vietovaizdžio(*„Landscape“*) formatu, tai vėl pakeis elementų išsidėstymą.



4pav. Išmaniųjų įrenginių ekranų tipai 2008 metais [4]

Dėl šios priežasties, siūloma kurti reaguojančias vartotojo sąsajas[5] ir dizainus (*„Responsivedesign“*). Pasinaudojant šiuo metodu galimasukurti vieną

vartotoją sąsają vienam ekrano dydžiui, o *„CSS“*pagalba nurodyti kaip elementai turėtų elgtis(būti praplečiami/sutraukiami ar „apkarpomi“), priklausomai nuo kintančio ekrano dydžio. Šio metodo pavyzdys pateikiamas 5 paveikslėlis.

Raudonai pažymėtame bloke matome puslapio logotipą ir kaip jis pasikeičia peržiūrint tą patį puslapį mažesniu ekranu. Paprastai tokiu atveju visas logotipas būtų sumažinamas proporcingai, to pasekoje,jo pavadinimas taptų sunkiai įskaitomas. Šiuo atveju paveiksliuko šonai yra nukerpami, o pavadinimas išlieka tokio pačio dydžio ir yra aiškiai įskaitomas.

5pav. Reaguojančio dizaino pavyzdys [5]

Tą patį galime matyti ir su pateikiamu tekstu bei autorių fotografijomis. Teksto dydis nesumažinamas, tik pasikeičia jo išdėstymas. Autorių nuotraukos sumažinamos, tačiau vardai išlieka tokio pačio dydžio.

Šitokio tipo *„CSS“* taisyklės padeda sukurti modernią vartotojo sąsają, kuri tinka įvairių dydžių išmaniųjų įrenginių ekranams.

- **Liečiami ekranai ir klaviatūros**

Viena iš populiariausių išmaniųjų telefonų technologijų - liečiamas ekranas.Visgi, kaikurie vartotojai mėgsta valdyti telefoną mygtukų pagalba arba, dėl tam tikrų priežaščių, negali naudotis liečiamu ekranu ir turi pasinaudoti mygtukais. Projektuojant vartotojo sąsają reikia į tai atsižvelgti ir pasistengti, jog sistemos funkcionalumas būtų kuo galima panašesnis tiek naudojantis liečiamu ekranu, tiek mygtukais. Žemiau išvardinti punktai į kuriuos reikėtų atsižvelgti:

- Patvirtinimo mygtukas(*„OK“*) turi veikti įdentiškai su ekrano paspaudimu.
- Visos interaktyvios sąsajos vietos turi būti pasiekiamos naudojant kryptį valdančiais mygtukais.
- Svarbu,jog naviguojama būtų nuosekliai. Jei vartotojas nori pasirinkti sekantį sąsajos mygtuką, jis neturėtų būti numetamas iš IS viršutinės turinio dalies į apatinę, praleidžiant dalį mygtukų.

Siekiant maksimaliai išnaudoti ekrano turinį kontekstinį meniu galima paslėpti ir parodyti jį tik kai vartotojas paspaudžia ekraną atitinkamoje vietoje. Standartinėse vartotojo sąsajose skirtose stacionariems kompiuteriams gan dažnai naudojamas šis metodas kai, pavyzdžiui, tik užvedus pelyte ant nuotraukos pasirodo jos trynimo ir redagavimo mygtukai. Taip sutaupoma vietos ir ne taip apkraunama vartotojo sąsaja nevisada reikalingais elementais. Šis metodas rekomenduojamas dėl ekrano vietos svarbos išmaniuosiuose telefonuose.

- **Teksto rašymas**

Teksto rašymas išmaniuoju įrenginiu užtrunka, sąlyginai, ilgai ir vartotojas gali įvesti klaidingą tekstą dėl kurio kils papildomų nepatogumų. Šią problemą padeda spresti populiarėjantys automatinio užbaigimo(*„Autocomplete“*) laukai, kuriuose pasiūlomas teksto užbaigimas pagal vartotojo įvestą teksto dalį. Išmaniesiems telefonams ši problema yra ypatingai aktuali dėl nepatogaus ir neefektyvus teksto įvedimo, todėl būtina minimizuoti įvedimo laukų skaičių, o kur jų negalima išvengti naudoti automatinį užbaigimą. Taip pat siūlomanaudoti *„x“* mygtuką (ar jo atitikmenį) prie įvedamų laukų kurį paspaudus būtų automatiškai išvalomas įvestas tekstas, šitaip paspartinant vartotojo darbą.

- **Naudojimo aplinka**

Priešingai negu stacionarūs (ar nešiojamieji) kompiuteriai, išmanieji įrenginiai naudojami ne tik darbo aplinkoje, bet ir einant gatve, važiuojant viešuoju transporto ar kitose, aktyviose aplinkose. Dėl šios priežasties vartotojas dažnai gali turėti ribotą apšvietą ar nepatogias naudojimui sąlygas. Į tai atsižvelgiant reikėtų vengti garsinių pranešimų pritaikomoje sistemoje, nes vartotojas, priklausomai nuo esamos aplinkos,gali būti išjungęs garsą. Projektuojant vartotojo sąsają, reikia kurti gan didelius mygtukus ir apie nuorodas,*„CSS“* pagalba, palikti ryškius tarpus, kurie

reaguotų į paspaudimą paliekant didelį „paklaidos“ plotą. Šitaip nereikalaujama iš vartotojo didelio preciziškumo.

- **Socialinė įtaka ir dėmesio stoka**

Išmaniųjų telefonų vartotojai jais paprastai naudojasi kai turi keletą laisvų minučių eidami į susitikimą, bendraudami su kolegomis ar vairuojant mašiną. Dėl šių priežasčių, vartotojai dažniausiai negali skirti pilno savo dėmesio išmaniojo telefono ekrane pateikiamai informacijai. Svarbu jog pagrindinė sistemos informacija būtų pateikiama iškarto ir aiškiai – vartotojui nereikėtų vertikaliai slankioti keletą kartų,orų prognozių sistemos turinio, ieškant šios dienos orų prognozės. Šioje vietoje dažnai daroma klaida: vartotojui nesvarbios informacijos išdėstymas(didelis puslapio logotipas, navigacijos blokai, reklamos ir t.t.) matomiausioje ekrano vietoje (pirmajame užkrautame lange, jo viršuje), ko pasėkoje vartotojas gaišta laiką ir, greitai neradęs informacijos, ateityje gali nebesinaudoti nuvylusia sistema.

Grįžtamasis ryšis kai sistema kraunasi ar atlieka tam tikrus skaičiavimus išmaniesiems įrenginiams yra dar svarbesnis negu stactionariųjų kompiuterių vartotojo sąsajose, nes vartotojas yra mažiau linkęs palaukti ir sužinoti ar sistema „užlūžo“, ar atlieka tam tikrus skaičiavimus.

Išsprendus pagrindines problemas privalu atsižvelgti ir į keletą antraeilių vartotojo sąsajos uždavinių:

- **Specialūs simboliai ir daltonizmu sergantys žmonės**

Dauguma vartotojo sąsajų pateikia įvairią informaciją vaizdiniu pavidalu. Pavyzdžiui raudona spalva dažniausiai reiškia kritinį veiksmą, kurio pasekmės gali būti neatstatomos. Taip pat dažnai naudojami papildomi simboliai, kaip „+“ simbolis mygtkui sufleruojantis, jog paspaudus jį bus atliekamas pridėjimo veiksmas. Šios praktikos yra teisingos ir jų reikia laikytis, tačiau svarbu atsižvelgti į daltonizmu sergančius žmones. Papildomi simboliai ir atsargus spalvų parinkimas šią problemą dalinai išsprendžia, tačiau dauguma išmaniųjų telefonų turi papildomą funkcionalumą skirta šių problemų sprendimui.

Papildomi panaudojamumo įrankiai garsiai perskaito informaciją pateiktą apie tam tikrą vartotojo sąsajos elementą ar išvedamos informacijos paskirti. Tai atliekama naudojantis specialiais atributais, kuriuos naudoja tik šie įrankiai. Eiliniam vartotojui, kuris nėra jų aktyvavęs,jie nemaišo ir nesukelia jokių trugdžių dirbant su sistema.

- **Pritaikomos sistemos nuoseklumas jau sukurtų sistemų atžvilgiu**

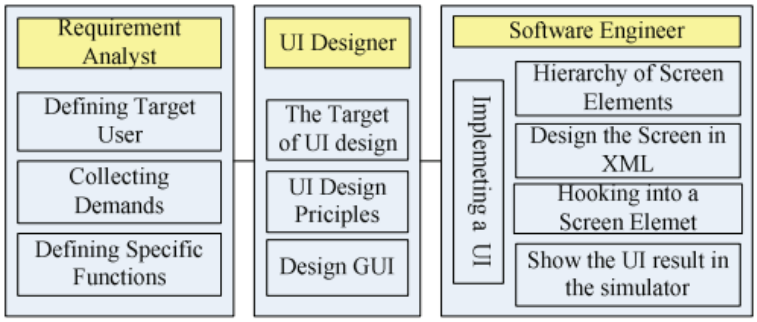
Tarp stacionarių ir nešiojamų kompiuterių savininkų yra plačiai paplitęs darbalaukio ir naršyklių pritaikymas pagal savo poreikius bei norus. Išmanieji įrenginiai šiuo atžvilgiu yra mažiau lankstūs ir daugumos vartotojų įrenginiai yra gan panašus, skiriasi tik įdiegtomis taikomosiomis programėlėmis ir ekrano uždangomis. Dėl šios priežastis svarbu išlaikyti bendrą vartotojo sąsajos stilistiką kurią naudoja išmaniojo telefono operacinė sistema. Jeigu pritaikyta sistema atrodys ir elgsis radikaliai skirtingai negu pats išmanusis telefonas (ar dauguma jo įrankių), pasisekimo tikėtis neverta, nes vartotojai pripratę prie jau nusistovėjusių standartų. Net patogesnė vartotojo sąsaja gali pasirodyti ne tokia patogi kaip standartinė dėl vartotojų familiarumo.

- **Unikalaus funkcionalumo išnaudojimas**

Dažna pritaikytų sistemų problema yra, jog vartotojas jomis naudodamasis jaučiasi, kad naudoja primityvią IS, kurioje veikia visi originaliosios sitemos vartotojo sąsajos sprendimai bei funkcinės galimybės, o naujos platformos teikiami privalumai nepanaudoti. Vartotojas naudodamas sistemą neturi pastebėti, jog tai yra pritaikyta sistema, jam turi atrodyti jog ši sistema ir buvo kurta išmaniajam įrenginiui.

Daugumai sistemų gali praversti giroskopiniai sensoriai, vaizdo kameros išnaudojimais pritaikytoje sistemoje. Pavyzdžiui restoranų sistema gali nustatyti kur šiuo metu yra vartotojas ir pirmajame lange jam siūlyti artimiausią restoraną. Tai smarkiai paspartina vartotojo darbą. Eksperimentinės sistemos atveju galima išnaudoti integruotą kamerą, jog nufotografuoti daiktai iš karto būtų įkeliami į sistemą ir į nurodytą albumą.

Vartotojo sąsajos kūrimo procesui siūloma[6] metodologija matoma 6 paveikslėlyje. Ji yra orientuota ne į egzistuojančios sistemos pritaikymą, o į naujos kūrimą. Ją reikėtų papildyti procesais, reikalingais nustatymui, kuriuos vartotojo sąsajos elementus bei funkcionalumą reikia perkelti iš jau egzistuojančios sistemos.



6pav. Vartotojo sąsajos kūrimo procesas [6]

1.6.2. Operacinė sistema*„Android“*

Operacinė sistema*„Android“* yra greičiausiai besiplečianti išmaniųjų įrenginių operacinė sistema. Jos plėtros statistika didžiosiose Europos šalyse [7] pateikiama 1 lentelėje, o pardavimų kiekis pasauliniu mastu pateikiamas 2 lentelėje.Dėl savo žemesnės kainos, lyginant su konkurentais, labiausiai tikėtina jog ateityje ji įsitvirtins ir Lietuvos rinkoje. Pritaikant IS išmaniesiems telefonams svarbu atsižvelgti į šios OS galimybes bei trūkumus.

1 lentelė. Populiariausios operacinės sistemos didžiosiose Europos šalyse.

Operacinė sistema	**2010m**	**2011m**	**Pokytis**
Symbian	53.9%	37.8%	-16.1
Android	6.0%	22.3%	16.2
IOS	19.0%	20.3%	1.2
BlackBeary	8%	9.4%	1.5
Windows Phone	11.5%	6.7%	-4.8

2 lentelė. Išmaniųjų įrenginių, pagal operacines sistemas, pardavimai, milijonais.

Ketvirtis	**Android**	**IOS**	**Symbian**	**BlackBeary OS**	**Windows Phone**	**Other**
2012 4	159.8	47.8	-	7.4	6	6.8
2012 3	136	26.9	4.1	7.7	3.6	2.8
2012 2	104.8	26	6.8	7.4	5.4	3.6
2012 1	89.9	35.1	10.4	9.7	3.3	3.9
2011 4	83.4	36.3	18.3	12.8	2.4	4.6
2011 3	67.7	16.3	17.3	11.3	1.4	4

Vienas iš svarbiausiu aspektų pritaikant (ar kuriant) IS yra duomenų bazė. Operacinė sistema*„Android“*, kaip numatyta, naudoja reliacinę duomenų bazę*„SQLite“*. Dėl savo mažo dydžio, ši duomenų bazė plačiai naudojama daugomoje išmaniųjų telefonų OS bei interneto naršyklėse. Yra palaikomos ir kitos duomenų bazės, tačiau, šiai dienai, išmaniųjų įrenginių siūloma įvairovė nusileidžia stacionarioms platformoms šiuo aspektu. Pritaikant sistemą gali kilti papildomų

problemų jeigu sistemoje naudojama duomenų bazė nėra palaikoma išmaniojo įrenginio operacinės sistemos.

Informacinių sistemų kūrimui naudojama programavimo kalba*„Java“*. Sistemų kūrimas, kaip numatyta, remiamasi[8]modifikuotu *„MVC“*(*Model-View-Controller*) modeliu.

Programavimo kalba *„Java“*sukurtos sistemos paprastai yra interpretuojamos „Javos virtualiaja mašina”–*„JVM“*. Tačiau operacinė sistema*„Android“* jos nenaudoja, šiam tikslui buvo sukurta virtuali mašina*„Dalvik“*. Ši virtuali mašina atlieka tas pačias funkcijas, tačiau yra integruota į sistemą*„Android“* ir pritaikyta prie išmaniųjų įrenginių galimybių– ribotas baterijos gyvavimo laikas, ribotas atminties kiekis, lyginant su stacionariaisiais įrenginiais, ir t. t.

