KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMATIKOS STUDIJŲ PROGRAMA

AUDRIUS MIČIULIS

PROGRAMINĖS ĮRANGOS KŪRIMO PRIEMONIŲ MOBILIOSIOMS PLATFORMOMS TYRIMAS

Magistro darbas

Darbo vadovas
lekt. dr. K. Jankauskas

KAUNAS, 2013

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMATIKOS STUDIJŲ PROGRAMA

AUDRIUS MIČIULIS

PROGRAMINĖS ĮRANGOS KŪRIMO PRIEMONIŲ MOBILIOSIOMS PLATFORMOMS TYRIMAS

Magistro darbas

Darbo vadovas:
lekt. dr. K. Jankauskas
2013-05-23

Recenzentas:
lekt. dr. K. Kapočius
2013-05-23

Atliko:
IFM-1/1 gr. studentas
Audrius Mičiulis
2013-05-23

KAUNAS, 2013

AUTORIŲ GARANTINIS RAŠTAS
DĖL PATEIKIAMO KŪRINIO

2013-05-21

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)
_____ (vardas, pavardė)	_____ (parašas)
_____ (vardas, pavardė)	_____ (parašas)

SANTRAUKA

Šiuo metu viena iš labiausiai bei sparčiausiai besivystančių technologijos krypčių yra mobiliesiems telefonams, planšetiniams kompiuteriams bei multimedijos įrenginiams skirta programinė įranga. Vartotojų susidomėjimas šia sritimi bei mobiliųjų įrenginių techninės įrangos tobulėjimas skatina vis daugiau IT kompanijų plėtoti savo veiklą šioje srityje. Viena iš didžiausių klaidų, kurias yra sunkiausia ištaisyti, yra pirminėje projekto stadijoje neteisingai pasirinktos technologijos.

Šiame darbe apžvelgiamos ir palyginamos trijų populiariausių, į mobiliuosius įrenginius diegiamų operacinių sistemų: *Android, iOS* bei *Windows Phone* programų kūrimo sąsajos ir nustatomi jų privalumai bei trūkumai. Vertinimai yra atliekami atsižvelgiant į operacinės sistemos architektūrą, programų kūrimo įrankius, suderinamumą su technine įranga, programų derinimo galimybėmis, saugumo užtikrinimu, grafinės vartotojo sąsajos kūrimu, daugiakalbystės palaikymu, animacijos, garsų bei video medžiagos pateikimu, HTML turinio atvaizdavimu, integracija su žemėlapiais, vartotojo buvimo vietos nustatymu, sensorių bei tinklų pasiekiamumu, foninių procesų kūrimu, duomenų išsaugojimo bei apsikeitimo su kitomis programomis galimybėmis.

Tyrimo metu nustatyta, kad *Android* turi geriausias sąsajas kurti foninius procesus bei integruotis su kitomis programomis. *iOS* turi geriausius įrankius bei grafinės vartotojo sąsajos kūrimo priemones, o Windows Phone turi patogiausia karkasą darbui su multimedija, žemėlapiais bei vartotojo buvimo vietos nustatymu.

SUMMARY

At a present time, one of the most developing area of technology is software for mobile phones, tablets and multimedia devices. Growing numbers of users and improved hardware makes more and more companies to expand their activity in this market. One of the biggest mistakes, that is very hard to solve, is badly chosen technologies in a beginning stage of the projects.

The main objective of this project is to compare three of the most popular operating systems Android, iOS and Windows Phone software development kits and to determinate their advantages and disadvantages. Analysis is made based on operating system architecture, tools, compatibility with hardware, debugging, security, graphical user interface, multilanguage support, animation, sounds, video, HTML preview, interaction with maps and user location, sensors, networks, background processes, data persistence and integration with another software.

In this research was identified that Android has best interfaces to create background processes and share data between programs. iOS has best tools and components for graphical user interface and Windows Phone has best framework to work with multimedia, maps and user location services.

Turinys

LENTELIŲ SĄRAŠAS

PAVEIKSLŲ SĄRAŠAS

TERMINŲ IR SANTRUMPŲ ŽODYNĖLIS

Android – Google Inc. kompanijos Linux pagrindu sukurta operacinė sistema skirta mobiliesiems telefonams ir planšetiniams kompiuteriams.

iOS – Apple Inc. kompanijos sukurta operacinė sistema skirta mobiliesiems telefonams, planšetiniams kompiuteriams ir nešiojamiems multimedijos įrenginiams.

Windows Phone – Microsoft Inc. kompanijos sukurta operacinė sistema skirta mobiliesiems telefonams ir planšetiniams kompiuteriams.

SDK *(angl. Software Development Kit)* – programavimo priemonių rinkinys, skirtas kurti programinę įrangą tam tikrai programavimo kalbai, terpei, platformai, operacinei sistemai ar konsolei.

API *(angl. Application Programming Interface)* – programos kodu pagrįsta sąsaja, specifikuojanti skirtingų programų tarpusavio komunikavimą.

HTTP *(angl. HyperText Transfer Protocol)* – protokolas nusakantis metodus, kaip turi bendrauti kliento-serverio sistemos paskirstytame tinkle.

HTTPS *(angl. Hypertext Transfer Protocol Secure)* – saugų ryšį paskirstytame tinkle užtikrinantis protokolas.

FTP *(angl. File Transfer Protocol)* – standartizuotas tinklo protokolas, naudojamas bylų persiuntimui paskirstytame tinkle.

HTML *(angl. HyperText Markup Language)* – kompiuterinė kalba, naudojama pateikti turinį internete.

XML *(angl. Extensible Markup Language)* – kompiuterinė bendros paskirties duomenų struktūrų bei jų turinio aprašomoji kalba.

URL *(angl. Uniform resource locator)* – iš simbolių sudarytas adresas, kuriuo galima pasiekti resursus internete.

Java – nuo kompiuterio architektūros nepriklausanti, objektiškai orientuota programavimo kalba.

JavaScript – objektiškai orientuota išplėtimų programavimo kalba, naudojama internetinių puslapių interaktyvumui realizuoti.

UML *(angl. Unified Modeling Language)* – modeliavimo ir specifikacijų kūrimo kalba, skirta specifikuoti, atvaizduoti ir konstruoti objektiškai orientuotų programų dokumentus.

OpenGL *(angl. Open Graphics Library)* – nuo operacinės sistemos nepriklausoma, programos kodu pagrįsta sąsaja, leidžianti atvaizduoti 2D bei 3D objektus.

HVGA *(angl. Half-size VGA)* - 3:2 (480x320 taškų), 4:3 (480x360 taškų), 16:9 (480x272) arba 8:3 kraštinių santykio (640:240 taškų) ekrano skiriamoji geba.

WVGA *(angl. Wide Video Graphics Array)* –5:3 (800x480 taškų) arba 16:9 kraštinių santykio (852:480 taškų) ekrano skiriamoji geba.

A-GPS *(angl. Assisted GPS)* – palydovinė įrenginio buvimo vietos nustatymo sistema.

LINQ *(angl. Language Integrated Query) – Microsoft Inc.* kompanijos sukurta ir *.NET* platformoje naudojama užklausų formavimo kalba duomenų mainams su reliacinėmis duomenų bazėmis.

SQL *(angl. Structured Query Language)* – standartizuota užklausų formavimo kalba duomenų mainams su reliacinėmis duomenų bazėmis.

ĮVADAS

Problemos aktualumas

Šiuo metu viena iš labiausiai bei sparčiausiai besivystančių technologijos krypčių yra mobiliesiems telefonams, planšetiniams kompiuteriams bei multimedijos įrenginiams skirta programinė įranga. Bendros metinės pajamos iš mobiliosioms platformoms skirtos programinės įrangos pakilo nuo 1100 mln. JAV dolerių iki 8500 mln. JAV dolerių 2009 – 2011 metų laikotarpyje ir toliau auga[[1]]. Vartotojų susidomėjimas šia sritimi bei mobiliųjų įrenginių techninės įrangos tobulėjimas skatina vis daugiau IT kompanijų plėtoti savo veiklą šioje srityje. Šiuo metu nedaug IT specialistų turi žinių ir patirties, reikalingos tokiai programinei įrangai kurti. Jų perkvalifikavimui gali prireikti nemažai laiko. Be to, skirtingi mobiliųjų technologijų gamintojai kuria ir į savo įrenginius diegia skirtingas operacines sistemas, kuriose programinę įrangą galima kurti tik su tai operacinei sistemai pritaikyta programavimo kalba. Viena iš didžiausių klaidų, kurias yra sunkiausia ištaisyti, yra pirminėje projekto stadijoje neteisingai pasirinktos technologijos. Neteisingai pasirinkus programavimo priemones, gali būti neįmanoma įgyvendinti visų numatytų funkcinių bei nefunkcinių reikalavimų, o sprendimas pakeisti technologijas projekto eigoje gali kainuoti viso esamo kodo perrašymą kita programavimo kalba. Todėl, prieš pradedant realizuoti projektą su naujomis, dar nenaudotomis technologijomis, būtina atlikti šių technologijų analizę.

Tyrimo sritis ir objektas

Tyrimo sritis – mobiliosios technologijos.

Tyrimo objektas – programavimo priemonės skirtos darbui *iOS, Android* bei *Windows Phone* operacinėse sistemose. Nagrinėjami ir palyginami programų kūrimo priemonių rinkiniai *(angl. Software Development Kit)*, jų privalumai ir trūkumai.

Tyrimo tikslas ir uždaviniai

Tyrimo tikslas – palyginti ir įvertinti *iOS, Android* bei *Windows Phone* operacinių sistemų programų kūrimo sąsajas.

Atsižvelgiant į tyrimo tikslą buvo suformuoti tokie tyrimo uždaviniai:

1. Nustatyti *Android SDK 4.0.0* programų kūrimo priemonių rinkinio privalumus ir trūkumus.
2. Nustatyti *iOS SDK 5.0.1* programų kūrimo priemonių rinkinio privalumus ir trūkumus.
3. Nustatyti *Windows Phone SDK 7.8* programų kūrimo priemonių rinkinio privalumus ir trūkumus.
4. Sukurti vertinimo metodiką, palyginti ir įvertinti *Android, iOS* bei *Windows Phone* programų kūrimo sąsajas 100 balų sistemoje.

1. PROGRAMINĖS ĮRANGOS KŪRIMO PRIEMONIŲ MOBILIOSIOMS PLATFORMOMS ANALIZĖ

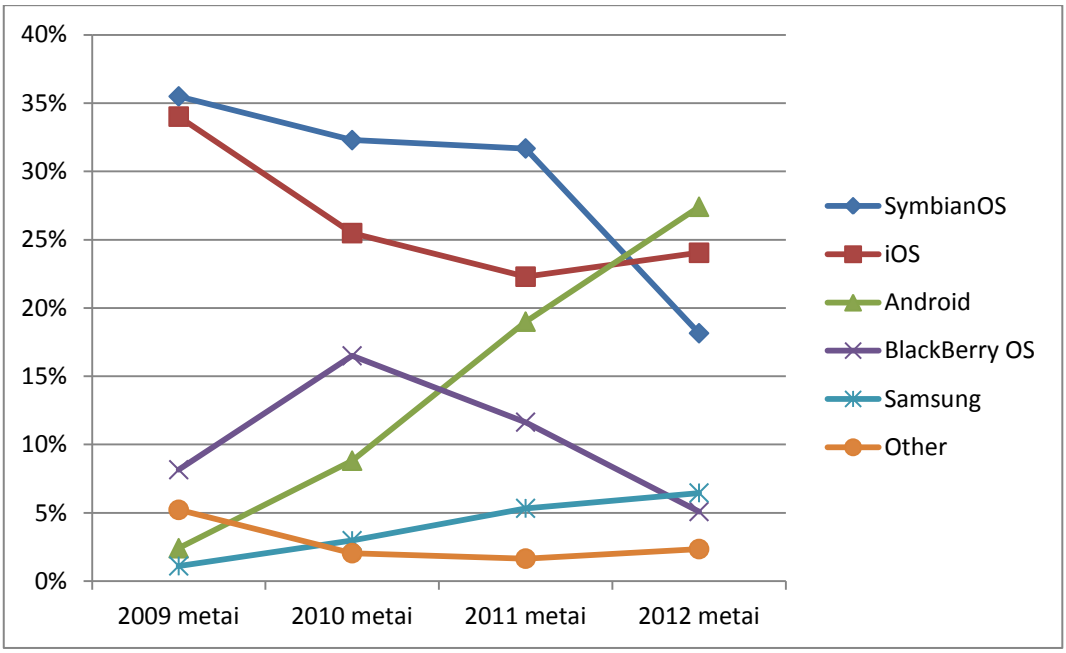
Šiame skyriuje bus aprašytos populiariausios į mobiliuosius įrenginius diegiamos operacinės sistemos. Apibrėžti programinės įrangos kūrimo sąsajų vertinimo kriterijai. Apžvelgti alternatyvūs sprendimai, kuriuos galima naudoti norint kurti nuo operacinės sistemos nepriklausančią programinę įrangą ir ištirti tokių sprendimų privalumai ir trūkumai.

1.1. Situacijos rinkoje apžvalga

Šiuo metu labiausiai paplitusios yra šios į mobilius įrenginius diegiamos operacinės sistemos:

1. *Android (Google Inc.)*
2. *iOS (Apple Inc.)*
3. *Symbian OS (Symbian Foundation)*
4. *BlackBerry (Research In Motion)*
5. *Windows Phone (Microsoft Corporation)*
6. *Bada (Samsung Electronic)*

Statistiką renkančios kompanijos *GlobalStats* duomenimis populiariausios yra *iOS* bei *Android* operacinės sistemos, kurios užima daugiau kaip 50% visos mobiliųjų įrenginių rinkos *(žr. Pav. 1.1)* [[2]]. Taip pat prognozuojama, kad *Microsoft Corporation* kompanijos rinkos dalis turėtų padidėti išpopuliarėjus *Windows Phone* operacinei sistemai, o *Symbian OS* populiarumas ir toliau kris.



Pav. 1.1 - „Išmaniųjų telefonų“ rinkos dalis pagal operacines sistemas

Android – atviro kodo, kompanijos *Google Inc*. išleista operacinė sistema, sukurta *Linux* operacinės sistemos pagrindu. Ji 2003 metų spalį buvo pradėta kurti kompanijos *Android Inc.*, o 2005 metų rugpjūtį ją įsigijo *Goggle Inc*. Su sistemos pardavimu į kompaniją perėjo ir pagrindiniai sistemos kūrėjai. Šiuo metu ši operacinė sistema yra labiausiai paplitusi rinkoje. Jos darbalaukio vaizdas yra pateiktas paveikslėlyje – Pav. 1.2.

iOS – kompanijos *Apple Inc*. sukurta operacinė sistema. Operacinės sistemos licencija neleidžia jos diegti į ne *Apple Inc.* kompanijos sukurtus produktus. Labai didelė dalis naujovių susijusių su grafinės sąsajos valdymu, programinės įrangos kūrimu ir platinimu atsirado būtent šioje operacinėje sistemoje. Operacinės sistemos darbalaukis pateiktas paveikslėlyje – Pav. 1.2.

Symbian OS – kompanijos *Symbian Foundation* sukurta operacinė sistema, labai ilgą laiką buvusi populiariausia tarp mobiliųjų įrenginių, tačiau pastaruoju metu netenkanti savo pozicijų. Ši

sistema buvo kuriama dar prieš „išmaniųjų telefonų“ atsiradimą ir diegiama kaip grafinė (arba tekstinė) sąsaja mobiliesiems telefonams. Kompanijos *Symbian Foundation* akcijos priklauso mobiliųjų telefonų gamintojai *Nokia*, todėl ši operacinė sistema buvo diegiama į visus šios kompanijos gaminamus telefonus iki pat 2011 metų vasario, kai Nokia pasirašė sutartį su *Microsoft Corporation*, įsipareigodama naudoti jos kuriamą operacinę sistemą *Windows Phone*. Šios operacinės sistemos darbalaukis yra pateiktas paveikslėlyje – Pav. 1.2.

Pav. 1.2 – *Android*, *iOS* ir *Symbian* operacinių sistemų darbalaukiai

Windows Phone – kompanijos *Microsoft Corporation* sukurta operacinė sistema, kuri pakeitė kitą šios kompanijos produktą *Windows Mobile*. Šią sistemą buvo pradėta kurti 2009 metais, o pirmoji jos versija išleista 2010 metų spalį. Programinė įranga, kurta *Windows Mobile* sistemai neturi suderinamumo su *Windows Phone* ir turi būti perrašyta su nauju *SDK*. Pagrindinis šios operacinės sistemos darbalaukis pateiktas paveikslėlyje – Pav. 1.3.

BlackBerry OS – kompanijos *Research In Motion* (*trump. RIM*) sukurta operacinė sistema, kuri yra diegiama į kompanijos *BlackBerry* „išmaniuosius telefonus“. Šios operacinės sistemos darbalaukis yra pateiktas paveikslėlyje – Pav. 1.3.

Bada – kompanijos *Samsung Electronic* sukurta ir 2010 metų rugpjūtį išleista operacinė sistema. Šalia „išmaniųjų telefonų“, ši sistema yra diegiama ir į „išmaniuosius televizorius“ (*angl. SmartTV*). Pagrindinis šios operacinės sistemos darbalaukis pateiktas paveikslėlyje – Pav. 1.3.

Pav. 1.3 - *Windows Phone*, *BlackBerry OS* ir *Bada* operacinių sistemų darbalaukiai

Visos šios operacinės sistemos turi programavimo priemonių rinkinį *(angl. Software Development Kit)* bei programų kūrimo sąsają *(angl. Application Programming Interface)* skirtą kurti programinę įrangą būtent toje operacinėje sistemoje. Su šiomis priemonėmis sukurtos programos negali būti perneštos ar perkompiliuotos į kitos operacinės sistemos kodą.

1.2. Analizės metodai

Tam, kad galėtume palyginti programavimo priemones skirtingose operacinėse sistemose turime apsibrėžti problemines sritis ir vertinimo kriterijus. Tuomet kiekvienai iš šių sričių sukuriami funkciniai bei nefunkciniai reikalavimai kurie bus realizuojami su eksperimentine programine įranga kiekvienoje iš analizuojamų operacinių sistemų. Testuojant sistemas bei atliekant eksperimentą bus laikoma, kad techninės įrangos galimybės yra identiškos ir pakankamos įvykdyti visus numatytus reikalavimus visose operacinėse sistemose.

1.2.1. Probleminės sritys

Norint įvertinti programavimo priemonių rinkinį privalome apsibrėžti problemines sritis ir kiekvienai iš šių sričių sukurti eksperimentinę programinę įrangą. Kadangi probleminės sritys nėra lygiavertės, kiekvienai iš jų yra priskiriamas svarbos koeficientas. Svarbos koeficientai parinkti taip, kad visų jų suma yra lygi 100. Probleminių sričių sąrašas pateiktas lentelėje – Lent. 1.1.