Vienas iš pagrindinių operacinės sistemos*„Android“* išskirtinumų yra tai, jog komponentai ir įvairios sistemos savybės turi būti aprašytos faile*„androidManifest.xml“*. Tai savotiškas turinys, kuriuo naudojasi operacinė sistema, valdydama sukurtą informacinę sistemą. Šis turinys užtikrina aplinkoje*„Android“* veikiančių sistemų komunikaciją ir suteikia galimybę pasinaudoti kitų platformos *„Android“* sistemų funkcionalumu[8], [9].Šio failo fragmentas pateikiamas7 paveikslėlyje.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest>
    <application>
        <activity>
            <intent-filter>
                <action />
                <category />
                <data />
            </intent-filter>
            <meta-data />
        </activity>
        <activity-alias>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </activity-alias>
        <service>
            <intent-filter> . . . </intent-filter>
            <meta-data/>
        </service>
        <receiver>
            <intent-filter> . . . </intent-filter>
            <meta-data />
        </receiver>
        <provider>
            <grant-uri-permission />
            <meta-data />
        </provider>
        <uses-library />
    </application>
</manifest>
```

7pav. *„androidManifest.xml“* šablonas

Pavyzdys yra neužpildytas ir neatvaizduoja visų galimų parametrų, tai tik jų dalis. Iš viso galimi 26 skirtingi *„xml“* elementai[10] saugantys informaciją apie kuriamą sistemą ir kiekvieną jos komponentą. Šiame turinyje negalima grupuoti

panašių komponentų ir aprašyti juos vienu įrašu– būtina kiekvieną komponentą aprašinėti atskirai. Pavyzdžiui kiekvieno lango kuriame vartotojas gali atlikti veiksmus aprašymui yra naudojamas elementas *„<activity>“*, kuris turi keletą žemesnio lygio elementų aprašančiųjo savybes bei galimas sąveikas su kitomis, įrenginyje esančiomis, sistemomis.

1.6.3. Egzistuojančios ir pritaikytos sistemos sąveikų tipai

Pritaikant sistemą išmaniajam įrenginiui neišvengiamai reikalingos sąveikos su originaliaja sistema, kurias, priklausomai nuo sistemos specifikos, galima išskirti į tris grupes.

- **Duomenys labiau skaitomi, negu rašomi ar atnaujinami**

Naujajai platformai pritaikyta sistema daugiausiai naudojama informacijos skaitymui ir kito originaliosios sistemos turinio peržiūrai. Ši sąveika yra pati paprasčiausia, nes į pritaikytą išmaniojo įrenginio sistemą pakanka perduoti duomenis iš internetinės sistemos, naudojant *„web”* paslaugas ar kitas panašias technologijas.

- **Dažnai naudojami tam tikros srities duomenys**

Priklausomai nuo sistemos paskirties, poreikis naudoti dalį duomenų išmaniojo įrenginio aplinkoje gali būti daug dažnesnis, pavyzdžiui, naudoti tam tikrą vartotojo registracijos informaciją. Siekiant paspartinti sistemų sąveikos našumą gali būti tikslinga, priklausomai nuo sistemos specifikos, dalį tokių dažnai naudojamų duomenų saugoti išmaniajame įrenginyje.

Tokius duomenys galima saugoti arba lokalioje duomenų bazėje, arba laikinojoje atmintyje, priklausomai nuo jų apimties ir panaudojimo dažnumo.

- **Duomenys naudojami asmeniniams tikslams, ir vartotojai yra pagrindiniai sistemos turinio šaltiniai**

Šiuo atveju didžioji dalis informacinių sistemų turinio yra generuojama pačių vartotojų. Vartotojams gali būti aktualu pasiekti savo duomenis, kai išmaniajam įrenginiui nepasiekiamas interneto ryšys arba dėl tam tikrų priežasčių originaliosios sistemos serveriai nėra pasiekiami. Tokiu atveju reikalinga, jog pritaikyta sistema galėtų veikti nepriklausomai nuo originaliosios internetinės informacinės sistemos. Veikimui užtikrinti būtina, jog lokali sistema turėtų savo duomenų bazę su reikiamų duomenų kopijomis. Siekiant išvengti duomenų neatitikimų tarp internetinės ir išmaniojo įrenginio sistemos, reikalinga atlikti duomenų sinchronizaciją, kai tik internetinės sistemos duomenų bazių valdymo sistemos (DBVS) serveriai tampa pasiekiami.

1.6.4. Duomenų sinchronizacija

Duomenims sinchronizuoti yra siūloma įvairių sprendimų [11], [12], [13],tačiau dauguma jų veikia korektiškai arba tik su identiškomis, arba to paties tipo DBVS (pavyzdžiui, sunku sinchronizuoti *„SQL“* ir *„NoSQL“* duomenų bazes). Reikia įvertinti, kad išmaniųjų įrenginių platformos palaiko ne visas duomenų bazes, todėl, priklausomai nuo sistemos specifikos, gali reikėti atlikti skirtingų DBVS sinchronizaciją. Bendro sprendimo tinkančio visiems atvejams nėra, kadangi priklausomai nuo kiekvienos sistemos specifikos bei sąveikos tipo, tarp originaliosios ir pritaikytos sitemos, gali reikėti skirtingų sprendimų. Dėl šių priežaščių, šiame darbe duomenų sinchronizavimo klausimas plačiau nėra aptariamas, o sprendimas priimtas eksperimentinės sistemos atžvilgiu pateikiamas pirmame priede.

1.6.5. Karkasų *„Zend Framework“* ir *„Android“*analizė

Kiekvienas programinis karkasas turi gan aiškiai apibrėžtą sistemos struktūrą – kur kokio tipo komponentai saugomi ir už kokius sistemos veiksmus jie atsakingi. Šiuo metu smarkiai populiarėja ir dominuoja *„MVC“* (*Model-View-Controller*)programinio kodo architektūros modelis. Dėl šių dviejų priežaščių galima pastebėti panašumų tarp įvairių programinių karkasų, kurie skirti skirtingoms programavimo kalboms, ar net skirtingomis platformomis.

Pasirinkta operacinė sistema naudoja savo specifinį programinį karkasą *„Android“*, kuris pilnai neatitinka *„MVC“* principų. Visų pirma –jis neturi *„model“* tipo klasių ir yra ginčytina ar egzistuojantis logikos (*„controller“*) bei vaizdavimo (*„view“*) atskyrimas yra teisingas. Nepaisant to, galima pastebėti nemažai panašumų su kitais *„MVC“* karkasais. Trūkstamas logikos vietas galima papildyti, taip panaikinant kai kuriuos struktūrinius skirtumus, lyginant su labiau standartinės struktūros programiniais karkasais.

Eksperimentinė sistema sukurta naudojant programavimo kalbos *„PHP“* karkaso*„Zend Framework“* pirmąją versiją. Šis karkasas labiau atitinka standartinį*„MVC“* modelį. Analizuojant abu karkasus pastebėti komponentai turintys atitikmenis abejuose karkasuose.

- *„MVC“* struktūra naudojama *„Zend Framework“* karkaso leidžia susieti kiekviena veiksmą*„controller“* (puslapis ir jo logika) su jam priklausančia *„view“* klase (vaizdavimas). Analogiškai karkase*„Android“*galima išskirti*„activity“*tipo komponentus ir jiems priklausančius *„xml“* failus, kurie turi tą pačią paskirtį – puslapis/logika ir jo vaizdavimas.

- Programinis karkasas *„Android“*, kaip jau minėta, neturi „model“ klasių atitikmens. Šiame programiniame karkase nedraudžiama sukurti patiems*„model“* tipo klasių ir jas naudoti identiškai, kaip ir tokio tipo komponentus karkase*„Zend Framework“*.
- Duomenų įvedimo formų laukai, karkase *„Zend Framework“*, paprastai aprašomi strukturizuotai naudojant*„Zend_Form“* klasę.Karkase *„Android“* tokio tipo duomenys strukturizuotai aprašomi*„xml“* failuose.
- Analizuojant *„Zend Framework“* pagrindu sukurtas sistemas, galima identifikuoti dažnai naudojamus komponentus (pavyzdžiui, šoninį meniu, kuris pateikiamas visuose puslapiuose). Atpažinimo požymis gali būti, jog šios klasės yra įterpiamos (*„include“*) kitose klasėse arba naudojamas vienas iš siūlomų metodų šitokių klasių panaudojimui. Tokio tipo komponentai, karkase *„Android“*, realizuojami klasėmis*„fragment“*.

Lyginant kitus programinius karkasus taip pat galima pastebėti panašumų ir atitkmenų, tačiau, šiame darbe jie nėra plačiau aptarinėjami siekiant apriboti darbo apimtį.

1.7. Vartotojų analizė

Galimų, pritaikytos sistemos vartotojų aibę sudaro dviejų tipų vartotojai, kurių savybės pateikiamos 3 lentelėje.

3lentelė. Vartotojų savybių aprašymas

Vartotojo tipas	Savybės
Originaliosios sistemos lankytojai	Tai įvairaus amžiaus bei techninių gebėjimų žmonės, kuriuos vienija tik tai, jog jie naudojosi originalia sistema ir nusprendžia išbandyti naują, išmaniesiems įrenginiams pritaikytą sistemos versiją. Šie vartotojai turi išankstinius lūkeščius ir tikisi jog pritaikyta sistema veiks panašiai kaip ir originalioji sistema
Atsitiktiniai, nauji lankytojai	Tai nauji vartotojai, prieš tai nesusidurę su originalia sistema. Priešingai negu originaliosios sistemos vartotojai jie neturi išankstiniu lūkeščių.

Keletas galimų vartotojų problemų, kurios būtų išsprendžiamos pritaikant sistemą išmaniesiems įrenginiams,pateikiamos 4 lentelėje.

4lentelė. Vartotojų problemų aprašymas

Problema	Kaip viskas vyksta dabar	Sprendimas
Sistemos teikiamos paslaugos naudingos kai stacionarus (ar nešiojamas)	Vartotojas privalo pasinaudoti stacionariu kompiuteriu norėdamas gauti informaciją, kuri naudinga esant išvykus.	Išmanųjį įrenginį patogiau nešiotis su savimi ir gauti reikiamą informaciją greitai bei patogiai.

kompiuteris nepasiekiamas.	Pavyzdžiui, autobusų maršruto grafikas.	
Adreso/ gyvenamosios vietos svarbumas dirbant su sistema.	Ieškant tam tikros informacijos (pavyzdžiui, parduotuvės filialų adresų) reikia pačiam vartotojui nurodyti kuriame mieste/ rajone esantys objektai jį domina.	Išmaniajame įrenginyje esanti sistemos versija gali pasinaudoti giroskopiniais sensoriais ir automatiškai filtruotireikiamą informaciją pagal buvimo vietą.
Kitos problemos.	Yra aibė galimų problemų, kuriospriklauso nuo sistemos specifikos. Jas dalinai galima spręsti pasinaudojant unikaliomis išmaniųjų įrenginių savybėmis.	

1.8. Esamų sprendimų analizė ir siekiamas sprendimas

Nors išmanieji įrenginiai šiuo metu ir yra labai populiarūs, o sistemų, skirtų jiems, skaičius auga kiekvieną dieną, tačiau nėra aiškių sprendimų kaip sėkmingai pritaikyti egzistuojančias sistemas naujoms platformoms. Šiuo metu galima išskirti tris pritaikymo būdus.

Kogero seniausias (ir paprasčiausias) iš jų, tačiau vis dar labai populiarus, tai sistemos vartotojo sąsajos adaptavimas priklausomai nuo to, kokio dydžio yra įrenginio ekranas [14], [15]. Pagrindinis šio būdo trūkumas yra, tas jog sistema neišnaudoja unikalaus naujos platformos funkcionalumo. Taip pat stacionariesiems kompiuteriams skirta vartotojo sąsaja gali būti nepatogi išmaniąjam įrenginiui, net jeigu ji ir atitinkamai sumažinta. Šis sąsajos pritaikymas pagal ekrano dydį labiau tinka norint išspręsti skirtingų išmaniųjų įrenginių ekranų dydžio problemą, kai jau sistema yra pritaikyta išmaniajam įrenginiui, o ne pačiam sistemos pritaikymui.

Šiuo metu smarkiai populiarėja paslaugos (įvarūs kodo kompiliatoriai/ interpretatoriai ir *„HTML5“* technologija) leidžiančios parašyti vieną sistemos versiją, kuri veiktų visiems išmaniųjų įrenginių tipams. Tai efektyvus būdas laiko prasme, kadangi nereikia kurti atskirų sistemų visoms operacinėms sistemoms,kurias norime pasiekti. Deja, neišvengiamai kyla problema, jog skirtingos operacinės sistemos veikia skirtingai, ir kodas, parašytas bendrai visoms joms, neišnaudos jų optimaliai. Priklausomai nuo sistemos specifikos, gali kilti atvejų jog reikėsskirtingo, tam tikrų funkcijų, veikimo skirtingose operacinėse sistemose (kai kurios funkcijos gali būti nepalaikomos tam tikroje operacinėje sistemoje), programinio kodo bazė neišvengiamai taps sudėtinga ir paini. Taipogi šios paslaugos tampa papildomu

technologijos sluoksniu, kuris gali kelti problemų palaikant sukurtas sistemas. Pavyzdžiui, dėl tam tikrų pakeitimų iš operacinės sistemos kūrėjų reikia laukti naujos kompiliatoriaus versijos, o tuo metu sukurta sistema veikia nekorektiškai. Vienas iš galimų įrankiu: <http://www.coronalabs.com/products/corona-sdk/> leidžiantis parašyti kodą vieną kartą ir kompiliuoti jį skirtingoms išmaniųjų įrenginių platformoms.

Trečiasis būdas – kurti atskiras sistemas kiekvienai platformai, kurios pritaikytų egzsituojančios sistemos funkcines galimybes bei papildytu jas unikaliomis, platformai būdingomis funkcijomis. Šiuo būdu kokybiškiausiai pritakomos sistemos, tačiau trūksta metodologijos kaip sėkmingai, tai atlikti. Pagrindinis šio darbo tikslas - sukurti koncepciją kuri palengvintų ši procesą.

1.9. Analizės išvados

Analizės metu apibrėžti tyrimo tikslai, ribos. Galimos problemos bei apibrėžta eksperimentui naudojama internetinė informacinė sistema.

Analizuojant išmaniųjų įrenginių vartotojo sąsajos savitumus pastebėta, jog tiesiogiai pritaikyti vartotojo sąsają iš stacionarioms sistemos skirto sprendimo - nėra teisinga, kadangi nebus išnaudojamas unikalios šios platformos savybės. Taip pat nebus atsižvelgiama į išmaniųjų įrenginių apribojimus/ trūkumus ir dėlto gali kilti naujų problemų su vartotojo sąsaja.

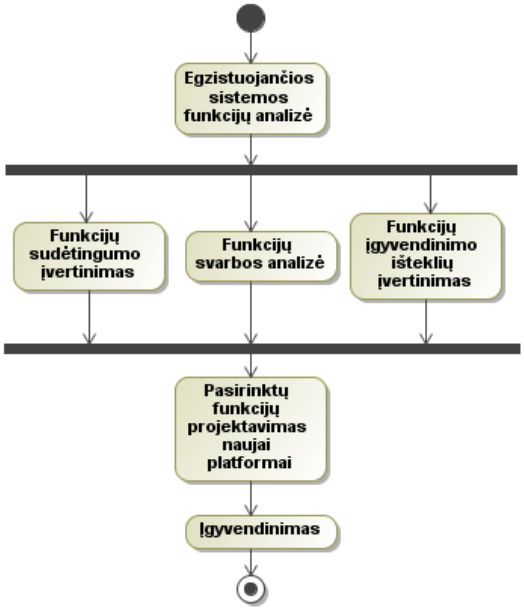
Techninės analizės metu nustatytos naujos sistemos galimybės. Analizuojant eksperimentinės sistemos ir naujos platformos programinius karkasus pastebėta panašumų, kuriais remiantis bus kuriama šio darbo siūlomo sprendimo koncepcija. Sprendimo tikslas palengvinti ir paspartinti internetinių sistemų pritaikymą išmaniesiems įrenginiams remiantis egzsituojančios sistemos kodo baze.

2. Internetinių sistemų reinžinerijos išmaniesiems įrenginiams koncepcija

Kuriant koncepciją buvo įvertinta ir jau esamų sprendimų patirtis bei logika [14], kurie buvo taikomi ir senesnės kartos mobiliesiems sprendimams [15].Visų pirma, prieš perkeliant egzistuojančią sistemą į naują platformą būtina įvertinti, kurias funkcijas tikslinga pritaikyti naujajai sistemai. Taip pat atsižvelgti į naujos programų kūrimo aplinkos programinio projekto struktūrą.

- **Reikiamo funkcionalumo atrinkimas**

Pritaikant esamą sistemą išmaniesiems įrenginiams, pirmiausia reikėtų pasinaudoti sistemos turimomis reikalavimų ir projektinėmis specifikacijomis. Pagal šias specifikacijas paprasta nustatyti, kuris funkcionalumas turi būti perkeliamas į išmaniajam įrenginiui kuriamą taikomąją programą. 8paveikslėlyje pateikiamas koncepcinis sistemos reinžinerijos procesas, kuriame detalizuojamas perkeliamų į naują platformą funkcijų parinkimas.



8pav. Perkeliamo funkcionalumo atrinkimas

Pirmiausia analizuojamas egzistuojančios sistemos funkcionalumas. Atsižvelgiama į tris pagrindinius aspektus:

- Reikia įvertinti kiekvieną sistemos funkcinį panaudojimo atvejį, ar šis funkcionalumas reikalingas išmaniajame įrenginyje. Nebūtino funkcionalumo pavyzdys gali būti sistemos administratoriaus funkcijos.
- Svarbu atsižvelgti į turimus, sistemai kurti skirtus išteklius. Reikia atskirti funkcionalumą, kuris nėra būtinas naujajai sistemai, kuriam įgyvendinti, gali būti panaudota daug sistemai kurti skirtų išteklių.
- Svarbu nustatyti, ar pritaikomas funkcionalumas labai sudėtingas. Šis klausimas vertinamas tiek techniškai, tiek iš vartotojo pozicijos. Realizuoti techniškai sudėtingą funkcionalumą gali užtrukti ilgiau, negu planuojama, o

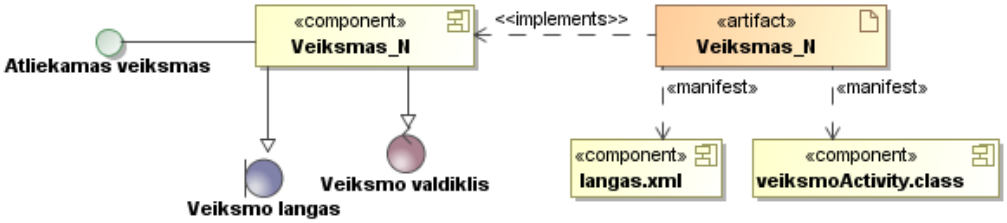
įgyvendintas sudėtingas funkcionalumas vartotojui gali pasirodyti per daug sudėtingas ir nepatogus, todėl vartotojas jo nenaudos.

Remiantis šiais vertinimais sudaromas funkcijų prioritetinis sąrašas. Iš sąrašo empiriškai pašalinamos jo gale esančios funkcijos.

Po šios reikalavimų analizės turime pradinį sąrašą funkcijų,naujajai sistemai kurti.

- **Programinio kodo struktūros perkėlimas**

Žinant perkeliamų funkcijų aibę, galima atlikti programinio kodo analizę. Kaip jau minėta analizės dalyje, kai kurie programinio karkaso *„Android“* komponentai turi atitikmenis kituose *„MVC“*tipo karkasuose. Komponentai *„activity“* veikia panašiai kaip valdikliai, pakartotinio panaudojimo komponentai veikia kaip *„Android“* komponentai *„fragments“*, o vaizdai aprašomi *„xml“* failuose. Šių komponentų įdiegimo klasių modelis matomas9 paveikslėlyje.

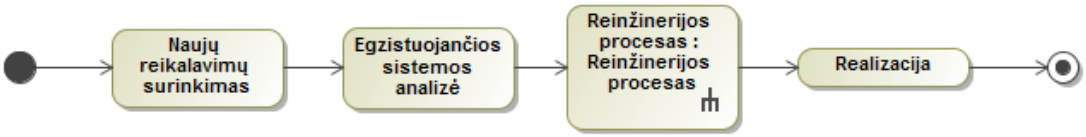


9pav. Komponentų įdiegimo klasių modelis

Atsižvelgiant į šiuos panašumus galima automatizuotai sukurti išmaniojo įrenginio taikomosios programos išeities programinio kodo projektą, kurį sudarytų dalis reikalingų failų, kuriuose saugomos programinės klasės ir kiti programinio kodo epizodai. Turint jau dalinę programinio projekto struktūrą lengviau testi IS kurimo darbus.

Išlaikant panašią programinę struktūrą tarp skirtingų sistemos versijų lengviau jas palaikyti.

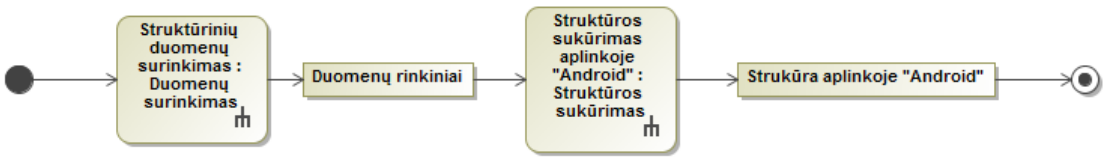
Šio darbo siūlomas sprendimas yra dalinai automatizuoti reinžinerijos procesą. Jis matomas apibendrintame modelyje, atvaizduojančiame egzistuojančios sistemos pritaikymą išmaniesiems įrenginiams, 10 paveikslėlyje.



10pav. Sistemos pritaykmo aukščiausio lygio modelis

Detalesnis reinžinerijos proceso modelis pateikiamas 11 paveikslėlyje. Modelyje procesas padalijamas į dvi pagrindines dalis: struktūrinių duomenų

surinkimą iš egzistuojančios sistemos ir naujos struktūros sukūrimąaplinkoje*„Android“* (remiantis surinktais duomenimis).



11pav. Reinžinerijos proceso abstraktus modelis

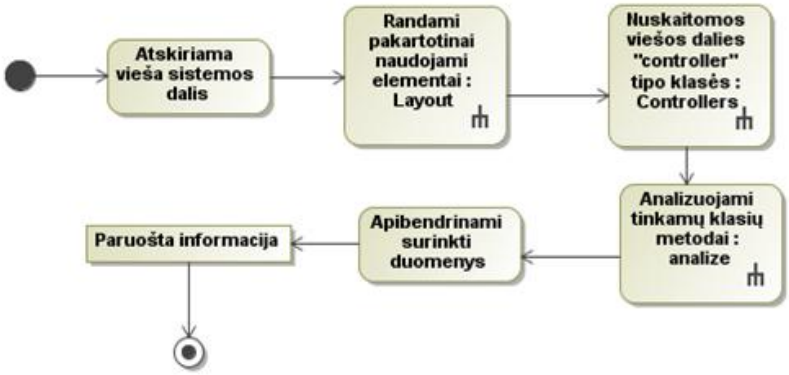
Kadangi kiekvienas programavimo karkasas turi gana aiškiai apibrėžtą sistemos struktūrą, kuri parodo, kur ir kokio tipo komponentai saugomi, šį procesą galima iš dalies automatizuoti ir automatiškai sukurti reikiamus komponentus, o jų informacijąsurašyti į failą *„androidManifest.xml“*.

Detaliam koncepcijos įgyvendinimui buvo pasirinktas programinio karkaso *„Zend Framework“* pagrindu sukurtų internetinių informacinių sistemų pritaikymo platformai *„Android“* uždavinys. Remiantis karkasų analizės rezultatais buvo išskirti šie struktūrų sąsajų principai:

- *„MVC“* struktūra leidžia susieti veiksmus su jiems priklausančiomis*„view“* klasėmis. Tiesiogiai perkeliant šį ryšį į programinį karkasą *„Android“* galima iš karto sugeneruoti *„activity“* klases ir jų *„xml“* failus, kurie turi tą pačią paskirtį.
- Formoms kurti paprastai naudojama *„Zend_Form“* klasė, struktūrizuotai aprašanti formos elementus (įvedimo laukai, jų pavadinimai, mygtukai). Nustačius, jog naudojama *„Zend_Form“* klasė,galima surinkti informaciją apie formos elementus ir sukurti atitinkamus elementus aplinkoje *„Android“* užpildant atitinkamą *„xml“* failą.
- Rastiems *„model“* tipo komponentams galima sukurti atitikmenis aplinkoje*„Android“*nepaisant to, jog, kaip numatyta, karkasas *„Android“* neturi *„MVC“* architektūros*„model“* klasių atitikmens.
- Internetinėse informacinėse sistemose logika dažnai atskiriama į viešą bei administracinę dalis). Pritaikant sistemą išmaniesiems įrenginiams netikslinga perkelti administracinę dalį, todėl ši sistemos dalis dažniausiai turėtų būti neliečiama.
- Analizuojant *„Zend Framework“* pagrindu sukurtas sistemas, galima identifikuoti dažnai naudojamas klases,analizuojant *„layout“* tipo komponentus. Radus pakartotinio panaudojimo komponentus galima sukurti *„fragment“* tipo klases aplinkoje *„Android“*.

- Sugeneruotų klasių informaciją galima automatiškai surašyti į failą *„androidManifest.xml“*,

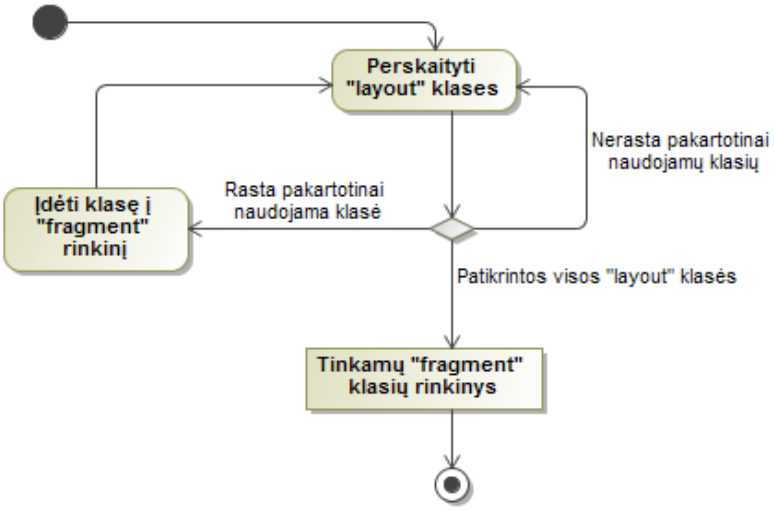