Lent. 1.1 - Probleminių sričių sąrašas

Sritis	Aprašymas	Svarba
Architektūra	Operacinės sistemos architektūra: sluoksniai, bibliotekos, standartinė programinė įranga, aplikacijos gyvavimo ciklas	8
Įrankiai	Programinės įrangos kūrimo įrankiai, jų diegimas ir atnaujinimas	5
Techninės įrangos emuliatorius	Galimybė išbandyti kuriamą programinę įrangą personaliniame kompiuteryje panaudojant techninės įrangos emuliavimo priemones	5
Suderinamumas su technine įranga	Programinės įrangos pritaikymo skirtingiems techniniams įrenginiams priemonės	5
Derinimas	Programinės įrangos derinimo (*angl. debugging*) priemonės	4
Saugumas	Programinės įrangos prieinamų techninių resursų kontroliavimas	4
Grafinė vartotojo sąsaja	Grafinės vartotojo sąsajos kūrimo priemonės: teksto įvedimas ir atvaizdavimas, mygtukai, sąrašai, paveiksliukai, žymimieji langeliai ir akutės	5
Meniu	Kontekstinio meniu sukūrimas ir valdymas	2
Dialogai	Dialogų kūrimas bei valdymas	2
Daugiakalbystė	Programinės įrangos adaptavimas keliomis skirtingomis kalbomis	4
Animacija	Animacijos kūrimas dvimatėje aplinkoje	6
Audio	Programinės įrangos papildymas muzika bei garso efektais	4
Video	Programinės įrangos papildymas vaizdine medžiaga	4
HTML	HTML turinio atvaizdavimas	4

Sritis	Aprašymas	Svarba
Žemėlapiai	Programinės įrangos papildymas žemėlapiais	5
Vartotojo buvimo vietos nustatymas	Vartotojo vietos nustatymas	4
Kamera	Kameros valdymas	3
Sensoriai	Duomenų gavimas iš sensorių	5
Tinklai	Informacijos perdavimas per tinklus (TCP/IP, Bluetooth)	5
Foniniai procesai	Foninių (*angl. background*) procedų kūrimas ir valdymas	5
Duomenų apsikeitimas	Duomenų apsikeitimas su kitomis programomis	5
Duomenų išsaugojimas	Informacijos išsaugojimas failinėje sistemoje bei integruotoje (*angl. embedded*) duomenų bazėje	6

1.2.2. Vertinimo kriterijai

Kiekvienai probleminiai sričiai išanalizuoti parengiami funkciniai ir nefunkciniai reikalavimai, kurie realizuojami visose pasirinktose operacinėje sistemoje su eksperimentine programine įranga. Tuomet kiekvienai programai pagal Lent. 1.2 lentelėje esančius vertinimo kriterijus ir (2.1) formulę suteikiamas įvertinimas kuris gali kisti nuo 0 iki 10.

$$V = \frac{\sum_{i=1}^{n} S_i * V_i}{\sum_{i=1}^{n} S_i * 10} * 10; \quad (2.1)$$

čia V – srities įvertinimas, S_i – vertinimo kriterijaus svarbumo koeficientas, V_i –įvertinimo balas, n – vertinimo kriterijų skaičius.

Kiekvienas vertinimo kriterijus įvertinamas 10 (dešimties) balų sistemoje, kur 10 (dešimt) reiškia, kad programa pilnai išpildo visus reikalavimus, o 0 (nulis) kad šie reikalavimai negali būti realizuoti persirinktoje sistemoje arba programa visiškai neatitinka nurodyto kriterijaus.

Suminis programavimo priemonių rinkinio įvertinimas yra apskaičiuojamas pagal (2.2) formulę ir jis gali kisti nuo 0 iki 100.

$$K = \frac{\sum_{i=1}^{n} S_i * V_i}{10}; \quad (2.2)$$

čia K – suminis vertinimo balas, S_i – srities svarbumo koeficientas, V_i – srities įvertinimas, n – sričių skaičius.

Lent. 1.2 - Vertinimo kriterijai

Kodas	Vertinimo kriterijus	Aprašymas	Svarba
VK1	Reikalavimų išpildymas	Įvertinimo balas nustatomas pagal tai, kiek funkcinių bei nefunkcinių reikalavimų galima pilnai įgyvendinti standartinėmis priemonėmis	20
VK2	Kodo eilučių skaičius	Įvertinimo balas nustatomas pagal tai, kiek programos kodo eilučių reikia parašyti, kad pilnai išpildytume visus numatytus funkcinius ir nefunkcinius reikalavimus	30
VK3	Patikimumas	Reikia įvertinti balu tikimybę, kad programinė įranga atliks jai priskirtas užduotis ar priešingai – neatliks jai nepriskirtų funkcijų	10

Kodas	Vertinimo kriterijus	Aprašymas	Svarba
VK4	Efektyvumas	Įvertinimo balas nustatomas pagal tai, kiek ir kokių kompiuterio resursų reikia, kad pilnai išpildytume visus numatytus funkcinius ir nefunkcinius reikalavimus	5
VK5	Integralumas	Įvertinama balais duomenų ir programinės įrangos funkcionalumo dalis, kurią gali pakeisti ar sugadinti neautorizuoti vartotojai	5
VK6	Palaikomumas	Įvertinimo balas nustatomas pagal tai, kiek reikia pastangų ir laiko atlikti pataisymams veikiančioje sistemoje	5
VK7	Testuojamumas	Įvertinimo balas nustatomas pagal tai, kiek reikia pastangų ištestuoti programinę įrangą	5
VK8	Lankstumas	Įvertinimo balas nustatomas pagal tai, kiek reikia pastangų norit pakeisti ar papildyti programinės įrangos funkcionalumą	5
VK9	Pernešamumas	Įvertinimo balas nustatomas pagal tai, kiek reikia laiko ir pastangų perketi sistemą iš vienos aparatūrinės konfigūracijos ar platformos į kitą	5
VK10	Pakartotinis panaudojamumas	Įvertinimo balas nustatomas pagal tai, kokios yra galimybės sukurtą programinę įrangą ar jos dalį panaudoti kuriant ar modifikuojant kitas sistemas	5
VK11	Sąveikos galimybės	Įvertinimo balas nustatomas pagal tai, kiek reikia laiko ir pastangų norint programinę įrangą apjungti su kita	5

1.2.3. Reikalavimai programos kodui

Kadangi vienas iš vertinimo kriterijų yra programos kodo eilučių skaičius, būtina apsibrėžti reikalavimus programos kodui. Kompanija *Sun Microsystems* 1997 m. rugsėjo 12 d. apibrėžė taisykles, kuriomis turi vadovautis programuotojai rašydami programos kodą, ir jas aprašė dokumente „*Java Code Conventions*“ [3]. Sukurta programinė įranga turi pilnai atitikti šiame dokumente aprašytas programavimo rekomendacijas. Papildomai būtina laikytis šių reikalavimų:

1. Tarp paketo pavadinimo ir importuojamų bibliotekų privalo būti vienos eilutės tarpas.
2. Tarp importuojamų bibliotekų ir klasės pavadinimo privalo būti vienos eilutės tarpas.
3. Tarp klasių privalo būti vienos eilutės tarpas.
4. Tarp metodų privalo būti vienos eilutės tarpas.
5. Tarp programos blokų privalo būti vienos eilutės tarpas.
6. Vienoje eilutėje negali būti daugiau nei vienas programos sakinys *(angl. statement).*
7. Sąlygos sakiniai, net jeigu jie sudaryti iš vieno sakinio, privalo turėti sakinio pradžios ir pabaigos žymas.
8. Jeigu ta pati išraiška metode yra naudojama daugiau nei vieną kartą, jos rezultatas privalo būti išsaugotas laikiname kintamajame.
9. Kiekvienas kintamasis turi būti aprašomas atskiroje eilutėje.

10. Klasių, metodų ar blokų atskyrimui negalima naudoti daugiau nei vienos iš eilės sekančios tuščios eilutės.

1.3. Alternatyvių sprendimų analizė

Standartinės, operacinės sistemos gamintojų pasiūlytos programinės įrangos kūrimo priemonės, nėra vienintelis būdas kurti programinę įrangą. Šiame skyriuje bus apžvelgtos kitos technologijos, leidžiančios kurti programinę įrangą, nepriklausančią nuo jos paleidimo aplinkos.

1.3.1. HTML

Programinė įranga gali būti aprašoma naudojant HTML žymėjimo kalbą ir paleidžiama su specialiu klientu – naršykle. Šias naršykles turi visos mobiliųjų įrenginių operacinės sistemos. HTML kalba buvo sukurta norint pateikti turinį internete ir yra orientuota į grafinės vartotojo sąsajos kūrimą ir kliento-serverio bendravimą HTTP protokolu. Norint pateikti interaktyvų turinį dažniausiai papildomai yra naudojama JavaScript išplėtimų kalba, leidžianti apdoroti kliento atliekamus veiksmus ir dinamiškai keičianti turinį.

Tačiau norint pilnai išnaudoti HTML teikiamus privalumus reikalingas HTTP serveris ir jame sukurta speciali programa leidžianti apdoroti kliento siunčiamas užklausas ir pateikianti rezultatus. Taigi tokiam modeliui reikalingas nuolatinis interneto ryšys. Be to programų galimybes ribotų ir tai, kad HTML kalba neturi priemonių darbui su įrenginio aparatūrine įranga, todėl nėra galimybės išnaudoti tokius mobiliojo įrenginio resursus kaip akselerometras arba fotoaparatas.

1.3.2. Java

Tai nuo kompiuterio architektūros nepriklausanti, objektiškai orientuota programavimo kalba. Šia kalba sukurtas programos kodas yra verčiamas į baitų kodą, kuriam paleisti reikalinga speciali programa – Java virtuali mašina *(angl. Java Virtual Machine)*. Šias virtualias mašinas turi visos populiariausios operacinės sistemos, taigi sukurtą programinę įrangą galime paleisti praktiškai bet kur.

Šios kalbos pagrindinis trūkumas yra tai, kad ji neturi priemonių dirbti su mobiliesiems įrenginiams skirta aparatūrine įranga.

1.3.3. C++

Tai žemo lygio programavimo kalba, kuri gali pasiekti ir valdyti visus mobiliojo įrenginio techninius resursus. Tačiau šia kalba sukurtą programinę įrangą galime paleisti tik ant vienos operacinės sistemos, kadangi darbui su skirtingomis operacinėmis sistemomis yra skirtos skirtingos programos kodo bibliotekos.

Šios kalbos pranašumas prieš operacinės sistemos gamintojų pateikiamas standartines programinės įrangos kūrimo priemones yra tai, kad nors ir programos kodo bibliotekos skirtingoms operacinėms sistemoms skiriasi ir programos kodą reikia perrašyti kiekvienai sistemai atskirai, pačios kalbos sintaksė išlieka ta pati.

1.4. Analizės išvados

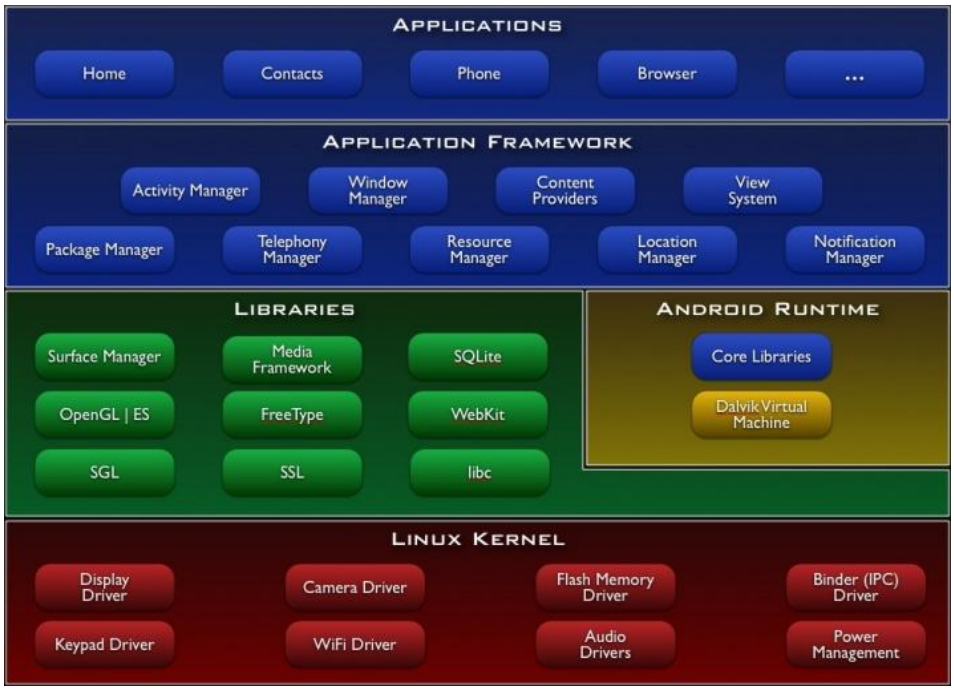
1. Šiuo metu populiariausios į mobiliuosius įrenginius diegiamos operacinės sistemos yra *Android, iOS* bei *Symbian*.
2. Visos operacinės sistemos turi standartines programinės įrangos kūrimo priemones, kuriomis galima kurti programas tik vienai operacinei sistemai.
3. Egzistuoja ir kitos, nuo operacinės sistemos nepriklausomos programų kūrimo priemonės, tačiau jos nėra pritaikytos dirbti su specifiniais mobiliojo įrenginio resursais, tokiais kaip kamera ar akselerometras.

2. TYRIMO OBJEKTŲ ANALIZĖ

2.1. Android

2.1.1. Architektūros apžvalga

Android – tai programinės įrangos paketas, skirtas mobiliesiems įrenginiams, į kurį įeina operacinė sistema, programinės įrangos kūrimo priemonės bei gamintojo sukurtos taikomosios programos. *Android* architektūrinė schema pateikta paveikslėlyje – Pav. 2.1[[4]].



Pav. 2.1 - *Android* architektūrinė schema

Android OS yra platinama su bazine taikomąją programine įranga *(angl. applications)* skirta valdyti mobilųjį įrenginį. Dažniausiai vartotojų naudojamos yra skambinimo, žinučių, el. pašto skaitymo bei siuntimo programos, žadintuvas, kalendorius, žemėlapiai, naršyklė, kontaktų valdymo programos ir t.t. Visos šios programos parašytos su *Java* programavimo kalba ir turi informacijos apsikeitimo su kitomis programomis sąsajas.

Integruota programinės įrangos kūrimo terpė *(angl. Application Framework)* suteikia galimybę kuriamoms sistemoms pasiekti įrenginio techninius resursus, nustatyti vartotojo buvimo vietą, paleisti foninius procesus, įdėti priminimus į užduočių juostą bei atlikti kitus veiksmus su sisteminiais procesais. Pagrindinės programų kūrimo paslaugos *(angl. services)* yra:

- *View System* – grafinės vartotojo sąsajos kūrimo priemonės.
- *Content Providers* – sąsajos, leidžiančios keistis informacija su kitomis programomis.
- *Resource Manager* – paslauga leidžianti gauti sistemos resursus esančius failinėje sistemoje.
- *Notifcation Manager* – ši paslauga leidžia įdėti priminimus į užduočių juostą.
- *Location Manager* – ši paslauga leidžia nustatyti vartotojo buvimo vietą.
- *Activity Manager* – šios paslaugos funkcijos yra užtikrinti programos gyvavimo ciklą (*angl. lifecycle*).

Android OS kūrėjai yra sukūrę specializuotų bibliotekų (*angl. libraries*) rinkinį, skirtą darbui su specifinėmis sritimis. Pagrindinės *Android* bibliotekos yra:

- *System C Library* – C programavimo kalbos bibliotekos *Linux* operacinei sistemai.
- *Media libraries* – bibliotekos skirtos darbui su garsu, statiniais paveiksliukais ir vaizdo bylomis (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG).

- *LibWebCore* – žiniatinklio naršyklės variklis, kurio pagalba galima atvaizduoti žiniatinklio (*HTML*, *CSS*, *JavaScript*) turinį tiek per integruotąją *(angl. embedded)* naršyklę, tiek ir tiesiai į grafinės vartotojo sąsajos elementus.
- *SGL* – 2D grafikos variklis.
- *3D libraries – OpenGL ES 1.0 API* realizacija darbui su 3D grafika.
- *SQL Lite* – duomenų bazių valdymo sistema.

Android paleidimo modulis *(angl. Android Runtime)* tai *Java* bibliotekų rinkinys programinės įrangos paleidimui užtikrinti. Kiekviena programa susikuria savo giją kuri yra paleidžiama su *Dalvik* virtualia mašina. Programos kompiliavimui naudojamas *Java* kompiliatorius ir *Android „dx“* įrankis. Sukompiliavus programą gaunama .dex plėtinio byla.

Android OS sukurtas *Linux 2.6* sistemos pagrindu, kuri užtikrina programų saugumą, atminties paskirstymą, gijų valdymą, tinklo funkcionalumą, techninės įrangos tvarkyklių diegimą. Tai yra pats žemiausias *Android* techninės architektūros sluoksnis.

2.1.2. Įrankiai

Norint kurti programinę įrangą *Android* operacinei sistemai reikia parsisiųsti *ADT Bundle* programavimo priemonių paketą arba į turimą *Eclipse IDE* distribuciją įdiegti *ADT* įskiepį, bei iš gamintojų tinklalapio parsisiųsti *Android SDK* bibliotekas.

ADT įskiepis leidžia susikurti virtualų mobilųjį įrenginį, kuriame galima išbandyti kuriamą programą. Šis įrenginys prastai dirba su animacija bei 3D grafika, o atlikus pakeitimus programos kode jį reikia iš naujo paleisti, kas užtrunka apie 2 min. (dirbant su kompiuteriu kuris turi 2,4 GHz procesorių bei 4Gb operatyvinės atminties). Be to šis įrenginys turi šiuos apribojimus[[5]]:

- Nėra galimybės skambinti ar priimti skambučių iš realių mobiliojo ryšio operatorių tinklų.
- Nepalaiko USB jungčių.
- Nėra fotografavimo/filmavimo funkcijų.
- Nepalaikomas mobiliojo tinklo būsenos nustatymas.
- Nėra galimybės nustatyti akumuliatoriaus įkrovimo lygio.
- Nėra galimybės simuliuoti SD kortelės nustatymo.
- Nepalaikomas Bluetooth įrenginys.

2.1.3. Derinimas

Programinės įrangos derinimui *(angl. debugging)* yra skirtas *LogCat* įrankis. Šis įrankis išsaugoja visus vartotojo bei aplikacijos atliekamus veiksmus, leidžia atrinkti pranešimus pagal pasirinktą aplikaciją, žymą (*angl. tag*), detalumo lygį (*angl. level*), proceso ID ar raktinį žodį.

2.1.4. Saugumas

Programinės įrangos kūrėjas privalo `AndroidManifest.xml` byloje aprašyti visus programos naudojamus techninius ir programinius resursus. Prieš aplikacijos instaliaciją vartotojas apie minėtų resursų naudojimą yra įspėjamas ir gali atsisakyti juos suteikti. Tokiu būdu yra apsaugoma nuo slapto vartotojo duomenų (pvz. adresų knygutės turinio) perdavimo tretiesiems asmenims.

2.1.5. Grafinė vartotojo sąsaja

Grafinei vartotojo sąsajai kuri *Android* kūrėjai yra realizavę standartinius komponentus. Grafinę vartotojo sąsają galima projektuoti naudojant *XML* aprašą arba aprašant visą struktūrą *Java* kodu. Dažniausiai naudojami komponentai yra aprašyti lentelėje – Lent. 2.1.

Lent. 2.1 - *Android* grafinės vartotojo sąsajos kūrimo komponentai

Komponentas	Klasės pavadinimas
Teksto atvaizdavimo laukas	`TextView`
Teksto įvedimo laukas	`EditText`

Komponentas	Klasės pavadinimas
Teksto įvedimo laukas (su automatiniu reikšmių užpildymu iš sąrašo)	`AutoCompleteTextView`
Mygtukas	`Button`
Sąrašo atvaizdavimo laukas	`ListView`
Iššokantis sąrašas	`Spinner`
Žyminčioji varnelė	`CheckBox`
Žyminčioji akutė	`RadioButton`
Nuotraukų sąrašo atvaizdavimo laukas	`Gallery`

2.1.6. Daugiakalbystė

Android turi priemones kurti aplikacijas keletu kalbų, kurių viena būtų parenkama priklausomai nuo vartotojo mobiliojo įrenginio nustatymų. Tokiu atveju reikia visus kalbinius resursus programos kode naudoti ne kaip konstantas, o kaip nuorodas į resursų bylas. Resursai, gali būti tiek kalbiniai, tiek ir paveiksliukai.