Esamos internetinės informacinės sistemos duomenų surinkimo ir perkėlimo veiklos modelis, kuris grindžiamas anksčiau aprašytais principais, pateikiamas 12 paveikslėlyje.



12pav. Esamos internetinės IS duomenų surinkimo ir perkėlimo veiklos modelis

Pirmiausia atskiriama vieša sistemos dalis nuo administracinės ir kitų sistemos dalių. Kadangi karkasas *„Zend Framework“*, kaip numatyta, neatskiria šių sistemos dalių, reikia papildomos informacijos norint nustatyti, kurias klases reikia perkelti, o kurių – ne. Šiuo atveju, galima pasinaudoti egzistuojančios sistemos projektine specifikacija.

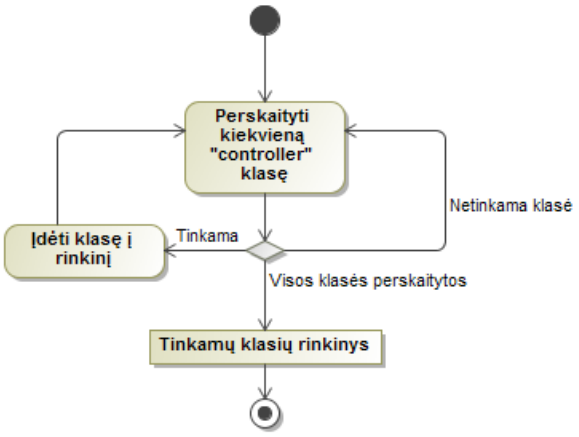
Pakartotinai naudojamus elementus galima rasti *„layout“* klasėse, ieškant komponentų, kurie yra į juos įtraukiami. Šių klasių analizės principas pateikiamas 13 paveikslėlyje.



13pav. „layout“ klasių analizė

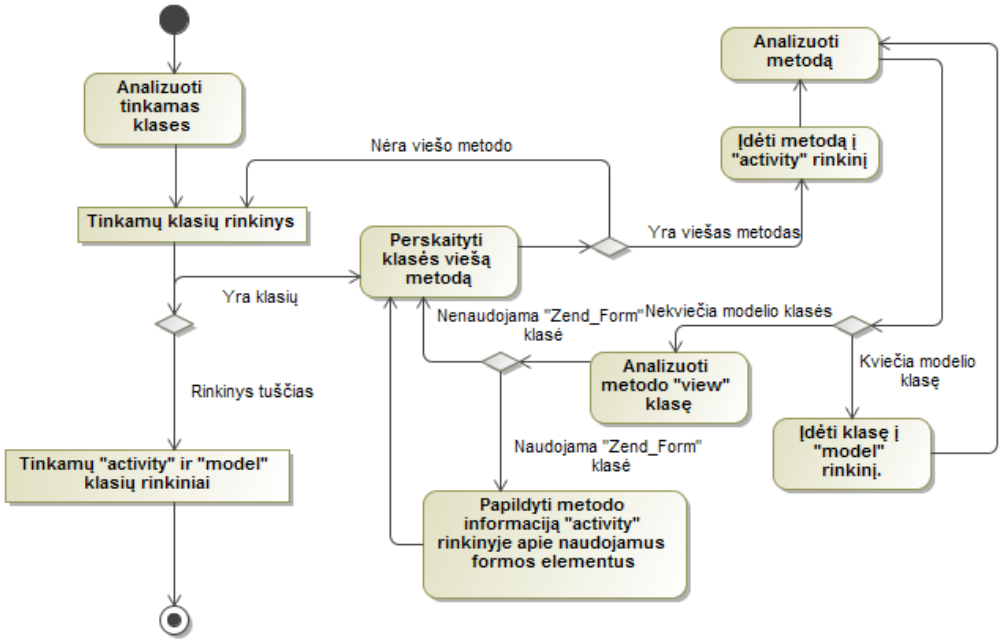
Sistemos veiklos logiką aprašo *„controller“* klasės, jų atrinkimo modelis matomas14 paveikslėlyje. Čia vartojama sąvoka „tinkama klasė“ – tai tokia klasė,

kuri yra skirta sistemos viešajai daliai ir turi bent vieną viešą (*„public“*) metodą (*„controller“* klasė, neturinti nė vieno viešo metodo, neturi prasmės, bandant į ją kreiptis bus grąžinama klaida). Perkelti tokias klases reinžinerijos metu nėra prasmės.



14pav. „Controller“ klasių surinkimas

Sekančiame žingsnyje analizuojamos surinktos *„controller“* klasės, analizės modelis matomas 15 paveikslėlyje. Pirmiausia, cikliniu būdu analizuojamas surinktų, perkelti tinkamų klasių rinkinys, kol pasiekiamas sąrašo galas.



15pav. Metodų analizės modelis

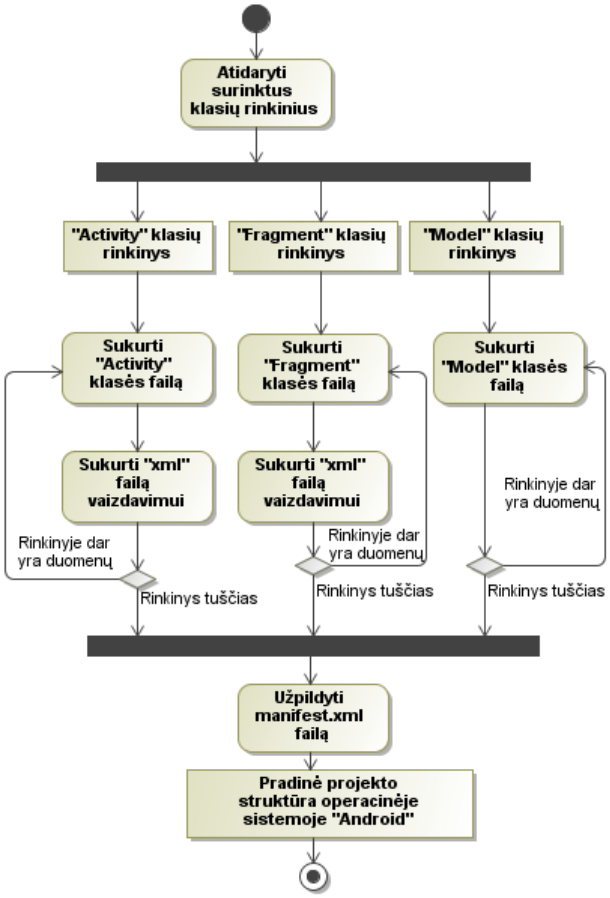
Ciklo metu yra analizuojami visi vieši *„controller“* klasės metodai – kiekvienas iš jų atitinka tam tikrą sistemos puslapį, kuris turi vaizdavimo būdą ir savo veiklos logiką.

Kiekvienas iš šių metodų gali kreiptis į modelio klasę (pavyzdžiui, reikia gauti duomenų iš duomenų klasės). Radus kreipinį, modelio klasės informacija yra įrašoma į *„models“* klasių rinkinį.

Kiekvienam *„controller“* klasės viešam metodui būtina turėti *„view“* klasę (kitokiu atveju bandant kreiptis į metodą bus grąžinama klaida), jeigu jame panaudota *„Zend_Form“* klasė (pavyzdžiui, kontaktų forma), tada galima nuskaityti elementus, esančius formoje, ir juos sukurti aplinkoje *„Android“*.

Atlikus šiuos veiksmus gaunamas duomenų rinkinys, kuris naudojamas paskutiniame šio reinžinerijos proceso etape.

Surinkus visą reikalingą informaciją iš egzistuojančios sistemos struktūros yra sukuriama nauja struktūra aplinkoje *„Android“*. Struktūros kūrimo modelis pateikiamas 16 paveikslėlyje. Lygiagrečiai kiekvienam duomenų rinkiniui yra kuriamos atitinkamos klasės ir *„xml“* failai aplinkoje *„Android“*.



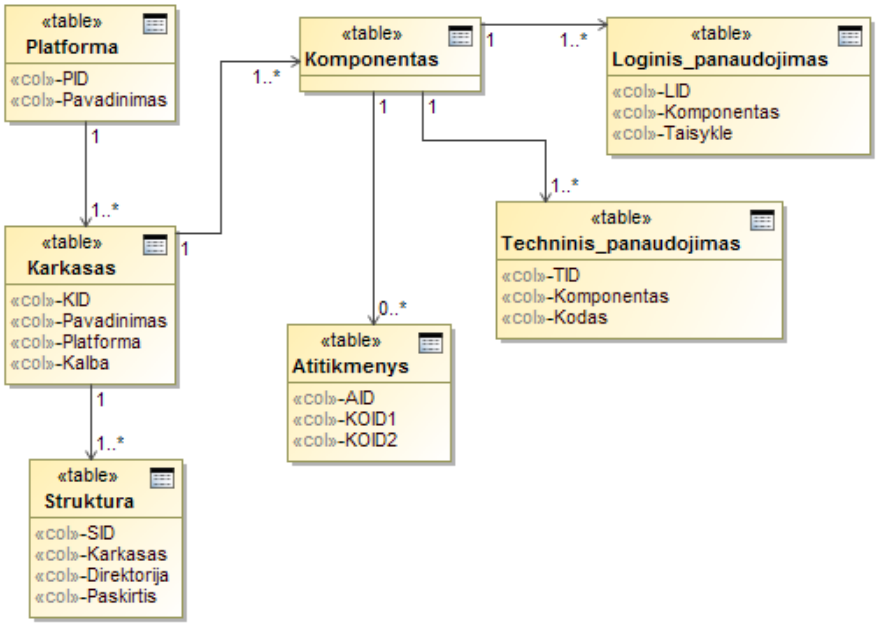
16pav. Naujos struktūros kūrimo veiklos modelis

Sukūrus visus failus, jų informacija surašoma į failą *„androidManifest.xml“*, kuriame saugoma pagrindinė informacija apie sistemos struktūrą.

Atlikus visus šiuos veiksmus gaunamas pradinis sistemos programinio kodo projektas aplinkoje *„Android“*, kurį toliau programuotojas papildo specifine veiklos logika, ir suteikia išmaniesiems įrenginiams būdingų funkcinių elementų bei sprendimų.

- **Taisyklių duomenų saugykla**

Ateityje analizuojant kitus programinius karkasus ir praplečiant sprendimą neišvengiamai kils poreikis saugoti taisykles duomenų bazėje. Suprojektuota duomenų bazė, kurioje būtų galima saugoti taisykles, pagal kurias atliekamas kodo generavimas, matoma17 paveikslėlyje. Lentelių specifikacija pateikiama 5 – 11 lentelėse. Svarbu paminėti jog projektuojant duomenų bazę remtasi dabartine patartimi, analizuojant kitus karkasus tikėtina, jog schema būtų praplečiama norint kaupti platesnio tipo taisykles.



17 pav. Duomenų bazės schema

5lentelė. „Platforma“ lentelės specifikacija

Reikšmė	**Paskirtis**
PID	Platformos identifikatorius.
Pavadinimas	Platformos pavadinimas.

6lentelė. „Karkasas“ lentelės specifikacija

Reikšmė	**Paskirtis**
KID	Karkaso identifikatorius.
Pavadinimas	Karkaso pavadinimas.
Platforma	Platformos, kuriai skirtas karkasas, identifikatorius
Kalba	Programavimo kalba naudojama karkase.

7lentelė. „Komponentai“ lentelės specifikacija

Reikšmė	**Paskirtis**
KID	Komponento identifikatorius.

Pavadinimas	Karkaso, kuriam priklauso komponentas, identifikatorius.

8lentelė. „Platforma“ lentelės specifikacija

Reikšmė	**Paskirtis**
AID	Atitinkančių komponentų identifikatorius.
KID1	Poros pirmojo komponento identifikatorius.
KID2	Poros antrojo komponento identifikatorius.

9lentelė. „Loginis_panaudojimas“ lentelės specifikacija

Reikšmė	**Paskirtis**
LID	Loginio panaudojimo taisyklės identifikatorius.
Komponentas	Komponento identifikatorius.
Taisykle	Kokiomis sąlygomis (iš loginės pusės) naudojamas komponentas.

10lentelė. „Techninis_panaudojimas“ lentelės specifikacija

Reikšmė	**Paskirtis**
TID	Techninio panaudojimo taisyklės identifikatorius.
Komponentas	Komponento identifikatorius.
Kodas	Kodo pavyzdys, kaip iš techninės pusės panaudojamas komponentas.

11 lentelė. „Struktura“ lentelės specifikacija

Reikšmė	**Paskirtis**
SID	Struktūros taisyklės identifikatorius.
Karkasas	Karkaso identifikatorius.
Direktorija	Direktorijos kelias.
Paskirtis	Direktorijos paskirtis nurodanti, kokio tipo komponentai ten saugomi.

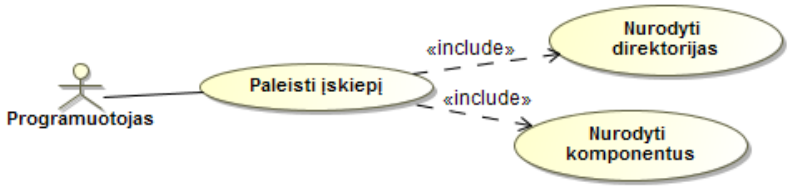
3. Koncepcijos prototipo įskiepio projektas

Anksčiau aprašytai koncepcijai įgyvendinti sukurtas programos prototipas, kuris veikia kartu su programavimo aplinkos *„Eclipse“* įskiepiu, skirtu kurti aplinkos *„Android“* taikomosioms programoms. Įskiepis kuriamas naudojant programavimo kalbą*„Java“*. Šiuo metu veiksmai atliekami tik su karkasu*„Zend_Framework“*, tačiau ateityje galima apimti ir kitus programinius karkasus, praplečiant taisyklių rinkinį.

Pirmiausia naudojant įrankį *„Eclipse“* yra sukuriamas naujas *„Android“* projektas su numatyta failų struktūra ir konfigūracija.O antrojo žingsnio metu paleidžiamas sukurtas įskiepis, nurodant pritaikomos internetinės sistemos programinio kodo adresą ir tikslinį *„Android“* projektą, kuris užpildomas perkeliama informacija iš pasirinktos sistemos. Naujajam projektui sugeneruojamos pagrindinės programinės klasės, dalis vartotojo sąsajos elementų ir užpildomas failas *„androidManifest.xml“*.

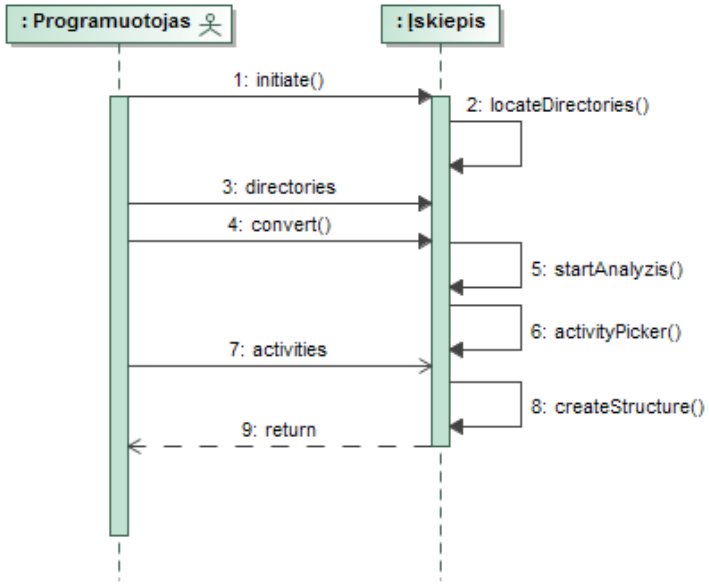
3.1. Panaudojimo atvejai ir vartotojo sąsaja

Įskiepio naudotojas, šiuo atveju programuotojas, minimaliai bendrauja su vykstančiais procesais. Jo panaudojimo atvejai apsiriboja įskiepio paleidimu, direktorijų, kuriuose yra originaliosios sistemos kodo bazė ir naujasis *„Android“* projektas, ir perkeliamų komponentų nurodymu. Šio panaudojimo atvejo modelis matomas 18 paveikslėlyje.



18pav. Įskiepio panaudojimo atvejai

Panaudojimo atvejų specifikacijos pateikiamos 12 - 14 lentelėse, o bendra sekų diagrama 19 paveikslėlyje.



19pav. Panaudojimo atvejų „Paleisti įskiepi“ ir „Nurodyti direktorijas“ sekų diagrama

12lentelė. „Paleisti įskiepi“ PA specifikacija

PA „Paleisti įskiepi“		
Tikslas. Paleisti *„Eclipse“* aplinkos įskiepi		
Aprašymas		
Prieš sąlyga		Programuotojas yra įsijungęs programavimo aplinką *„Eclipse“* ir yra sukūręs naują *„Android“* projektą.
Aktorius		Programuotojas.
Sužadinimo sąlyga		Programuotojas nori konvertuoti programos kodą.
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	„Nurodyti direktorijas“, „Nurodyti komponentus“
	Specializuoja PA	
Pagrindinis įvykiu srautas		Sistemos reakcija į sprendimus

1. Vartotojas pasirenka rėžimo perjungimą.	Įskiepisaktyvuoja pasirinkimo langus ir pereina į antrą žingsnį.
2. Sistema baigia PA	

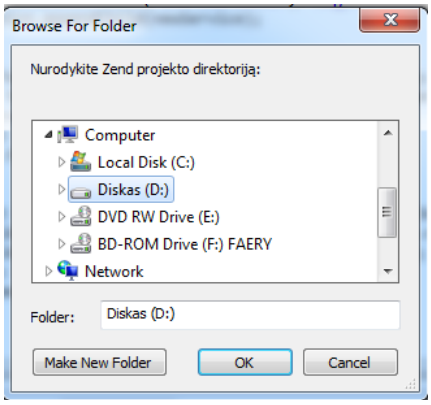
13lentelė. „Nurodyti direktorijas“ PA specifikacija

PA „Nurodyti direktorijas“		
Tikslas. Nurodyti originalios sistemos direktoriją ir parinkti *„Android“* projektą.		
Aprašymas		
Prieš sąlyga		Įskiepis paleistas.
Aktorius		Programuotojas.
Sužadinimo sąlyga		Programuotojas paleidžia *„Eclipse“* įskiepį.
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykiu srautas		Sistemos reakcija į sprendimus
1. Vartotojas nurodo originalios sistemos direktoriją		Įskiepis išsaugo informaciją ir atidaro *„Android“* projekto pasirinkimo langą.
2. Vartotojas nurodo *„Android“* projektą.		Įskiepis išsaugo informaciją ir pereina į trečią žingsnį.
3. Sistema baigia PA		
Alternatyvūs scenarijai		
1b. Vartotojas nurodo ne programinio kodo direktoriją.		Įskiepis gražina klaidą ir prašo nurodyti direktoriją dar kartą.
2b. Vartotojas nurodo ne *„Android“* projektą.		Įskiepis gražina klaidą ir prašo nurodyti projektą dar kartą.

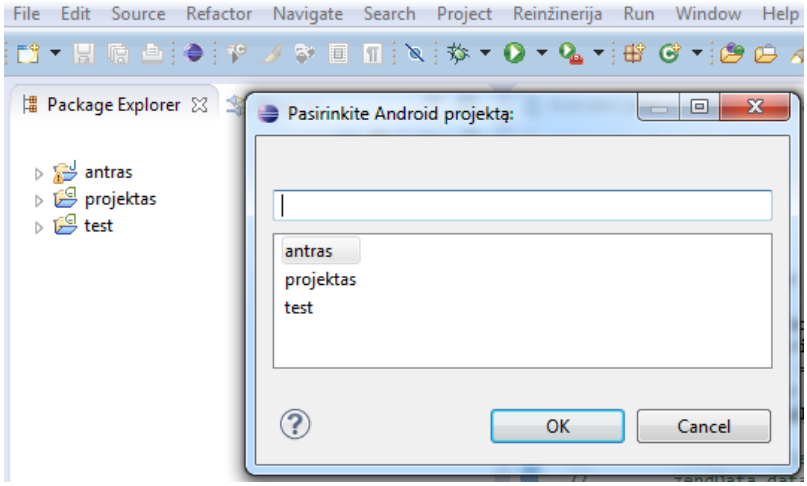
14lentelė. „Nurodyti komponentus“ PA specifikacija

PA „Nurodyti direktorijas“		
Tikslas. Nurodyti kuriuos komponentus perkelti į naująją platformą.		
Aprašymas		
Prieš sąlyga		Įskiepis paleistas.
Aktorius		Programuotojas.
Sužadinimo sąlyga		Programuotojas paleidžia *„Eclipse“* įskiepį.
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykiu srautas		Sistemos reakcija į sprendimus
1. Vartotojas nurodo kuriuos komponentus perkelti.		Įskiepis išsaugo informaciją ir pereina į antrą žingsnį.
2. Sistema baigia PA		
Alternatyvūs scenarijai		

Kadangi įskiepis didžiąją dalį veiksmų atlieka be vartotojo papildomo įsikišimo todėl atskira vartotojo sąsaja jam nėra kuriama. Pradinių duomenų įvedimui naudojami standartiniai *„Eclipse“* įrankio langai, jų pavyzdžiai matomi 20 ir 21 paveikslėliuose.



20pav. Originalios sistemos kodo bazės nurodymo langas

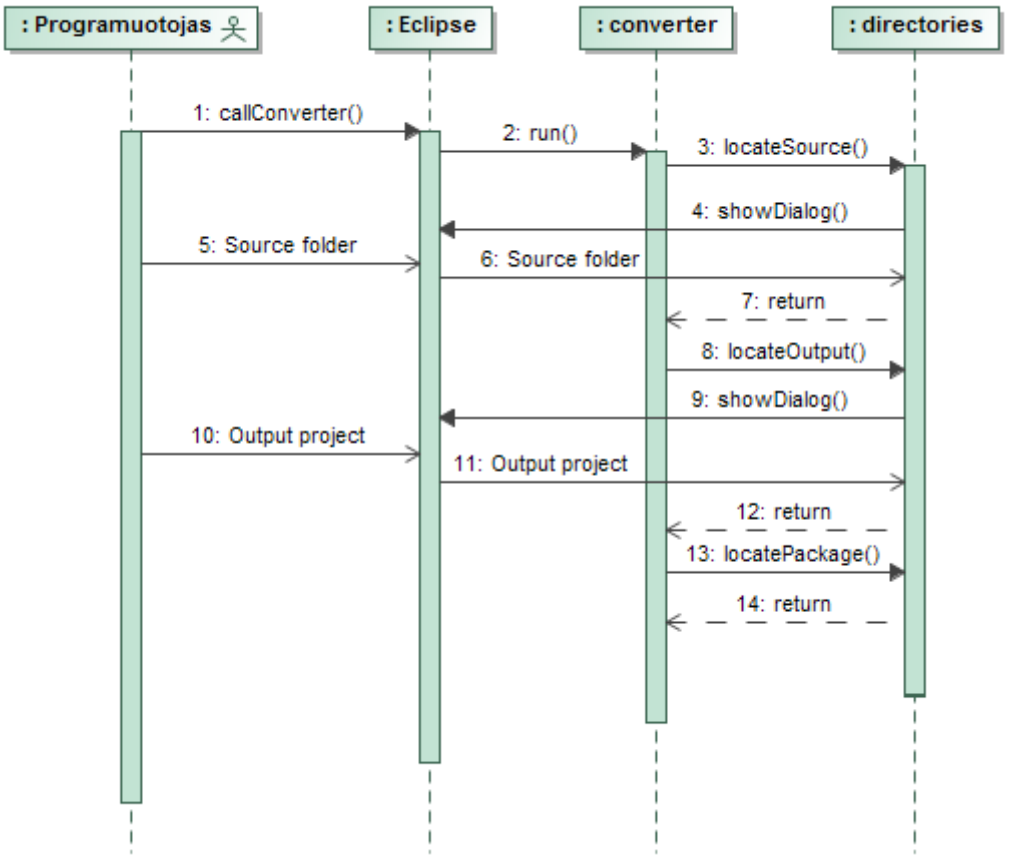


21pav. *„Android“* projekto pasirinkimo langas

3.2. Įskiepio projektas

3.2.1. Įskiepio elgsenos modelis

Įskiepio veiklą galima išskaidyti į keletą etapų. Pirmasis iš jų, tai informacijos apie originaliosios sistemos ir *„Android“* projekto direktorijų surinkimas. Šios veiklos sekų diagrama matoma 22 paveikslėlyje. Programuotojas, naudodamas įrankį *„Eclipse“*,paleidžia sukurtą įskiepį ir nurodo reikiamą informaciją.

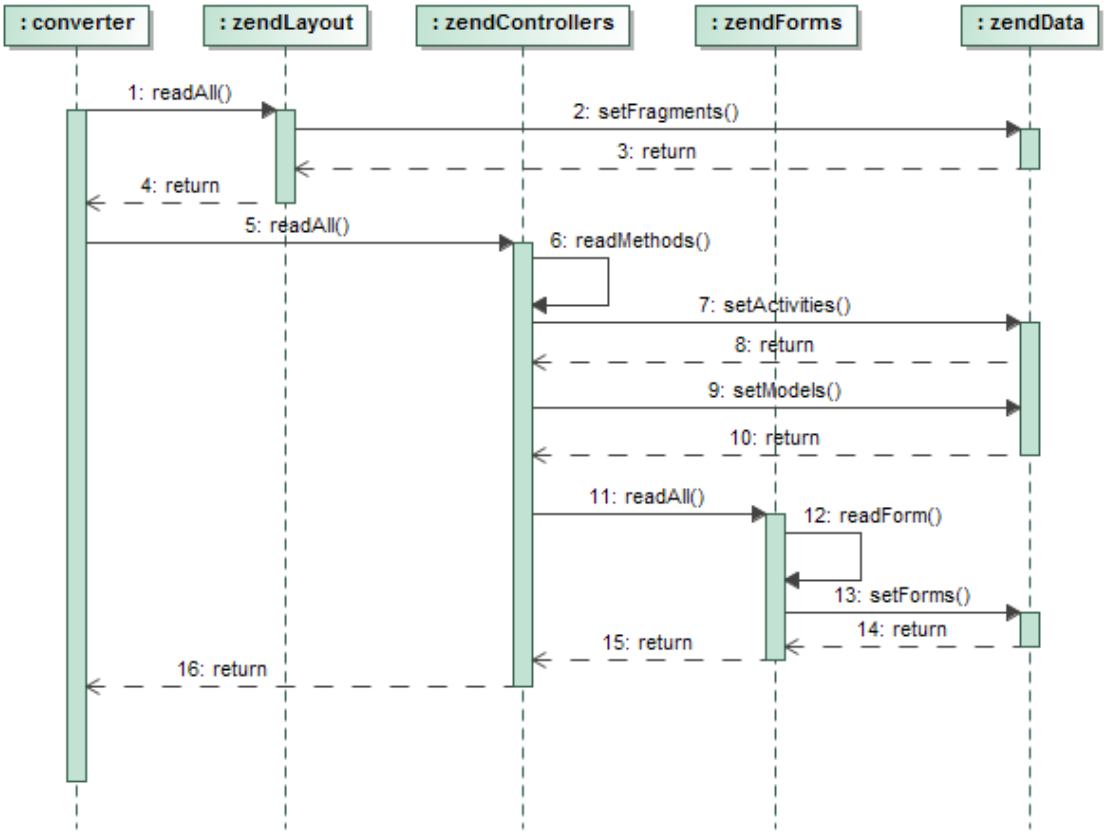


22pav. Informacijos apie direktorijas surinkimo sekų diagrama

Sekančiame žingsnyje įskiepis analizuoja nurodytą originaliosios sistemos direktoriją, ieškodamas tinkamų duomenų perkėlimui. Šio etapo sekų diagrama matoma 23 paveikslėlyje.

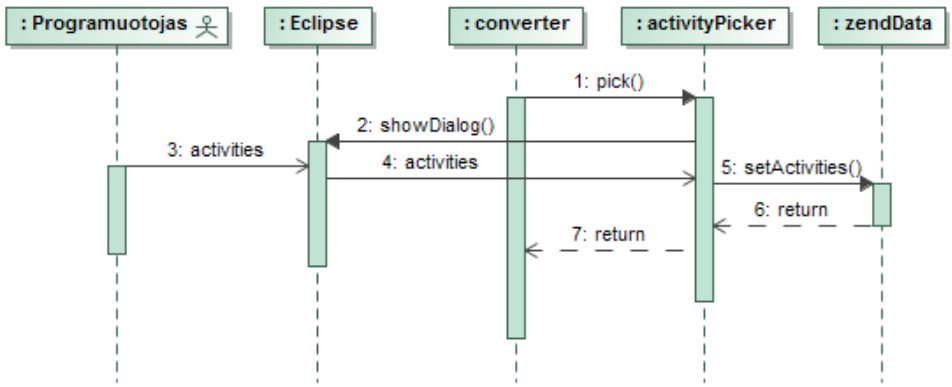
Pirmiausia analizuojamos *„layout“* klasės ieškant pakartotinai panaudotų komponentų. Radus jie įrašomi į duomenų klasę *„zendData“*.

Analogiškai analizuojamos ir *„controller“* klasės. Jose ieškoma viešų metodų, kreipinių į *„model“* tipo klases bei strukturizuotų formų. Radus strukturizuotų formų, iškviečiama*„zendForms“* klasė, kuri papildomai analizuoja formas ir surenka jų duomenis. Visa rasta informacija surašoma į jau minėtą duomenų klasę.



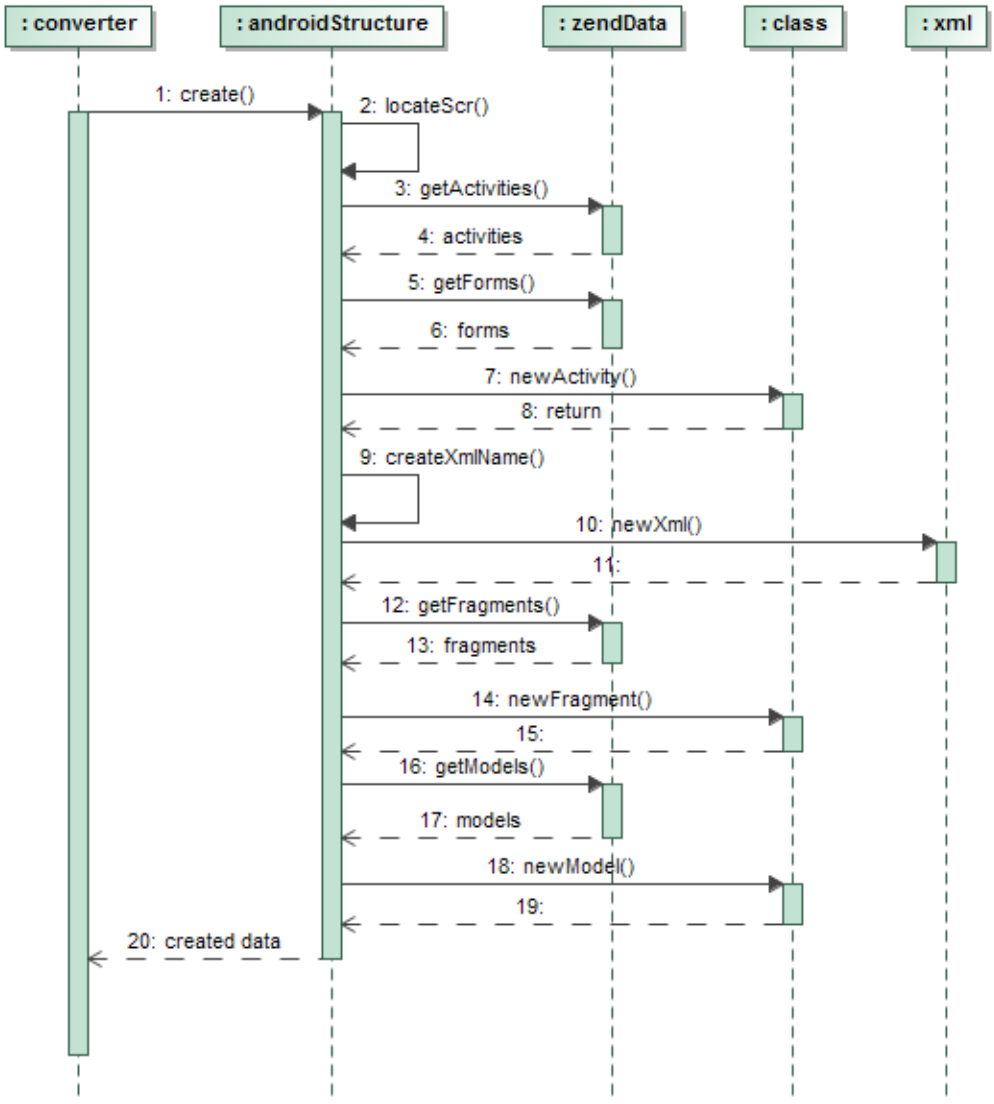
23pav. Originaliosios sistemos programinio kodo analizė

Trečiajame žingsnyje programuotojui duodama pasirinkti kuriuos *„activity“* komponentus jis nori sukurti naujojoje platformoje. Šio veiksmo sekų diagrama matoma 24 paveikslėlyje.



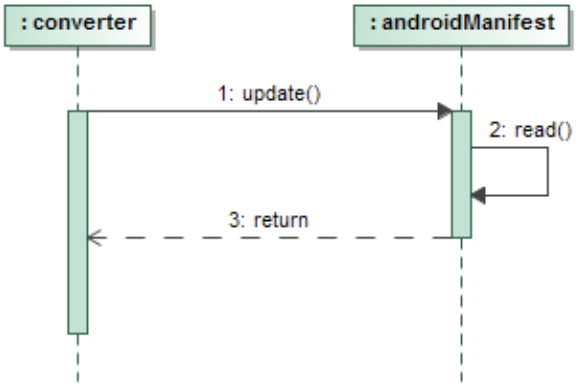
24pav. Komponentų parinkimo sekų diagrama

Sekančiame žingsnyje sukuriami komponentai *„Android“* projekte, remiamasi antrajame etape surinktais duomenimis, kurie buvo įrašyti į duomenų klasę *„zendData“*. Šis procesas matomas25 paveikslėlyje.



25pav. Naujos struktūros sukūrimas *„Android“* projekte

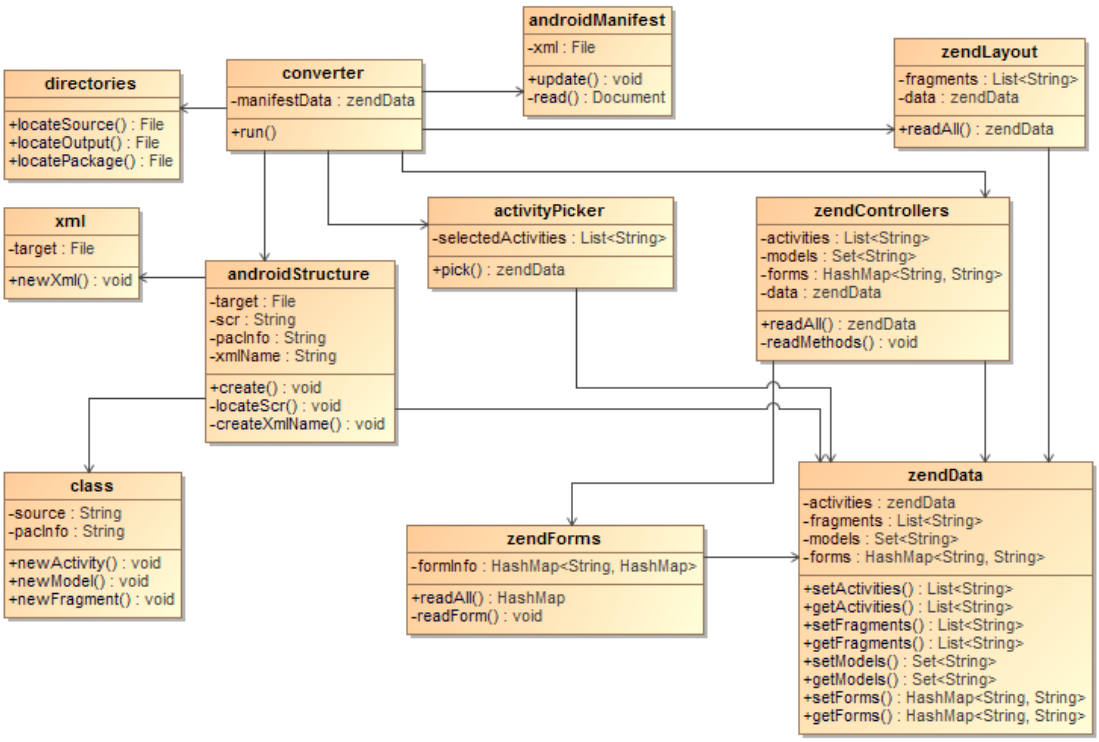
Paskutiniame etape, remiantis sukurtais komponentais, atnaujinamas *„androidManifest.xml“* failas ir įskiepis baigia darbą. Šis procesas matomas 26 paveikslėlyje.



26pav. *„androidManifest.xml“* failo atnaujinimas

3.2.2. Detalus projektas

Detalus klasių modelis, naudojamas visiems įskiepio veiksmams, pateikiamas 27 paveikslėlyje, o klasių specifikacijos pateikiamos 15-25 lentelėse.



27pav. Detalus klasių modelis

15lentelė. „converter“ klasės specifikacija

Kintamasis	**Paskirtis**
manifestData	Nurodo kokios klasės buvo sugeneruotos ir remiantis šia informacija yra atnaujinamas failas *„androidManifest.xml“*.
Metodas	**Paskirtis**
run()	Paleidžiamas įskiepis.

16lentelė. „directories“ klasės specifikacija

Metodas	**Paskirtis**
locateSource()	Nustatoma originaliosios sistemos kodo bazės direktorija.
locateOutput()	Nustatomas naujas *„Android“* projektas.
locatePackage()	Nustatomas naujojo projekto paketas.

17lentelė. „zendControllers“ klasės specifikacija

Kintamasis	**Paskirtis**
Activities	Duomenys apie projekte rastus *„activities“* komponentų atitikmenis.
Models	Duomenys apie projekte rastus *„models“* komponentų atitikmenis.
Forms	Duomenys apie projekte rastas struktūrizuotas formas.
Data	Surinkta informacija.
Metodas	**Paskirtis**
readAll()	Skaitomi visi originalios sistemos *„controller“* komponentai.
readMethods()	Skaitomi visi originalios sistemos, *„controller“* klasės metodai.

18lentelė. „zendLayout“ klasės specifikacija

Kintamasis	**Paskirtis**
Fragments	Duomenys apie projekte rastus *„fragments“* komponentų atitikmenis.
Data	Surinkta informacija.
Metodas	**Paskirtis**
readAll()	Skaitomi visi originalios sistemos *„layout“* komponentai.

19lentelė. „zendForms“ klasės specifikacija

Kintamasis	**Paskirtis**
formInfo	Informacija apie projekte rastas strutūrizuotas formas.
Metodas	**Paskirtis**
readAll()	Skaitomos visos struktūrizuotos formos.
readForm()	Analizuojama struktūrizuota forma.

20lentelė. „androidStructure“ klasės specifikacija

Kintamasis	**Paskirtis**
target	Projekto direktorija.
scr	Vieta kur reikia įrašyti komponentus.
pacInfo	*„Java“*paketo informacija.
xmlName	Failo *„xml“* pavadinimas.

Metodas	**Paskirtis**
Create()	Inicijuojamas naujos struktūros kūrimas.
locateScr()	Nustatoma vieta kur reikia įrašyti naujus komponentus.
createXmlName()	Pagal *„activity“* komponento vardą sukuriamas jam reikalingo *„xml“* failo vardas.

21lentelė. „class“ klasės specifikacija

Kintamasis	**Paskirtis**
Source	Šaltinis kuriame reikia sukurti naują klasę.
packInfo	Paketo informacija kuriame kuriama nauja klasė.
Metodas	**Paskirtis**
newActivity()	Sukuriamas naujas *„activity“* komponentas.
newModel()	Sukuriamas naujas *„model“* komponentas.
newFragment()	Sukuriamas naujas „fragment“ komponentas.

22lentelė. „xml“ klasės specifikacija

Kintamasis	**Paskirtis**
Target	Šaltinis kuriame reikia sukurti naują failą.
Metodas	**Paskirtis**
newXml()	Sukuriamas naujas *„xml“* failas ir užpildomas formų informacija jei ji buvo pateikta.

23lentelė. „androidManifest“ klasės specifikacija

Kintamasis	**Paskirtis**
Xml	*„androidManifest.xml“* dokumentas.
Metodas	**Paskirtis**
update()	Atnaujinamas *„androidManifest.xml“* failo turinys.
read()	Perskaitomas *„androidManifest.xml“* failo turinys.

24lentelė. „zendData“ klasės specifikacija

Kintamasis	**Paskirtis**
Activities	Informacija apie kuriamus *„activity“* komponentus.
Fragments	Informacija apie kuriamus *„fragment“* komponentus.
Models	Informacija apie kuriamus *„model“* komponentus.
Forms	Informacija apie struktūrizuotas formas.
Metodas	**Paskirtis**
setActivities()	Atnaujinama kintamojo *„activities“* informacija.
getActivities()	Gražina kintamojo *„activities“* informacija
setFragments()	Atnaujinama kintamojo *„fragments“* informacija.
getFragments()	Gražina kintamojo *„fragments“* informacija
setModels()	Atnaujinama

	kintamojo„*models*“informacija.
getModels()	Gražina kintamojo „*models*“ informacija
setForms()	Atnaujinama kintamojo „*forms*“ informacija.
getForms()	Gražina kintamojo „*forms*“ informacija

25lentelė. „activityPicker“ klasės specifikacija

Kintamasis	**Paskirtis**
selectedActivities	Vartotojo pasirinkti komponentai.
Metodas	**Paskirtis**
Pick()	Atidaro langą kuriame vartotojas nurodo kuriuos komponentus perkelti.

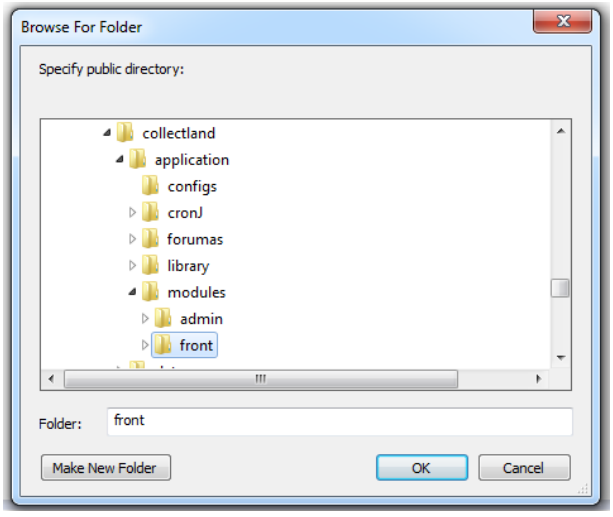
Prototipas naudoja 17 paveikslėlyje (antras skyrius) pateiką „*MySQL*“duomenų bazę.

4. Įskiepio eksperimentinis tyrimas

Atliekant eskperimentinį tyrimą naudojama kolekcionierių internetinė sistema. Eksperimente naudojama siūloma koncepcija, pagal siūlomus aspektus atrenkamos funkcijos, kurias planuojama perkelti į naująją sistemą.Kolekcionierių internetinės IS detalusprojektas pateiktas 1 priede.Turint atrinktų funkcijų sąrašą ir panaudojant sukurtą prototipą, kuris veikia programavimo aplinkoje „*Eclipse*“, automatiškai sugeneruojama dalis komponentų.

Iš pradžių susikuriame naują, tuščią„*Android*“ projektą pavadinimu„*Android project*“. Atlikus šį veiksmą viršutiniame „*Eclipse*“ meniu pasirenkame „*Reinžinerija->Kovertuoti*“ meniu punktą.

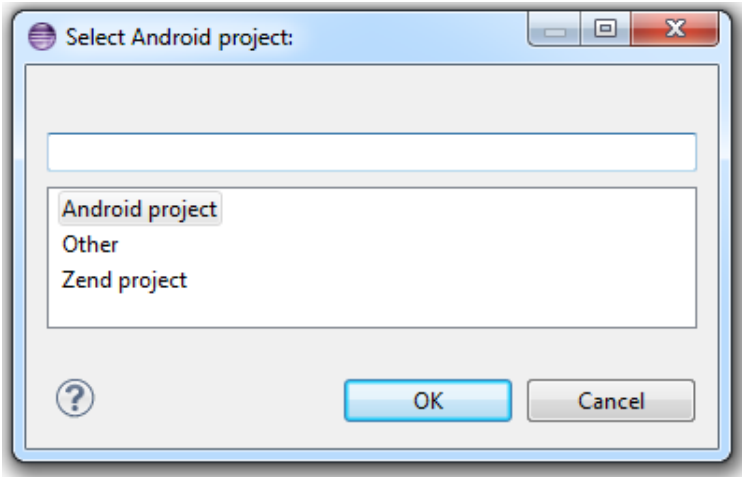
Originaliosis sitemos programinio kodo direktorijos nurodymas matomas 28 paveikslėlyje.



28pav. Viešos sistemos dalies direktorijos nurodymas

Kadangi eksperimentinėje IS (*„collectland“*) naudojamas viešos dalies atskyrimas nuo administracinės, programinio kodo šaltinį nurodome viešosios dalies direktoriją, o ne bendro projekto direktoriją.

Sekančiame žingsnyje nurodome susikurtą *„Android“* projektą, kuriame norime sugeneruoti dalį originaliosios sistemos komponentų, šiuo atveju *„Android project“*. Šis veiksmas matomas 29 paveikslėlyje.

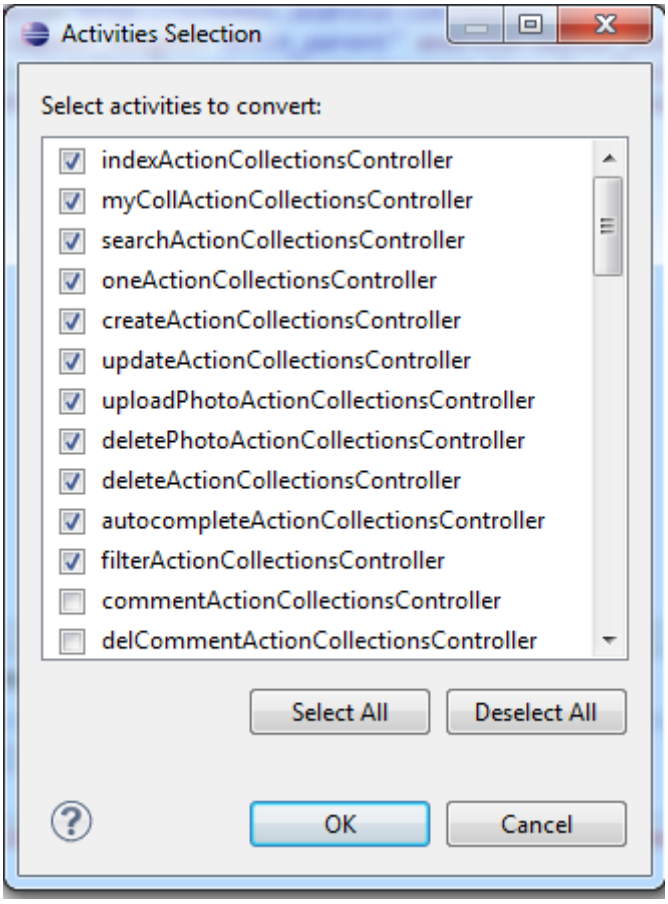


29pav. *„Android“* projekto pasirinkimas

Nurodžius tiek originaliosios sistemos direktoriją, tiek pasirinkus naująjį *„Android“* projektą įskiepis atlieka pirminę analizę ir surenka visus galimus *„activity“* tipo komponentus, kuriuos galima automatiškai sugeneruoti, remiantis originalios sistemos kodo baze.

Pasirinkimas, kuriuos *„activity“* komponentus reikia sugeneruoti, pateikiamas sąrašu(30 paveikslėlis) su pažymimomis varnelėmis.Pavadinimo paskutinė dalis nurodo kuriuo *„controller“* komponentu remiantis siūloma sukurti atitinkamus *„activity“* komponentus, pavyzdžiui *„CollectionsController“*. Pasinaudojus pirmąjameetape susidarytų reikiamų funkcijų sąrašu, pažymime reikiamus komponentus ir patvirtiname pasirinkimą.

Eksperimentinės sistemos atveju pasirinktos tik su kolekcijomis ir vartotojo informacija susijusios funkcijos, tačiau atsisakyta komentarų. Iš viso pasirinkti 26*„activity*“ komponentai.



30pav. „*Activity*“ komponentų pasirinkimo sąrašas

Po šio žingsnio vartotojas su įskiepiu daugiau nebendrauja ir atliekamas komponentų generavimas. Sukuriami visi pažymėti *„activity“* komponentai, kiekvienam iš jų sukuriamas *„xml“* failas (ir formos elementai jei jų rasta), jų naudojami *„model“* komponentai ir rasti *„fragment“* tipo komponentai.

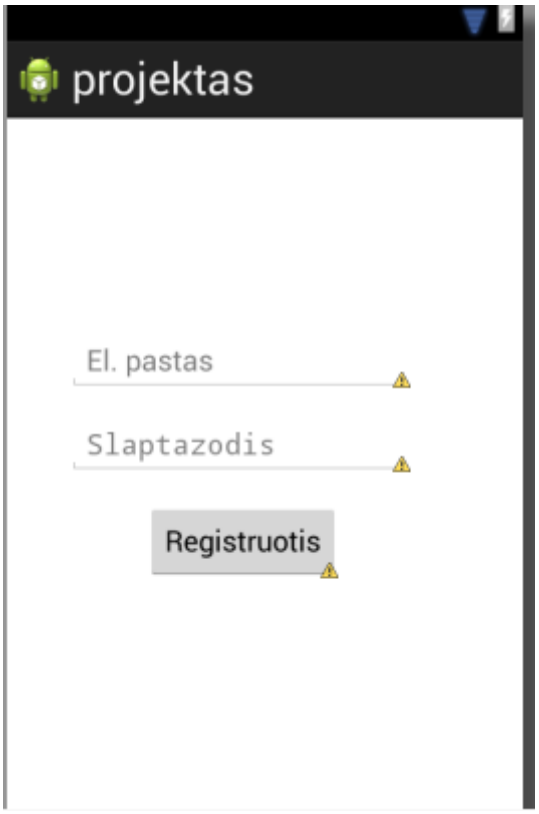
Generuojami komponentai programiniu kodu užpildomi minimaliai, tik standartiniais karkaso *„Android“* elementais ir nurodomas jam sukurtas *„xml“* failas skirtas vaizdavimui. Vieno iš sugeneruotų *„activity“* komponentų pavyzdys matomas 31 paveikslėlyje.

```
package com.example.projektas;

import android.os.Bundle;

public class createActionCollectionsController extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.create_action_collections_controller);
    }
}
```

31pav. Sugeneruoto kodo pavyzdys.

Dėl minimalaus kodo generavimo išvengiama grėsmės sugeneruoti klaidingą kodą, kurio didelę dalį programuotojas turėtų koreguoti užbaigus generavimą. Taip pat sudėtinga perkelti programinę logiką į kitą programavimo kalbą ir kitą aplinką (internetiniai serveriai ir išmaniųjų įrenginių platformos).

Eksperimento metu rastos trys strukturizuotai aprašytos formos, kurių įvedimo laukai ir mygtukai automatiškai perkelti į naująją aplinką. Vienos iš jų vaizdas„*Eclipse*“ vartotojo sąsajos redaktoriuje, matomas 32 paveikslėlyje. Kaikurių formų perkelti nepavyko dėl nestandartinio panaudojimo.



32pav. Registracijos forma su automatiškai sugeneruotais įvedimo laukais

Iš viso eksperimento metu sugeneruoti 59 komponentai: 26 *„activity“* komponentai, kiekvienam iš jų *„xml“* failas ir vienas *„fragment“* tipo komponentas su atitinkamu *„xml“* failu. 27 iš jų informacija surašyta į *„androidManifest.xml“* failą.

Kadangi nėra bandoma perkleti programinės logikos, sugeneruotas kodas neturi klaidų, naująjį projektą pavyksta iš karto sukompiliuoti naudojantis *„Eclipse“* įrankiu. Taip pat, dėl minimalaus generuojamų komponentų turinio, generavimo procesas (po direktorijų ir norimų *„activity“* komponentų pasirinkimo) užtrunka iki 5 sekundžių.

Pakartojus eksperimentą keleta kartų keičiant duomenų imtį, gaunami panašūs rezultatai. Pirmoji koncepcijos dalis padeda nuosekliau pasirinkti reikiamas funkcijas naujajai platformai, o realizuotas įskiepis paspartina mechaninį komponentų kūrimo procesą.Priklausomai nuo sistemos apimties šis procesas gali užimti nemažai laiko, kurį programuotojas galėtų išnaudoti efektyviau.

Siekiant praplėsti gautų rezultatų aibę, eksperimentas atliktas su dar keturiomis informacinėmis sistemomis. Kadangi darbo autorius nėra šių sistemų sąvininkas, detali jų specifikacija ir struktūra nepateikiama. Toliau patiekiami tik jų trumpi aprašymai.

Darbuotojų vertinimo informacinė sistema skirta įvertinti savo kolegų darbo rezultatus ir palengvinti valdybos darbą priimant sprendimus.

„Zend Framework“ karkaso pavyzdinė sistema. Sistema aiškios paskirties neturi, tai labiau pavyzdinės struktūros sistema.

Fotografijos IS skirta fotografams (ir fotografija besidomentiems) ir jų darbų publikavimui.

Buhalterinė IS skirta asmeninei vartotojų buhalterijai vesti.

Atlikus eksperimentą su visomis penkiomis IS gautas rezultatas ir pastebėjimai pateikiami 26 lentelėje.

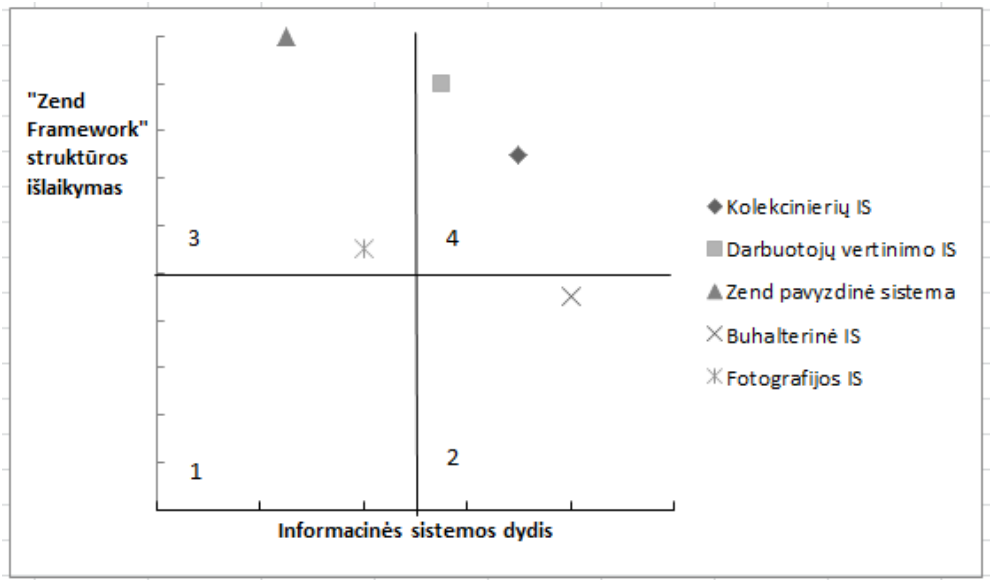
26 lentelė. Eksperimento rezultatai ir pastebėjimai

Pavadinimas	**IS dydis**	**Karkaso struktūros išlaikymas**	**Gautas rezultatas**	**Pastabos**
Kolekcionierių IS	Didesnė nei vidutinė	Aukštesnis nei vidutinis	Sugeneruoti 59 komponentai. 27 informacija surašyta į failą „androidManifest.xml“.	Dalis formų nenuskaitytos dėl nestandartinio panaudojimo.
Darbuotojų vertinimo IS	Vidutinė	Aukštas	Sugeneruoti 42 komponentai. 20 iš jų informacija surašyta į failą „androidManifest.xml“.	Kai kurie *„controller“* komponentai nuskaityti klaidingai dėl nestandartinio panaudojimo
Pavyzdinė sistema	Maža	Labai aukštas	Sugeneruoti 22 komponentai. 10 iš jų	Problemų nekilo.

			informacija surašyta į failą „androidManifest.xml“.	
Fotografijos IS	Mažesnė nei vidutinė	Vidutinis	Sugeneruoti 35 komponentai. 16 iš jų informacija surašyta į failą „androidManifest.xml“.	Nenaudojamos struktūrizuotos formos, todėl formų perkelti nepavyko.
Buhalterinė IS	Didelė	Žemesnis nei vidutinis	Sugeneruoti 55 komponentai. 25 iš jų informacija surašyta į failą „androidManifest.xml“.	Perkelta daug komponentų, tačiau didelė dalis buvo neperkelti dėl nestandartinio panaudojimo.

Remiantis gautais rezultatais sudarytas grafikas matomas 33 paveikslėlyje. Informacinės sistemos, su kuriomis atliktas eksperimentas, išdėstytos pagal dvi ašis: karkaso struktūros išlaikymą ir IS dydį. Grafikas suskirstytas į keturias dalis pagal kurias galima klasifikuoti informacines sistemas.

Eksperimentas buvo atliktas su nedidele informacinių sistemų aibe, tačiau pagrindinis aspektas yra šių sistemų struktūra. Kadangi eksperimentas atliekamas tik su *„Zend Framework“* struktūros besilaikančiomis sistemomis pakanka išanalizuoti keletą iš jų – programinio kodo struktūra išlieka panaši, o galimus nukrypimus galima nuspėti.



33 pav. Informacinių sistemų išsidėstymas

Pirmasis ketvirtis, tai mažos apimties informacinės sistemos, kurios neišlaiko karkaso standartinės struktūros. Eksperientuojama su šio tipo informacinės sistemomis nebuvo, tačiau, atsižvelgiant į antro ir trečio ketvirčio tipo sistemas, galima daryti tam tikras prielaidas. Šio darbo siūlomas sprendimas šioms sistemos gali nepasiteisinti,nes nebus sugeneruojamas didelis skaičius komponentų, o dalis jų bus apskritai praleidžiami. Siūlomas funkcijų atrinkimas taip pat gali būti nelabai naudingas, kadangi esant nedideliam funkcijų skaičiui jas galima lengvai klasifikuoti be papildomų metodologijų.

Antras ketvirtis, tai didelės apimties informacinės sistemos, kurios neišlaiko karkaso standartinės struktūros. Funkcijų atrinkimas šiuo atveju naudingas, tačiau, priklausomai nuo struktūrinių nukrypimų, didelė dalis komponentų gali būti nesugeneruoti ir prireiks programuotojų rankinio darbo norint perkelti visus reikiamus komponentus.

Trečiasis ketvirtis, tai mažos apimties informacinės sistemos, kurios išlaiko karkaso standartinę struktūrą. Šios sistemos didžioji dalis komponentų (esant labai aukštam struktūros išlaikymui ir visi komponentai) sėkmingai perkeliami į naują aplinką. Reikalingų funkcijų atrinkimas gali būti mažiau reikalingas, nes esant mažai informacinės sistemos apimčiai galima jas nesudėtingai prioritezuoti ir be papildomų metodologijų. Naudojantis prototipu gaunanamas rezultatas yra teisingas ir paspartinantis programuotojo darbą.

Ketvirtasis ketvirtis, tai didelės apimties informacinės sistemos, kurios išlaiko karkaso standartinę struktūrą. Šiuo atveju naudingas tiek perkeliamų funkcijų atrinkimo metodas, tiek automatinis komponentų sukūrimas naujojoje aplinkoje. Į šį ketvirtį patenkančioms informacinėms sistemoms siūlomas sprendimas tinka labiausiai.

Apibendrinant eksperimento rezultatus, galima teigti, jog informacinėms sistemoms, patenkančioms į antrąjį ir trečiąjį ketvirčius, naudinga pasinaudoti darbe siūlomu sprendimu. Į ketvirtąjį ketvirtį patenkančių IS pritaikymui naująjai platformai siūlomas sprendimas tinka labiausiai, o į pirmąjį ketvirti patenkančiomis IS naudotis siūlomu sprendimu nerekomenduojama.

5. Išvados

1. Buvo atlikta išmaniųjų įrenginių funkcinių galimybių analizė, kuri parodė, kad identiškas funkcijų perkėlimas iš įprastinės internetinės sistemos į išmaniajam įrenginiui skirtą aplinką gali neišnaudoti išmaniesiems įrenginiams būdingų vartotojo sąsajos funkcinių elementų bei perkrauti pačią sistemą nereikalingomis funkcijomis.
2. Darbe buvo aptarti internetinės sistemos ir jos pagrindu sukurtos išmaniojo įrenginio informacinės sistemos sąveikos modeliai bei aptartos galimos duomenų sinchronizacijos problemos.
3. Darbe buvo pateiktas karkaso „*Zend Framework*“ pagrindu sukurtos internetinės informacinės sistemos pritaikymo išmaniajam įrenginiui koncepcinis sprendimas, kuris padaro programuotojo darbą našesnį ir atliekamo sistemos pritaikymo proceso rezultatą kokybiškesnį.
4. Pateikto koncepcinio sprendimo pagrindu buvo sukurtas programinis prototipas, kuris išbandytas su kolekcionierių internetine informacine sistema. Eksperimentinis tyrimas parodė, kad adaptavimo procesas gali būti sėkmingai automatizuotas priklausomai nuo struktūros išlaikymo ir informacinės sistemos dydžio.
5. Tolimesniuose tyrimuose šį koncepcinį sprendimą norima integruoti į projektavimo įrankį „*MagicDraw*“ siekiant automatizuoti perkeliamų funkcijų pasirinkimo ir perėjimo prie programinio kodo generavimo žingsnį.
6. Taip pat atsižvelgiant į įvairių „*PHP*“ programavimo karkasų panašumus bus siekiama išplėsti sprendimą, kad jis apimtų daugiau negu vieną („*Zend Framework*“) programinį karkasą.

6. Literatūros sąrašas

1. M. Hinz, Z. Fiala, F. Wehner, Personalization-Based Optimization of Web Interfaces for Mobile Devices, Mobile Human-Computer Interaction-MobileHCI, 2004, pp. 204-215.
2. SunSpider testas, atnaujinta 2013, [interaktyvus]. Prieiga per internetą: <http://www.webkit.org/perf/sunspider/sunspider.html>.
3. J. Tidwell, Designing Interfaces. 2-oji laida. Kanada, 2010.
4. M. Hjerde, Mobile Screen Size Trends, 2008. Prieiga per internetą: <http://sender11.typepad.com/sender11/2008/04/mobile-screen-s.html>.
5. K. Knight, Responsive Web Design: What It Is and How To Use It, 2011. Prieiga per internetą: <http://coding.smashingmagazine.com/2011/01/12/guidelines-for-responsive-web-design/>.
6. M. Song, H. Song, X. Fu, Methodology of user interfaces design based on Android, Multimedia Technology (ICMT), 2011 International Conference, 2011 (July), pp. 408-411.
7. B. Block, Android Captures #2 Ranking Among Smartphone Platforms in EU5, 2011.Prieiga per internetą: <http://www.comscore.com/Press_Events/Press_Releases/2011/9/Android_Captures_number_2_Ranking_Among_Smartphone_Platforms_in_EU5>.
8. L. Vogel, Android Development Tutorial, 2011, [interaktyvus]. Prieiga per internetą: <http://www.vogella.de/articles/Android/article.html>.
9. J. Stark, Building Apps with HTML, CSS, and JavaScript. 1-oji laida. JAV, 2010.
10. Android API Guides, atnaujinta 2013, [interaktyvus]. Prieiga per internetą: <http://developer.android.com/guide/topics/manifest/manifest-intro.html>.
11. M.-Y. Choi, E.-A. Cho, D.-H. Park, C.-J. Moon, D.-K. Baik, A Database Synchronization Algorithm for Mobile Devices, Knowledge Creation Diffusion Utilization, 2010, vol. 56, pp. 392-398.
12. S. Gansemer, U. Gröner, M. Maus, Database Classification of Mobile Devices, IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems Technology and Applications, 2007(September), pp. 699-703.
13. I.McDowall, Programming PC connectivity applications for Symbian OS. 1-oji laida. JAV, 2005.
14. W. Höpken, M. Fuchs, M. Zanker, T. Beer, "Context-Based Adaptation of Mobile Applications in Tourism", Information Technology & Tourism, 2010, vol. 12, no. 2, pp. 175-195.
15. P. Cotter, B. Smyth, Content Personalisation for WAP-Enabled Devices, Adaptive Hypermedia and Adaptive Web-Based Systems, Lecture Notes in Computer Science, 2000, vol. 1892, pp. 98-108.

7. Priedai

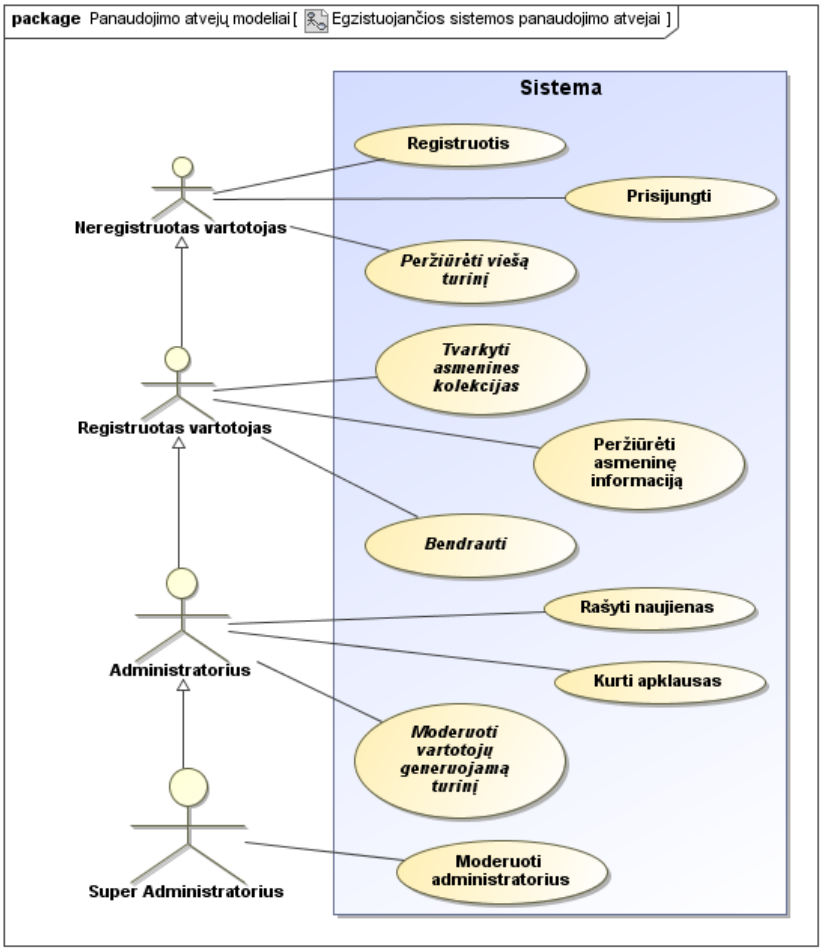
1. Priedas: Eksperimentinės sistemos reikalavimų specifikacija

1.1. Reikalavimų specifikacija

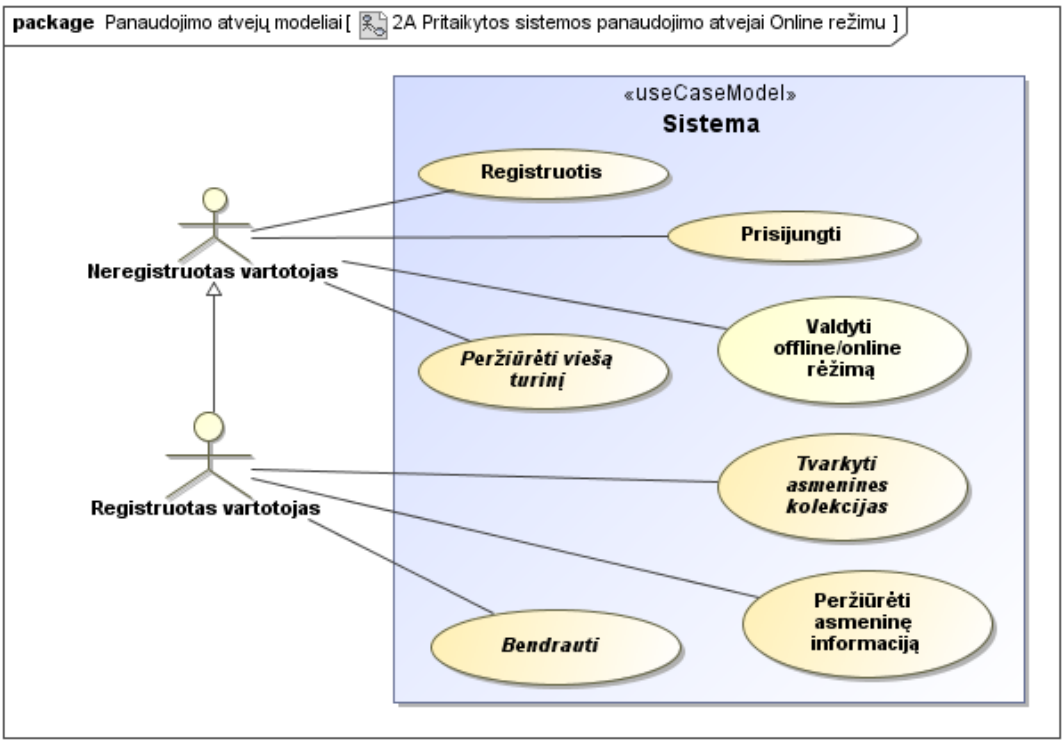
1.1.1. Funkciniai reikalavimai

Pritaikant sistemą visų pirma sudaromaabstrakti egzistuojančios sistemos panaudojimo atvejų diagrama, kuri matoma 1 paveikslėlyje. Panaudojimo atvejai nedetalizuojami. Šiame etape nustatoma, kuriuos panaudojimo atvejus reikia pašalinti, o kuriuos pridėti atsižvelgiant į naujos platformos galimybes bei trūkumus.

Pritaikytos sistemos abstrakti panaudojimo atvejų diagrama matoma2 paveikslėlyje.



1 pav. Egzistuojančios sisteos abstraktūs panaudojimo atvejai

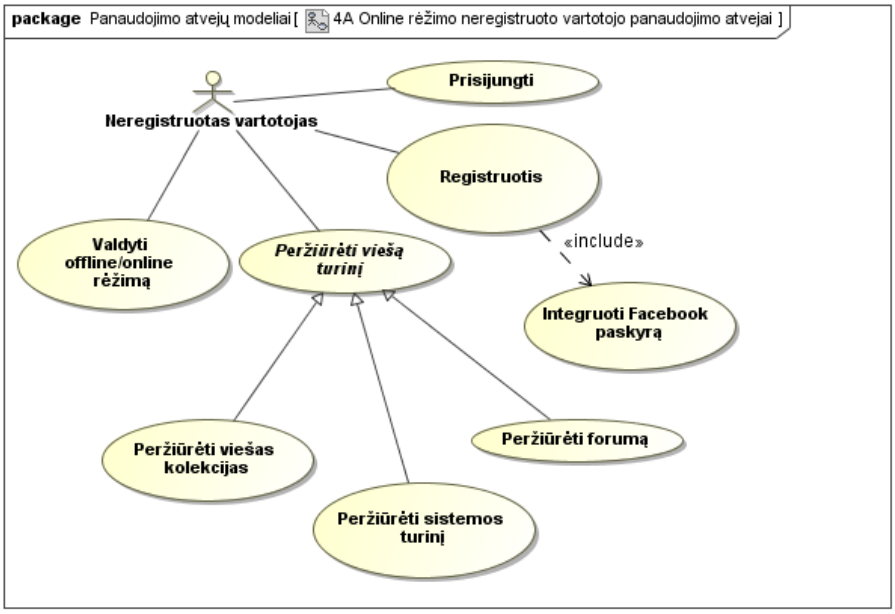


2 pav. Pritaikytos sistemos abstraktūs panaudojimo atvejai

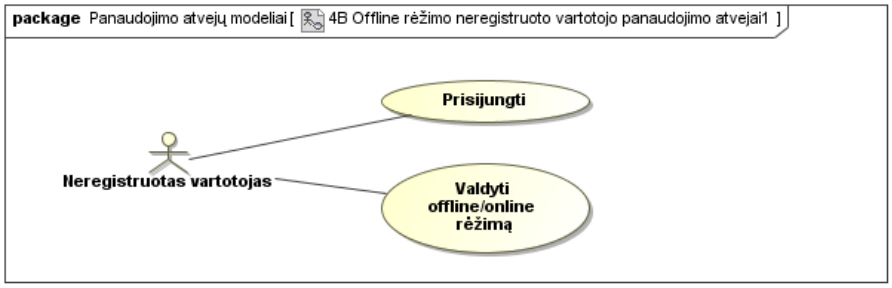
Pritaikant sistemą atsisakyta „Administratorius“ ir „Super Administratorius“ rolių bei jų panaudojimo atvejų. Tradiciškai administracinės dalies funkcionalumo patogumui skiriama mažiau resursų negu viešos dalies vystimui. Taip pat administratorių funkcionalumas yra sudėtingesnis negu paprastų vartotojų. Atsižvelgiant į tai jog, išmaniesiems telefonams ypač svarbu patogi vartotojo sąsaja ir nesudėtingas funkcionalumas, buvo nuspręsta administracinės dalies pritaikant IS atsisakyti.

Likusios dvi rolės bei jų funkcionalumas išlaikytas ir papildytas nauju panaudojimo atveju „Valdyti offline/online rėžimą“ skirtu IS atjungimui nuo interneto ir apriboto funckionalumo naudojimui.

Lygiagrečiai detalizuoti neregistruoto vartotojo panaudojimo atvejai ir kai IS jungiasi prie interneto, ir kai ji veikia lokaliai. Šių panaudojimo atvejų diagramos atitinkamai matomos 3 ir 4 paveikslėliuose.