2.1.6.1. Kalbiniai resursai

Kalbiniai resursai yra saugomi `res\values\string.xml` byloje. Norint, kad resursai būtų pasiekiami tik tiems vartotojams, kurių įrenginiuose yra nustatyta konkreti kalba ar regionas reikia sukurti resursų katalogą `res\values-xx-yy` (kur xx – kalbos sutrumpinimas, yy – regiono sutrumpinimas). Resursų paieškoje galioja paveldimumo taisyklė, t.y. jeigu resursas nerastas lokalizuotoje byloje, jis ieškomas bendruosiuose resursuose.

2.1.6.2. Paveikslėliai

Kartais neužtenka išversti vien tik tekstinius resursus. Dalis kalbinės informacijos gali būti pateikta paveikslėliuose. Paveiksliukų lokalizavimo principas išlieka toks pats kaip ir tekstinių resursų. Skirtingoms kalboms ar regionams reikia sukurti papildomus katalogus pagal šabloną `res\drawable-xx-yy` (kur xx – kalbos sutrumpinimas, yy – regiono sutrumpinimas). Paveiksliukus, kurie neturi jokios kalbinės informacijos rekomenduojama dėti į bendrą `res\drawable` katalogą.

2.1.7. Animacija

Android turi keletą priemonių animacijai atvaizduoti. Nesudėtingai animacijai (pvz. animuotam logotipui parodyti) *Android* sistema turi komponentą, kuris leidžia atvaizduoti paveikslėlių seką, taip sukurdamas judesio iliuziją. Kitas animacijos kūrimo būdas yra grafinės vartotojo sąsajos objektų savybių (pvz. dydžio, vietos, pasukimo kampo) reikšmių keitimas laike. Sudėtinga animacija realizuojama *Java* kalbos kodu praplečiant *(angl. extend)* `SurfaceView` klasę kurioje galima modeliuoti įvairių objektų judėjimą keičiant šių objektų pozicijas laike. Tai yra pats sudėtingiausias iš visų paminėtų variantų ir dažniausiai yra naudojamas kompiuterinių žaidimų kūrimui.

2.1.8. Multimedija

Paveiksliukams, muzikai, garsams bei video medžiagai įrašyti bei atkurti *Android* programinės įrangos kūrimo paketas turi `MediaPlayer` klasę. Šios klasės palaikomi formatai pateikti lentelėje – Lent. 2.2[[5]].

Lent. 2.2 - *Android* MediaPlayer klasės palaikomi formatai

Tipas	Formatas, kodeksas	Įrašymas	Atkūrimas	Bylų plėtiniai
Garsas	AAC LC	Yra	Yra	3GPP (.3gp)

Tipas	Formatas, kodeksas	Įrašymas	Atkūrimas	Bylų plėtiniai
	HE-AACv1	Yra	Yra	MPEG-4 (.mp4, .m4a) ADTS raw AAC MPEG-TS
	HE-AACv2	Nėra	Yra	
	AAC ELD	Yra	Yra	
	AMR-NB	Yra	Yra	3GPP (.3gp)
	AMR-WB	Yra	Yra	3GPP (.3gp)
	FLAC	Nėra	Yra	FLAC (.flac)
	MP3	Nėra	Yra	MP3 (.mp3)
	MIDI	Nėra	Yra	Type 0 and 1 (.mid, .xmf, .mxmf) RTTTL/RTX (.rtttl, .rtx) OTA (.ota) iMelody (.imy)
	Vorbis		Yra	Ogg (.ogg) Matroska (.mkv)
	PCM/WAVE	Yra	Yra	WAVE (.wav)
Video	H.263	Yra	Yra	3GPP (.3gp) MPEG-4 (.mp4)
	H.264 AVC	Yra	Yra	3GPP (.3gp) MPEG-4 (.mp4) MPEG-TS (.ts)
	MPEG-4 SP	Nėra	Yra	3GPP (.3gp)
	VP8	Nėra	Yra	WebM (.webm) Matroska (.mkv)

2.1.9. HTML

Android programinės įrangos kūrimo paketas turi `WebView` komponentą, kuris leidžia atvaizduoti *HTML* kalba parašytos programos turininį. Šis komponentas turi metodus `loadUrl(String url)` ir `loadData(String data, String mimeType, String encoding)`, kurie leidžia atvaizduoti turinį pagal jo *URL* adresą arba tiesiogiai perduodant *HTML* programos kodą. *HTML* kodas gali būti papildytas *JavaScript* bei *CSS* turiniu.

2.1.10. Žemėlapiai

Android programinės įrangos kūrimo paketas neturi priemonių žemėlapių integracijai į kuriamą programinę įrangą. Norint kurti aplikacijas, kuriose butų atvaizduojami žemėlapiai rekomenduojama įdiegti *Google Play Services SDK* bibliotekas.

2.1.11. Vartotojo buvimo vietos nustatymas

Vartotojo buvimo vietai nustatyti *Android* skirta `LocationManager` klasė. Ši klasė leidžia nustatyti vartotojo buvimo vietą pagal ryšio tiekėjo tinklą arba pagal GPS tinklą. Programinės įrangos sąsaja leidžia nustatyti minimalų intervalą sekundėmis arba vartotojo buvimo vietos poslinkį metrais ir tik tuomet atnaujinti duomenis žemėlapyje. Tai padeda taupyti mobiliojo įrenginio akumuliatorių.

2.1.12. Kamera

Norint įdiegti fotografavimo ar filmavimo funkcijas *Android* aplikacijoje programuotojai turi du pasirinkimus: naudoti integruotą *Android* programėlę arba programuoti visas funkcijas pačiam

naudojant `Camera` komponentą. Pasirinkus pirmąjį variantą kai programai prireikia fotografavimo funkcijos iškviečiama standartinė *Android* programėlė, o vartotojui nufotografavus ar nufilmavus norimą vaizdą ir nuspaudus *„OK“* mygtuką, programėlė grąžina rezultatą (nuotrauką arba vaizdo klipą) atgal į ją iškvietusią aplikaciją. Antruoju variantu visas fotografavimo ar filmavimo funkcijas, kaip priartinimą, fokusavimą ar blykstės įjungimą reikia įgyvendinti pačiam.

2.1.13. Sensoriai

Norint gauti informaciją iš sensorių, reikia įgyvendinti `SensorEventListener` sąsają. Tuomet reikia aplikacijoje užregistruoti naudojamus sensorius sisteminėje `SensorManager` klasėje. Šios klasės pagalba galima sužinoti kokius sensorius turi naudojamas įrenginys, kokiu tikslumu jis gali pateikti duomenis ir kokios tam reikalingos energijos sąnaudos. *Android* palaikomų sensorių sąrašas pateiktas lentelėje – Lent. 2.3.

Lent. 2.3 - *Android* palaikomų sensorių sąrašas

Sensoriaus konstanta	Tipas	Aprašymas	Panaudojimas
`TYPE_ACCELEROMETER`	Aparatūrinis	Išmatuoja pagreitį (m/s^2), kuriuo juda įrenginys, įvertinant visas tris ašis (x, y, z) bei įvertinant gravitacijos jėgą	Judesių nustatymui (pasvyrimas, kratymas ir t.t.)
`TYPE_AMBIENT_TEMPERATURE`	Aparatūrinis	Išmatuoja patalpos temperatūrą (°C)	Oro temperatūros nustatymui
`TYPE_GRAVITY`	Aparatūrinis arba programinis	Išmatuoja (apskaičiuoja) gravitacijos jėgą įvertinant visas tris ašis (x, y, z)	Judesių nustatymui (pasvyrimas, kratymas ir t.t.)
`TYPE_GYROSCOPE`	Aparatūrinis	Išmatuoja įrenginio sukimosi greitį (rad/s) apie visas tris ašis (x, y, z)	Sukimosi nustatymui (pavertimui, sukimuisi ir t.t.)
`TYPE_LIGHT`	Aparatūrinis	Išmatuoja aplinkos apšvietimą (lx)	Ekrano šviesumo nustatymui
`TYPE_LINEAR_ACCELERATION`	Aparatūrinis arba programinis	Išmatuoja (apskaičiuoja) pagreitį (m/s^2), kuriuo juda įrenginys, įvertinant visas tris ašis (x, y, z) bet neįvertinant gravitacijos jėgos	Judesių nustatymui (pasvyrimas, kratymas ir t.t.)
`TYPE_MAGNETIC_FIELD`	Aparatūrinis	Išmatuoja aplinkos magnetinį lauką (μT)	Kompasas
`TYPE_ORIENTATION`	Programinis	Apskaičiuoja įrenginio pasisukimo laipsnį apie visas tris ašis	Įrengimo pozicijos nustatymui
`TYPE_PRESSURE`	Aparatūrinis	Išmatuoja aplinkos oro slėgį (hPa arba mbar)	Oro slėgio nustatymui
`TYPE_PROXIMITY`	Aparatūrinis	Išmatuoja atstumą (cm) nuo įrenginio ekrano iki artimiausio objekto	Dažniausiai naudojama nustatyti ar įrenginys pridėtas prie klausytojo ausies
`TYPE_RELATIVE_HUMIDITY`	Aparatūrinis	Išmatuoja oro drėgnumą (%)	Nustatyti rasai, santykinį ir absoliutų drėgnumą
`TYPE_ROTATION_VECTOR`	Aparatūrinis arba programinis	Išmatuoja (apskaičiuoja) įrenginio posūkio vektorius apie visas tris ašis	Judesio ir posūkio nustatymui

Sensoriaus konstanta	Tipas	Aprašymas	Panaudojimas
`TYPE_TEMPERATURE`	Aparatūrinis	Išmatuoja įrenginio temperatūrą (°C)	Temperatūrų nustatymui

2.1.14. Tinklai

Šiuo metu beveik visi mobilieji įrenginiai turi prieigą prie interneto. *Android* programinės įrangos kūrimo paketas turi `URLConnection` klasę kurios dėka galima užmegzti ryšį su nutolusiu kompiuteriu naudojantis *FTP*, *HTTP* arba *HTTPS* protokolu.

2.1.15. Foniniai procesai

Sudėtingus ar ilgai užtrunkančius veiksmus, kuriuose nėra reikalingas nuolatinis vartotojo įsikišimas rekomenduojama vykdyti foniniuose procesuose. Šiems procesams kurti *Android* platformoje reikia praplėsti `IntentService` klasę. Paleidus šį procesą, (pvz. įjungus groti muziką), vartotojas gali sėkmingai dirbti su kitomis programomis. Fone esanti programa, gali vartotojui palikti pranešimą užduočių juostoje *(angl. status bar)*. Šiems pranešimams valdyti yra skirta `NotificationManager` klasė.

2.1.16. Duomenų apsikeitimas su kitomis programomis

Duomenų apsikeitimui tarp skirtingų aplikacijų *Android* programinės įrangos kūrimo paketas turi `ContentProvider` ir `ContentResolver` klases.

2.1.16.1. Duomenų pateikimas

Norint leisti tam tikrais savo programos duomenimis naudotis kitoms aplikacijoms *Android* platformoje yra skirta `ContentProvider` klasė. Tai yra abstrakti klasė ir programos kūrėjas privalo praplėsti ją parašydamas šiuos metodus:

- `onCreate()` – šis metodas iškviečiamas visada startavus duomenis teikiančią aplikaciją;
- `query(Uri uri, String[] projection, String selection, String[] selectionArgs, String sortOrder)` – šis metodas leidžia teikti duomenis kitoms aplikacijoms;
- `insert (Uri uri, ContentValues values)` – šis metodas leidžia įterpti duomenis;
- `delete (Uri uri, String selection, String[] selectionArgs)` – šis metodas leidžia šalinti duomenis;
- `update (Uri uri, ContentValues values, String selection, String[] selectionArgs)` – šis metodas leidžia atnaujinti duomenis;

Duomenys yra saugomi lentelių formoje. Visuose metoduose (išskyrus `onCreate()`) perduodamas `Uri` parametras nurodo, kurį duomenų rinkinį (lentelę ar eilutę) vartotojas nori gauti ar koreguoti.

2.1.16.2. Duomenų gavimas

Gauti ar išsaugoti duomenims tokioje aplikacijoje yra skirta `ContentResolver` klasė. Sukūrę šios klasės objektą, ir žinodami mums dominančio duomenų rinkinio *URI* adreso reikšmę, mes galime kviesti ankščiau aprašytus metodus. Sisteminės *Android* aplikacijos (pvz. adresų knygutė) turi konstantas su savo duomenų lentelių adresais. Toks savo duomenų lentelių pavadinimų iškėlimas į viešas *(angl. public)* konstantas yra rekomenduotinas ir kuriant vartotojo aplikacijas.

2.1.17. Duomenų išsaugojimas

Duomenų išsaugojimui *Android* platformoje yra skirti šie komponentai:

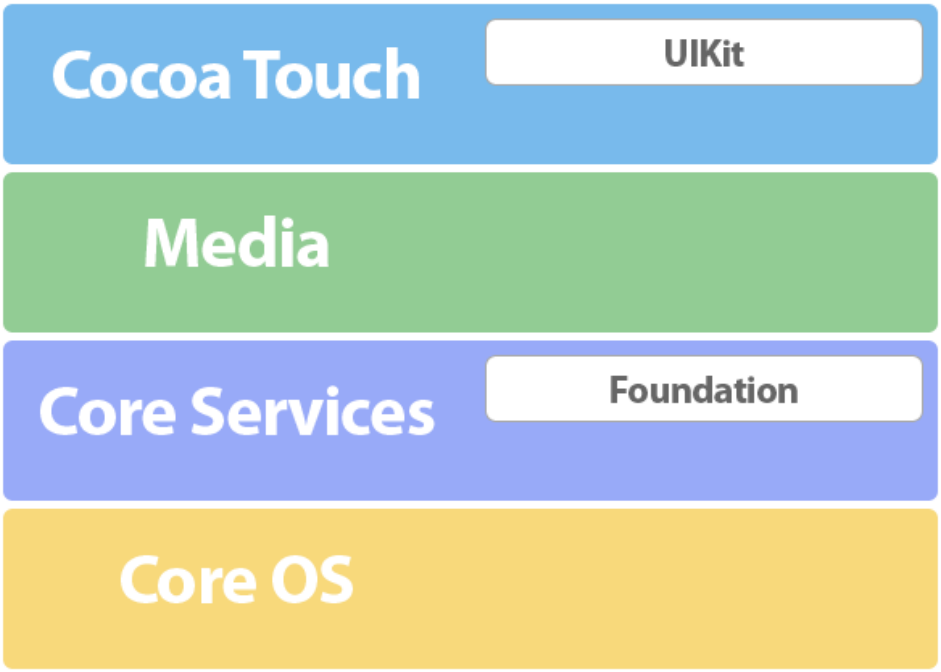
- *SharedPreferences* – nesudėtingos, tik aplikacijai prieinamos informacijos išsaugojimas rakto-reikšmės *(angl. key-value)* tipo žemėlapiuose *(angl. map).*
- Internal Storage – tik aplikacijai prieinamos informacijos išsaugojimas vidinėje atmintyje (bylose).
- External Storage – viešai prieinamos informacijos išsaugojimas išorinėje atmintyje (bylose).
- SQLite Databases – struktūrinės informacijos išsaugojimas duomenų bazėje.

Labiausiai paplitęs duomenų saugojimo būdas yra reliacinės duomenų bazės. Jos ne tik leidžia išsaugoti informaciją, bet ir užtikrina šių duomenų integralumą. *Android* programinės įrangos kūrimo rinkinys turi integruotą *SQLite* duomenų bazę su kuria bendraujama per `SQLiteDatabase` tipo objektą. Šis objektas turi metodus tiek duomenų gavimui, įterpimui, koregavimui ar šalinimui tiek ir darbui su duomenų bazės struktūra bei transakcijomis.

2.2. iOS

2.2.1. Architektūros apžvalga

iOS – tai *Apple Inc.* kompanijos sukurta operacinė sistema, kuri yra diegiama į jos kuriamus mobiliuosius telefonus, planšetinius kompiuterius ir multimedijos įrenginius. *iOS* programinės įrangos kūrimo paketo architektūra sudaryta iš 4 sluoksnių: *Cocoa Touch, Media, Core Services* ir *Core OS*. Ši architektūrinė schema yra pateikta paveikslėlyje – Pav. 2.2[[6]].



Pav. 2.2 - *iOS* architektūrinė schema

Cocoa Touch Layer – tai pats aukščiausias *iOS* architektūrinis sluoksnis. Jame realizuoti grafinės vartotojo sąsajos kūrimo elementai bei visi pagrindiniai karkaso komponentai skirti gestų atpažinimui, daugiaprogramiam režimui *(angl. multitasking)*, įvykių valdymui *(angl. events handling)* ir t.t. Jeigu tik yra galimybė, visuomet pirmiausiai reikia naudotis priemonėmis, kurios yra pateiktos šiame architektūriniame sluoksnyje.

Media Layer – šis sluoksnis skirtas multimedijos funkcionalumui pateikti. Jame yra realizuotos bibliotekos darbui su *Core Graphic, OpenGL ES, OpenAL, AV Foundation* bei *Core Media* technologijomis. Šių technologijų pagalba galima kurti animaciją bei sudėtingą 3D grafiką turinčius žaidimus. Šiame sluoksnyje taip pat pateiktos sąsajos video bei nuotraukų gavimui iš standartinių įrenginio katalogų.

Core Services Layer – šiame sluoksnyje yra realizuotos visos bazinės *Objective-C* kalbos klasės, kuriomis remiasi visi aukštesni architektūros sluoksniai. Būtent šiame sluoksnyje yra realizuotos tokios klasės kaip `NSArray`, `NSDictionary`, `NSDate` ar `NSObject`, skirtos sąrašų, aibių, reikšmių žemėlapių ar datų funkcionalumui realizuoti.

Core OS Layer – tai pats žemiausias architektūrinis sluoksnis ir dauguma jo sąsajų yra paslėptos nuo vartotojų dėl saugumo užtikrinimo. Šiame sluoksnyje yra realizuotos komunikavimo su UNIX branduoliu bei technine įranga sąsajos.

2.2.2. Įrankiai

Įdiegus *iOS SDK* automatiškai yra įdiegiami ir visi įrankiai, kurių reikia norint kurti programinę įrangą *iOS* operacinei sistemai. Pats svarbiausias įrankis yra *Xcode*, tai integruota programavimo aplinka, kurioje yra kuriami projektai, rašomas programos kodas bei projektuojama grafinė vartotojo sąsaja.

Programų paleidimui galima naudoti *iPhone Simulator* virtualų mobilųjį įrenginį. Šis įrenginys paleidžia per 10-15s. (dirbant su kompiuteriu kuris turi 2,4 GHz procesorių bei 4Gb operatyvinės atminties) ir gali pakeisti beveik visas tikrojo įrenginio funkcijas. Pakeitus programos kodą būtina perkrauti programą emuliatoriuje.

2.2.3. Derinimas

Programinės įrangos derinimui yra naudojamos *XCode* priemonės. Šios priemonės leidžia sustabdyti programos veikimą norimoje vietoje, peržiūrėti ar pakeisti kintamųjų reikšmes. *iOS SDK* turi papildomą įrankį *Instruments*, kurio pagalba galima stebėti sunaudojamus resursus, tvarkomas bylas, objektus, kas padeda nustatyti programos silpnas vietas ar aptikti atminties praradimus *(angl. memory leaks)*.

2.2.4. Saugumas

Programinės įrangos kūrėjas privalo `Info.plist` byloje aprašyti visus programos naudojamus techninius resursus. Tačiau ši informacija yra naudojama tik techninės įrangos suderinamumui patikrinti ir vartotojas apie juos nėra informuojamas. Vieninteliai resursai, apie kuriuos yra perspėjamas vartotojas, yra pranešimų kūrimas ir vartotojo vietos nustatymas.