3 pav. Neregistruoto vartotojo PA diagrama, kuomet IS jungiasi prie interneto



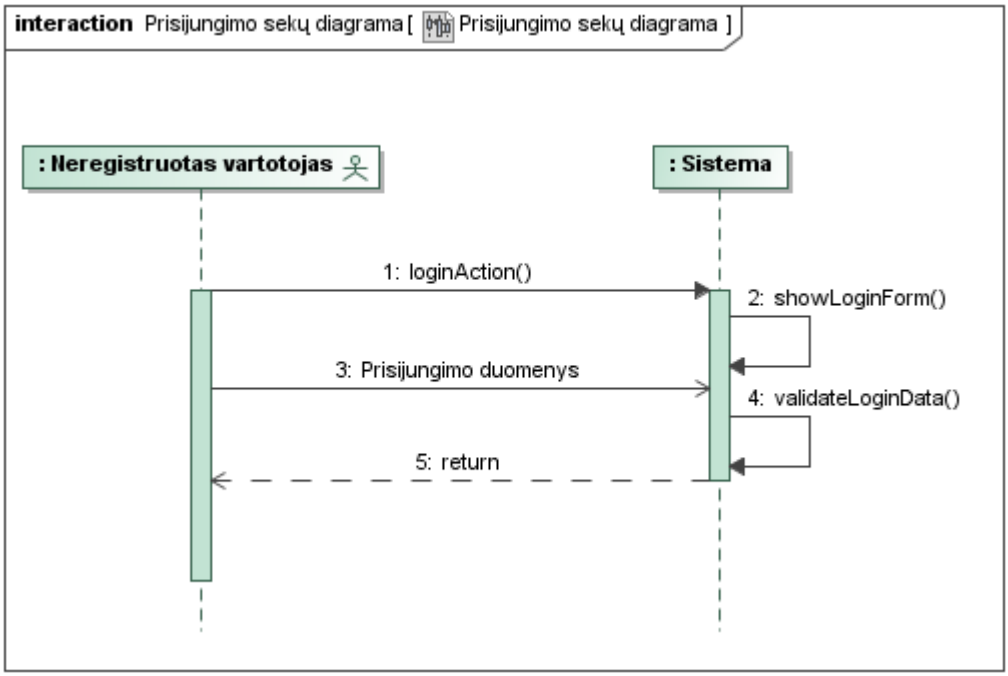
4 pav. Neregistruoto vartotojo PA diagrama, kuomet IS veikia lokaliai

Toliau pateikiamos pagrindinių panaudojimo atvejų specifikacijos. Atskiros specifikacijos kai sistema veikia lokaliai ir kai jungiasi prie tinklo nepateikiamos.

Panaudojimo atvejo „Prisijungti“ specifikacija pateikiama 1 lentelėje, o sekų diagrama 5 paveikslėlyje.

1 lentelė. „Prisijungti“ PA specifikacija

PA „Prisijungti“		
Tikslas. Leisti varotojui prisijungti prie sistemos		
Aprašymas Šis PA apima vartotojų prisijungimą		
Prieš sąlyga		Neregistruotas vartotojas naudojasi sistema.
Aktorius		Neregistruotas vartotojas
Sužadinimo sąlyga		Neregistruotas vartotojas nori prisijungti prie sistemos
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykiu srautas		Sistemos reakcija į sprendimus
1. Vartotojas įveda prisijungimo duomenis		Sistema prijungia vartotoją prie sistemos ir pereina į 2 žingsnį.
2. Sistema baigia PA		
Alternatyvūs scenarijai		
1a. Vartotojo įvesti duomenis klaidingi		Sistema perduoda pranešimą „Pateikti duomenys neteisingi“
Po sąlyga:		Neregistruotas vartotojas tampa registruotu vartotoju

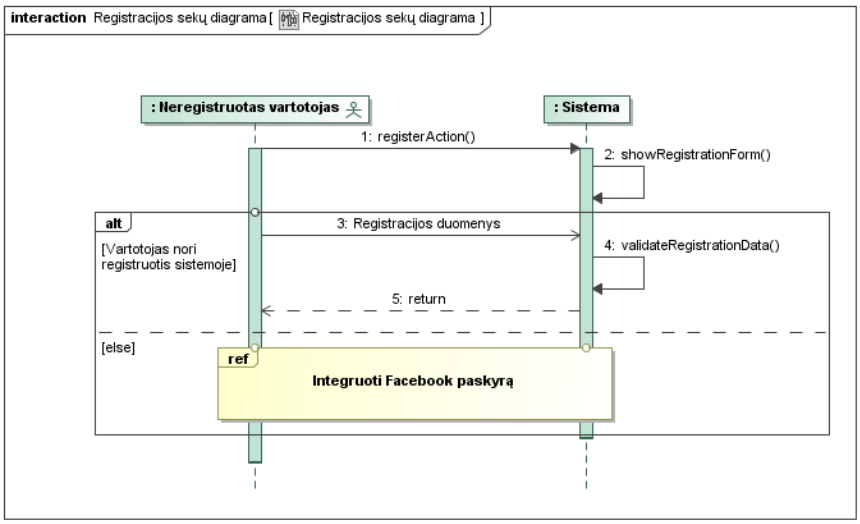


5 pav. „Prisijungti“ PA sekų diagrama

Panaudojimo atvejo „Registruotis“ specifikacija pateikiama 2 lentelėje, o sekų diagrama 6 paveikslėlyje.

2 lentelė. „Registruotis“ PA specifikacija

PA „Registruotis“		
Tikslas. Neregistruoto vartotojo registracija, taip pat apima ir Facebook integracija PA		
Aprašymas Šis PA apima vartotojų registracija sistemoje ir forume bei Facebook integracija		
Prieš sąlyga		Neregistruotas vartotojas naudojasi kuriama sistema, sistema jungiasi prie interneto.
Aktorius		Neregistruotas vartotojas
Sužadinimo sąlyga		Neregistruotas vartotojas nori registruotis sistemoje
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	Integruoti Facebook paskyrą
	Specializuoja PA	
Pagrindinis įvykiu srautas		Sistemos reakcija į sprendimus
1. Vartotojas įveda reikiamus registracijai duomenis		
2. Vartotojas tvirtina registraciją		Sistema išsaugo vartotooi prisijungimo duomenis duomenų bazėje ir pereina į 3 žingsnį
3. Sistema baigia PA		
Alternatyvūs scenarijai		
1a. Vartotojo įvesti duomenis netinkami registracijai		Sistema perduoda pranešimą „Pataisykite registracijos duomenis“.
1b. Vartotojas nori naudoti savo facebook prisijungimo duomenis		Sistema įvygdo PA „Integruoti Facebook paskyrą“ ir pereina į 3 žingsnį.
Po sąlyga:		Vartotojas užregistruojamas sistemoje
Pastabos		
1. Netinkami duomenys- toks prisijungimo vardas jau egzistuoja duomenu bazėje, nesaugus slaptažodis, nesutampa pakartotinai įvestas slaptažodis.		

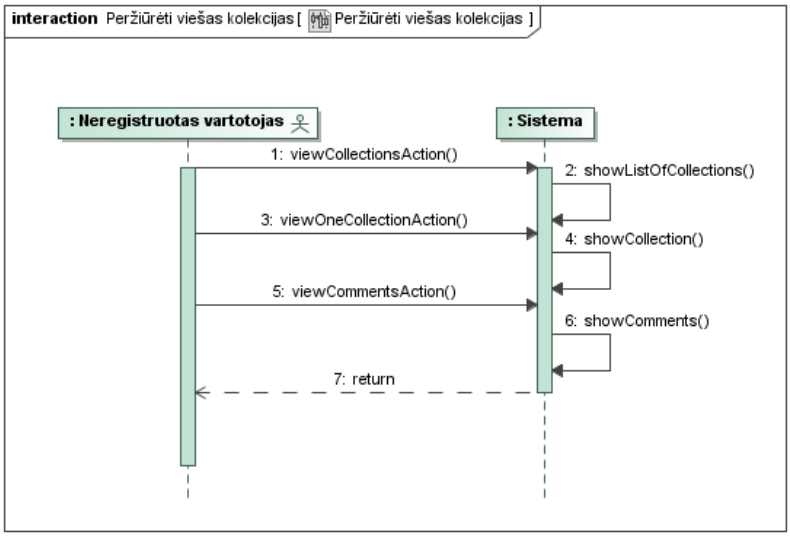


6 pav. „Registruotis“ PA sekų diagrama

Panaudojimo atvejo „Peržiūrėti viešas kolekcijas“ specifikacija pateikiama 3 lentelėje, o sekų diagrama 7 paveiksle.

3 lentelė. „Peržiūrėti viešas kolekcijas“ PA specifikacija

PA „Peržiūrėti viešas kolekcijas“		
Tikslas. Peržiūrėti viešas kolekcijas		
Aprašymas Šis PA yra „Peržiūrėti viešą turinį“ dalis		
Prieš sąlyga		Neregistruotas vartotojas naudojasi sistema, sistema jungiasi prie interneto.
Aktorius		Neregistruotas vartotojas
Sužadinimo sąlyga		Neregistruotas vartotojas peržiuri kolekcijas
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	Peržiūrėti viešą turinį
Pagrindinis įvykiu srautas		Sistemos reakcija į sprendimus
1. Vartotojas pasirenka kolekcijų peržiūra		Sistema pateikia kolekcijų sąrašą
2. Vartotojas pasirenka norimą kolekciją		Sistema pateikia pasirinktą kolekciją
3. Vartotojas skaito kolekcijos komentarus		Sistema pateikia kolekcijos komentarus
4. Vartotojas baigia PA		
Alternatyvūs scenarijai		
Pastabos		

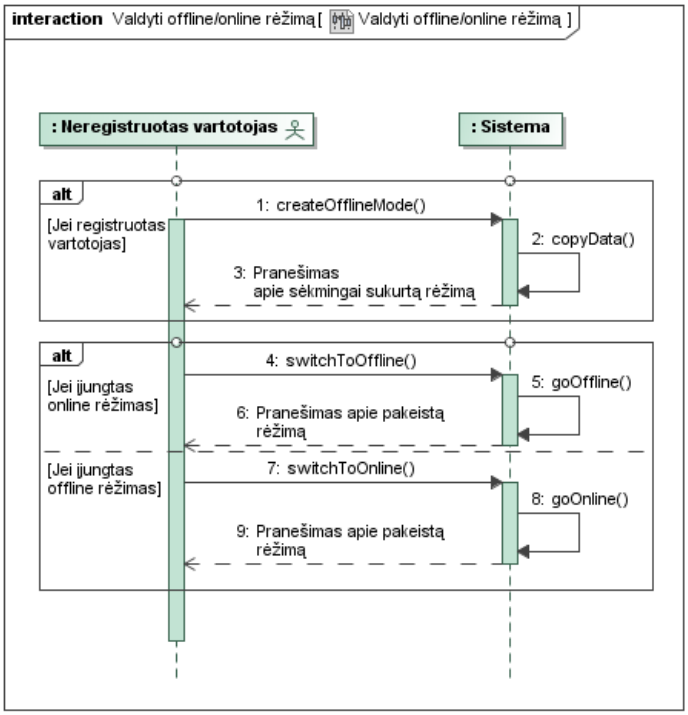


7 pav. „Peržiūrėti viešas kolekcijas“ PA specifikacija

Panaudojimo atvejo „Valdyti offline/online rėžimą“ specifikacija pateikiama 4 lentelėje, o sekų diagrama 8 paveiksle.

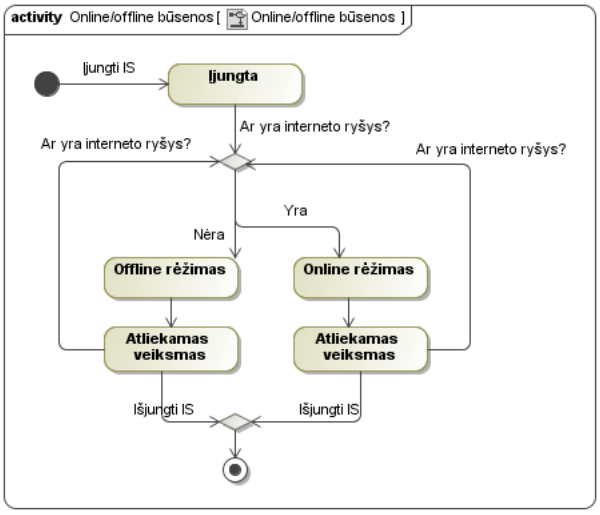
4 lentelė. „Valdyti offline/online rėžimą“ PA specifikacija

PA „Valdyti offline/online rėžimą“		
Tikslas. Pakeisti veikimo rėžimą		
Aprašymas		
Prieš sąlyga		Neregistruotas vartotojas naudojasi sistemas
Aktorius		Neregistruotas vartotojas, registruotas vartotojas
Sužadinimo sąlyga		Neregistruotas vartotojas atsidaro rėžimo valdymo langą
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykiu srautas		Sistemos reakcija į sprendimus
1. Vartotojas pasirenka rėžimo perjungimą.		Sistema pateikia valdymo langą
2. Vartotojas įjungia „Offline“ rėžimą		Sistema pereina į lokalujį rėžimą ir pereina į 3 žingsnį..
3. Sistema baigia PA		
Alternatyvūs scenarijai		
2a. Vartotojas įjungia „Online“ rėžimą		Sistema bando jungtis prie egzistuojančios IS serverio ir pereina į 3 žingsnį.
2b. Registruotojas vartotojas spaudžia „Offline“ paskyros sukūrimo mygtuką.		Sistema parsiunčia reikiamus duomenis iš serverio ir išsaugo juos lokaliai, pereina į 3 žingsnį.
Pastabos		
1. Persijungimas į „Offline“ rėžimą neveikia tol kol registruotas vartotojas nesukuria „offline“ rėžimo ir sistema neparsisiunčia reikiamų duomenų.		



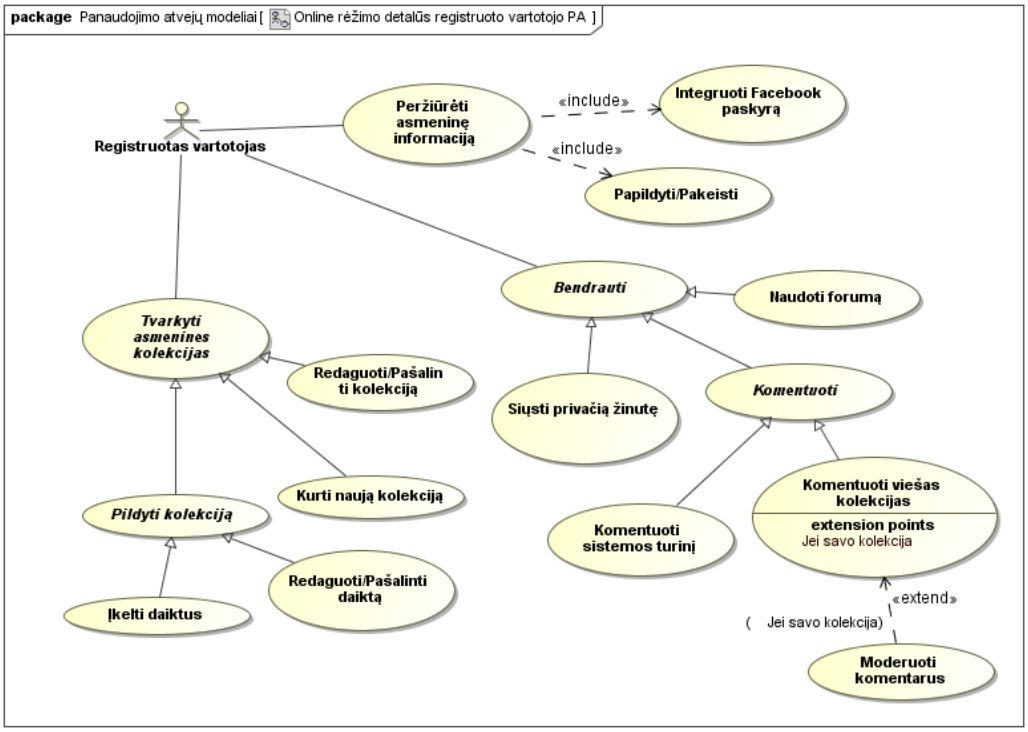
8 pav. „Valdyti offline/online rėžimą“ PA specifikacija

Sistema nuolat tikrina ar yra interneto ryšys ir jam dingus automatiškai perjungia sistemą į lokalųjį rėžimą. Šio tikrinimo būsenų diagrama pateikiama9 paveikslėlyje.

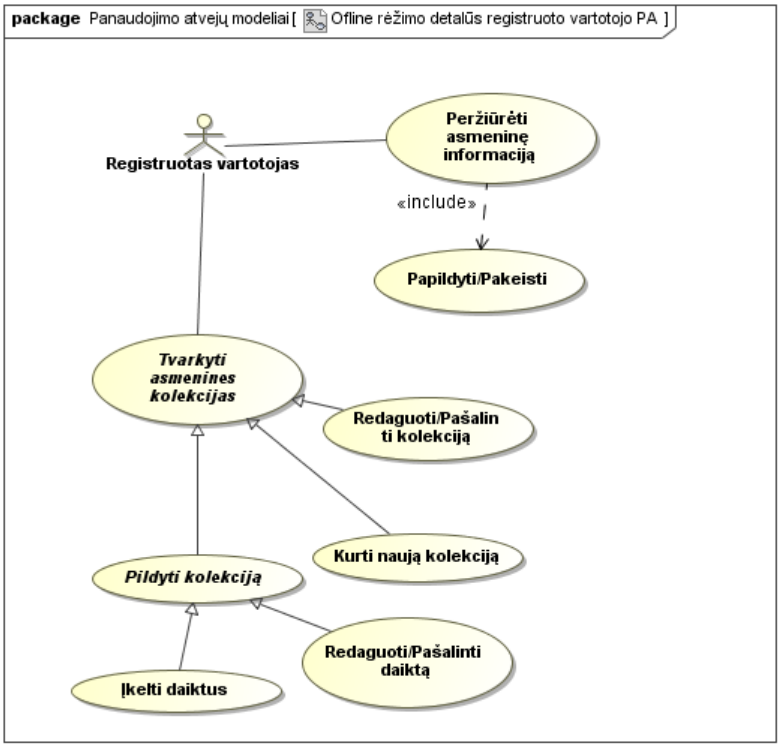


9 pav. Ryšio tikrinimo būsenų diagrama

Sudaryti detalūs registruoto vartotojo panaudojimo atvejų modeliai, taip pat ir kai sistema jungiasi prie egzistuojančios IS serverio, ir kai sistema veikia lokaliai. Jos matomos 10 ir 11 paveikslėliuose. Atskiros specifikacijos bei sekų diagramos nesudarytos, nes iš vartotojo pusės veikimas nesikeičia.



10 pav. Registruotojo vartotojo PA modelis kai sistema jungiasi prie serverio

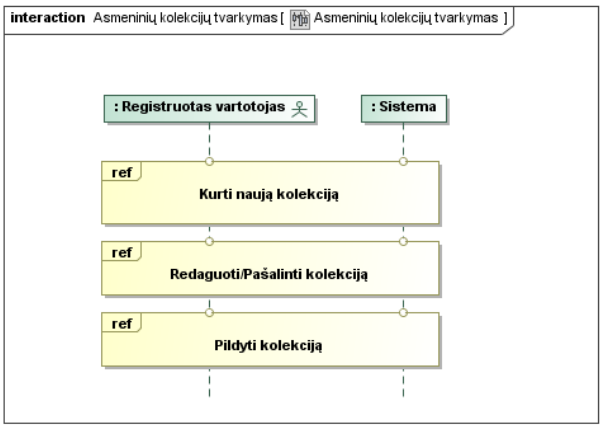


11 pav. Registruoto vartotojo PA modelis kai sistema veikia lokaliai

Abstraktaus panaudojimo atvejo „Tvarkyti asmenines kolekcijas“ specifikacija pateikiama 5 lentelėje, o sekų diagrama 12 paveikslėlyje.

5 lentelė. „Tvarkyti asmenines kolekcijas“ PA specifikacija

PA „Tvarkyti asmenines kolekcijas“		
Tikslas. Apibendrintai pavaizduoti kolekcijų kūrimo, redagavimo, pildymo ir naikinimo PA		
Aprašymas Šis PA apima kolekcijų kūrima, redagavimą, pildyma ir naikinimą		
Prieš sąlyga		Vartotojas sekmingai prisijungęs prie sistemos
Aktorius		Registruotas vartotojas
Sužadinimo sąlyga		Vartotojas nori tvarkyti savo kolekciją
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	Kurti naują kolekciją, Pildyti kolekciją, Redaguoti/Pašalinti kolekciją
	Specializuoja PA	
Pagrindinis įvykiu srautas		Sistemos reakcija į sprendimus
1. Vartotojas nori susikurti naują kolekciją		Sistema įvygdo PA „Kurti naują kolekciją“ ir pereina į 4 žingsnį
2. Vartotojas nori pridėti naują daiktą į kolekciją		Sistema įvygdo PA „Pildyti kolekciją“ ir pereina į 4 žingsnį
3. Vartotojas nori redaguoti arba naikinti kolekciją		Sistema įvygdo PA „Redaguoti/Pašalinti kolekciją“ ir pereina į 4 žingsnį
4. Sistema baigia PA		
Alternatyvūs scenarijai		
Pastabos		

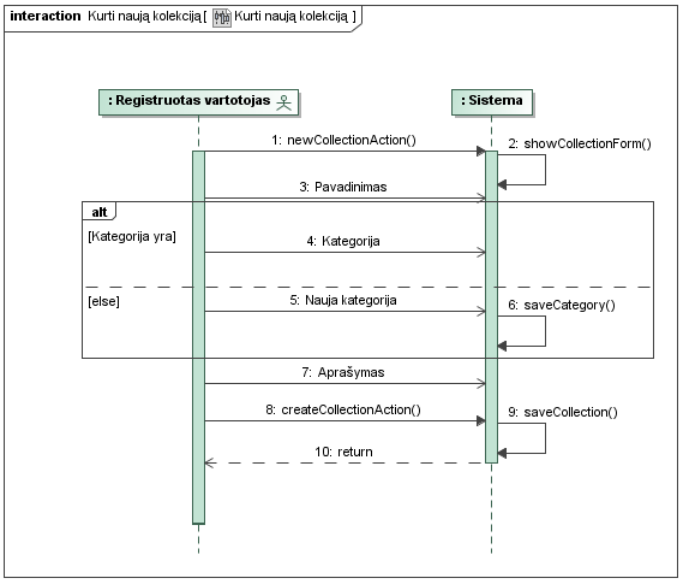


12 pav. „Tvarkyti asmenines kolekcijas“ PA sekų diagrama

Panaudojimo atvejo „Kurti naują kolekciją“ specifikacija pateikiama 6 lentelėje, o sekų diagrama 13 paveiksle.

6 lentelė. „Kurti naują kolekciją“ PA specifikacija

PA „Kurti naują kolekciją“		
Tikslas. Sukurti naują kolekciją		
Aprašymas Šis PA yra „Tvarkyti asmenines kolekcijas“ dalis		
Prieš sąlyga		Vartotojas sekmingai prisijungęs prie sistemos
Aktorius		Registruotas vartotojas
Sužadinimo sąlyga		Vartotojas nori kurti naują kolekciją(iš PA „Tvarkyti asmenines kolekcijas“)
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	Tvarkyti asmenines kolekcijas
Pagrindinis įvykiu srautas		Sistemos reakcija į sprendimus
1. Vartotojas nurodo kolekcijos pavadinimą		
2. Vartotojas renkasi kolekcijos kategoriją		
3. Vartotojas pateikia kolekcijos aprašymą		
4. Vartotojas tvirtina kolekciją		Sistema sukuria kolekciją ir pereina į 5 žingsnį
5. Sistema baigia PA		
Po sąlyga:		Duomenų bazėje išsaugomi nauja kolekcija
Alternatyvūs scenarijai		
2a. Vartotojas neranda reikiamos kategorijos ir sukuria naują		Sistema sukuria naują kategoriją

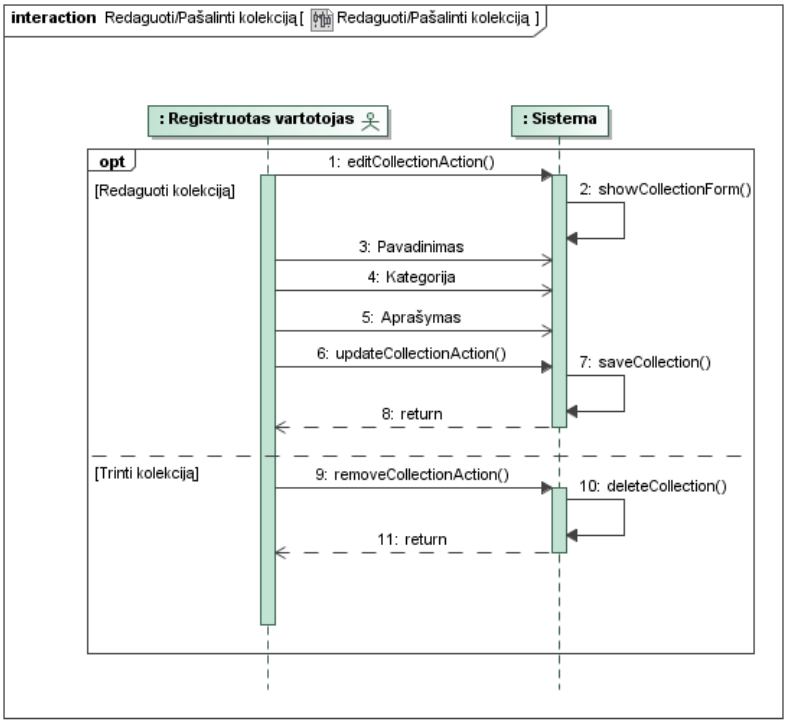


13 pav. „Kurti naują kolekciją“ PA sekų diagrama

Panaudojimo atvejo „Redaguoti/Pašalinti kolekciją“ specifikacija pateikiama 7 lentelėje, o sekų diagrama 14 paveikslėlyje.

7 lentelė. „Redaguoti/Pašalinti kolekciją“ PA specifikacija

PA „Redaguoti/Pašalinti kolekciją“		
Tikslas. Modifikuoti egzistuojančią kolekciją		
Aprašymas Šis PA yra „Tvarkyti asmenines kolekcijas“ dalis		
Prieš sąlyga		Vartotojas sekmingai prisijungęs prie sistemos, vartotojas turi bent vieną kolekciją
Aktorius		Registruotas vartotojas
Sužadinimo sąlyga		Vartotojas nori modifikuoti jau sukurtą kolekciją(iš PA „Tvarkyti asmenines kolekcijas“)
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	Tvarkyti asmenines kolekcijas
Pagrindinis įvykiu srautas		Sistemos reakcija į sprendimus
1. Vartotojas pasirenka redaguojamą kolekciją		Sistema pateikia informaciją apie kolekciją
2. Vartotojas keičia kolekcijos pavadinimą		
3. Vartotojas keičia kolekcijos aprašymą		
4. Vartotojas keičia kolekcijos kategoriją		
5. Vartotojas išsaugo pakeitimus		Sistema išsaugo pateiktus pakeitimus
6. Vartotojas baigia PA		
Po sąlyga:		Duomenų bazėje išsaugomi nauji kolekcijos duomenys
Alternatyvūs scenarijai		
1a. Vartotojas ištrina kolekciją		Sistema pašalinia kolekciją iš duomenų bazės
Pastabos		
1. Vartotojas pirmiausia atlieka žingsni „1.”, 2-4 žingsniai atliekami betkokia tvarka ir užbaigiami 5 žingsiu, vartotojas gali nutraukti darbą kada nori.		

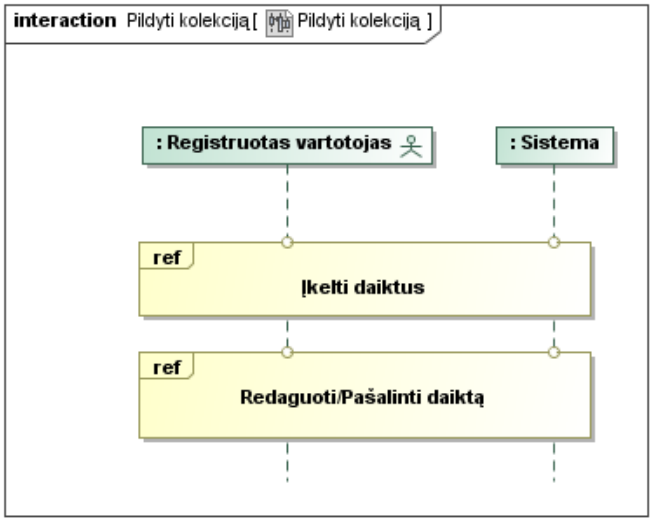


14 pav. „Redaguoti/Pašalinti kolekciją“ PA sekų diagrama

Abstraktaus panaudojimo atvejo „Pildyti kolekciją“ specifikacija pateikiama 8 lentelėje, o sekų diagrama 15 paveikslėlyje

8 lentelė. „Pildyti kolekciją“ PA specifikacija

PA „Pildyti kolekciją“		
Tikslas. Papildyti kolekciją naujais daiktais		
Aprašymas Šis PA yra „Tvarkyti asmenines kolekcijas“ dalis		
Prieš sąlyga		Vartotojas sekmingai prisijungęs prie sistemos, vartotojas turi bent vieną kolekciją
Aktorius		Registruotas vartotojas
Sužadinimo sąlyga		Vartotojas nori papildyti kolekciją daiktais(iš PA „Tvarkyti asmenines kolekcijas“)
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	Įkelti daiktus, Redaguoti/Pašalinti daiktą
	Specializuoja PA	Tvarkyti asmenines kolekcijas
Pagrindinis įvykiu srautas		Sistemos reakcija į sprendimus
1. Vartotojas nori įkelti naują daiktą		Sistema įvygdo PA „Daiktų įkelimas“ ir pereina į 3 žingsnį
2. Vartotojas nori modifikuoti įkeltus daiktus		Sistema įvygdo PA „Daiktų įkelimas“ ir pereina į 3 žingsnį
3. Sistema baigia PA		



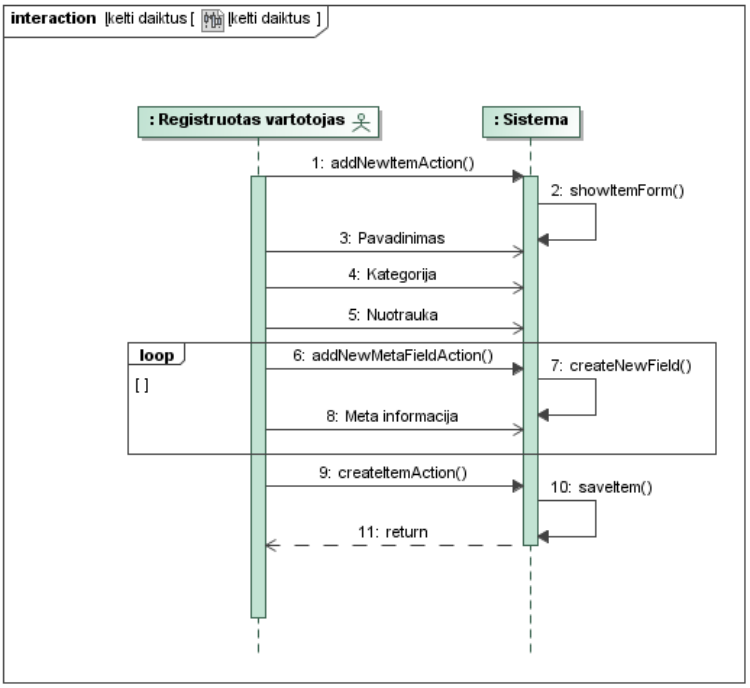
15 pav. „Pildyti kolekciją“ PA sekų diagrama

Panaudojimo atvejo „Įkelti daiktus“ specifikacija pateikiama 9 lentelėje, o sekų diagrama 16 ir 17 paveikslėliuose.

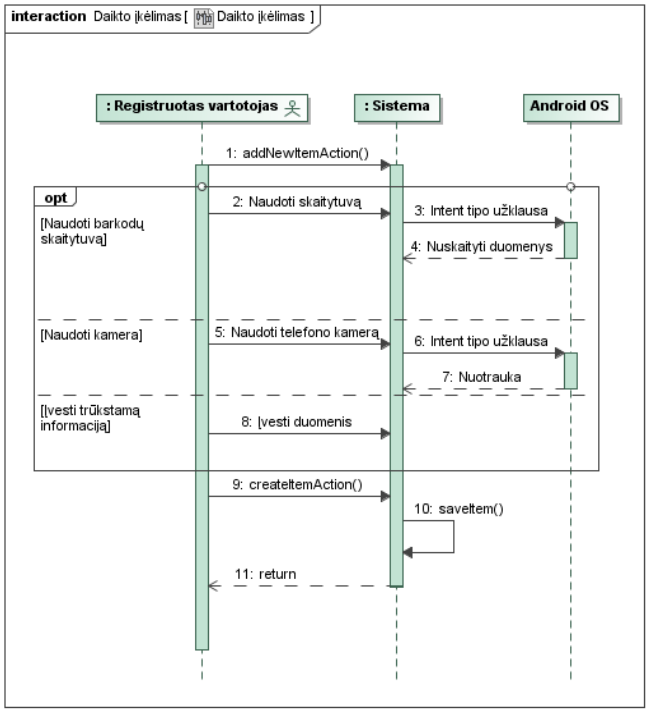
9 lentelė. „Įkelti daiktus“ PA specifikacija

PA „Įkelti daiktus“		
Tikslas. Įkelti naują daiktą į kolekciją		
Aprašymas Šis PA yra „Pildyti kolekciją“ dalis		
Prieš sąlyga		Vartotojas sekmingai prisijungęs prie sistemos, vartotojas turi bent vieną kolekciją
Aktorius		Registruotas vartotojas
Sužadinimo sąlyga		Vartotojas nori įkelti naują daiktą į kolekciją(iš PA „Pildyti kolekciją“)
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	Pildyti kolekciją
Pagrindinis įvykiu srautas		Sistemos reakcija į sprendimus
1. Vartotojas nurodo daikto pavadinimą		
2. Vartotojas nurodo daikto kategoriją		Sistema pateikia galimas kategorijas
3. Vartotojas nurodo daikto nuotrauką		
4. Vartotojas pateikia jam svarbią meta informaciją apie daiktą		

5. Vartotojas tvirtina naują daiktą	Sistema sukuria daiktą ir pereina į 6 žingsnį
6. Sistema baigia PA	
Alternatyvūs scenarijai	
1a. Jei įrenginyje veikia foto kamera ar yra barkodų skaitytuvo sistema vartotojas gali šių IS pagalba pildyti daikto duomenis.	Sistema iškviečia kitą įrenginyje esančią programą galinčia atlikti reikiamus veiksmus.