2.2.5. Grafinė vartotojo sąsaja

Grafinei vartotojo sąsajai kuri *iOS* kūrėjai yra realizavę standartinius komponentus. Grafinę vartotojo sąsają galima projektuoti naudojant *Interface Builder* įrankį arba aprašant visą struktūrą *Objective-C* kodu. Dažniausiai naudojami komponentai yra aprašyti lentelėje – Lent. 2.4.

Lent. 2.4 - *iOS* grafinės vartotojo sąsajos kūrimo komponentai

Komponentas	Klasės pavadinimas
Teksto atvaizdavimo laukas	`UILabel`
Teksto įvedimo laukas	`UITextField`
Mygtukas	`UIButton`
Sąrašo atvaizdavimo laukas	`UIPickerView`
Jungiklis (įjungta/išjungta)	`UISwitch`
Auselė	`UISegmentedControl`
Paveiksliukas	`UIImageView`
Datos pasirinkimas	`UIDatePicker`

2.2.6. Daugiakalbystė

iOS turi priemones kurti aplikacijas keletu kalbų, kurių viena būtų parenkama priklausomai nuo vartotojo mobiliojo įrenginio nustatymų. Resursai, gali būti tiek kalbiniai, tiek ir paveiksliukai. Kiekvienai kalbai yra sukuriamas atskiras katalogas, kuriame bus saugomi tekstai bei paveikslėliai. Katalogas turi būti pavadintas pagal šabloną `xx.lproj` (kur xx – kalbos sutrumpinimas).

2.2.6.1. Kalbiniai resursai

Kalbiniai resursai yra saugomi `Localizable.strings` byloje, rakto-reikšmės principu. Resurso raktas gali būti sudarytas ne tik iš raidžių, bet ir skaičių, tarpų bei kitų specialiųjų simbolių, todėl yra įprasta, kad raktas ir jo reikšmė pagrindinės programos kalboje sutampa.

2.2.6.2. Paveikslėliai

Paveiksliukų lokalizavimo principas išlieka toks pats kaip ir tekstinių resursų. Skirtingoms kalboms skirtus paveiksliukus reikia patalpinti į atskirus lokalizuotus katalogus kartu su tekstiniais resursais. Paveiksliukų bylos pavadinimas visuose kataloguose turi sutapti.

2.2.7. Animacija

Animacijai kurti *iOS* operacinėje sistemoje yra sukurti du karkasai: *UIKit* ir *Core Animation*. *UIKit* karkasas dirba tik su `UIView` tipo objektais. Tai reiškia, kad galima animuoti grafinės vartotojo sąsajos elementus: priversti juos judėti, keisti dydį ar spalvą. *Core Animation* karkasas leidžia atlikti sudėtingus animacijos veiksmus tiek dvimatėje, tiek ir trimatėje erdvėje pasinaudojant transformacijos matricomis. Ši animacija dažniausiai yra naudojama kompiuterinių žaidimų kūrime.

2.2.8. Multimedija

Muzikai ir garsams įrašyti bei atkurti *iOS* operacinėje sistemoje yra skirtos `AVAudioPlayer` bei `AVAudioRecorder` klasės. Video medžiagai atvaizduoti skirta `MPMoviePlayerController` klasė. Šių klasių palaikomi formatai pateikti lentelėje – Lent. 2.5[[7]].

Lent. 2.5 - *iOS SDK* palaikomi audio bei video formatai

Tipas	Formatas, kodeksas	Įrašymas	Atkūrimas
Garsas	AAC	Yra	Yra
	HE-AAC	Nėra	Yra
	Apple Lossless (ALAC)	Nėra	Yra
	A-law	Yra	Yra
	μ-law	Yra	Yra
	AMR (Adaptive Multi-rate)	Nėra	Yra
	iLBC (internet Low Bitrate Codec)	Yra	Yra
	IMA4 (IMA/ADPCM)	Yra	Yra
	Linear PCM (uncompressed)	Yra	Yra
	MP3	Nėra	Yra
Video	H.264	Yra	Yra
	MPEG-4	Yra	Yra

2.2.9. HTML

iOS SDK turi `UIWebView` komponentą, kuris leidžia atvaizduoti *HTML* kalba parašytos programos turininį. Šio komponento metodai `loadRequest(NSURLRequest)` ir `loadHTMLString(HTMLData) baseURL(NSURLRequest)` leidžia atvaizduoti turinį pagal jo *URL*

adresą arba tiesiogiai perduodant *HTML* programos kodą. *HTML* kodas gali būti papildytas *JavaScript* bei *CSS* turiniu.

2.2.10. Žemėlapiai

Žemėlapių atvaizdavimui *iOS* aplikacijose yra skirtas *Apple MapKit* karkasas. Šis karkasas leidžia nustatyti žemėlapio koordinates, priartinimo lygį, parodyti vartotojo buvimo vietą, uždėti ant žemėlapio specialias žymas, be to labai svarbi ir išskirtina šio karkaso savybė yra ta, kad jis leidžia išsisaugoti *(angl. cache)* visus parsisiųstus žemėlapius su galimybe panaudoti juos ateityje[[8]]. Ši savybė leidžia sutaupyti tinklo resursus ar netgi leisti naudotis žemėlapių funkcionalumu laikinai neturint interneto ryšio.

2.2.11. Vartotojo buvimo vietos nustatymas

Vartotojo buvimo vietai nustatyti *iOS* skirta `CLLocationManager` klasė. Programinės įrangos sąsaja leidžia nustatyti minimalų vartotojo buvimo vietos pokytį metrais ir tik tuomet atnaujinti duomenis žemėlapyje. Tai padeda taupyti mobiliojo įrenginio akumuliatorių.

2.2.12. Kamera

iOS turi komponentą kuris leidžia iškviesti integruotą fotografavimo ar filmavimo programėlę. Pasinaudojus šia programėle rezultatas apdorojimui grąžinamas į pagrindinę programą. Sisteminė programėlė jau turi visas pagrindines fotografavimo ar filmavimo funkcijas, tokias kaip fokusavimas ar baltos šviesos srauto nustatymas, tačiau toks sprendimas netinka kai norima, kad fotografavimo funkcija užimtų tik dalį viso lango ar norima sukurti individualią kameros valdymo sąsaja.

2.2.13. Sensoriai

Kiekvienas sensorius *iOS* sąsajoje turi klasę, kurios pagalba galima gauti jo duomenis. Akcelerometro duomenims gauti yra skirta `CMAccelerometerData`, giroskopo - `CMRotationRate`, o kompaso duomenims `CMMagneticField` klasė. Sukuriant sensoriaus duomenis grąžinančią klasę reikia nurodyti laiko intervalą, kuriuo bus atnaujinami gaunami duomenys.

2.2.14. Tinklai

Šiuo metu beveik visi mobilieji įrenginiai turi prieigą prie interneto. *iOS* programinės įrangos kūrimo paketas turi `NSURLConnection` klasę kurios dėka galima užmegzti ryšį su nutolusiu kompiuteriu naudojantis *FTP, HTTP, HTTPS* arba *FILE* protokolu.

2.2.15. Foniniai procesai

Foninius procesus *iOS* sistemoje galima suskirstyti į dvi grupes: priminimai *(angl. notifications)* ir ilgai besitęsiančios foninės užduotys *(angl. long-running background tasks).*

2.2.15.1. Priminimai

Priminimai leidžia aplikacijai susisiekti su vartotoju ir parodyti jam pranešimą, kaip aplikacija yra išjungta arba nėra aktyvi. Priminimas – tai tekstinis iššokantis langas, kuris gali būti papildytas piktograma bei muzika ar garsu. Iššokantis langas gali turėti mygtuką, kurį nuspaudus paleidžiama pranešimą atsiuntusi programa. Šio tipo pranešimams kurti yra skirta `UILocalNotification` klasė.

2.2.15.2. Ilgai besitęsiančios foninės užduotys

iOS operacinė sistema riboja ilgai besitęsiančių foninių užduočių naudojimą, nes tai apkrauna įrenginio resursus bei akumuliatorių. Šiuos foninius procesus galima naudoti tik šiais atvejais[[7]]:

- Aplikacijoms, kurios groja muziką (pvz. muzikos grotuvas).
- Aplikacijoms, kurios fiksuoja vartotojo buvimo vietą (pvz. navigacijos aplikacija).
- Aplikacijoms, kurios teikia telefonijos (VoIP) paslaugas.
- Naujienų pranešimų aplikacijoms.

- Aplikacijoms, kurioms reikia tam tikru periodiškumu prisijungti prie išorinių įrenginių ar duomenų šaltinių. Tokiu atveju turi būti įgyvendinta galimybė vartotojui įjungti bei išjungti šią funkciją. Vienas aplikacijos prisijungimas prie išorinių šaltinių negali būti ilgesnis nei 10 sekundžių.

iOS programinės įrangos kūrimo sąsajoje nėra priemonių ar vartotojas tikrai panaudojo foninius procesus vienam iš aukščiau aprašytų tikslų pasiekti, tačiau jeigu vartotojo aplikacija pažeidė šiuos reikalavimus, ji gali būti pašalinta iš *Apple* programėlių parduotuvės *(anlg. Apple App Store)*.

2.2.16. Duomenų apsikeitimas su kitomis programomis

Kiekviena *iOS* programa pasileidžia izoliuotoje, tik tai aplikacijai prieinamoje aplinkoje ir nėra galimybių apsikeisti duomenimis su kitomis aplikacijomis (net jei duomenys saugomi integruotoje duomenų bazėje). *iOS SDK* yra sąsajos gauti duomenis tik iš tam tikrų sisteminių programėlių (pvz. adresų knygutės).

2.2.17. Duomenų išsaugojimas

Duomenų išsaugojimui *iOS* operacinėje sistemoje yra skirtos šios technologijos[[9]]:

- *Property Lists* – tai objektų masyvas arba sąrašas, kurį galima nusiskaityti bei išsisaugoti kiekvieną kartą paleidžiant ar išjungiant aplikaciją. Masyve galima saugoti tik `NSString`, `NSNumber`, `NSData`, `NSDate`, `NSArray` bei `NSDictionary` tipo objektus.
- *Core Data* – šis karkasas leidžia išsaugoti bet kokio tipo objektus duomenų bazėje, *XML* bylose ar sąrašuose. Programos kūrėjui nereikia pačiam projektuoti duomenų bazės ar *XSD* schemos struktūros, kurią automatiškai sukuria pats karkasas.
- *User Defaults* – šis karkasas leidžia išsaugoti vartotojo nustatymus ar grafinės vartotojo sąsajos laukų reikšmes ir automatiškai jas atstatyti kai vartotojas kitą kartą įsijunks tą pačią programėlę.
- *SQLite* – leidžia išsaugoti struktūrinę informaciją reliacinėje duomenų bazėje.

2.3. Windows Phone

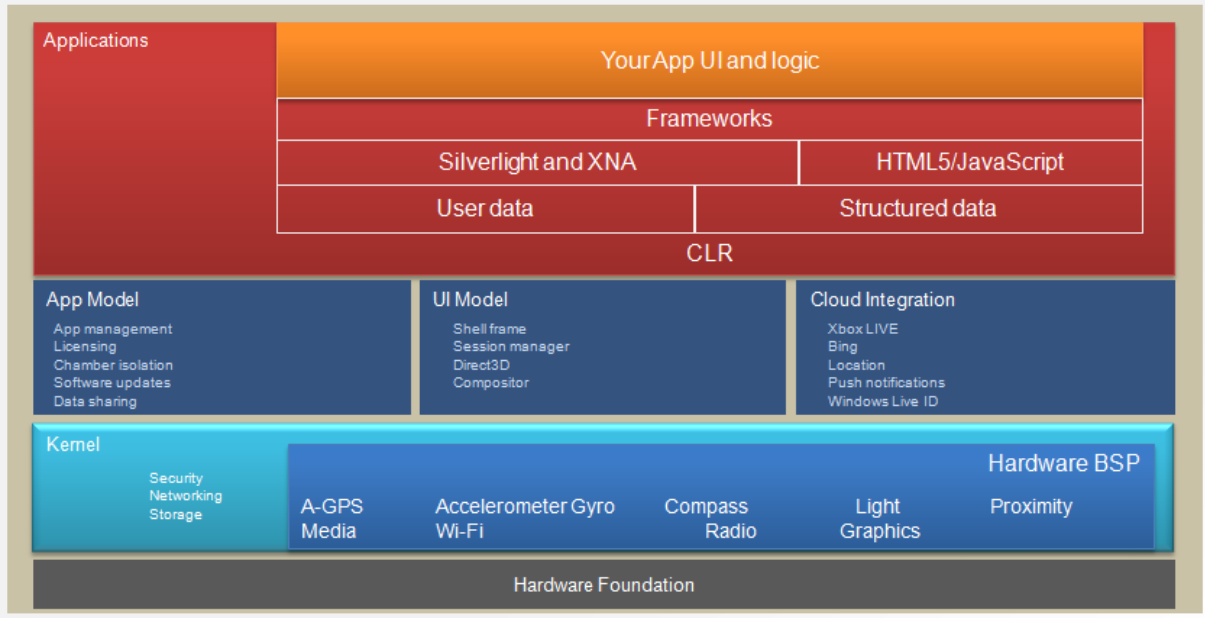
2.3.1. Architektūros apžvalga

Windows Phone 7 – Microsoft Inc. kompanijos sukurta operacinė sistema. Tai pirmoji Windows operacinė sistema turinti *Metro* grafinę vartotojo sąsają. Techninės įrangos gamintojams *Microsoft Inc.* kompanija nustatė šiuos minimalius techninės įrangos reikalavimus[[10]]:

- *ARMv7 Cortex/Scorpion* arba galingesnis procesorius;
- lietimui jautrus, *WVGA* (800×480) arba *HVGA* (480×320) raiškos ekranas;
- DirectX9 tvarkykles palaikantis grafinis procesorius;
- 512 Mb operatyvios atminties;
- 4 Gb išorinė atmintinė *(angl. flash memory)*;
- vartotojo vietos nustatymo įrengints A-GPS;
- akselerometras;
- kompasas;
- šviesumo nustatymo sensorius;
- atstumo iki objekto nustatymo sensorius *(angl. proximity sensor)*;
- 5 arba daugiau megapikselių kamera;
- „Start“, „Search“ ir „Back“ mygtukai.

Vieninga techninė įranga garantuoja, kad ištestavus kuriamą aplikaciją su vienu įrenginiu galime tikėtis analogiško veikimo su visais kitais modeliais.

Windows Phone architektūrinė schema pateikta paveikslėlyje – Pav. 2.3.



Pav. 2.3 - *Windows Phone* architektūrinė schema

Žaidimų kūrimui yra skirtas *XNA* karkasas, kuris yra naudojamas ir *Microsoft Inc.* sukurtame *XBOX* žaidimų kompiuteryje. Interaktyviai grafinei vartotojo sąsajai kurti yra dedikuotas *Silverlight* karkasas. Interaktyviam turiniui kurti galima naudoti integruotą *Internet Explorer* naršyklę, kuri palaiko *HTML5* bei *JavaScript* standartus.

Standartinės programinės įrangos kūrimo priemonės užtikrina programos gyvavimo ciklą, duomenų apsaugą, licencijavimą, grafinės vartotojo sąsajos valdymo priemones, integraciją su debesų kompiuterija, žemėlapiais ar vietos nustatymu. Pavyzdžiui programų kūrėjas gali nustatyti nemokamą 15 dienų programos bandomąjį laikotarpį ir operacinė sistema automatiškai uždraus šios programos naudojimą po nustatyto laikotarpio, net jeigu vartotojas sumanys ją įdiegti iš naujo.

Žemiausiame architektūriniame lygyje yra techninių įrenginių tvarkyklės. Daugumą tvarkyklių yra parašiusi pati *Microsoft Inc*. kompanija ir techninės įrangos gamintojams nereikia jomis rūpintis[[11]].

2.3.2. Įrankiai

Norint kurti programinę įrangą *Windows Phone* operacinei sistemai reikia įsidiegti *Microsoft Visual Studio 2010 Express for Windows Phone* įrankį. Šis įrankis yra nemokamas, tačiau jame nėra daugumos naudingų funkcijų, kurios pateikiamos tik su mokama *Professional* įrankio versija.

Programų paleidimui galima naudoti *Windows Phone Emulator* virtualų mobilųjį įrenginį. Šis įrenginys paleidžia per 10-15s. (dirbant su kompiuteriu kuris turi 2,4 GHz procesorių bei 4Gb operatyvinės atminties) ir gali pakeisti praktiškai visas tikrojo įrenginio funkcijas. Pakeitus programos kodą būtina perkrauti programą emuliatoriuje.

2.3.3. Derinimas

Programinės įrangos derinimui yra naudojamos standartinės *Microsoft Visual Studio 2010* priemonės. Šios priemonės leidžia sustabdyti programos veikimą norimoje vietoje, peržiūrėti ar pakeisti kintamųjų reikšmes.

2.3.4. Saugumas

Programinės įrangos kūrėjas privalo `WMAppManifest.xml` byloje aprašyti visus programos naudojamus techninius ir programinius resursus. Prieš aplikacijos instaliaciją vartotojas apie minėtų resursų naudojimą yra įspėjamas ir gali atsisakyti juos suteikti. Tokiu būdu yra apsaugoma nuo slapto vartotojo duomenų (pvz. adresų knygutės turinio) perdavimo tretiesiems asmenims.

2.3.5. Grafinė vartotojo sąsaja

Grafinei vartotojo sąsajai kuri *Windows Phone* kūrėjai yra realizavę standartinius komponentus. Grafinę vartotojo sąsają galima projektuoti naudojant *XML* aprašą arba aprašant visą struktūrą *C#* arba *VisualBasic* kodu. Dažniausiai naudojami komponentai yra aprašyti lentelėje – Lent. 2.6.

Lent. 2.6 - *Windows Phone* grafinės vartotojo sąsajos kūrimo komponentai

Komponentas	Klasės pavadinimas
Teksto atvaizdavimo laukas	`TextBlock`
Teksto įvedimo laukas	`TextBox`
Slaptažodžio įvedimo laukas	`PasswordBox`
Mygtukas	`Button`
Sąrašo atvaizdavimo laukas	`ListBox`
Žyminčioji varnelė	`CheckBox`
Žyminčioji akutė	`RadioButton`
Paveiksliukas	`Image`
Figūra	`Canvas`
Nuoroda	`HyperlinkButton`

2.3.6. Daugiakalbystė

Windows Phone turi priemones tekstinių resursų pateikimui keletu kalbų, kurių viena būtų parenkama priklausomai nuo vartotojo mobiliojo įrenginio nustatymų. Kalbiniai resursai yra saugomi `.resx` tipo bylose, kurias redaguoti galima tik *Microsoft Visual Studio* priemonėmis. Resursų byla turi būti pavadinta pagal šabloną `pavadinimas.xx-yy` (kur xx – kalbos sutrumpinimas, yy – regiono sutrumpinimas). Resursų paieškoje galioja paveldimumo taisyklė, t.y. jeigu resursas nerastas lokalizuotoje byloje, jis ieškomas bendruosiuose resursuose. Resursai gali būti tik tekstiniai, *Windows Phone SDK* neturi standartinių priemonių paveiksliukų pateikimui keletu kalbų.

2.3.7. Animacija

Nesudėtingą animaciją *Windows Phone* platformoje galima realizuoti sukūrus vieną ar daugiau `Canvas` tipo objektų ir keičiant jų pozicijas ekrane. Kompiuterinių žaidimų kūrimui *Microsoft Inc.* kompanija yra sukūrusi *XNA* karkasą, kuris ne tik leidžia kurti sudėtingus trimačius žaidimus, bet ir integruoti į juos audio, video intarpus, pridėti žaidimo tinkle galimybes. Šis karkasas taip pat yra naudojamas kompiuterinių žaidimų kūrimui *XBox 360* platformoje.

2.3.8. Multimedija

Muzikai, garsams bei video medžiagai įrašyti bei atkurti *Windows Phone* turi `MediaElement` komponentą. Šios klasės palaikomi formatai pateikti lentelėje – Lent. 2.7[[12]].

Lent. 2.7 - *Windows Phone* MediaElement komponento palaikomi formatai

Tipas	Formatas, kodeksas	Įrašymas	Atkūrimas	Bylų plėtiniai
Garsas	PCM / WAVE	Yra	Yra	WAVE (.wav)

Tipas	Formatas, kodeksas	Įrašymas	Atkūrimas	Bylų plėtiniai
	Microsoft Windows Media Audio v7, v8 and v9.x Standard (WMA Standard)	Yra	Yra	WMA (.wma)
	Microsoft Windows Media Audio v9.x and v10 Professional (WMA Professional)	Yra	Yra	WMA (.wma)
	MP3	Yra	Yra	MP3 (.mp3)
	AAC	Yra	Yra	3GPP (.3gp) MPEG-4 (.mp4, .m4a)
	AMR-WB	Yra	Yra	3GPP (.3gp)
Video	Raw Video	Yra	Yra	
	RGBA	Yra	Yra	
	YV12 - YCrCb(4:2:0)	Yra	Yra	
	Windows Media Video	Yra	Yra	WMV1, WMV2, WMV3, WMVA (.wmv)
	H.263	Yra	Yra	3GPP (.3gp) MPEG-4 (.mp4)
	H.264	Yra	Yra	3GPP (.3gp) MPEG-4 (.mp4) MPEG-TS (.ts)
	MPEG-4	Yra	Yra	3GPP (.3gp)

2.3.9. HTML

Windows Phone turi `WebView` komponentą, kuris leidžia atvaizduoti *HTML* kalba parašytos programos turininį. Šis komponentas turi metodus `Navigate(Uri uri)` ir `NavigateToString(String html)`, kurie leidžia atvaizduoti turinį pagal jo *URL* adresą arba tiesiogiai perduodant *HTML* programos kodą. *HTML* kodas gali būti papildytas *JavaScript* bei *CSS* turiniu.

2.3.10. Žemėlapiai

Žemėlapių atvaizdavimui *Windows Phone* aplikacijose yra skirtas `Map` komponentas. Šis komponentas leidžia nustatyti žemėlapio koordinates, priartinimo lygį, įjungti arba išjungti valdymo mygtukus, 3D peržiūros režimą, pėsčiųjų žymas (perėjas, laiptus ir t.t.). Žemėlapis turi kelių, ortofoto, mišrų bei reljefo režimus. Norint naudoti žemėlapius savo aplikacijoje būtina užsiregistruoti *Microsoft Inc.* svetainėje ir gauti nemokamą aktyvavimo raktą.

2.3.11. Vartotojo buvimo vietos nustatymas

Vartotojo buvimo vietai nustatyti *Windows Phone* skirta `GeoCoordinateWatcher` klasė. Ši klasė leidžia nustatyti vartotojo buvimo vietą pagal ryšio tiekėjo tinklą arba pagal *GPS* tinklą. Programinės įrangos sąsaja leidžia nustatyti minimalų vartotojo buvimo vietos poslinkį metrais ir tik tuomet atnaujinti duomenis žemėlapyje. Tai padeda taupyti mobiliojo įrenginio akumuliatorių.

2.3.12. Kamera

Windows Phone turi komponentą kuris leidžia iškviesti integruotą fotografavimo ar filmavimo programėlę. Pasinaudojus šia programėle rezultatas apdorojimui grąžinamas į pagrindinę programą. Sisteminė programėlė jau turi visas pagrindines fotografavimo ar filmavimo funkcijas, tokias kaip fokusavimas ar priartinimas, tačiau toks sprendimas nepatogus kai norima sukurti individualią kameros valdymo sąsaja.

2.3.13. Sensoriai

Windows Phone SDK palaiko trijų tipų sensorius: akselerometrą, giroskopą bei kompasą. Norint gauti jų duomenis reikia sukurti `Accelerometer`, `Gyroscope` arba `Compass` klasės objektus. Įrenginiai su *Windows Phone* operacine sistema privalo turėti ir šviesumo bei atstumo iki objekto nustatymo sensorius, tačiau standartinės programinės įrangos kūrimo priemonės dar neturi sąsajų darbui su šiais sensoriais.

2.3.14. Tinklai

Šiuo metu beveik visi mobilieji įrenginiai turi prieigą prie interneto. *Windows Phone* programinės įrangos kūrimo paketas turi `URLConnection` klasę kurios dėka galima užmegzti ryšį su nutolusiu kompiuteriu naudojantis *FTP*, *HTTP* bei *HTTPS* protokolais.

2.3.15. Foniniai procesai

Foninius procesus *Windows Phone* sistemoje galima suskirstyti į dvi grupes: suplanuotos užduotys *(angl. scheduled tasks)* ir daug resursų reikalaujančios užduotys *(angl. resource intensive tasks)*.

2.3.15.1. Suplanuotos užduotys

Suplanuotos užduotys – tai periodinės, trumpos užduotys, kurias automatiškai paleidžia operacinė sistema. Dažniausiai šie procesai naudojami nedidelių duomenų sinchronizacijai. Užduotys negali trukti ilgiau nei 25 sekundes, priešingu atveju jos yra nutraukiamos. Minimalus užduočių pasileidimo intervalas – 30 minučių. Šie procesų paleidimas gali būti sustabdytas jeigu įrenginio akumuliatorius yra prie išsikrovimo ribos arba tokių foninių procesų yra daugiau kaip 6.

2.3.15.2. Daug resursų reikalaujančios užduotys

Resursų reikalaujančios užduotys – tai ilgai trunkančios arba daug procesoriaus, tinklo pralaidumo ar energijos reikalaujančios periodinės užduotys. Tokios užduotys yra paleidžiamos tik tuomet, kai vartotojas nesinaudoja mobiliuoju įrenginiu (įrenginio ekranas privalo būti užrakintas). Be to įrenginys privalo būti prijungtas prie išorinio energijos tiekimo tinklo arba akumuliatoriaus įkrovimas turi būti ne mažiau nei 90%. Užduotys negali trukti ilgiau nei 10 min., priešingu atveju jos yra nutraukiamos.

2.3.16. Duomenų apsikeitimas su kitomis programomis

Kiekviena *Windows Phone* programa pasileidžia izoliuotoje, tik tai aplikacijai prieinamoje aplinkoje ir nėra galimybių apsikeisti duomenimis su kitomis aplikacijomis (net jei duomenys saugomi integruotoje duomenų bazėje). *Windows Phone SDK* yra sąsajos gauti duomenis tik iš tam tikrų sisteminių programėlių (pvz. adresų knygutės).

2.3.17. Duomenų išsaugojimas

Duomenų išsaugojimui *Windows Phone* platformoje yra skirti šie komponentai:

- *IsolatedStorageSetings* – nesudėtingos, tik aplikacijai prieinamos informacijos išsaugojimas rakto-reikšmės tipo žemėlapiuose.
- *IsolatedStorage* – tik aplikacijai prieinamos informacijos išsaugojimas vidinėje ir išorinėje atmintyje (bylose).
- *SQL CE* reliacinė duomenų bazė – struktūrinės informacijos išsaugojimui.

Duomenis esančius *SQL CE* duomenų bazėje galima pasiekti naudojant specialią užklausų kalbą *LINQ* (*angl. Language Integrated Query*). Ši kalba taip pat yra naudojama ir duomenų bazės struktūrai kurti bei redaguoti. *SQL CE* duomenų bazė nepalaiko standartinių *SQL* užklausų.

2.4. Išvados

1. Visi programinės įrangos kūrimo paketai turi įrankius palengvinančius kurti programinę įrangą tai operacinei sistemai. *Android* programoms kurti yra skirtas *Android ADT Bundle, iOS XCode*, o *Windows Phone Visual Studio* įrankis. Visos programavimo aplinkos turi virtualius įrenginius, kuriuose galima išbandyti kuriamą programinę įrangą.
2. *Android* programos kuriamos *Java* programavimo kalba, *iOS Object-C*, o *Windows Phone* programas galima kurti *C#* arba *Visual Basic* kalbomis.
3. *Android* programinės įrangos kūrimo paketas neturi priemonių darbui su žemėlapiais. Šiam funkcionalumui įgyvendinti, *Android* kūrėjai rekomenduoja naudoti *Google Play Services SDK* bibliotekas.
4. *iOS* ir *Windows Phone* programinės įrangos kūrimo paketai turi labai griežtus reikalavimus foniniams procesams, dėl kurių gali nepavykti įgyvendinti visus norimus programos funkcinius bei nefunkcinius reikalavimus.

3. REIKALAVIMAI EKSPERIMENTINEI PROGRAMINEI ĮRANGAI

Šiame skyriuje bus pateikti reikalavimai eksperimentinei programinei įrangai. Funkciniai ir nefunkciniai reikalavimai bus parinkti taip, kad būtų galima įvertinti kiekvieną apsirašytą probleminę sritį. Kiekvienas iš funkcinių reikalavimų turės būti realizuotas atskira, pilnai funkcionuojančia programa. Visos programos bus realizuotos su visomis trimis platformomis.

Kadangi tam tikri vertinimo kriterijai kaip lankstumas, pernešamumas ar pakartotinis panaudojamumas negali būti įvertinti su tokiomis mažomis aplikacijomis, papildomai bus aprašyti reikalavimai didesnei, kompleksinei užduočiai.

3.1. Funkciniai reikalavimai

Programėlių funkciniai reikalavimai yra aprašyti lentelėje – Lent. 3.1. Viena programėlė įgyvendina lygiai vieną funkcinį reikalavimą.

Lent. 3.1 - Programėlių funkciniai reikalavimai

Reikalavimo numeris	Testuojama sritis	Aprašymas
FR-1-1	Grafinė vartotojo sąsaja	Sukurti formą, kurioje būtų vienas teksto atvaizdavimo laukas, vienas teksto įvedimo laukas bei mygtukas. Nuspaudus mygtuką vartotojo įvestas tekstas išvalomas.
FR-1-2	Meniu	Sukurti formą su vienu teksto atvaizdavimo lauku. Nuspaudęs meniu mygtuką vartotojui parodomas kontekstinis meniu iš dviejų variantų: „Tęsti“ ir „Baigti“. Pasirinkus „Tęsti“ grįžtama į pradinę būseną, pasirinkus „Baigti“ aplikacija išjungiama.
FR-1-3	Dialogai	Sukurti formą, kurioje būtų vienas teksto atvaizdavimo laukas, vienas teksto įvedimo laukas, bei mygtukas. Nuspaudus mygtuką vartotojo įvestas tekstas turi būti atvaizduojamas iššokančiame lange.
FR-1-4	Daugiakalbystė	Sukurti formą, kurioje būtų vienas teksto atvaizdavimo laukas su užrašu „Sveiki“. Užrašas turi būti pateiktas tokia kalba, kuri numatyta mobiliajame įrenginyje pagal nutylėjimą. Turi būti realizuotas 3 kalbos: lietuvių, anglų ir vokiečių. Jeigu nei viena iš šių kalbų nėra numatyta įrenginyje, turi būti pateiktas lietuviškas užrašas.
FR-1-5	Animacija	Sukurti kamuoliuką kuris juda ekrane ir atšoka nuo ekrano kraštų.
FR-1-6	Audio	Sukurti formą su vienu mygtuku. Nuspaudus mygtuką sugrojama muzikėlė.
FR-1-7	Video	Sukurti formą su vienu mygtuku. Nuspaudus mygtuką parodomas video klipas.
FR-1-8	HTML	Sukurti formą su vienu teksto įvedimo lauku bei mygtuku. Nuspaudus mygtuką atvaizduojama svetainė su įvestu adresu.
FR-1-9	Žemėlapiai, vartotojo buvimo vietos nustatymas	Sukurti formą, su žemėlapiu. Žemėlapis turi parodyti vartotojo buvimo vietą.
FR-1-10	Kamera	Sukurti formą su vienu mygtuku. Nuspaudus mygtuką nufotografuojamas ir atvaizduojamas kameros vaizdas.

Reikalavimo numeris	Testuojama sritis	Aprašymas
FR-1-11	Sensoriai	Sukurti formą su trimis laukais. Šie laukai turi parodyti ir nuolat atnaujinti mobiliojo įrenginio akselerometro parodymus.
FR-1-12	Tinklai	Sukurti formą, kurioje būtų vienas teksto atvaizdavimo laukas, vienas teksto įvedimo laukas, bei mygtukas. HTTP protokolu nurodžius svetainės adresą ir nuspaudus mygtuką atvaizduojamas svetainės *HTML* kodas.
FR-1-13	Foniniai procesai	Sukurti procesą, kuris kas 10 sekundžių parodytų vartotojui laiką. Aplikacija turi būti nematoma vartotojui. Taip pat turi būti priemonės leidžiančios įjungti ir išjungti šį procesą.
FR-1-14	Duomenų apsikeitimas	Sukurti formą su vienu teksto atvaizdavimo lauku ir mygtuku. Nuspaudus mygtuką į tekstinį lauką išvedami visi asmenys esantys mobiliojo įrenginio kontaktų sąraše.
FR-1-15	Duomenų išsaugojimas	Sukurti formą, kurioje būtų teksto įvedimo laukas, teksto atvaizdavimo laukas bei mygtukas. Nuspaudus mygtuką vartotojo įvestas tekstas išvalomas ir įrašomas į duomenų bazės lentelę, teksto atvaizdavimo laukas išvalomas ir atvaizduojamas visas duomenų bazės lentelės turinys. Aplikacija turi turėti priemones duomenų bazės sukūrimui jos pirmojo startavimo metu.

3.2. Nefunkciniai reikalavimai

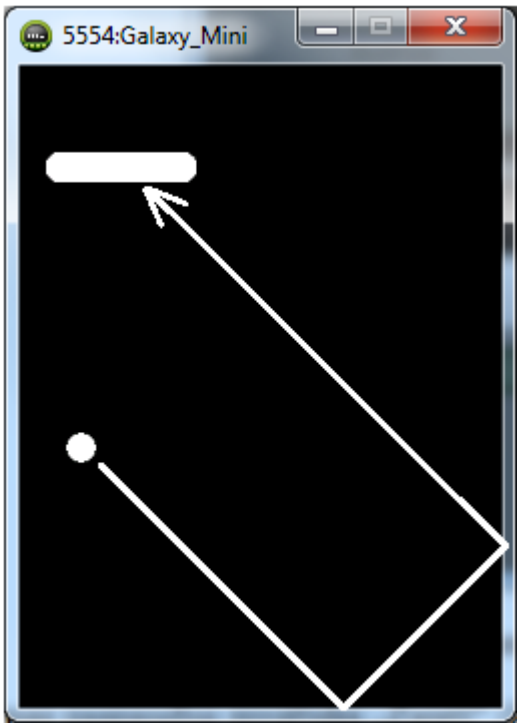
Programėlių nefunkciniai reikalavimai yra aprašyti lentelėje Lent. 3.2. Visos programėlės turi pilnai įgyvendinti visus nefunkcinius reikalavimus.

Lent. 3.2 - Programėlių nefunkciniai reikalavimai

Reikalavimo numeris	Aprašymas
NFR-1-1	Programos kodas turi remtis tik standartinėmis programinės įrangos kūrimo priemonės. Negalima naudoti jokių papildomų bibliotekų.
NFR-1-2	Programos kodas turi būti parašytas pagal 2.2.3 skyriuje aprašytus reikalavimus.

3.3. Kompleksinė užduotis

Kompleksine užduotim pasirinktas žaidimas. Žaidimo tikslas kuo daugiau kartų išmušinėti kamuoliuką naudojant specialų įrenginį – tiltelį. Kamuoliukas atšoka atsimušęs į apatinę ir šonines ekrano sieneles, o žaidėjui reikia apsaugoti viršutinę sieną. Tiltelis yra valdomas akselerometro pagalba. Už kiekvieną kamuoliuko atmušimą skiriamas vienas taškas ir kamuoliuko greitis padidėja. Grafinis tokio žaidimo prototipas yra pateikta paveikslėlyje – Pav. 3.1.

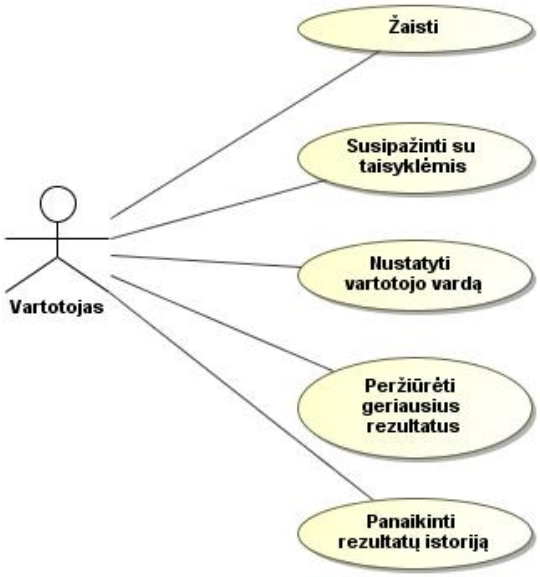


Pav. 3.1 - Žaidimo prototipas

Visos žaidimo funkcijos bus aprašytos panaudos atvejų diagrama ir detalizuotos funkciniuose bei nefunkciniuose reikalavimuose.

3.3.1. Panaudos atvejai

Žaidimo panaudos atvejai yra pavaizduoti paveikslėlyje – Pav. 3.2.



Pav. 3.2 - Panaudos atvejų diagrama

Įsijungęs žaidimą, pirmiausiai vartotojas turi nustatyti savo vardą. Kol nėra įvestas vartotojo vardas, programa neturi leisti pradėti žaidimo. Kad neerzintų vartotojo, programa privalo įsiminti vartotojo vardą ir įjungus žaidimą dar kartą nebeturi jo reikalauti. Žaidėjas taip pat gali susipažinti su taisyklėmis, peržiūrėti ar panaikinti geriausių rezultatų istoriją.

3.3.2. Funkciniai reikalavimai

Žaidimo funkciniai reikalavimai yra aprašyti lentelėje – Lent. 3.3.

Lent. 3.3 - Kompleksinės užduoties funkciniai reikalavimai

Reikalavimo numeris	Panaudos atvejis	Aprašymas
FR-2-1	Žaisti	Vartotojas akselerometro pagalba valdo tiltelį ir stengiasi atmušti nuo ekrano kraštų atšokantį kamuoliuką. Po kiekvieno atmušimo kamuoliuko greitis padidėja.
FR-2-2		Vartotojui praleidus kamuoliuką, žaidimas suskaičiuoja kiek kartų jis buvo atmuštas ir parodo vartotojui. Už kiekvieną atmušimą skiriamas vienas taškas. Rezultatas išsaugojamas žaidimo duomenų bazėje.
FR-2-3		Jeigu nėra nustatytas vartotojo vardas vartotojui parodomas iššokantis pranešimas „Privalote įvesti savo vardą“.
FR-2-4	Susipažinti su taisyklėmis	Žaidimo lange pateikiamos žaidimo taisyklės.
FR-2-5	Nustatyti vartotojo vardą	Žaidimo lange pateikiama forma su užrašu „Vartotojo vardas“, teksto įvedimo lauku ir mygtuku „Išsaugoti“. Pagal nutylėjimą teksto įvedimo lauke parodomas šiuo metu nustatytas vartotojo vardas. Jeigu vartotojas nėra įvedęs jokio vardo, sistema turi parodyti pagalbos tekstą „Įveskite savo vardą“.
FR-2-6	Peržiūrėti geriausius rezultatus	Vartotojui išvedamas dešimties daugiausiai taškų surinkusių žaidėjų sąrašas ir jų rezultatai.
FR-2-7	Panaikinti rezultatų istorija	Geriausių rezultatų lange turi būti mygtukas „Ištrinti“ leidžiantis vartotojui panaikinti rezultatų istoriją.
FR-2-8		Mygtukas leidžiantis panaikinti rezultatų istoriją rodomas tik tada, jeigu yra išsaugotas bent vienas rezultatas.