Pastabos	
1. Meta informacijos kieki ir turini nurodo pats vartotojas todėl 4 žingsnis kartojamas tiek kartų kiek vartotojas nori.	



16 pav. „Įkelti daiktus“ PA sekų diagrama

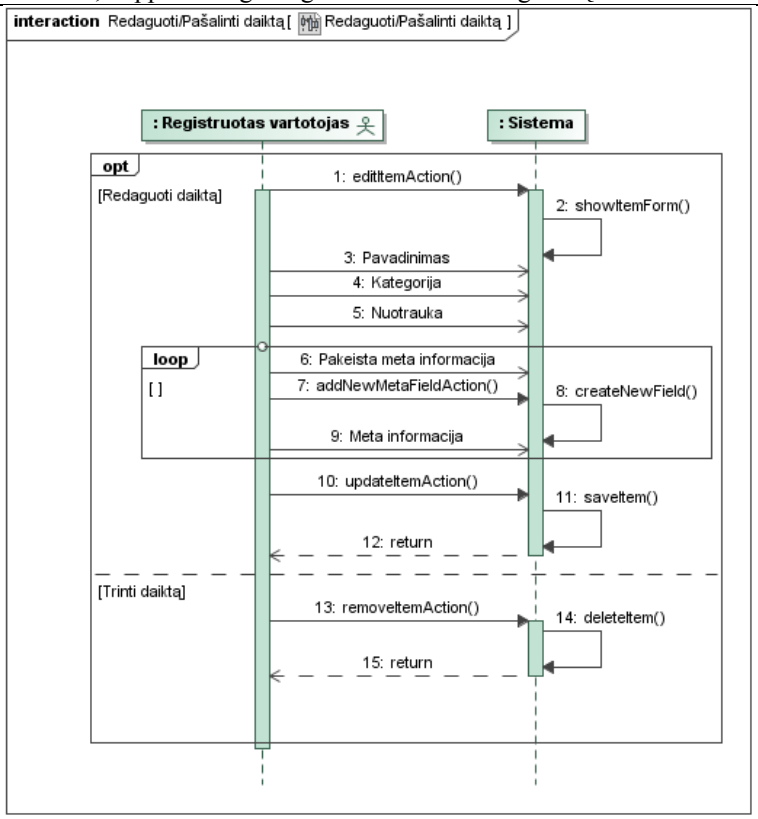


17 pav. „Įkelti daiktus“ PA sekų diagrama

Panaudojimo atvejo „Redaguoti/Pašalinti daiktą“ specifikacija pateikiama 10 lentelėje, o sekų diagrama 18 paveikslėlyje.

10 lentelė. „Redaguoti/Pašalinti daiktą“ PA specifikacija

PA „Redaguoti/Pašalinti daiktą“		
Tikslas. Modifikuoti egzistuojančius kolekcijoje daiktus		
Aprašymas Šis PA yra „Pildyti kolekciją“ dalis		
Prieš sąlyga		Vartotojas sekmingai prisijungęs prie sistemos, vartotojas turi bent vieną kolekciją, vartotojas turi bent vieną daiktą
Aktorius		Registruotas vartotojas
Sužadinimo sąlyga		Vartotojas nori kurti modifikuoti jau įkeltą daiktą(iš PA „Pildyti kolekciją“)
Susiję panaudojimo atvejai	**Išplečia PA**	
	Apima PA	
	Specializuoja PA	Pildyti kolekciją
Pagrindinis įvykiu srautas		Sistemos reakcija į sprendimus
1. Vartotojas pasirenka daiktą		Sistema pateikia informaciją apie daitką
2. Vartotojas keičia daikto pavadinimą		
3. Vartotojas keičia daikto kategoriją		
4. Vartotojas keičia keičia daikto meta informaciją		
5. Vartotoajas tvirtina naujus duomenis		Sistema išsaugo pateiktus naujus duomenis
6. Vartotojas šalina daiktą		Sistema pašalina daiktą iš duomenų bazės
7. Vartotojas baigia PA		
Po sąlyga:		Duomenų bazėje išsaugomi nauji kolekcijos daikto duomenys
Alternatyvūs scenarijai		
1b. Vartotojas pašalina daiktą		Sistema pašalina daiktą iš kolekcijos
Pastabos		
1. Vartotojas pirmiausia atlieka žingsnį „1.”, 2-4 žingsniai atliekami betkokia tvarka ir vartotojas gali nutraukti darbą kada nori, taippat 4 žingsnis gali būti atliktas daug kartų.		

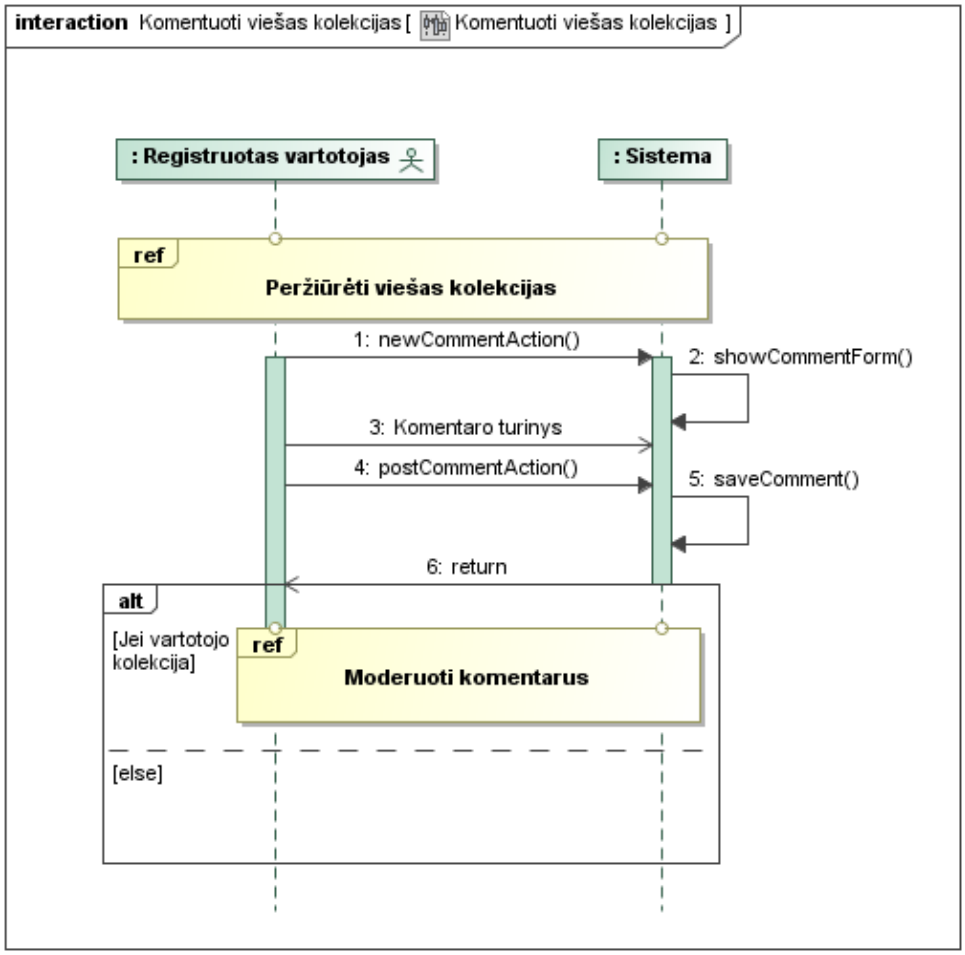


18 pav. „Redaguoti/Pašalinti daiktą“ PA sekų diagrama

Panaudojimo atvejo „Komentuoti viešas kolekcijas“ specifikacija pateikiama 11 lentelėje, o sekų diagrama 19 paveikslėlyje.

11 lentelė. „Komentuoti viešas kolekcijas“ PA specifikacija

PA „Komentuoti viešas kolekcijas“		
Tikslas. Pakomentuoti šiuo metu peržiurima kolekciją		
Aprašymas Šis PA yra „Komentuoti“ dalis		
Prieš sąlyga		Vartotojas sekmingai prisijungęs prie sistemos, sistema veikia „Online“ rėžimu.
Aktorius		Registruotas vartotojas
Sužadinimo sąlyga		Vartotojas nori komentuoti peržiūrima kolekciją
Susiję panaudojimo atvejai	**Išplečia PA**	Viešų kolekcijų peržiūra
	Apima PA	
	Specializuoja PA	Komentuoti
Pagrindinis įvykiu srautas		Sistemos reakcija į sprendimus
1. Vartotojas įveda komentaro turinį		
2. Vartotojas skelbia komentarą		Sistema išsaugo vartotojo pateikta komentarą ir pereina į 3 žingsnį
3. Sistema baigia PA		
Po sąlyga:		Komentaras pridedamas prie atitinkamos kolekcijos komentarų sąrašo
Alternatyvūs scenarijai		
Pastabos		
1. Jei kolekcija priklauso vartotojui jis gauna moderatoriaus teises joje.		

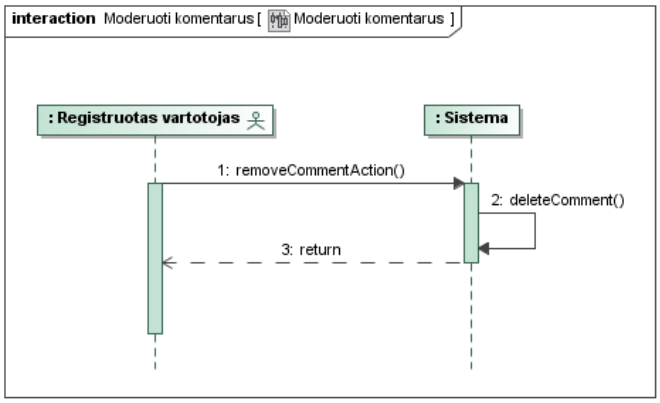


19 pav. „Komentuoti viešas kolekcijas“ sekų diagrama

Panaudojimo atvejo „Moderuoti komentarus“ specifikacija pateikiama 12 lentelėje, o sekų diagrama 20 paveikslėlyje.

12 lentelė. „Moderuoti komentarus“ PA sspecifikacija

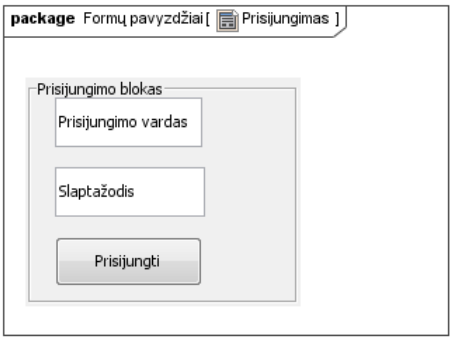
PA „Moderuoti komentarus“		
Tikslas. Vartotojas gali moderuoti savo kolekcijų komentarus		
Aprašymas Šis PA išplečia Komentuoti viešas kolekcijas PA		
Prieš sąlyga		Vartotojas sekmingai prisijungęs prie sistemos, vartotojas turi bent vieną kolekciją, sistema veikia „Online“ rėžimu.
Aktorius		Registruotas vartotojas
Sužadinimo sąlyga		Vartotojas nori ištrinti komentarą iš savo kolekcijos
Susiję panaudojimo atvejai	**Išplečia PA**	Komentuoti viešas kolekcijas
	Apima PA	
	Specializuoja PA	
Pagrindinis įvykiu srautas		Sistemos reakcija į sprendimus
1. Vartotojas spaudžia trinimo mygtuka prie norimo komentaro		Sistema perduoda pranešimą „Ar tikrai norite ištrinti šį komentarą?“
2. Vartotojas patvirtina jog nori ištrinti pasirinkta komentarą		Sistema pašalina pasirinktą komentarą iš vartotojo kolekcijos komentarų sąrašo.
3. Vartotojas baigia PA		
Alternatyvūs scenarijai		
Pastabos		



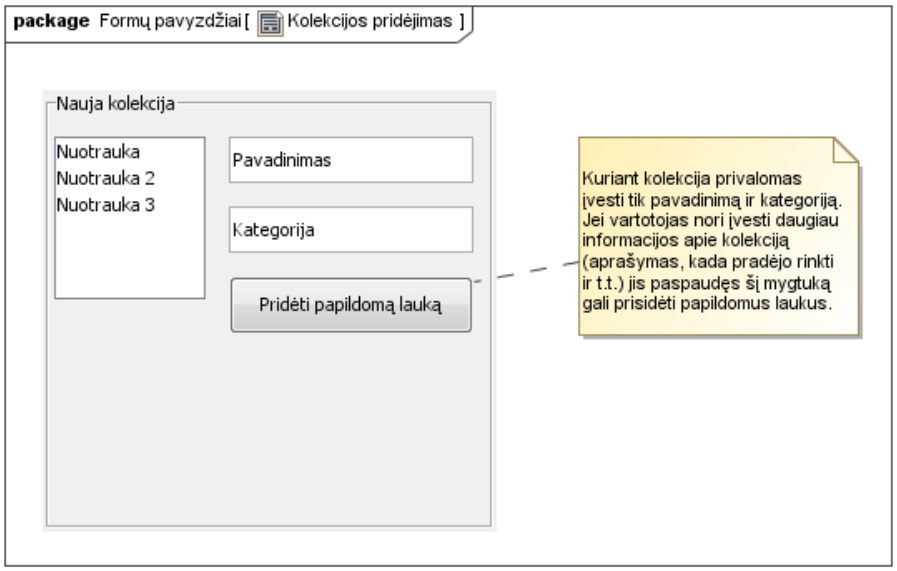
20 pav. „Moderuoti komentarus“ sekų diagrama

1.1.2. Vartotojo sąsajos prototipas

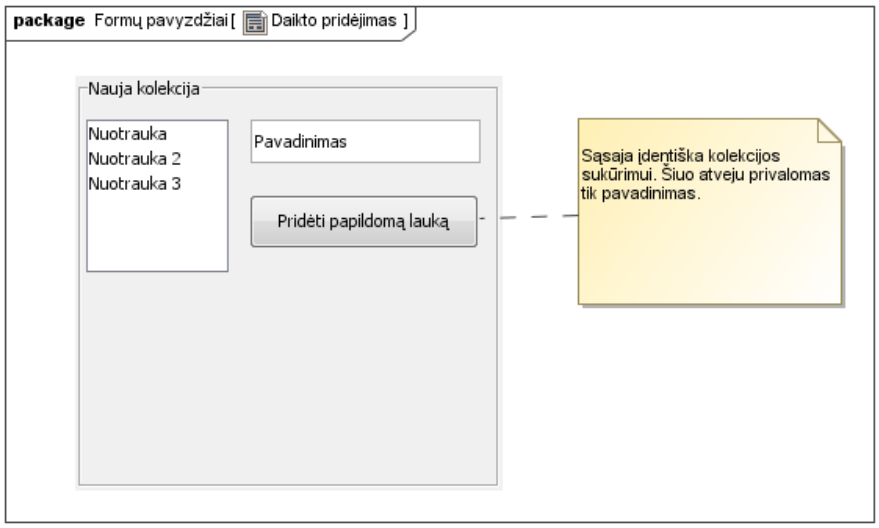
Sudaryti keli prototipiniai vartotojo sąsajos langai, jie matomi 21-25 paveikslėliuose.



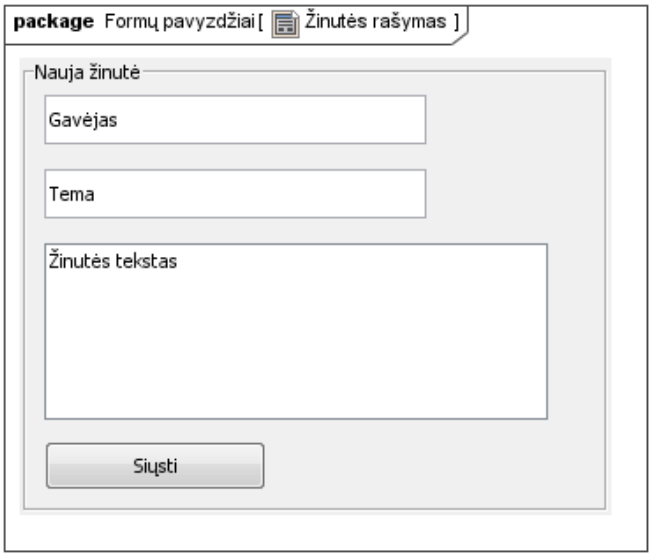
21 pav. Prisijungimo formos prototipas



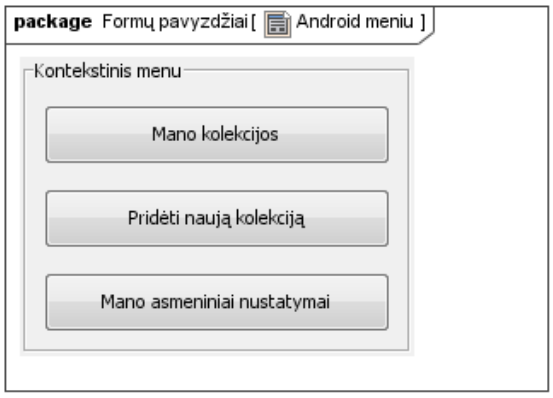
22 pav. Kolekcijos pridėjimo formos prototipas



23 pav. Daikto pridėjimo formos prototipas

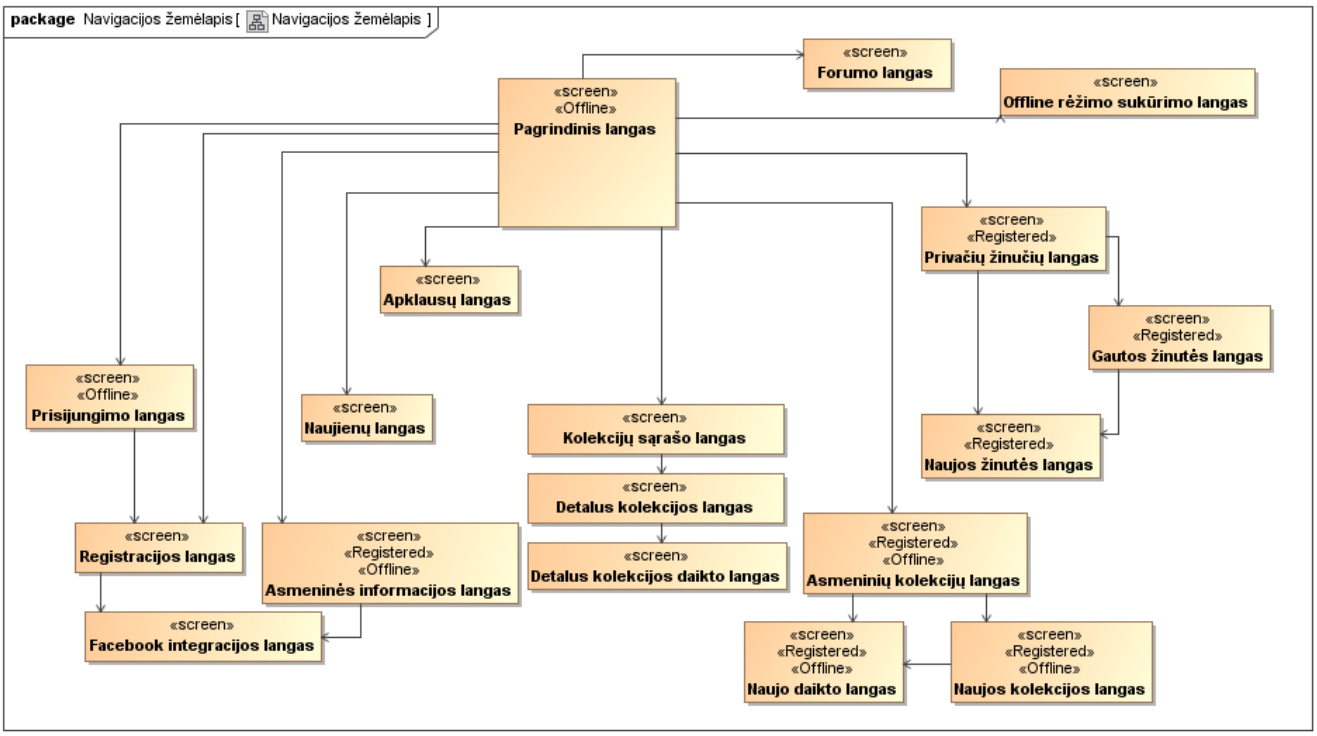


24 pav. Asmeninių žinučių rašymo formos prototipas



25 pav. „Android“ OS kontekstinio meniu prototipas

Sudarytas vartotojo sąsajos navigavimo prototipas matomas 26 paveikslėlyje. <<screen>> stereotipas nurodo, jog klasė atvaizduoja langą, kurį mato vartotojas. Kaikurie langai papildomai turi stereotipą <<registered>>, kuris nurodo jog šis langas pasiekiamas tik registruotiems vartotojams.<<Offline>> stereotipas nurodo ar langas yra pasiekiamas kai sistema veikia lokaliai. Visi vartotojai sistemoje darbą pradeda pagrindiniame lange. Vienas iš pagrindinių modeliuojamos sąsajos tikslų- minimizuoti vartotojų praleidžiamą laiką navygacijai, todėl kiekviename lange, viršuje, matomas sistemos logo per kurį gryžtama į pradinį langą. Atgalinys ryšys į pradinį langą modelyje nevaizduojamas.



26 pav. Vartotojo sąsajos navigacijos prototipas

1.1.3. Nefunkciniai reikalavimai

Pasinaudojant *„Volere“* šablono metodika sudaryti nefunkciniai reikalavimai.

Išvaizdos ir patrauklumo (Look and Feel) reikalavimai:

- Išnaudoti standartinį android stilių. Išmanieji įrenginiai pasižymi mažesne įvairovę nei asmeniniai ar nešiojamieji kompiuteriai, todėl svarbu, jog vartotojui sistema atrodytų kaip *„Android“* operacinės sistemos dalis.
- Išlaikomas originalios sistemos stilius. Vartotojui pritaikyta sistema turi sietis su egzistuojančia, ne tik tuo pačiu funkcionalumu, bet ir išvaizdos elementais.

Naudojimo (Usability):

- Minimizuoti teksto rašymą. Išmaniaisiais įrenginiais netaip patogu rašyti tekstą kaip standartinėmis klaviatūromis, todėl pritaikant sistemą reikia minimizuoti įvedamo teksto kiekį.
- Naudojami standartiniai *„Android“* operacinės sistemos sąsajos elementai. Kaip ir su stiliumi taip ir su panaudojamumu- vartotojai tikisi rasti panašumų tarp visų įrenginio sistemų.

Našumo (Perfomance):

- Duomenų sinchronizavimas norint išsaugoti jų tikslumą.Kitu atveju, kiltų problemų dėl skirtingų duomenų skirtingose sistemos rėžimuose.
- Asmeninė kolekcija saugoma telefono atmintyje, kitokiu atveju sistema neveiktų lokiu rėžimu.

Veikimo (Operational) reikalavimai:

- Išmanieji įrenginiai dažnai naudojami dinaminėje aplinkoje- kintantis apšvietimas, darbas viena ranka, socialinė įtaka ir t.t. Sistemos sąsaja turi būti tolerantiška šioms kliūtimis.
- Sąveika su kitomis telefone esančiomis sistemomis. Išmanieji įrenginiai turi didelę įvairovę sistemų, atliekančių įvairias funkcijas ir *„Android“* operacinė sistema leidžia šioms sistemomis bendrauti tarpusavyje. Išnaudojant šį funkcionalumą galima palengvinti vartotojo darbą su pritaikoma sistema.

Priežiūros ir mobilumo (Maintainability and Portability):

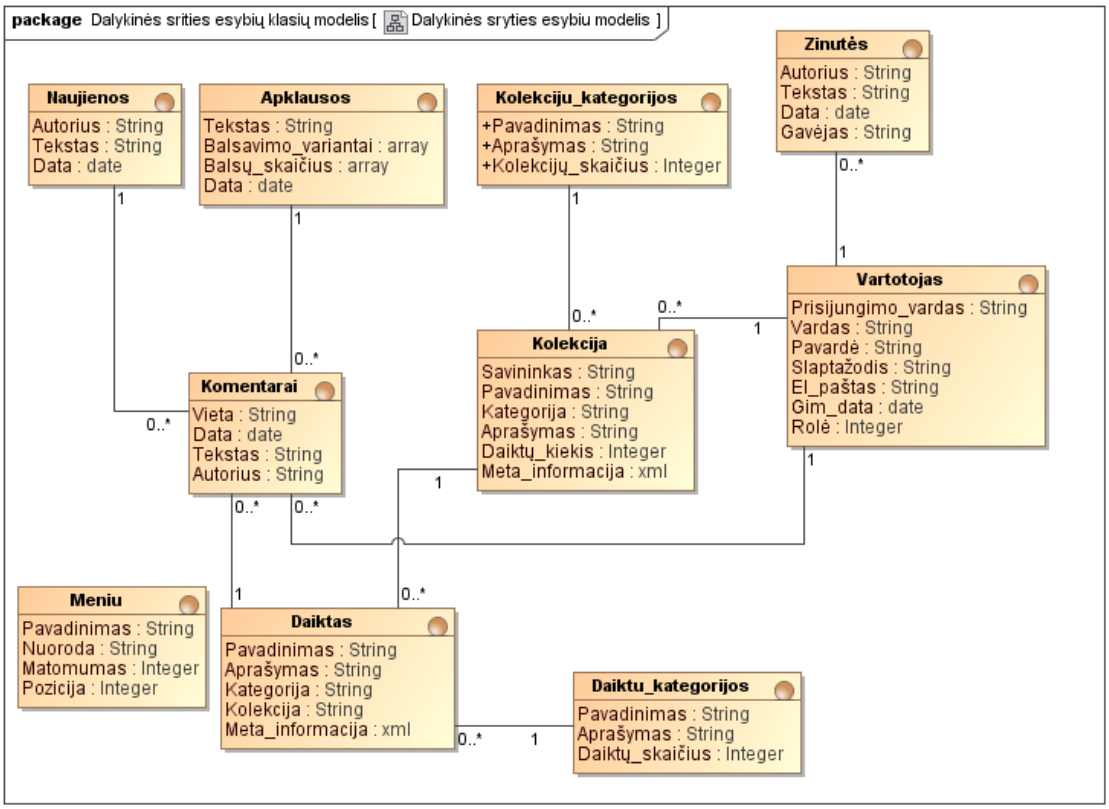
- Lygiagreti priežiūra su egzistuojančia sistema. Pritaikyta sistema neturi prasmės be egzistuojančios sistemos, todėl būtina jas abi palaikyti.

Saugumo (Security) reikalavimai:

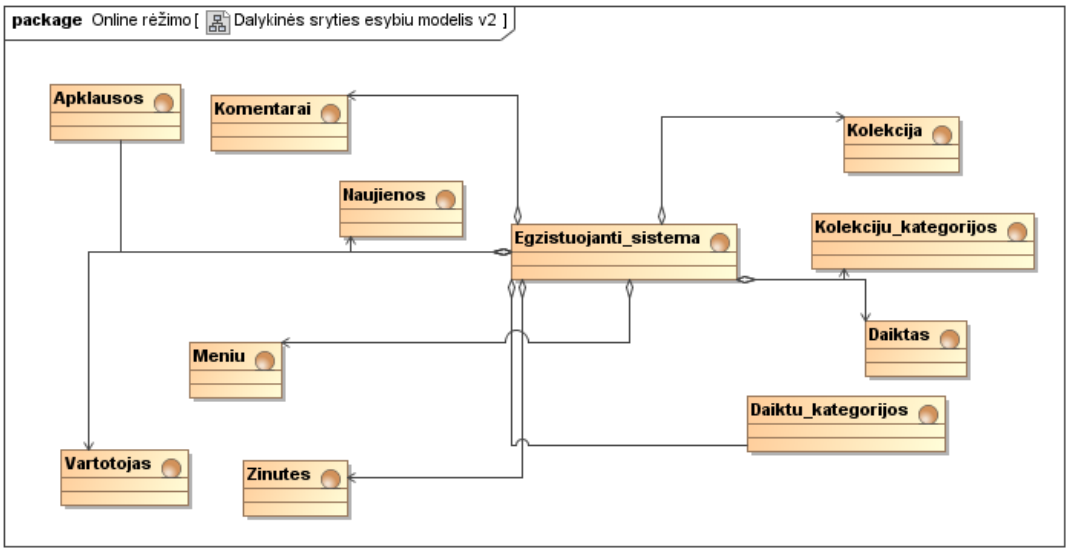
- Asmeninės informacijos konfidencialumas.
- Privatumas- nematomos asmeninės kolekcijos, jei vartotojas nenurodo jog jos viešos.

1.2. Dalykinės srities esybių modelis

Sudarytas dalykinės srities esybių modelis pateikiamas 27 paveikslėlyje. Kadangi sistema bendrauja tiesiogiai su egzistuojančia sistema, sudarytas ir antras esybių modelis matomas 28 paveikslėlyje.



27 pav. Dalykinės srities esybių klasių modelis



28 pav. „Online“ rėžimo esybių modelis

Pritaikyta sistema gauna duomenis per servisą iš egzistuojančios sistemos, kurios esybė buvo pridėta antrajame esybių modelyje. Kadangi sistema tiesiogiai nebendrauja su šiomis esybės, nėra svarbios jų charakteristikos bei savybės, svarbu tik

žinoti, jog jos visos sudaro vieną „egzistuojanti_sistema“ esybę su kuria ir bendrauja pritaikyta sistema.

1.3. Reikalavimų analizės apibendrinimas

Atlikus reikalavimų analizę buvo sudaryti egzistuojančios sistemos abstraktūs panaudojimo atvejai. Atsižvelgiant į projekto tikslus bei galimybes atsisakyta keletos panaudojio atvejų ir modelis papildytas nauju, sistemos rėžimo valdymo panaudojimo atveju. Likę panaudojimo atvejai specifikuoti lentelėmis bei sekų diagramomis.

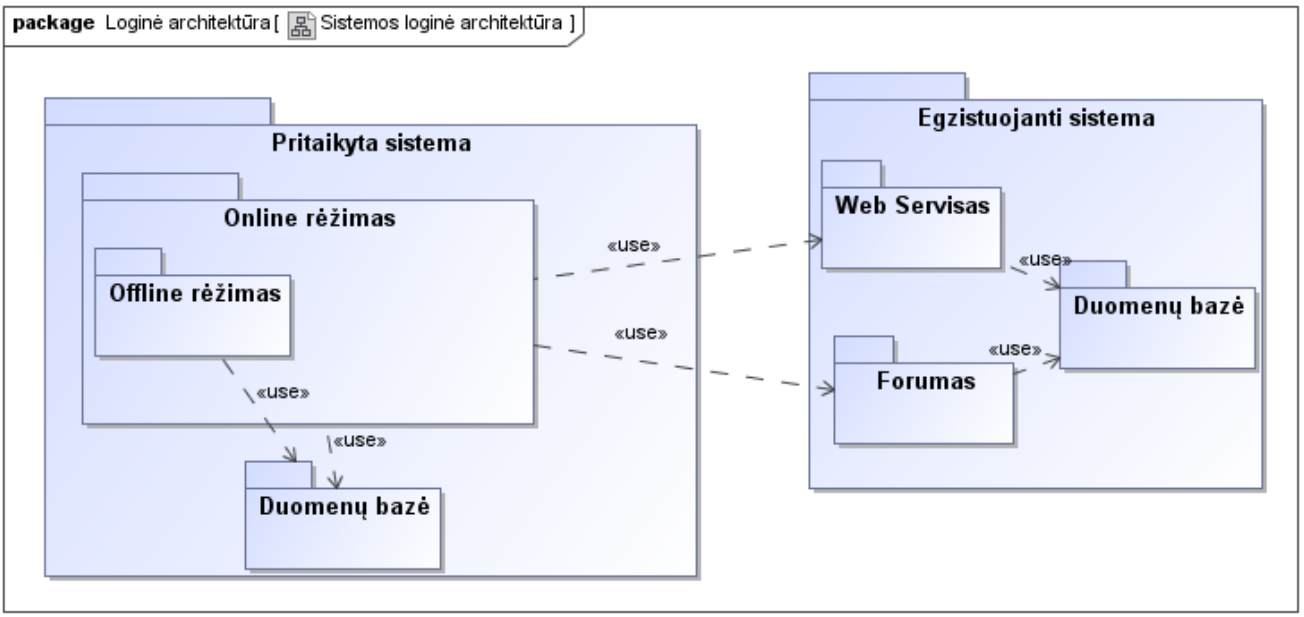
Sudarytas vartotojo sąsajos navigavimo prototipas, keletas formų prototipų ir nefunkciniai reikalavimai.

Pateikti du dalykinės srities esybių modeliai, pirmas detalus, o antrasis abstraktus, parodantis, jog egzistuojančią sistemą, kurią pritaikome išmaniesiems telefonams galime traktuoti kaip atskirą esybę, apimančia visas likusias dalykinės srities esybes.

1.4. Sistemos architektūros planas

1.4.1. Pritaikomos sistemos loginė architektūra

Suprojektuota sistemos loginė architektūra matomta 29 paveikslėlyje.



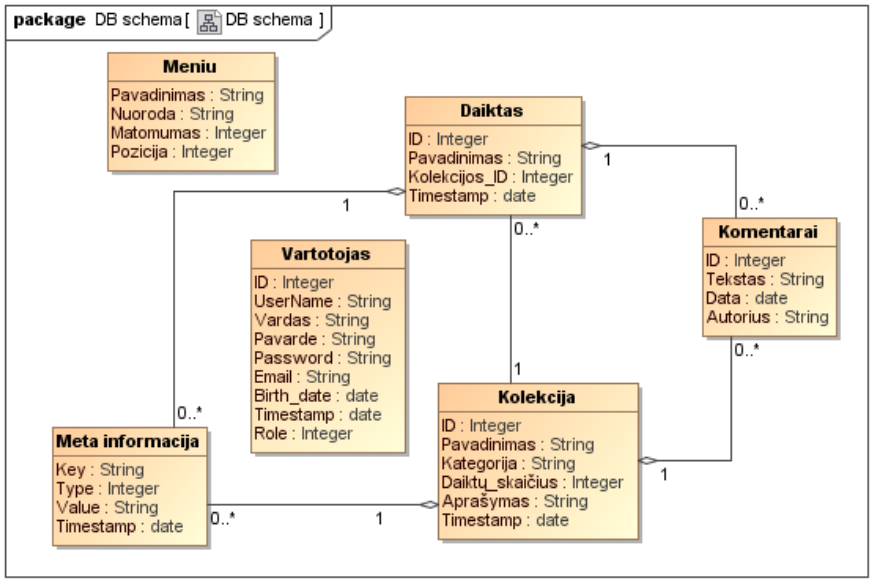
29 pav. Sistemos loginė architektūra

Pritaikyta sistema turi du posistemius, „Online rėžimas“ posistemis apima visus sistemos panaudojimo atvejus ir bendrauja su egzistuojančios sistemos tinklo servisu bei forumu esančiu egzistuojančioje sistemoje. Antras posistemis „Offline rėžimas“ yra „Online rėžimas“ posistemio dalis– jame realizuojama dalis panaudojimo atvejų, kurie veikia ir „Online rėžimo“ metu. Abu posistemiai bendrauja

su lokalia duomenų baze. „Online rėžimas“ posistemis naudoja lokalią duomenų bazę tik kai yra sukuriama „Offline“ rėžimo paskyra ir nėra interneto ryšio, o „Offline rėžimo“ posistemis visą laiką naudoja lokalią duomenų bazę.

1.5. Duomenų bazės schema

Sudaryta duomenų bazės schema matoma 30 paveikslėlyje, o jos lentelių specifikacijos pateikiamos 13–18 lentelėse. Pritaikyta sistema naudos nereliacinę *„CouchDB“* duomenų bazę, todėl nėra nubraižytas reliacinis duomenų bazės modelis. Nepavyko rasti aiškios metodologijos kaip braižyti nereliacinių duomenų bazių schemas, todėl buvo braižyta naudojantis klasių diagramos modeliu.



30 pav. Duomenų bazės schema

13 lentelė. Duomenų bazės lentelės „Vartotojas“ specifikacija.

Laukas	Tipas	Paskirtis
Id	Integer	Tai CouchDB raktinis atributas.
UserName	String	Vartotojo prisijungimo vardas
Vardas	String	Vartotojo vardas.
Pavarde	String	Vartotojo pavardė.
Email	String	Vartotojo elektroninis paštas.
Password	Md5 string	Prisijungimo slaptažodis užkoduotas md5 koduote.
Birth_date	Date	Vartotojo gimimo data.
Role	Integer	Vartotojo rolė sistemoje.
Timestamp	Date	Nurodo kada paskutinį kartą buvo atlikti keitimai įrašams.

14 lentelė. Duomenų bazės lentelės „Kolekcija“ specifikacija.

Laukas	Tipas	Paskirtis
ID	Integer	CouchDB dokumento identifikatorius.
Pavadinimas	String	Kolekcijos pavadinimas.
Kategorija	String	Kolekcijos kategorija.
Aprašymas	String	Kolekcijos aprašymas.
Timestamp	Date	Nurodo kada paskutinį kartą buvo atlikti keitimai įrašams.
Daiktų_skaičius	Int	Kolekcijai priklausančių daiktų kiekis.

15 lentelė. Duomenų bazės lentelės „Daiktas“ specifikacija.

Laukas	Tipas	Paskirtis
ID	Integer	CouchDB identifikatorius.
Pavadinimas	String	Daikto pavadinimas.
Kolekcijos_id	Integer	Nurodo kolekcijos identifikatorių kuriam priklauso daiktas
Timestamp	Date	Nurodo kada paskutinį kartą buvo atlikti keitimai įrašams.

16 lentelė. Duomenų bazės lentelės „Komentarai“ specifikacija.

Laukas	Tipas	Paskirtis
ID	Integer	Komentaro identifikatorius.
Tekstas	String	Komentaro turinys.
Data	Date	Laikas kada buvo parašytas komentaras.
Autorius	String	Komentaro autoriaus vardas ir pavardė.

17 lentelė. Duomenų bazės lentelės „Meta informacija“ specifikacija.

Laukas	Tipas	Paskirtis
Key	String	Meta informacijos lauko pavadinimas
Type	Integer	Meta informacijos lauko tipas, pvz ar tai metai, skaičius, data ar tekstinis aprašymas
Value	String	Meta informacijos lauko reikšmė
Timestamp	Date	Nurodo kada paskutinį kartą buvo atlikti keitimai įrašams.

18 lentelė. Duomenų bazės lentelės „Meniu“ specifikacija

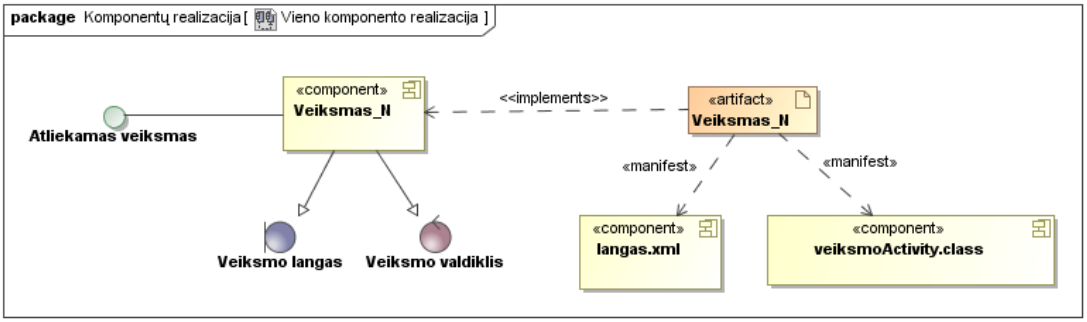
Laukas	Tipas	Paskirtis
Pavadinimas	String	Meniu lauko pavadinimas
Nuoroda	String	Nuoroda kur vartotojas bus nukreipiamas paspaudęs ant meniu punkto
Matomumas	Integer	Nurodo kurios vartotojų grupės mato šį meniu, pavyzdžiui jei ši reikšmė 1, tai meniu punktą mano tik registruoti vartotojai, o jeigu 0, tai ir registruoti ir neregistruoti vartotojai.
Pozicija	Integer	Nurodo meniu punktų eiliškumą.

Žemiau, 31 paveikslėlyje, pateikiama priklausomybių matrica parodanti, kurios esybes realiziuojamos duomenų bazės lentelėmis, o kurios lieka egzistuojančios sistemos ribose.

31 pav. Duomenų bazės priklausomybių lentelė nuo dalykines srities esybių modelio

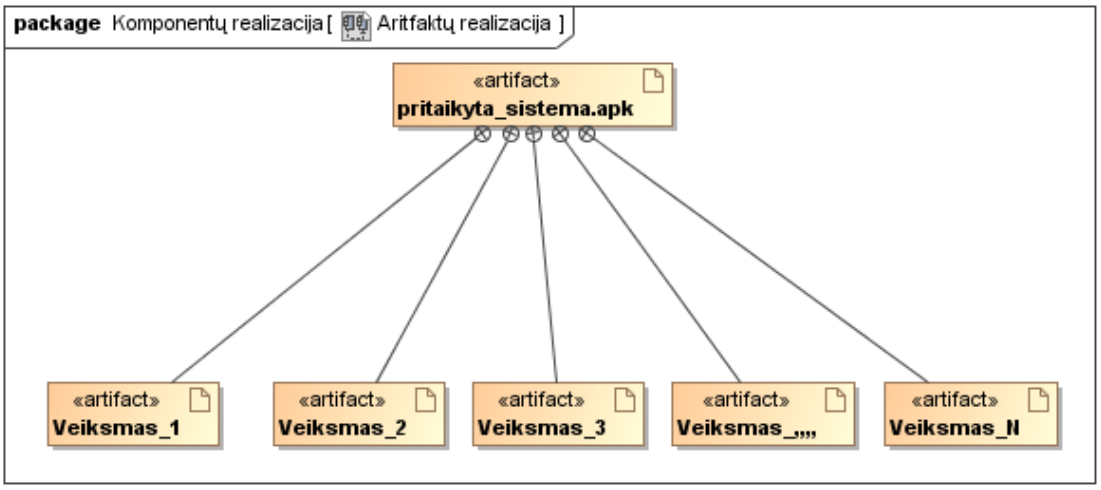
1.6. Realizacijos modelis

Atliekant visus veiksmus sistemoje yra naudojami langai ir valdikliai, jie sudaro kiekvieno veiksmo komponentą. *„Android“* operacinės sistemos aplinkoje šie komponentai realizuojami *„xml“* failais ir *„Activity“* tipo klasėmis. Pavyzdys matomas 32 paveikslėlyje.



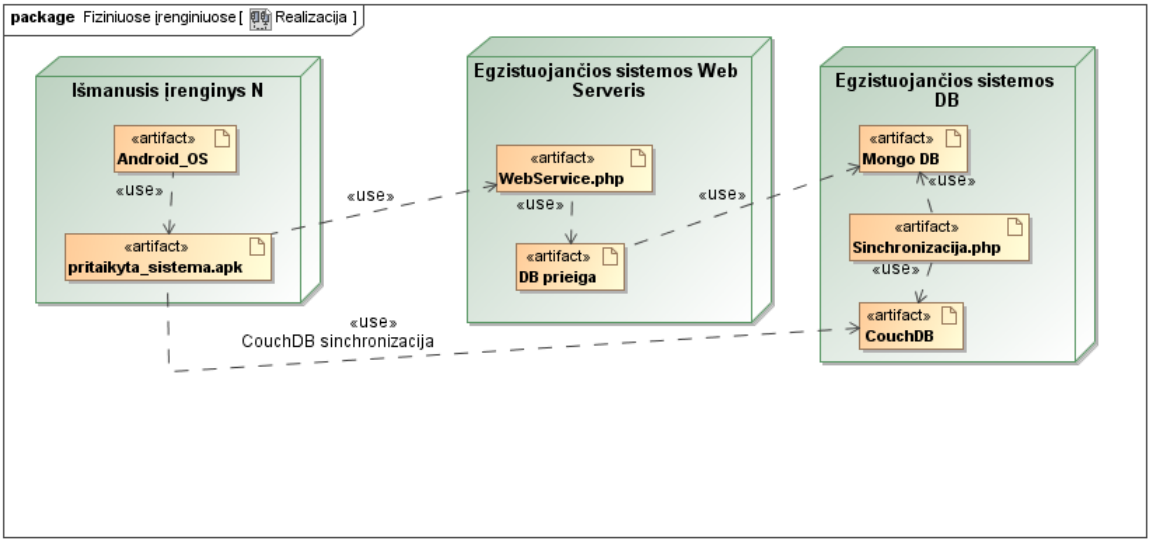
32 pav. Komponento realizacija

Kadangi visa sistema realizuojama tokiu pačiu principu nėra sudaromos realizacijos diagramos kiekvienam veiksmui. Visi šitaip realizuoti komponentai tampa *„.apk“* failo dalimi(33paveikslėlis), kurį naudoja *„Android“* operacinė sistema sistemų paleidimui.



4.33 pav. „*.apk*“ failo sudėtis

Diegimo modelis įrenginiuose pateikiamas 4.34 paveikslėlyje. Sistema gali būti įdiegta daugelyje išmaniųjų įrenginių, diagramoje pavaizduotas tik vienas, siekiant aiškumo.



4.34 pav. Diegimo modelis

Kai sistema veikia „online“ rėžimu, ji tiesiogiai bendrauja su egzistuojančios sistemos serveriu per *„WebServe.php“* klasę, o veikdama „offline“ rėžimu dirba lokaliai ir atlieka sinchronizacijas tiesiogiai su duomenų baze.

2. Priedas: Publikacija konferencijos pranešimų medžiagoje

Pateikiamo straipsnio informacija: V. Žitkus, „Karkaso "Zend Framework" pagrindu sukurtų internetinių sistemų pritaikymo išmaniesiems įrenginiams technologinio sprendimo sudarymas ir tyrimas“, Informacinė visuomenė ir universitetinės studijos (IVUS 2013), 2013, pp. 205-209.

Karkaso „Zend Framework“ pagrindu sukurtų internetinių sistemų pritaikymo išmaniesiems įrenginiams technologinio sprendimo sudarymas ir tyrimas

Voldemaras Žitkus
Kauno technologijos universitetas
Informacijos sistemų katedra
Kaunas, Lietuva
voldes@gmail.com

Santrauka. Darbe sprendžiamas esamų internetinių sistemų pritaikymo išmaniesiems įrenginiams uždavinys. Pasiūlytas sprendimas, kuris palengvina internetinių sistemų pritaikymą ir palaikymą mobiliųjų įrenginių platformose. Pateiktame sprendime siūloma automatiškai generuoti dalį kodo struktūros panaudojant egzistuojančios internetinės sistemos kodo bazę. Pasiūlyto sprendimo koncepcija parengta programinio karkaso *„Zend Framework“* ir operacinės sistemos *„Android“* pagrindu. Darbe taip pat aptariamas ir duomenų sinchronizavimo tarp esamos internetinės sistemos ir išmaniųjų įrenginių posistemio aspketas. Pateikto sprendimo prototipas sukurtas, kaip programavimo kalbos aplinkos *„Eclipse“* įskiepio *„Android“* papildinys.

Reikšminiai žodžiai: Internetinė sistema, išmanusis įrenginis, „Android“ operacinė sistema, „Zend Framework“, programinis karkasas.

Įvadas

Kasdien augantis išmaniųjų įrenginių vartotojų skaičius ženkliai plečia ir siūlomų paslaugų, skirtų šiems įrenginiams skaičių. Šiuo metu ryškėja tendencija, kad kiekvieną naują, internetinę informacinę sistemą lydi ir išmaniesiems įrenginiams skirtas sprendimas. Tokia tendencija lemia poreikį pritaikyti internetines sistemas pritaikymą išmaniųjų įrenginių. Galima būtų teigti, kad naujausių išmaniųjų įrenginių interneto naršyklės gana gerai susidoroja su esamos internetinės sistemos turinio vaizdavimu ir funkcijų palaikymu. Arba galima internetinių sistemų vartotojo sąsają pritaikyti, pasinaudojant vienu iš galimų sprendimų [1] ir taip išspręsti panaudojamumo problemas. Tačiau reikia nepamiršti, kad didžioji dauguma šiuolaikinių išmaniųjų įrenginių pasižymi įvairiomis funkcinėmis savybėmis, tokiomis kaip giroskopinis sensorius, globalios pozicijos nustatymo sistema, liečiamas ekranas, kurios suteikia šiems įrenginiams panaudojamumo pranašumą, palyginti su asmeniniais kompiuteriais. Todėl kalbant apie esamų internetinių sistemų pritaikymą išmaniesiems įrenginiams reikia įvertinti ir išnaudoti vartotojo sąsajos kūrimo bei funkcines galimybes.

Pagrindinės problemos pritaikant internetines sistemas išmaniesiems įrenginiams kyla dėl skirtingų programinių platformų ir, programinio kodo palaikymo sudėtingumo. Šiame darbe pateikiamas sprendimas, kuris padeda automatizuotai išspręsti dalį minėtų problemų. Siekiant iki galo įgyvendinti darbe siūlomą konceptualų sprendimą buvo pasirinkti keli apribojimai: pritaikoma internetinė sistema turi būti sukurta programinio karkaso *„Zend Framework“* pagrindu, o pritaikomi – tie įrenginiai, kurių pagrindas yra operacinė sistema *„Android“*. Sukurto sprendimo eksperimentinis tyrimas buvo atliktas naudojant kolekcionieriams skirtą internetinę informacinę sistemą.

Išmaniųjų įrenginių ir jų programinių plaformų išskirtinumas palyginti su asmeniniais kompiuteriais.

Lyginti išmaniuosius įrenginius su asmeniniais kompiuteriais galima dviem aspektais – pagal techninę charakteristiką ir ergonomiką.

Išmanieji įrenginiai techniškai tobulėja kiekvieną dieną, tad tampa vis patrauklesni vartotojams beto, vyksta ir pačių įrenginių inovacijų progresas. Bet vis dar galima teigti, jog vidutinis išmanusis įrenginys gerokai nusileidžia pajėgumu vidutinam stacionariam kompiuteriui. Šiuo metu asmeniniai kompiuteriai lengvai apdoroja internetinių informacinių sistemų pateikiamą turinį, o nemažai rinkoje esančių išmaniųjų įrenginių negali nepriekaištingai susidoroti su pažangių internetinių sistemų pateikiamu turiniu. Todėl esamas internetines informacines sistemas aktualu pritaikyti ir optimizuoti išmaniesiems įrenginiams, sukuriant taikomąsias programėles, ir taip siekiant išlaikyti sistemos patrauklumą bei išlikti tarp išmaniųjų įrenginių naudotojų.

Operacinę sistemą *„Android“* naudojantiems įrenginiams informacinės sistemos yra kuriamos programavimo kalba *„Java“*. Taip pat naudojamas programinis karkasas, pritaikytas kurti sistemas *„Android“*. Vienas iš pagrindinių operacinės sistemos *„Android“* išskirtinumų yra tai, jog komponentai ir įvairios sistemos savybės turi būti aprašytos faile „*androidManifest.xml“*, tai savotiškas turinys, kuriuo naudojasi operacinė sistema, valdydama sukurtą informacinę sistemą. Šis turinys užtikrina aplinkoje *„Android“* veikiančių

sistemų komunikaciją ir suteikia galimybę pasinaudoti kitų platformos *„Android“* sistemų funkcionalumu [2], [3].

Mažas išmaniųjų įrenginių ekranas, palyginti su stacionarių kompiuterių monitoriais, yra vienas didžiausių skirtumų, vertinant ergonomikos aspektu. Dėl šios priežasties būtina optimizuoti informacijos kiekį, kuris pateikiamas išmaniajame įrenginyje – pateikti tik būtiniausią informaciją.

Taip pat svarbus aspektas yra naudojimo aplinka: išmaniaisiais įrenginiais vartotojai dažnai naudojasi eidami gatve, važiuodami viešuoju transportu ar kitoje, tokiai veiklai neįprastoje, aplinkoje. Atsižvelgiant į šiuos aspektus, sistema turi būti nesudėtinga, nereikalauti iš vartotojo didelio navigacinio tikslumo ir patogi naudotis viena ranka. Žinoma, kai sistema skirta naudoti namų aplinkoje (ar kitoje ramesnėje aplinkoje), anksčiau paminėti aspektai gali pasirodyti ne tokie aktualūs, bet tai – tik prielaidos, kurias reikia vertinti atsargiai.

Vienas iš pagrindinių išmaniųjų įrenginių privalumų yra jutiklinis ekranas, tačiau kartu tai – ir trūkumas, apsunkinantis vartotojo sąsajos kūrimą. Kadangi dauguma veiksmų vartotojas atlieka liesdamas ekraną, būtina sukurti vartotojo sąsają, pritaikytą būtent šiam techniniam sprendimui [4]. Internete yra daug oficialios medžiagos [5], kurioje pateikiama aibė rekomendacijų, kaip panaudoti įvairius sąsajos elementus (mygtukų dydis, tarpai tarp nuorodų ir t. t.) ir derinti su sistemos funkcionalumu.

Dėl šių ypatumų, nėra teisinga tiesiogiai perkelti egzistuojančios internetinės informacinės sistemos vartotojo sąsają į išmaniajam įrenginiui skirtą sprendimą. Galima perkelti dalį sąsajos elementų, o po to atlikti papildomas korekcijas, kad būtų priimtas optimalus sprendimas išmaniųjų įrenginių atžvilgiu.

Esamų internetių sistemų ir išmaniųjų įrenginių sąveika

Priklausomai nuo sistemos tipo bei paskirties, gali iškilti sąveikos tarp esamos internetinės informacinės sistemos ir išmaniųjų įrenginių platformoms pritaikytų sistemos versijų poreikis. Galima išskirti keletą pagrindinių sąveikų tipų.

Duomenys labiau skaitomi, negu rašomi ar atnaujinami

Naujajai platformai pritaikyta sistema daugiausiai naudojama informacijos skaitymui ir kito originaliosios sistemos turinio peržiūrai. Ši sąveika yra pati paprasčiausia, nes į pritaikytą išmaniojo įrenginio sistemą pakanka perduoti duomenis iš internetinės sistemos, naudojant *„web“* paslaugas ar kitas panašias technologijas.

Dažnai naudojami tam tikros srities duomenys

Priklausomai nuo sistemos paskirties, poreikis naudoti dalį duomenų išmaniojo įrenginio aplinkoje gali būti daug dažnesnis, pavyzdžiui, naudoti tam tikrą vartotojo registracijos informaciją. Siekiant paspartinti sistemų sąveikos našumą gali būti tikslinga, priklausomai nuo sistemos specifikos, dalį tokių dažnai naudojamų duomenų saugoti išmaniajame įrenginyje.

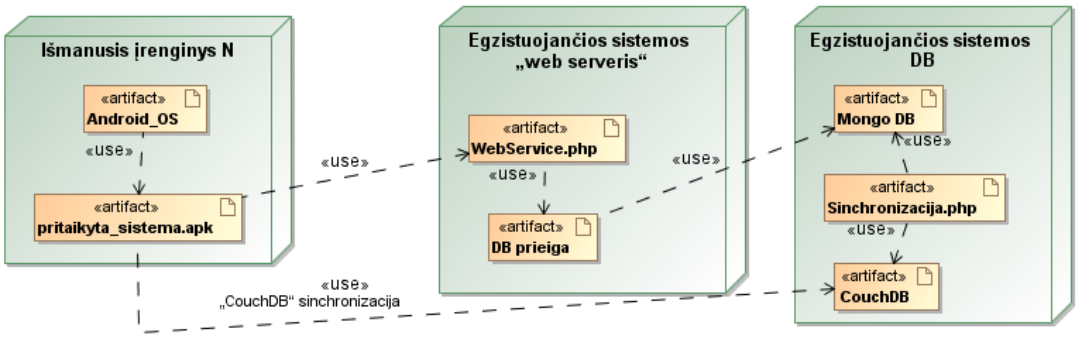
Tokius duomenys galima saugoti arba lokalioje duomenų bazėje, arba laikinojoje atmintyje, priklausomai nuo jų apimties ir panaudojimo dažnumo.

Duomenys naudojami asmeniniams tikslams, ir vartotojai yra pagrindiniai sistemos turinio šaltiniai

Šiuo atveju didžioji dalis informacinių sistemų turinio yra generuojama pačių vartotojų. Vartotojams gali būti aktualu pasiekti savo duomenis, kai išmaniajam įrenginiui nepasiekiamas interneto ryšys arba dėl tam tikrų priežasčių originaliosios sistemos serveriai nėra pasiekiami. Tokiu atveju reikalinga, jog pritaikyta sistema galėtų veikti nepriklausomai nuo originaliosios internetinės informacinės sistemos. Veikimui užtikrinti būtina, jog lokali sistema turėtų savo duomenų bazę su reikiamų duomenų kopijomis. Siekiant išvengti duomenų neatitikimų tarp internetinės ir išmaniojo įrenginio sistemos, reikalinga atlikti duomenų sinchronizaciją, kai tik internetinės sistemos duomenų bazių valdymo sistemos (DBVS) serveriai tampa pasiekiami.

Duomenims sinchronizuoti yra siūloma įvairių sprendimų [6], [7], [8], tačiau dauguma jų veikia korektiškai arba tik su identiškomis, arba to paties tipo DBVS (pavyzdžiui, sunku sinchronizuoti *„SQL“* ir *„NoSQL“* duomenų bazes). Taip pat reiktų įvertinti, kad išmaniųjų įrenginių platformos ne visas duomenų bazes, todėl priklausomai nuo sistemos specifikos gali reikėti atlikti skirtingų DBVS sinchronizaciją. Detaliau duomenų sinchronizavimo uždavinio sprendimas šiame darbe nenagrinėjamas.

Šiame tyrime buvo sinchronizuojami skirtingų duomenų bazių valdymo sistemų duomenys. Naudotas skirtingų DBVS duomenų sinchronizacijos architektūros modelis pateiktas 1 pav.



1 pav. Sistemų, turinčių skirtingas duomenų bazes, sąveika

Darbe eksperimentiniams tyrimams buvo naudojama kolekcinionierių internetinė sistema, sukurta panaudojant programinį karkasą *„Zend Framework“* ir duomenų bazių valdymo sistemą *„Mongo DB“*. Sistema buvo pritaikytą platformai *„Android“* naudojant duomenų bazę *„CouchDB“*. Ši duomenų bazė turi gerai parengtą atskirose platformose esančių sistemų sinchronizavimo mechanizmą. Kadangi abi duomenų bazės yra *„NoSQL“* tipo, jas galima sinchronizuoti atsižvelgiant į jų struktūrą.

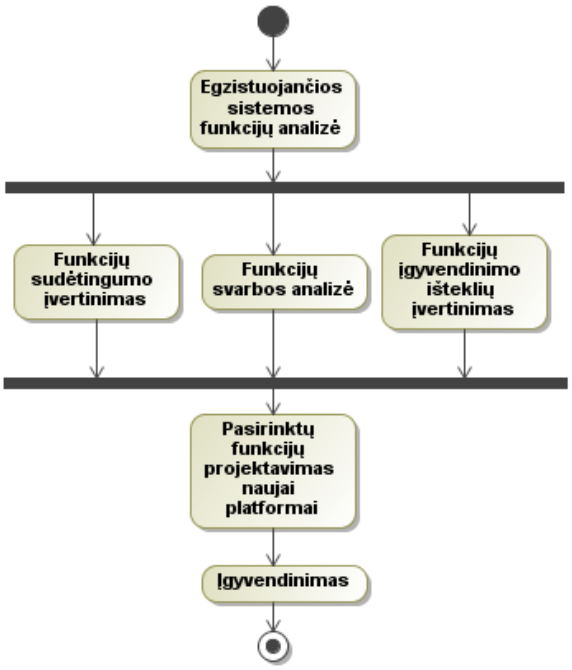
Bendro sprendimo, tinkančio visais atvejais nėra, tačiau pateiktą sinchronizavimo modulio architektūros principą su smulkiais pakeitimais galima pritaikyti įvairiems DBVS deriniams.

„Zend Framework“ pagrindu sukurtos internetinės IS pritaikymo išmaniesiems įrenginiams su os „Android“ koncepcija