3.3.3. Nefunkciniai reikalavimai

Žaidimo nefunkciniai reikalavimai yra aprašyti lentelėje – Lent. 3.4.

Lent. 3.4 - Kompleksinės užduoties nefunkciniai reikalavimai

Reikalavimo numeris	Aprašymas
NFR-2-1	Programos kodas turi remtis tik standartinėmis programinės įrangos kūrimo priemonės (*angl. SDK*). Negalima naudoti jokių papildomų bibliotekų.
NFR-2-2	Programos kodas turi būti parašytas pagal 2.2.3 skyriuje aprašytus reikalavimus.

3.4. Reikalavimų analizės išvados

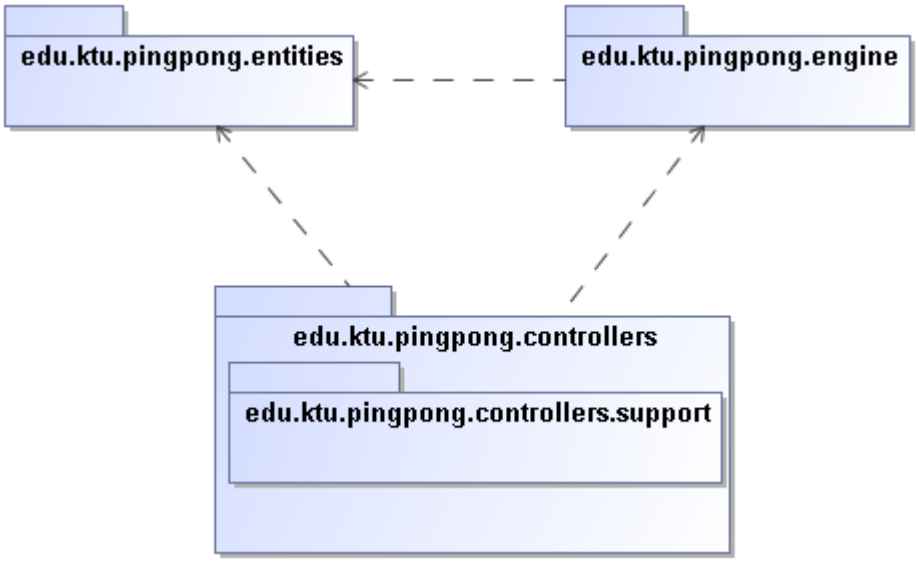
Šiame skyriuje buvo aprašyti reikalavimai eksperimentiniai programinei įrangai. Suformuluoti funkciniai ir nefunkciniai reikalavimai leidžia pilnai padengti visas nagrinėjamas problemines sritis ir įvertinti jas pagal apsibrėžtus vertinimo kriterijus.

4. EKSPERIMENTINĖS PROGRAMINĖS ĮRANGOS PROJEKTAVIMO MODELIS

Šiame skyriuje bus aprašytas kompleksinės užduoties projektavimo modelis, kurį sudaro paketų struktūra, klasių diagrama bei duomenų bazės projektas. Visoms trims programavimo platformoms pritaikius tą pačią architektūrą, nebus sukurtos sąlygos kažkuriai iš jų pasiekti geresnį rezultatą dėl geresnio sistemos suprojektavimo.

4.1. Paketų struktūra

Kompleksinės užduoties paketų diagrama pavaizduota paveikslėlyje – Pav. 4.1.



Pav. 4.1 - Kompleksinės užduoties paketų diagrama

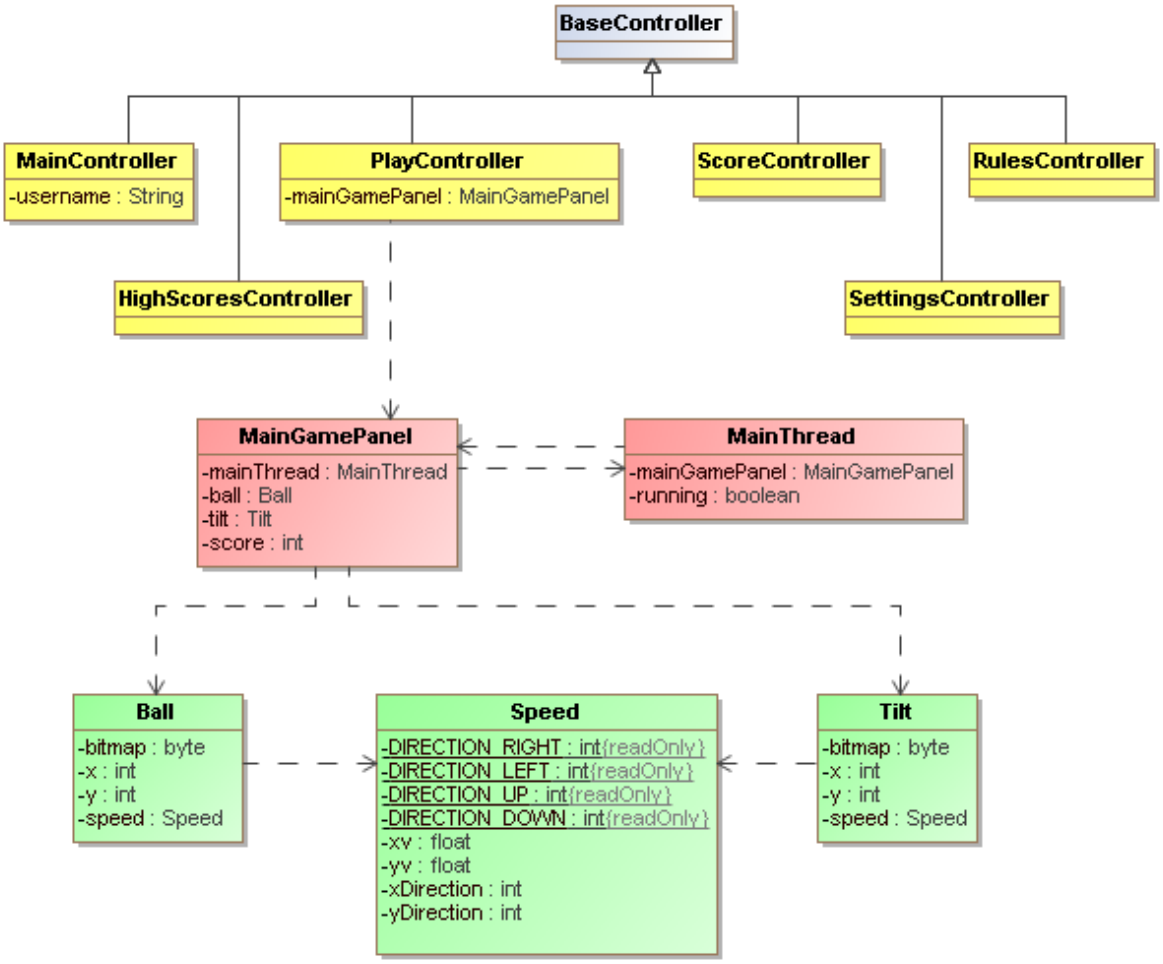
Šių paketų paskirtis aprašyta lentelėje – Lent. 4.1.

Lent. 4.1 - Kompleksinės užduoties paketų paskirtis

Paketo pavadinimas	Paskirtis
`edu.ktu.pingpong.entities`	Žaidime naudojamos esybės.
`edu.ktu.pingpong.engine`	Žaidimo logikos klasės.
`edu.ktu.pingpong.controllers`	Grafinės vartotojo sąsajos valdymo klasės.
`edu.ktu.pingpong.controllers.support`	Bazinės (abstrakčios) grafinės vartotojo sąsajos valdymo klasės.

4.2. Klasių diagrama

Kompleksinės užduoties klasių diagrama pavaizduota paveikslėlyje – Pav. 4.2.



Pav. 4.2 - Kompleksinės užduoties klasių diagrama

Šių klasių paskirtis aprašyta lentelėje – Lent. 4.2.

Lent. 4.2 - Kompleksinės užduoties klasių paskirtis

Klasės pavadinimas	Paskirtis
`BaseController`	Bazinė (abstrakti) grafinės vartotojo sąsajos valdymo klasė kurią paveldi visos sąsajos valdymo klasės.
`MainController`	Pagrindinio žaidimo meniu lango valdymo klasė.
`PlayController`	Žaidimo lango valdymo klasė.
`ScoreController`	Rezultato atvaizdavimo lango valdymo klasė.
`RulesController`	Taisyklių lango valdymo klasė.
`HighScoresController`	Geriausių rezultatų lango valdymo klasė.
`SettingsController`	Nustatymų lango valdymo klasė.
`MainGamePanel`	Žaidimo langas kuriame atvaizduojami visi grafiniai objektai.
`MainThread`	Gija, kuri suka amžiną ciklą, kol vyksta žaidimas.
`Ball`	Kamuoliuko esybė.
`Tilt`	Tiltelio esybė.
`Speed`	Greičio esybė.

4.3. Duomenų bazės projektas

Duomenų bazė bus sudaryta iš vienos lentelės "SCORES", kurioje bus saugomi rezultatai. Duomenų lentelės struktūra pateikta lentelėje – Lent. 4.3.

Lent. 4.3 - Duomenų bazės lentelės "SCORES" struktūra

Raktas	Stulpelio pavadinimas	Tipas	Apribojimai	Komentaras
PK	SCR_ID	NUMBER(10,0)	NOT NULL	Įrašo identifikatorius
	SCR_NAME	VARCHAR	NOT NULL	Žaidėjo vardas
	SCR_SCORE	NUMBER(5,0)	NOT NULL	Rezultatas
	SCR_DATE	DATE	NOT NULL	Data ir laikas

4.4. Programinės įrangos projektavimo išvados

Šiame skyriuje buvo aprašytas kompleksinės užduoties projektavimo modelis. Šis modelis leidžia pilnai įgyvendinti visus apsibrėžtus eksperimentinės programinės įrangos funkcinius reikalavimus.

5. EKSPERIMENTINĖS PROGRAMINĖS ĮRANGOS REALIZACIJOS MODELIS

Šiame skyriuje bus pateikti programų, kurios įgyvendina apsibrėžtus funkcinius reikalavimus, išeities kodai. Bus pateikiamas ir vertinamas tik tas kodas, kuris tiesiogiai įgyvendina funkcinį reikalavimą. Grafinę vartotojo sąsają įgyvendinantis kodas nebus pateikiamas ir vertinamas (išskyrus reikalavimus, kurie tiesiogiai vertina grafinę vartotojo sąsają). Taip pat nebus pateikiamas ir vertinamas kodas kurį sugeneruoja programavimo įrankiai (pvz. kodas kuris importuoja bibliotekas, paketų, vardų sričių apibrėžimai ir t.t.). Pilni visų programų išeities kodai yra pateikiami kompaktiniame diske.

5.1. FR-1-1

FR-1-1 reikalavimo aprašymas pateiktas lentelėje – Lent. 5.1.

Lent. 5.1 - FR-1-1 reikalavimo aprašymas

Reikalavimo numeris	Testuojama sritis	Aprašymas
FR-1-1	Grafinė vartotojo sąsaja	Sukurti formą, kurioje būtų vienas teksto atvaizdavimo laukas, vienas teksto įvedimo laukas bei mygtukas. Nuspaudus mygtuką vartotojo įvestas tekstas išvalomas.

5.1.1. Android

Android priemonėmis realizuotas reikalavimas:

```
<LinearLayout android:layout_width="match_parent"
              android:layout_height="match_parent"
              android:orientation="vertical"
              tools:context=".MainActivity" >
    <TextView android:id="@+id/textView"
              android:layout_width="wrap_content"
              android:layout height="wrap content"
              android:layout margin="10dp"
              android:text="Jūsų vardas:" />
    <EditText android:id="@+id/editText"
              android:layout_width="match_parent"
              android:layout_height="wrap_content"
              android:layout_margin="10dp">
        <requestFocus />
    </EditText>
    <Button android:id="@+id/button"
            style="android:buttonStyle"
            android:layout width="wrap content"
            android:layout height="wrap content"
            android:layout gravity="center horizontal"
            android:text="Išvalyti" />
</LinearLayout>
```

layout\main.xml

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new ButtonListener());
    }

    class ButtonListener implements OnClickListener {

        @Override
        public void onClick(View view) {
            TextView textView = (TextView) findViewById(R.id.editText);
            textView.setText("");
        }
    }
}
```

MainActivity.java

5.1.2. iOS

Grafinė vartotojo sąsaja *iOS* operacinėje sistemoje kuriama su *Interface Builder* įrankiu. Programos kodas, kuris apdoroja grafinės vartotojo sąsajos įvykius:

```
@interface ViewController : UIViewController

@property (strong, nonatomic) IBOutlet UITextField *textField;

- (IBAction)buttonPressed:(id)sender;

@end
```

ViewController.h

```
@implementation ViewController

@synthesize textField;

- (IBAction)buttonPressed:(id)sender {
    textField.text = @"";
    [textField resignFirstResponder];
}

- (BOOL)textFieldShouldReturn:(UITextField *) textField {
    [self.textField resignFirstResponder];
    return YES;
}

- (void)viewDidUnload {
    [self setTextField:nil];
    [super viewDidUnload];
}

@end
```

ViewController.m

5.1.3. Windows Phone

Windows Phone priemonėmis realizuotas reikalavimas:

```
<phone:PhoneApplicationPage>
    <Grid x:Name="LayoutRoot"
          Background="Transparent">
        <Grid x:Name="ContentPanel"
              Grid.Row="1"
              Margin="12,0,12,0">
            <TextBlock Name="textBlock"
                       Text="Jūsų vardas:"
                       VerticalAlignment="Top" />
            <TextBox Margin="0,30,0,0"
                     Name="textBox"
                     VerticalAlignment="Top" />
            <Button Content="Išvalyti"
                    Margin="0,110,0,0"
                    Name="button"
                    VerticalAlignment="Top"
                    Click="buttonClick" />
        </Grid>
    </Grid>
</phone:PhoneApplicationPage>
```

MainPage.xaml

```
namespace MasterWork {

    public partial class MainPage : PhoneApplicationPage {

        public MainPage() {
            InitializeComponent();
        }

        private void buttonClick(object sender, RoutedEventArgs e) {
            textBox.Text = "";
        }
    }
}
```

MainPage.xaml.cs

5.2. FR-1-2

FR-1-2 reikalavimo aprašymas pateiktas lentelėje – Lent. 5.2.

Lent. 5.2 - FR-1-2 reikalavimo aprašymas

Reikalavimo numeris	Testuojama sritis	Aprašymas
FR-1-2	Meniu	Sukurti formą su vienu teksto atvaizdavimo lauku. Nuspaudęs meniu mygtuką vartotojui parodomas kontekstinis meniu iš dviejų variantų: „Tęsti“ ir „Baigti“. Pasirinkus „Tęsti“ grįžtama į pradinę būseną, pasirinkus „Baigti“ aplikacija išjungiama.

5.2.1. Android

Android priemonėmis realizuotas reikalavimas:

```
<menu>
    <item android:id="@+id/item1"
          android:title="Tęsti"
          android:icon="@android:drawable/ic_menu_revert" />
    <item android:id="@+id/item2"
          android:title="Baigti"
          android:icon="@android:drawable/ic_menu_close_clear_cancel" />
</menu>
```

menu\menu.xml

```
public class MainActivity extends Activity {

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

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        if (item.getItemId() == R.id.item2) {
            this.finish();
        }
        return super.onOptionsItemSelected(item);
    }
}
```

MainActivity.java

5.2.2. iOS

iOS techninė įranga neturi dedikuoto meniu mygtuko, todėl nėra galimybės įgyvendinti šį reikalavimą.

5.2.3. Windows Phone

Windows Phone priemonėmis realizuotas reikalavimas:

```
<phone:PhoneApplicationPage>
    <phone:PhoneApplicationPage.ApplicationBar>
        <shell:ApplicationBar IsMenuEnabled="True"
                              IsVisible="True">
            <shell:ApplicationBar.MenuItems>
                <shell:ApplicationBarMenuItem Text="Tęsti" />
                <shell:ApplicationBarMenuItem Text="Baigti"
                                              Click="item2Click" />
            </shell:ApplicationBar.MenuItems>
```

```
        </shell:ApplicationBar>
    </phone:PhoneApplicationPage.ApplicationBar>
</phone:PhoneApplicationPage>
```

MainPage.xaml

```
namespace MasterWork {

    public partial class MainPage : PhoneApplicationPage {

        public MainPage() {
            InitializeComponent();
        }

        private void item2Click(object sender, EventArgs e) {
            throw new Exception("Exit");
        }
    }
}
```

MainPage.xaml.cs

5.3. FR-1-3

FR-1-3 reikalavimo aprašymas pateiktas lentelėje – Lent. 5.3.

Lent. 5.3 - FR-1-3 reikalavimo aprašymas

Reikalavimo numeris	Testuojama sritis	Aprašymas
FR-1-3	Dialogai	Sukurti formą, kurioje būtų vienas teksto atvaizdavimo laukas, vienas teksto įvedimo laukas, bei mygtukas. Nuspaudus mygtuką vartotojo įvestas tekstas turi būti atvaizduojamas iššokančiame lange.

5.3.1. Android

Android priemonėmis realizuotas reikalavimas:

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new ButtonListener());
    }

    class ButtonListener implements OnClickListener {

        @Override
        public void onClick(View view) {
            TextView textView = (TextView) findViewById(R.id.editText);
            AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
            builder.setMessage(textView.getText());
            builder.setNegativeButton("OK", new NegativeButtonListener());
            AlertDialog alertDialog = builder.create();
            alertDialog.show();
        }

        class NegativeButtonListener implements DialogInterface.OnClickListener {

            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        }
    }
}
```

MainActivity.java

5.3.2. iOS

iOS priemonėmis realizuotas reikalavimas:

```
@interface ViewController : UIViewController

@property (strong, nonatomic) IBOutlet UITextField *textField;

- (IBAction)buttonPressed:(id)sender;

@end
```

ViewController.h

```
@implementation ViewController

@synthesize textField;

- (IBAction)buttonPressed:(id)sender {
    UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"Pranešimas" message:textField.text
                          delegate:nil cancelButtonTitle:@"OK" otherButtonTitles:nil, nil];
    [alert show];
    [textField resignFirstResponder];
}

- (BOOL)textFieldShouldReturn:(UITextField *) textField {
    [self.textField resignFirstResponder];
    return YES;
}

- (void)viewDidUnload {
    [self setTextField:nil];
    [super viewDidUnload];
}

@end
```

ViewController.m

5.3.3. Windows Phone

Windows Phone priemonėmis realizuotas reikalavimas:

```
<UserControl d:DesignHeight="480"
             d:DesignWidth="480">
    <StackPanel x:Name="LayoutRoot"
                Width="480"
                Height="120"
                Background="LightBlue">
        <TextBlock Height="30"
                   Name="popUpTextBlock"
                   Margin="10" />
        <Button x:Name="popUpButtonOK"
                Content="OK" />
    </StackPanel>
</UserControl>
```

PopUpControl.xaml

```
namespace MasterWork {

    public partial class MainPage : PhoneApplicationPage {

        public MainPage() {
            InitializeComponent();
        }

        private void buttonClick(object sender, RoutedEventArgs e) {
            PopUpControl control = new PopUpControl();
            control.popUpTextBlock.Text = textBox.Text;

            Popup popup = new Popup();
            popup.VerticalOffset = 100;
            popup.Child = control;
            popup.IsOpen = true;

            control.popUpButtonOK.Click += (s, args) => {
                popup.IsOpen = false;
            };
        }
    }
}
```

MainPage.xaml.cs

5.4. FR-1-4

FR-1-4 reikalavimo aprašymas pateiktas lentelėje – Lent. 5.4.