Kuriant koncepciją buvo įvertinta ir jau esamų sprendimų patirtis bei logika [9], kurie buvo taikomi ir senesnės kartos mobiliesiems sprendimams [10]. Perkeliant sistemą į naują platformą būtina įvertinti, kurias funkcijas tikslinga pritaikyti naujajai sistemai, taip pat atsižvelgti į naujos programų kūrimo aplinkos programinio projekto struktūrą.

Reikiamo funkcionalumo atrinkimas

Pritaikant esamą sistemą išmaniesiems įrenginiams, reikėtų pirmiausia pasinaudoti sistemos turimomis reikalavimų ir projektinėmis specifikacijomis. Pagal šias specifikacijas paprasta nustatyti, kuris funkcionalumas turi būti perkeliamas į išmaniajam įrenginiui kuriamą taikomąją programą. 2 pav. pateikiamas koncepcinis sistemos reinžinerijos procesas, kuriame detalizuojamas perkeliamų į naują platformą funkcijų parinkimas.



2 pav. Perkeliamo funkcionalumo atrinkimas

Pirmiausia analizuojamas egzistuojančios sistemos funkcionalumas. Atsižvelgiama į tris pagrindinius aspektus:

- Reikia įvertinti kiekvieną sistemos funkcinį panaudojimo atvejį, ar šis funkcionalumas reikalingas išmaniajame įrenginyje. Nebūtino funkcionalumo pavyzdys gali būti sistemos administratoriaus funkcijos.
- Svarbu atsižvelgti į turimus, sistemai kurti skirtus, išteklius. Reikia atskirti funkcionalumą, kuris nėra būtinas naujajai sistemai, kuram įgyvendinti gali būti panaudota daug sistemai kurti skirtų išteklių.
- Taip pat svarbu nustatyti, ar pritaikomas labai sudėtingas funkcionalumas. Šis klausimas vertinamas tiek techniškai, tiek iš vartotojo pozicijos. Realizuoti techniškai sudėtingą funkcionalumą gali užtrukti daugiau laiko, negu planuojama, o įgyvendintas sudėtingas funkcionalumas vartotojui gali pasirodyti per daug sudėtingas ir nepatogus, todėl vartotojas jo nenaudos.

Remiantis šiais vertinimais sudaromas funkcijų prioritetinis sąrašas. Iš sąrašo empiriškai pašalinamos jo gale esančios funkcijos.

Po šios reikalavimų analizės turime pradinį sąrašą funkcijų reikalingų naujojoje sistemoje.

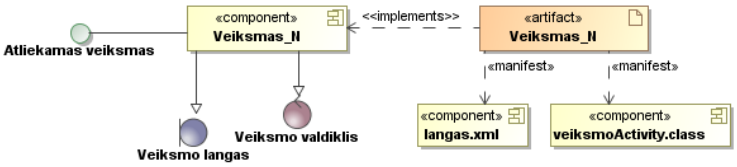
Programinio kodo struktūros perkėlimas

Žinant perkeliamų funkcijų aibę, galima atlikti programinio kodo analizę. Reikia pažymėti, jog kai kurie programinio karkaso *„Android“* komponentai turi atitikmenis kituose *„Model-view-controller“ (MVC)* tipo karkasuose. Komponentai *„Activity“* veikia panašiai kaip valdikliai, pakartotinio panaudojimo komponentai veikia kaip komponentai *„Fragments“*, o vaizdai aprašomi *„xml“* failuose. Šių komponentų įdiegimo klasių modelis matyti 3 pav.

Atsižvelgiant į šiuos panašumus galima automatizuotai sukurti išmaniojo įrenginio taikomosios programos išeities programinio kodo projektą, kurį sudarytų dalis reikalingų failų, kuriuose saugomos programinės klasės ir kiti programinio kodo epizodai. Turint jau dalinę programinio projekto struktūrą lengviau toliau kurti sistemą.

Išlaikant panašią programinę struktūrą tarp skirtingų sistemos versijų lengviau jas palaikyti.

Kadangi kiekvienas programavimo karkasas turi gana aiškiai apibrėžtą sistemos struktūrą, kuri parodo, kur ir kokio tipo komponentai saugomi, šį procesą galima iš dalies automatizuoti ir automatiškai sukurti reikiamus komponentus bei užpildyti jais failą *„androidManifest.xml“*.



3 pav. Komponentų įdiegimo klasių modelis

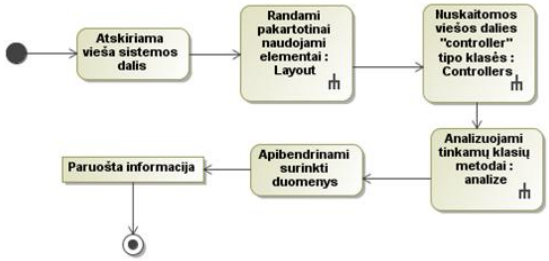
Detaliam koncepcijos įgyvendinimui buvo pasirinktas programinio karkaso *„Zend Framework“* pagrindu sukurtų internetinių informacinių sistemų pritaikymo platformai *„Android“* uždavinys. Analizuojant *„Zend Framework“* pagrindu kuriamų ir platformai *„Android“* skirtų informacinių sistemų programinio kodo projektų struktūrą buvo išskirti šie struktūrų sąsajų principai:

- *MVC* struktūra leidžia susieti kiekvieną veiksmą (puslapis bei jo logika) su jam priklausančia *„view“* klase (vaizdavimas). Tiesiogiai perkeliant šį ryšį į programinį karkasą *„Android“* galima iš karto sugeneruoti *„activity“* klases ir jų *„xml“* failus, kurie turi tą pačią paskirtį – puslapio logika ir jo vaizdavimas.
- Formoms kurti paprastai naudojama *„Zend_Form“* klasė, struktūrizuotai aprašanti formos elementus (įvedimo laukai, jų pavadinimai, mygtukai). Nustačius, jog atvaizdavimo klasėje (*„view“*) naudojama *„Zend_Form“* klasė, galima surinkti informaciją apie formos elementus ir sukurti atitinkamus elementus aplinkoje *„Android“*.
- Programinis karkasas *„Android“* kaip numatyta neturi *MVC „model“* klasių atitikmens, bet šiuo atveju šiame programiniame karkase nedraudžiama sukurti *„model“* tipo klasių ir jas naudoti.
- Internetinėse informacinėse sistemose logika dažnai atskyriama į viešą dalį (tai, ką mato eilinis vartotojas) ir administracinę dalį. Pritaikant sistemą išmaniesiems įrenginiams netikslinga perkelti administracinę dalį, todėl ši sistemos dalis dažniausiai turėtų būti neliečiama.
- Analizuojant *„Zend Framework“* pagrindu sukurtas sistemas, galima identifikuoti dažnai naudojamas klases (pavyzdžiui, šoninį meniu, kuris pateikiamas visuose puslapiuose). Atpažinimo požymis gali būti jog šios klasės yra įterpiamos (*„include“*) kitose klasėse. Jas atpažinus karkase *„Android“* galima sukurti *„fragment“* tipo klases, kurios taip pat skirtos naudoti pakartotinai.
- Sugeneruotų klasių informaciją galima automatiškai surašyti į failą *„androidManifest.xml“*.

Esamos internetinės informacinės sistemos duomenų surinkimo ir perkėlimo veiklos modelis, kuris grindžiamas anksčiau aprašytais principais, pateikiamas 4 pav.

Pirmiausia atskiriama vieša sistemos dalis nuo administracinės ir kitų sistemos dalių. Kadangi karkasas *„Zend Framework“*, kaip numatyta, neatskiria šių sistemos dalių, reikia papildomos informacijos norint nustatyti, kurias klases reikia perkelti, o kurių – ne. Šiuo atveju galima pasinaudoti egzistuojančios sistemos projektine specifikacija.

Pakartotinai naudojamus elementus galima rasti *„layout“* klasėse ieškant komponentų, kurie yra į juos įtraukiami.



4 pav. Esamos internetinės informacinės sistemos duomenų surinkimo ir perkėlimo veiklos modelis

Sistemos veiklos logiką aprašo *„controller“* klasės. Čia vartojama sąvoka „tinkama klasė“ – tai tokia klasė, kuri yra skirta sistemos viešajai daliai ir turi bent vieną viešą (*„public“*) metodą (*„controller“* klasė, neturinti nė vieno viešo metodo, neturi prasmės, bandant į ją kreiptis bus grąžinama klaida). Neprasminga perkelti tokias klases reinžinerijos metu.

Kitame žingsnyje analizuojamos surinktos *„controller“* klasės. Pirmiausia cikliniu būdu analizuojamas surinktų, perkelti tinkamų klasių rinkinys, kol pasiekiamas sąrašo galas. Po atliktos analizės gaunamas duomenų rinkinys pagal kurį bus kuriama sistemos struktūra aplinkoje *„Android“*.

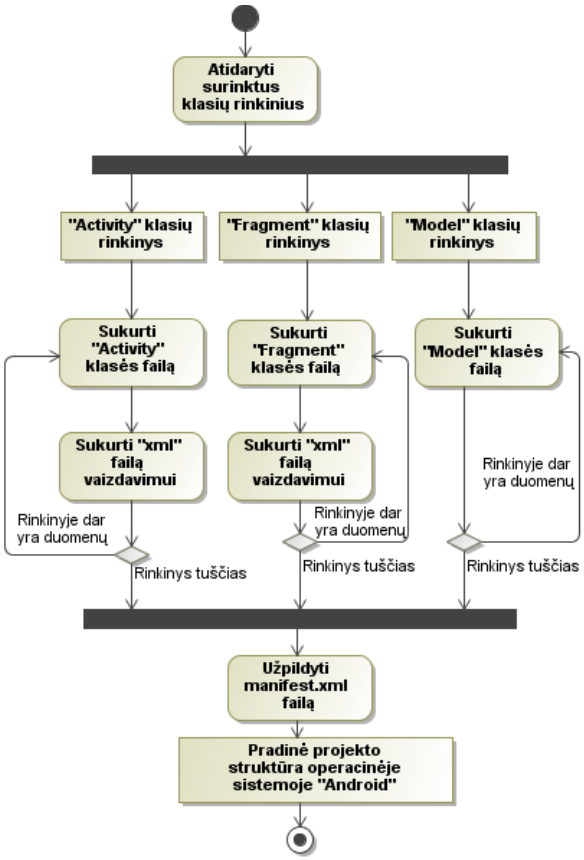
Ciklo metu yra analizuojami visi vieši *„controller“* klasės metodai – kiekvienas iš jų atitinka tam tikrą sistemos puslapį, kuris turi vaizdavimo būdą ir savo veiklos logiką.

Kiekvienas iš šių metodų gali kreiptis į modelio klasę (pavyzdžiui, reikia gauti duomenų iš duomenų klasės). Radus kreipinį, modelio klasės informacija yra įrašoma į *„models“* klasių rinkinį.

Kiekvienam *„controller“* klasės viešam metodui būtina turėti *„view“* klasę (kitokiu atveju bandant kreiptis į metodą bus grąžinama klaida), jeigu jame panaudota *„Zend_Form“* klasė (pavyzdžiui, kontaktų forma), tada galima nuskaityti elementus, esančius formoje, ir juos sukurti aplinkoje *„Android“*.

Atlikus šiuos veiksmus gaunamas duomenų rinkinys naudojamas paskutiniame šio reinžinerijos proceso žingsnyje.

Surinkus visą reikalingą informaciją iš egzistuojančios sistemos struktūros yra sukuriama nauja struktūra aplinkoje *„Android“*. Struktūros kūrimo modelis pateikiamas 5 pav. Lygiagrečiai kiekvienam duomenų rinkiniui yra kuriamos atitinkamos klasės ir *„xml“* failai aplinkoje *„Android“*.



5 pav. Naujos struktūros kūrimo veiklos modelis

Sukūrus visus failus, jų informacija surašoma į failą *„androidManifest.xml“*, kuriame saugoma pagrindinė informacija apie sistemos struktūrą.

Atlikus visus šiuos veiksmus gaunamas pradinis sistemos programinio kodo projektas aplinkoje *„Android“*, kurį toliau programuotojas papildo specifine veiklos logika, ir drauge suteikia išmaniesiems įrenginiams būdingų funkcinių elementų ir sprendimų.

Koncepcijos įgyvendinimas Aplinkoje „Eclipse“

Anksčiau aprašytai koncepcijai įgyvendinti sukurtas programos prototipas, kuris veikia kartu su programavimo aplinkos *„Eclipse“* įskiepiu, skirtu kurti aplinkos *„Android“* taikomosioms programoms. Pirmiausia naudojant įrankį *„Eclipse“* yra sukuriamas naujas *Android* projektas su numatyta failų struktūra ir konfigūracija, o antrojo žingsnio metu paleidžiama sukurta programa nurodant pritaikomos internetinės sistemos programinio kodo adresą ir tikslinį *Android* projektą, kuris užpildomas perkeliama informacija iš pasirinktos sistemos. Galiausiai naujajam projektui sugeneruojamos pagrindinės programinės klasės, dalis vartotojo sąsajos elementų ir užpildomas failas *„androidManifest.xml“*. Pastebėta, jog sukurtas ir eksperimentiškai išmėgintas kodo generavimas, paspartina esamų internetinių sistemų pritaikymo išmaniesiems įrenginiams procesą.

Išvados

Buvo atlikta išmaniųjų įrenginių funkcinių galimybių analizė, kuri parodė, kad identiškas funkcijų perkėlimas iš įprastinės internetinės sistemos į išmaniajam įrenginiui skirtą aplinką gali neišnaudoti išmaniesiems įrenginiams būdingų vartotojo sąsajos funkcinių elementų bei perkrauti pačią sistemą nereikalingomis funkcijomis.

Darbe buvo aptarti internetinės sistemos ir jos pagrindu sukurtos išmaniojo įrenginio informacinės sistemos sąveikos modeliai bei pateiktas sistemos architektūros modelis, kuris padeda spręsti duomenų sinchronizavimo problemas.

Darbe buvo pateiktas karkaso *„Zend Framework“* pagrindu sukurtos internetinės informacinės sistemos pritaikymo išmaniajam įrenginiui koncepcinis sprendimas, kuris padaro programuotojo darbą našesnį ir atliekamo sistemos pritaikymo proceso rezultatą kokybiškesni.

Pateikto koncepcinio sprendimo pagrindu buvo sukurtas programinis prototipas, kuris išbandytas su kolekcionierių internetine informacine sistema. Eksperimentinis tyrimas parodė, kad adaptavimo procesas gali būti automatizuotas ir gaunamas rezultatas yra korektiškas.

Tolimesniuose tyrimuose šį koncepcinį sprendimą norima integruoti į projektavimo įrankį *„MagicDraw“* siekiant automatizuoti perkeliamų funkcijų pasirinkimo ir perėjimo prie programinio kodo generavimo žingsnį.

Taip pat atsižvelgiant į įvairių *„PHP“* programavimo karkasų panašumus bus siekiama išplėsti sprendimą, kad jis apimtų daugiau negu vieną (*„Zend Framework“*) programinį karkasą.

Literatūra

[1] M. Hinz, Z. Fiala, F. Wehner, „Personalization-Based Optimization of Web Interfaces for Mobile Devices“, Mobile Human-Computer Interaction- MobileHCI 2004, pp. 204-215.

[2] J. Stark, Building Apps with HTML, CSS, and JavaScript. 1-oji laida. JAV, 2010.

[3] L. Vogel, Android Development Tutorial, 2011,[interaktyvus]. Prieiga per internetą: <http://www.vogella.de/articles/Android/article.html>.

[4] M. Song, H. Song, X. Fu, Methodology of User Interfaces Design Based on Android, Multimedia Technology (ICMT), 2011 International Conference, 2011 (July), pp. 408-411.

[5] Android API Guides [interaktyvus]. Prieiga per internetą: <http://developer.android.com/develop/index.html>.

[6] M.-Y. Choi, E.-A. Cho, D.-H. Park, C.-J. Moon, D.-K. Baik, A Database Synchronization Algorithm for Mobile Devices, Knowledge Creation Diffusion Utilization, 2010, vol. 56, pp. 392-398.

[7] S. Gansemer, U. Gröner, M. Maus, „Database Classification of Mobile Devices“, IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems Technology and Applications, 2007 (September), pp. 699-703.

[8] I. McDowall, Programming PC Connectivity Applications for Symbian OS. 1-oji laida. JAV, 2005.

[9] W. Höpken, M. Fuchs, M. Zanker, T. Beer, “Context-Based Adaptation of Mobile Applications in Tourism”, Information Technology & Tourism, 2010, vol. 12, no. 2, pp. 175-195.

[10] P. Cotter, B. Smyth, Content Personalisation for WAP-Enabled Devices, Adaptive Hypermedia and Adaptive Web-Based Systems, Lecture Notes in Computer Science, 2000, vol. 1892, pp. 98-108.