Lent. 5.4 - FR-1-4 reikalavimo aprašymas

Reikalavimo numeris	Testuojama sritis	Aprašymas
FR-1-4	Daugiakalbystė	Sukurti formą, kurioje būtų vienas teksto atvaizdavimo laukas su užrašu „Sveiki“. Užrašas turi būti pateiktas tokia kalba, kuri numatyta mobiliajame įrenginyje pagal nutylėjimą. Turi būti realizuotas 3 kalbos: lietuvių, anglų ir vokiečių. Jeigu nei viena iš šių kalbų nėra numatyta įrenginyje, turi būti pateiktas lietuviškas užrašas.

5.4.1. Android

Android priemonėmis realizuotas reikalavimas:

```
<TextView android:id="@+id/textView"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_margin="10dp"
          android:text="@string/text" />
```

layout\main.xml

```
<resources>
    <string name="appName">Magistro darbas</string>
    <string name="text">Sveiki</string>
</resources>
```

values\strings.xml

```
<resources>
    <string name="appName">Master work</string>
    <string name="text">Hello</string>
</resources>
```

values-en\strings.xml

```
<resources>
    <string name="appName">Master thesis</string>
    <string name="text">Hallo</string>
</resources>
```

values-de\strings.xml

```
public class MainActivity extends Activity {

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

MainActivity.java

5.4.2. iOS

Naudojantis *Interface Builder* įrankiu kiekvienai kalbai sukuriamas atskiras grafinės vartotojo sąsajos langas. Programos kodas, kuris apdoroja grafinės vartotojo sąsajos įvykius:

```
@interface ViewController : UIViewController

@property (strong, nonatomic) IBOutlet UILabel *label;

@end
```

ViewController.h

```
@implementation ViewController

@synthesize label;
```

```
- (void)viewDidUnload {
    [self setLabel:nil];
    [super viewDidUnload];
}

@end
```

ViewController.m

5.4.3. Windows Phone

Windows Phone priemonėmis realizuotas reikalavimas:

```
<TextBlock Name="textBlock"
           Text="{Binding Path=Resources.text, Source={StaticResource LStrings}}"
           VerticalAlignment="Top" />
```

MainPage.xaml

```
namespace MasterWork {

    public class LStrings {

        private static Resources resources = new Resources();

        public Resources Resources {
            get {
                return resources;
            }
        }
    }
}
```

LStrings.cs

```
<Application>
    <Application.Resources>
        <local:LStrings xmlns:local="clr-namespace:MasterWork"
                        x:Key="LStrings" />
    </Application.Resources>
</Application>
```

App.xaml

Taip pat reikia skurti kalbinių resursų bykas: Resources.resx, Resources.en-US.resx ir Resources.de-DE.resx, kuriose kalbiniai resursai pildomi *Microsoft Visual Studio 2010* priemonėmis.

5.5. FR-1-5

FR-1-5 reikalavimo aprašymas pateiktas lentelėje – Lent. 5.5.

Lent. 5.5 - FR-1-5 reikalavimo aprašymas

Reikalavimo numeris	Testuojama sritis	Aprašymas
FR-1-5	Animacija	Sukurti kamuoliuką kuris juda ekrane ir atšoka nuo ekrano kraštų.

5.5.1. Android

Android priemonėmis realizuotas reikalavimas:

```
<edu.ktu.informatics.AnimatedView android:id="@+id/anim_view"
                                  android:layout_height="fill_parent"
                                  android:layout_width="fill_parent" />
```

layout\main.xml

```
public class MainActivity extends Activity {

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

MainActivity.java

```
public class AnimatedView extends ImageView {

    private Context context;
    private int x = 100;
    private int y = 100;
    private int xDirection = 1;
    private int yDirection = 1;
    private Handler handler;

    public AnimatedView(Context context, AttributeSet attrs) {
        super(context, attrs);
        this.context = context;
        handler = new Handler();
    }

    private Runnable r = new Runnable() {
        @Override
        public void run() {
            invalidate();
        }
    };

    protected void onDraw(Canvas c) {
        BitmapDrawable b = (BitmapDrawable) context.getResources().getDrawable(R.drawable.b);

        x += xDirection;
        y += yDirection;

        if (x > this.getWidth() - b.getBitmap().getWidth()) {
            xDirection = -1;
        }

        if (x < 0) {
            xDirection = 1;
        }

        if (y > this.getHeight() - b.getBitmap().getHeight()) {
            yDirection = -1;
        }

        if (y < 0) {
            yDirection = 1;
        }

        c.drawBitmap(b.getBitmap(), x, y, null);
        handler.postDelayed(r, 30);
    }
}
```

MainThread.java

5.5.2. iOS

Naudojantis *Interface Builder* įrankiu sukuriamas kamuoliukas. Programos kodas, kuris apdoroja programos gyvavimo ciklą:

```
@interface ViewController : UIViewController {
    CGPoint pos;
}

@property (strong, nonatomic) IBOutlet UIImageView *ball;

@end
```

ViewController.h

```
@implementation ViewController

@synthesize ball;

- (void)onTimer {
    if (ball.center.x > 304 || ball.center.x < 16) {
        pos.x = -pos.x;
    }
    if (ball.center.y > 444 || ball.center.y < 16) {
        pos.y = -pos.y;
    }
    ball.center = CGPointMake(ball.center.x + pos.x, ball.center.y + pos.y);
}
```

```
- (void)viewDidLoad {
    [super viewDidLoad];
    [NSTimer scheduledTimerWithTimeInterval:0.05 target:self
             selector:@selector(onTimer) userInfo:nil repeats:YES];
    pos = CGPointMake(2.0, 2.0);
}

- (void)viewDidUnload {
    [self setBall:nil];
    [super viewDidUnload];
}

@end
```

ViewController.m

5.5.3. Windows Phone

Windows Phone priemonėmis realizuotas reikalavimas:

```
<Canvas x:Name="ContentGrid"
        Width="410"
        Height="560"
        HorizontalAlignment="Left">
    <Ellipse x:Name="el"
             Canvas.Left="200"
             Canvas.Top="220"
             Fill="#FF963C3C"
             Height="50"
             Width="50" />
</Canvas>
```

MainPage.xaml

```
namespace MasterWork {

    public partial class MainPage : PhoneApplicationPage {

        private int x = 1;
        private int y = 1;

        public MainPage() {
            InitializeComponent();
            CompositionTarget.Rendering += new EventHandler(AnimateRectangle);
        }

        void AnimateRectangle(object sender, EventArgs e) {
            if ((double) el.GetValue(Canvas.LeftProperty) >= ContentGrid.Width) {
                x = -1;
            }

            if ((double) el.GetValue(Canvas.LeftProperty) <= 0) {
                x = 1;
            }

            if ((double) el.GetValue(Canvas.TopProperty) >= ContentGrid.Height) {
                y = -1;
            }

            if ((double) el.GetValue(Canvas.TopProperty) <= 0) {
                y = 1;
            }

            el.SetValue(Canvas.LeftProperty, (double) el.GetValue(Canvas.LeftProperty) + x);
            el.SetValue(Canvas.TopProperty, (double) el.GetValue(Canvas.TopProperty) + y);
        }
    }
}
```

MainPage.xaml.cs

5.6. FR-1-6

FR-1-6 reikalavimo aprašymas pateiktas lentelėje – Lent. 5.6.

Lent. 5.6 - FR-1-6 reikalavimo aprašymas

Reikalavimo numeris	Testuojama sritis	Aprašymas
FR-1-6	Audio	Sukurti formą su vienu mygtuku. Nuspaudus mygtuką sugrojama muzikėlė.

5.6.1. Android

Android priemonėmis realizuotas reikalavimas:

```
public class MainActivity extends Activity {

    private MediaPlayer mediaPlayer;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        mediaPlayer = MediaPlayer.create(MainActivity.this, R.raw.audio);
        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new ButtonListener());
    }

    @Override
    protected void onStop() {
        mediaPlayer.release();
        super.onStop();
    }

    class ButtonListener implements OnClickListener {

        @Override
        public void onClick(View view) {
            mediaPlayer.start();
        }
    }
}
```

MainActivity.java

5.6.2. iOS

iOS priemonėmis realizuotas reikalavimas:

```
@interface ViewController : UIViewController <AVAudioPlayerDelegate>

@property (nonatomic, retain) AVAudioPlayer *player;

- (IBAction)buttonPressed:(id)sender;

@end
```

ViewController.h

```
@implementation ViewController

@synthesize player;

- (IBAction)buttonPressed:(id)sender {
    [self.player play];
}

- (void)viewDidLoad {
    [super viewDidLoad];

    NSString *path = [[NSBundle mainBundle] pathForResource:@"audio" ofType:@"mp3"];
    NSURL *file = [NSURL fileURLWithPath:path];
    AVAudioPlayer *p = [[AVAudioPlayer alloc] initWithContentsOfURL:file error:nil];
    self.player = p;

    [player prepareToPlay];
    [player setDelegate:self];
}

@end
```

ViewController.m

5.6.3. Windows Phone

Windows Phone priemonėmis realizuotas reikalavimas:

```
<MediaElement x:Name="soundPlayer"
              Source="audio.mp3"
              AutoPlay="False" />
```

MainPage.xaml

```
namespace MasterWork {

    public partial class MainPage : PhoneApplicationPage {

        public MainPage() {
            InitializeComponent();
        }

        private void buttonClick(object sender, RoutedEventArgs e) {
            soundPlayer.Play();
        }
    }
}
```

MainPage.xaml.cs

5.7. FR-1-7

FR-1-7 reikalavimo aprašymas pateiktas lentelėje – Lent. 5.7.

Lent. 5.7 - FR-1-7 reikalavimo aprašymas

Reikalavimo numeris	Testuojama sritis	Aprašymas
FR-1-7	Video	Sukurti formą su vienu mygtuku. Nuspaudus mygtuką parodomas video klipas.

5.7.1. Android

Android priemonėmis realizuotas reikalavimas:

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new ButtonListener());
    }

    class ButtonListener implements OnClickListener {

        @Override
        public void onClick(View view) {
            VideoView v = (VideoView) findViewById(R.id.videoView);
            v.setVideoURI(Uri.parse("android.resource://" + getPackageName() + R.raw.video));
            v.setMediaController(new MediaController(MainActivity.this));
            v.start();
            v.requestFocus();
        }
    }
}
```

MainActivity.java

5.7.2. iOS

iOS priemonėmis realizuotas reikalavimas:

```
@interface ViewController : UIViewController  {
    MPMoviePlayerController *moviePlayerController;
}

@property (strong, nonatomic) IBOutlet UIButton *videoFrame;

-(IBAction)buttonPressed:(id)sender;
```

```
@end
```

ViewController.h

```
@implementation ViewController

@synthesize videoFrame;

- (IBAction)buttonPressed:(id)sender {
    NSString *path = [[NSBundle mainBundle] pathForResource:@"video" ofType:@"m4v"];
    NSURL *file = [NSURL fileURLWithPath:path];
    moviePlayerController = [[MPMoviePlayerController alloc] initWithContentURL:file];

    [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(moviePlaybackComplete:)
                    name:MPMoviePlayerPlaybackDidFinishNotification object:moviePlayerController];
    [moviePlayerController.view setFrame:CGRectMake(videoFrame.frame.origin.x,
             videoFrame.frame.origin.y, videoFrame.frame.size.width, videoFrame.frame.size.height)];
    [self.view addSubview:moviePlayerController.view];
    [moviePlayerController prepareToPlay];
    [moviePlayerController play];
}

- (void)moviePlaybackComplete:(NSNotification *)notification {
    [[NSNotificationCenter defaultCenter] removeObserver:self
                    name:MPMoviePlayerPlaybackDidFinishNotification object:moviePlayerController];
    [moviePlayerController.view removeFromSuperview];
}

- (void)viewDidUnload {
    [self setVideoFrame:nil];
    [super viewDidUnload];
}

@end
```

ViewController.m

5.7.3. Windows Phone

Windows Phone priemonėmis realizuotas reikalavimas:

```
<MediaElement x:Name="videoPlayer"
              Margin="0,100,0,0"
              Source="video.mp4"
              AutoPlay="False" />
```

MainPage.xaml

```
namespace MasterWork {

    public partial class MainPage : PhoneApplicationPage {

        public MainPage() {
            InitializeComponent();
        }

        private void buttonClick(object sender, RoutedEventArgs e) {
            soundPlayer.Play();
        }
    }
}
```

MainPage.xaml.cs

5.8. FR-1-8

FR-1-8 reikalavimo aprašymas pateiktas lentelėje – Lent. 5.8.

Lent. 5.8 - FR-1-8 reikalavimo aprašymas

Reikalavimo numeris	Testuojama sritis	Aprašymas
FR-1-8	HTML	Sukurti formą su vienu teksto įvedimo lauku bei mygtuku. Nuspaudus mygtuką atvaizduojama svetainė su įvestu adresu.

5.8.1. Android

Android priemonėmis realizuotas reikalavimas:

```
<WebView android:id="@+id/webView"
         android:layout width="fill parent"
         android:layout_height="fill_parent" />
```

layout\main.xml

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new ButtonListener());
    }

    class ButtonListener implements OnClickListener {

        @Override
        public void onClick(View view) {
            TextView textView = (TextView) findViewById(R.id.editText);
            WebView webView = (WebView) findViewById(R.id.webView);
            webView.getSettings().setJavaScriptEnabled(true);
            webView.loadUrl(textView.getText().toString());
        }
    }
}
```

MainActivity.java

5.8.2. iOS

iOS priemonėmis realizuotas reikalavimas:

```
@interface ViewController : UIViewController

@property (strong, nonatomic) IBOutlet UITextField *textField;
@property (strong, nonatomic) IBOutlet UIWebView *webView;

- (IBAction)buttonPressed:(id)sender;

@end
```

ViewController.h

```
@implementation ViewController

@synthesize textField;
@synthesize webView;

- (IBAction)buttonPressed:(id)sender {
    NSURL *url = [NSURL URLWithString:textField.text];
    NSURLRequest *request = [NSURLRequest requestWithURL:url];
    [webView loadRequest:request];
    [textField resignFirstResponder];
}

- (BOOL) textFieldShouldReturn:(UITextField *) textField {
    [self.textField resignFirstResponder];
    return YES;
}

- (void)viewDidUnload {
    [self setTextField:nil];
    [self setWebView:nil];
    [super viewDidUnload];
}

@end
```

ViewController.m

5.8.3. Windows Phone

Windows Phone priemonėmis realizuotas reikalavimas:

```
<phone:WebBrowser Name="webBrowser"
```

```
                Margin="0,150,0,0"
                VerticalAlignment="Stretch"
                HorizontalAlignment="Stretch" />
```

MainPage.xaml

```
namespace MasterWork {

    public partial class MainPage : PhoneApplicationPage {

        public MainPage() {
            InitializeComponent();
        }

        private void buttonClick(object sender, RoutedEventArgs e) {
            string site = textBox.Text;
            webBrowser.Navigate(new Uri(site, UriKind.Absolute));
        }
    }
}
```

MainPage.xaml.cs

5.9. FR-1-9

FR-1-9 reikalavimo aprašymas pateiktas lentelėje – Lent. 5.9.

Lent. 5.9 - FR-1-9 reikalavimo aprašymas

Reikalavimo numeris	Testuojama sritis	Aprašymas
FR-1-9	Žemėlapiai, vartotojo buvimo vietos nustatymas	Sukurti formą, su žemėlapiu. Žemėlapis turi parodyti vartotojo buvimo vietą.

5.9.1. Android

Android standartines programinės įrangos kūrimo priemones (*angl. SDK*) neturi priemonių darbui su žemėlapiais. Norint dirbti su žemėlapiais būtina įgiegti Google Play Services bibliotekas. Kadangi papildomų bibliotekų naudojimą draudžia NFR-2-1 nefunkcinis reikalavimas, bus realizuota tik dalis šio funkcinio reikalavimo: nustatyta vartotojo buvimo vieta.

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        LocationManager m = (LocationManager) this.getSystemService(Context.LOCATION_SERVICE);
        LocationListener listener = new UserLocationListener();
        m.requestLocationUpdates(LocationManager.GPS PROVIDER, 0, 0, listener);
    }

    class UserLocationListener implements LocationListener {

        @Override
        public void onStatusChanged(String provider, int status, Bundle extras) {
        }

        @Override
        public void onProviderEnabled(String provider) {
        }

        @Override
        public void onProviderDisabled(String provider) {
        }

        @Override
        public void onLocationChanged(Location location) {
            TextView textViewLatitude = (TextView) findViewById(R.id.textViewLatitude);
            TextView textViewLongitude = (TextView) findViewById(R.id.textViewLongitude);
            textViewLatitude.setText("Platuma: " + location.getLatitude());
            textViewLongitude.setText("Ilguma: " + location.getLongitude());
        }
    }
}
```

MainActivity.java

5.9.2. iOS

iOS priemonėmis realizuotas reikalavimas:

```
@interface ViewController : UIViewController <CLLocationManagerDelegate, MKMapViewDelegate> {
    CLLocationManager *locationManager;
    IBOutlet MKMapView *mapView;
}

@property (nonatomic, retain) IBOutlet MKMapView *mapView;
@property (nonatomic, retain) IBOutlet UIWindow *window;

@end
```

ViewController.h

```
@implementation ViewController

@synthesize mapView;
@synthesize window;

- (void)mapView:(MKMapView *)mapView didUpdateUserLocation:(MKUserLocation *)userLocation {
    self.mapView.centerCoordinate = userLocation.location.coordinate;
}

- (void)mapView:(MKMapView*)mv didAddAnnotationViews:(NSArray *)views {
    MKAnnotationView *annotationView = [views objectAtIndex:0];
    id<MKAnnotation> mp = [annotationView annotation];
    MKCoordinateRegion region = MKCoordinateRegionMakeWithDistance([mp coordinate], 250, 250);
    [mv setRegion:region animated:YES];
}

- (void)viewDidLoad {
    [super viewDidLoad];
    self.mapView.delegate = self;

    locationManager = [[CLLocationManager alloc] init];
    [locationManager setDelegate:self];
    [locationManager setDistanceFilter:kCLDistanceFilterNone];
    [locationManager setDesiredAccuracy:kCLLocationAccuracyBest];

    [self.mapView setShowsUserLocation:YES];
    [self.window makeKeyAndVisible];
}

- (void)viewDidUnload {
    [self setMapView:nil];
    [super viewDidUnload];
}

@end
```

ViewController.m

5.9.3. Windows Phone

Windows Phone priemonėmis realizuotas reikalavimas:

```
<my:Map Name="map"
        HorizontalAlignment="Stretch"
        VerticalAlignment="Stretch"
        CredentialsProvider="********************">
</my:Map>
```

MainPage.xaml

```
namespace MasterWork {

    public partial class MainPage : PhoneApplicationPage {

        GeoCoordinateWatcher watcher;
        double latitued;
        double longitude;

        public MainPage() {
            InitializeComponent();

            latitued = 54.9058;
```

```
            longitude = 23.9567;
            map.Center = new GeoCoordinate(latitued, longitude);
            map.ZoomLevel = 15;
            map.ZoomBarVisibility = Visibility.Visible;
            watcher = new GeoCoordinateWatcher(GeoPositionAccuracy.High);
            watcher.PositionChanged +=
                new EventHandler<GeoPositionChangedEventArgs<GeoCoordinate>>(positionChanged);
            watcher.Start();
        }

        void positionChanged(object sender, GeoPositionChangedEventArgs<GeoCoordinate> e) {
            if (!e.Position.Location.IsUnknown) {
                latitued = e.Position.Location.Latitude;
                longitude = e.Position.Location.Longitude;
                map.Center = new GeoCoordinate(latitued, longitude);
            }
        }
    }
}
```

MainPage.xaml.cs

5.10. FR-1-10

FR-1-10 reikalavimo aprašymas pateiktas lentelėje – Lent. 5.10.

Lent. 5.10 - FR-1-10 reikalavimo aprašymas

Reikalavimo numeris	Testuojama sritis	Aprašymas
FR-1-10	Kamera	Sukurti formą su vienu mygtuku. Nuspaudus mygtuką nufotografuojamas ir atvaizduojamas kameros vaizdas.

5.10.1. Android

Android priemonėmis realizuotas reikalavimas:

```
public class MainActivity extends Activity {

    private Camera camera;
    private CameraPreview cameraPreview;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        camera = Camera.open();
        cameraPreview = new CameraPreview(this, camera);
        FrameLayout preview = (FrameLayout) findViewById(R.id.preview);
        preview.addView(cameraPreview);

        Button captureButton = (Button) findViewById(R.id.button);
        captureButton.setOnClickListener(new ButtonListener());
    }

    class ButtonListener implements OnClickListener {

        @Override
        public void onClick(View view) {
            camera.takePicture(null, null, new PictureCallback());
        }
    }

    class PictureCallback implements Camera.PictureCallback {

        @Override
        public void onPictureTaken(byte[] data, Camera camera) {
            Bitmap bitmap = BitmapFactory.decodeByteArray(data , 0, data.length);
            ImageView imageView = (ImageView) findViewById(R.id.imageView);
            imageView.setImageBitmap(bitmap);
        }
    }
}
```

MainActivity.java

```
public class CameraPreview extends SurfaceView implements SurfaceHolder.Callback {

    private SurfaceHolder holder;
    private Camera camera;

    public CameraPreview(Context context, Camera camera) {
        super(context);
        this.camera = camera;
        holder = getHolder();
        holder.addCallback(this);
        holder.setType(SurfaceHolder.SURFACE_TYPE_PUSH_BUFFERS);
    }

    public void surfaceCreated(SurfaceHolder holder) {
        try {
            camera.setPreviewDisplay(holder);
            camera.startPreview();
        } catch (IOException e) {
            Log.e("Klaida", "Nepavyko paruošti aplicacijos.");
        }
    }

    public void surfaceDestroyed(SurfaceHolder holder) {
        camera.stopPreview();
        camera.release();
    }

    public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {
    }
}
```

CameraPreview.java

5.10.2. iOS

iOS priemonėmis realizuotas reikalavimas:

```
@interface ViewController : UIViewController <UIImagePickerControllerDelegate,
UINavigationControllerDelegate> {
    UIImageView *imageView;
    BOOL newMedia;
}

@property (nonatomic, retain) IBOutlet UIImageView *imageView;

- (IBAction)buttonPressed:(id)sender;

@end
```

ViewController.h

```
@implementation ViewController

@synthesize imageView;

- (IBAction)buttonPressed:(id)sender {
    if ([UIImagePickerController isSourceTypeAvailable:UIImagePickerControllerSourceTypeCamera]) {
        UIImagePickerController *imagePicker = [[UIImagePickerController alloc] init];
        imagePicker.delegate = self;
        imagePicker.sourceType = UIImagePickerControllerSourceTypeCamera;
        imagePicker.mediaTypes = [NSArray arrayWithObjects:(NSString *) kUTTypeImage, nil];
        imagePicker.allowsEditing = NO;
        [self presentModalViewController:imagePicker animated:YES];
        newMedia = YES;
    }
}

- (void)imagePickerController:(UIImagePickerController *)picker
        didFinishPickingMediaWithInfo:(NSDictionary *)info {
    NSString *mediaType = [info objectForKey:UIImagePickerControllerMediaType];
    [self dismissModalViewControllerAnimated:YES];

    if ([mediaType isEqualToString:(NSString *)kUTTypeImage]) {
        UIImage *image = [info objectForKey:UIImagePickerControllerOriginalImage];
        imageView.image = image;

        if (newMedia) {
            UIImageWriteToSavedPhotosAlbum(image, self,
                    @selector(image:finishedSavingWithError:contextInfo:), nil);
```

```
        }
    }
}

- (void)image:(UIImage *)image finishedSavingWithError:(NSError *)error
        contextInfo:(void *)contextInfo {
    if (error) {
        UIAlertView *alert = [[UIAlertView alloc] initWithTitle: @"Klaida"
                              message: @"Nuotraukos išsaugoti nepavyko." delegate: nil
                              cancelButtonTitle:@"OK" otherButtonTitles:nil];
        [alert show];
    }
}

- (void)imagePickerControllerDidCancel:(UIImagePickerController *)picker {
    [self dismissModalViewControllerAnimated:YES];
}

- (void)viewDidUnload {
    [self setImageView:nil];
    [super viewDidUnload];
}

@end
```

ViewController.m

5.10.3. Windows Phone

Windows Phone priemonėmis realizuotas reikalavimas:

```
namespace MasterWork {

    public partial class MainPage : PhoneApplicationPage {

        CameraCaptureTask cameraTask = new CameraCaptureTask();

        public MainPage() {
            InitializeComponent();
            cameraTask.Completed += new EventHandler<PhotoResult>(_cameraCapture_Completed);
        }

        void _cameraCapture_Completed(object sender, PhotoResult e) {
            if (e.TaskResult == TaskResult.OK) {
                Image1.Source = new BitmapImage(new Uri(e.OriginalFileName));
            }
        }

        private void buttonClick(object sender, RoutedEventArgs e) {
            cameraTask.Show();
        }
    }
}
```

MainPage.xaml.cs

5.11. FR-1-11

FR-1-11 reikalavimo aprašymas pateiktas lentelėje – Lent. 5.11.

Lent. 5.11 - FR-1-11 reikalavimo aprašymas

Reikalavimo numeris	Testuojama sritis	Aprašymas
FR-1-11	Sensoriai	Sukurti formą su trimis laukais. Šie laukai turi parodyti ir nuolat atnaujinti mobiliojo įrenginio akselerometro parodymus.

5.11.1. Android

Android priemonėmis realizuotas reikalavimas:

```
public class MainActivity extends Activity implements SensorEventListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.main);
        SensorManager sm = (SensorManager) getSystemService(Context.SENSOR_SERVICE);
        Sensor a = sm.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
        sm.registerListener(this, a, SensorManager.SENSOR_DELAY_NORMAL);
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int arg) {
    }

    @Override
    public void onSensorChanged(SensorEvent event) {
        float x = event.values[0];
        float y = event.values[1];
        float z = event.values[2];

        TextView textViewX = (TextView) findViewById(R.id.textViewX);
        TextView textViewY = (TextView) findViewById(R.id.textViewY);
        TextView textViewZ = (TextView) findViewById(R.id.textViewZ);

        textViewX.setText(String.valueOf(x));
        textViewY.setText(String.valueOf(y));
        textViewZ.setText(String.valueOf(z));
    }

}
```

MainActivity.java

5.11.2. iOS

iOS priemonėmis realizuotas reikalavimas:

```
@interface ViewController : UIViewController <UIAccelerometerDelegate>

@property (strong, nonatomic) IBOutlet UILabel *labelX;
@property (strong, nonatomic) IBOutlet UILabel *labelY;
@property (strong, nonatomic) IBOutlet UILabel *labelZ;

@end
```

ViewController.h

```
@implementation ViewController

@synthesize labelX;
@synthesize labelY;
@synthesize labelZ;

- (void)accelerometer:(UIAccelerometer *)accelerometer didAccelerate:(UIAcceleration *)acceleration {
    labelX.text = [NSString stringWithFormat:@"%f", acceleration.x];
    labelY.text = [NSString stringWithFormat:@"%f", acceleration.y];
    labelZ.text = [NSString stringWithFormat:@"%f", acceleration.z];
}

- (void)viewDidLoad {
    [super viewDidLoad];
    UIAccelerometer *accelerometer = [UIAccelerometer sharedAccelerometer];
    accelerometer.delegate = self;
    accelerometer.updateInterval = 1.0f/30.0f;
}

- (void)viewDidUnload {
    [self setLabelX:nil];
    [self setLabelY:nil];
    [self setLabelZ:nil];
    [super viewDidUnload];
}

@end
```

ViewController.m

5.11.3. Windows Phone

Windows Phone priemonėmis realizuotas reikalavimas:

```
namespace MasterWork {

    public partial class MainPage : PhoneApplicationPage {
```

```
        Accelerometer accelerometer;

        public MainPage() {
            InitializeComponent();
            accelerometer = new Accelerometer();
            accelerometer.Start();
            accelerometer.ReadingChanged +=
                new EventHandler<AccelerometerReadingEventArgs>(valueChanged);
        }

        public void valueChanged(object sender, AccelerometerReadingEventArgs e) {
            this.Dispatcher.BeginInvoke(delegate() {
                textBlockX.Text = e.X.ToString();
                textBlockY.Text = e.Y.ToString();
                textBlockZ.Text = e.Z.ToString();
            });
        }
    }
}
```

MainPage.xaml.cs

5.12. FR-1-12

FR-1-12 reikalavimo aprašymas pateiktas lentelėje – Lent. 5.12.

Lent. 5.12 - FR-1-12 reikalavimo aprašymas

Reikalavimo numeris	Testuojama sritis	Aprašymas
FR-1-12	Tinklai	Sukurti formą, kurioje būtų vienas teksto atvaizdavimo laukas, vienas teksto įvedimo laukas, bei mygtukas. HTTP protokolu nurodžius svetainės adresą ir nuspaudus mygtuką atvaizduojamas svetainės *HTML* kodas.

5.12.1. Android

Android priemonėmis realizuotas reikalavimas:

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new ButtonListener());
    }

    class ButtonListener implements OnClickListener {

        @Override
        public void onClick(View view) {
            TextView textView = (TextView) findViewById(R.id.editText);

            try {
                URL url = new URL(textView.getText().toString());
                URLConnection connection = url.openConnection();
                InputStreamReader isr = new InputStreamReader(connection.getInputStream());
                BufferedReader reader = new BufferedReader(isr);

                String line = reader.readLine();
                while (line != null) {
                    textView.append(line);
                    line = reader.readLine();
                }
            } catch (IOException ex) {
            }
        }
    }
}
```

MainActivity.java

5.12.2. iOS

iOS priemonėmis realizuotas reikalavimas:

```
@interface ViewController : UIViewController {
    NSMutableData *myData;
    NSURLConnection *connection;
}

@property (strong, nonatomic) IBOutlet UITextField *textField;
@property (strong, nonatomic) IBOutlet UITextView *textView;

- (IBAction)buttonPressed:(id)sender;

@end
```

ViewController.h

```
@implementation ViewController

@synthesize textField;
@synthesize textView;

- (IBAction)buttonPressed:(id)sender {
    if (connection == nil) {
        myData = [NSMutableData new];

        NSURL *url = [NSURL URLWithString:textField.text];
        NSURLRequest *request = [NSURLRequest requestWithURL:url];
        connection = [NSURLConnection connectionWithRequest:request delegate:self];

        [textField resignFirstResponder];
    }
}

- (BOOL)textFieldShouldReturn:(UITextField *)textField {
    [self.textField resignFirstResponder];
    return YES;
}

- (void)connection:(NSURLConnection *)connection didReceiveData:(NSData *)data {
    [myData appendData:data];
}

- (void) connectionDidFinishLoading:(NSURLConnection *)connection {
    textView.text = [[NSString alloc] initWithData:myData encoding:NSASCIIStringEncoding];
}

- (void)viewDidUnload {
    [self setTextField:nil];
    [self setTextView:nil];
    [super viewDidUnload];
}

@end
```

ViewController.m

5.12.3. Windows Phone

Windows Phone priemonėmis realizuotas reikalavimas:

```
namespace MasterWork {

    public partial class MainPage : PhoneApplicationPage {

        public MainPage() {
            InitializeComponent();
        }

        private void buttonClick(object sender, System.Windows.RoutedEventArgs e) {
            WebClient webClient = new WebClient();
            webClient.DownloadStringCompleted +=
                new DownloadStringCompletedEventHandler(downloadCompleted);
            webClient.DownloadStringAsync(new System.Uri(textBox.Text));
        }

        private void downloadCompleted(object sender, DownloadStringCompletedEventArgs e) {
            textBlock.Text = e.Result;
        }
```

```
    }
}
```

MainPage.xaml.cs

5.13. FR-1-13

FR-1-13 reikalavimo aprašymas pateiktas lentelėje – Lent. 5.13.

Lent. 5.13 - FR-1-13 reikalavimo aprašymas

Reikalavimo numeris	Testuojama sritis	Aprašymas
FR-1-13	Foniniai procesai	Sukurti procesą, kuris kas 10 sekundžių parodytų vartotojui laiką. Aplikacija turi būti nematoma vartotojui. Taip pat turi būti priemonės leidžiančios įjungti ir išjungti šį procesą.

5.13.1. Android

Android priemonėmis realizuotas reikalavimas:

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button buttonStart = (Button) findViewById(R.id.buttonStart);
        buttonStart.setOnClickListener(new ButtonStartListener());
        Button buttonStop = (Button) findViewById(R.id.buttonStop);
        buttonStop.setOnClickListener(new ButtonStopListener());
    }

    class ButtonStartListener implements OnClickListener {

        @Override
        public void onClick(View view) {
            startService(new Intent(MainActivity.this, TimeService.class));
        }
    }

    class ButtonStopListener implements OnClickListener {

        @Override
        public void onClick(View view) {
            stopService(new Intent(MainActivity.this, TimeService.class));
        }
    }
}
```

MainActivity.java

```
public class TimeService extends Service {

    private Handler handler;
    private boolean isRunning;

    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }

    @Override
    public void onCreate() {
        super.onCreate();
        isRunning = true;
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        isRunning = false;
    }

    @Override
    public void onStart(Intent intent, int startId) {
        handler = new Handler() {
```

```
            @Override
            public void handleMessage(Message msg) {
                super.handleMessage(msg);
                DateFormat df = DateFormat.getTimeInstance();
                String message = df.format(new Date());
                Toast.makeText(TimeService.this, message, Toast.LENGTH SHORT).show();
            }
        };

        new Thread(new Runnable() {
            public void run() {
                while(isRunning) {
                    try {
                        Thread.sleep(5000);
                        handler.sendEmptyMessage(0);
                    } catch (InterruptedException e) {
                    }
                }
            }
        }).start();
    }
}
```

TimeService.java

5.13.2. iOS

iOS priemonėmis realizuotas reikalavimas:

```
@interface ViewController : UIViewController {
    UILocalNotification *localNotification;
}

- (IBAction)buttonStartPressed:(id)sender;
- (IBAction)buttonStopPressed:(id)sender;

@end
```

ViewController.h

```
@implementation ViewController

- (IBAction)buttonStartPressed:(id)sender {
    NSDateFormatter *formatter = [[NSDateFormatter alloc] init];
    [formatter setDateFormat:@"HH:mm"];
    NSString *dateString = [formatter stringFromDate:[NSDate date]];

    localNotification = [[UILocalNotification alloc] init];
    localNotification.fireDate = [NSDate dateWithTimeIntervalSinceNow:10];
    localNotification.timeZone = [NSTimeZone defaultTimeZone];
    localNotification.repeatInterval = NSMinuteCalendarUnit;
    localNotification.hasAction = NO;
    localNotification.alertBody = dateString;
    [[UIApplication sharedApplication] scheduleLocalNotification:localNotification];
}

- (IBAction)buttonStopPressed:(id)sender {
    [[UIApplication sharedApplication] cancelAllLocalNotifications];
}

@end
```

ViewController.m

5.13.3. Windows Phone

Windows Phone priemonėmis realizuotas reikalavimas:

```
namespace MasterWorkLauncher {

    public partial class MainPage : PhoneApplicationPage {

        PeriodicTask periodicTask;
        string periodicTaskName = "PeriodicAgent";

        public MainPage() {
            InitializeComponent();
        }
```

```
        private void buttonStartClick(object sender, RoutedEventArgs e) {
            periodicTask = ScheduledActionService.Find(periodicTaskName) as PeriodicTask;
            if (periodicTask != null) {
                ScheduledActionService.Remove(periodicTaskName);
            }

            periodicTask = new PeriodicTask(periodicTaskName);
            periodicTask.Description = "This demonstrates a periodic task.";
            ScheduledActionService.Add(periodicTask);
        }

        private void buttonStopClick(object sender, RoutedEventArgs e) {
            ScheduledActionService.Remove(periodicTaskName);
        }
    }
}
```

MainPage.xaml.cs

```
using System;
using System.Windows;
using Microsoft.Phone.Scheduler;
using Microsoft.Phone.Shell;

namespace MasterWork {

    public class ScheduledAgent : ScheduledTaskAgent {

        protected override void OnInvoke(ScheduledTask task) {
            ShellToast toast = new ShellToast();
            string message = DateTime.Now.ToString("HH:mm:ss");
            toast.Title = "Laikas: ";
            toast.Content = message;
            toast.Show();
            NotifyComplete();
        }
    }
}
```

ScheduledAgent.cs

5.14. FR-1-14

FR-1-14 reikalavimo aprašymas pateiktas lentelėje – Lent. 5.14.

Lent. 5.14 - FR-1-14 reikalavimo aprašymas

Reikalavimo numeris	Testuojama sritis	Aprašymas
FR-1-14	Duomenų apsikeitimas	Sukurti formą su vienu teksto atvaizdavimo lauku ir mygtuku. Nuspaudus mygtuką į tekstinį lauką išvedami visi asmenys esantys mobiliojo įrenginio kontaktų sąraše.

5.14.1. Android

Android priemonėmis realizuotas reikalavimas:

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new ButtonListener());
    }

    class ButtonListener implements OnClickListener {

        @Override
        public void onClick(View view) {
            TextView textView = (TextView) findViewById(R.id.textView);
            ContentResolver r = getContentResolver();
            Cursor c = r.query(ContactsContract.Contacts.CONTENT_URI, null, null, null, null);

            if (c.getCount() > 0) {
                while (c.moveToNext()) {
```

```
                    String name = c.getString(c.getColumnIndex("display_name"));
                    textView.append(name + "\n");
                }
                textView.append(".");
            }
        }
    }
}
```

MainActivity.java

5.14.2. iOS

iOS priemonėmis realizuotas reikalavimas:

```
@interface ViewController : UIViewController

@property (strong, nonatomic) IBOutlet UITextView *textView;

@end
```

ViewController.h

```
@implementation ViewController

@synthesize textView;

- (void)viewDidLoad {
    [super viewDidLoad];

    ABAddressBookRef addressBook = ABAddressBookCreate();
    CFArrayRef allPeople = ABAddressBookCopyArrayOfAllPeople(addressBook);
    CFIndex n = ABAddressBookGetPersonCount(addressBook);

    for (int i = 0; i < n; i++) {
        ABRecordRef ref = CFArrayGetValueAtIndex(allPeople, i);
        if (ABRecordGetRecordType(ref) == kABPersonType) {
            CFStringRef firstName = ABRecordCopyValue(ref, kABPersonFirstNameProperty);
            if (firstName) {
                NSString *fn = [NSString stringWithFormat:@"%@", firstName];
                fn = [fn stringByAppendingString:@"\n"];
                textView.text = [textView.text stringByAppendingString:fn];
            }
        }
    }
}

- (void)viewDidUnload {
    [self setTextView:nil];
    [super viewDidUnload];
}

@end
```

ViewController.m

5.14.3. Windows Phone

Windows Phone priemonėmis realizuotas reikalavimas:

```
namespace MasterWork {

    public partial class MainPage : PhoneApplicationPage {

        public MainPage() {
            InitializeComponent();
        }

        private void buttonClick(object sender, RoutedEventArgs e) {
            Contacts cons = new Contacts();
            cons.SearchCompleted +=
                new EventHandler<ContactsSearchEventArgs>(searchCompleted);
            cons.SearchAsync(String.Empty, FilterKind.None, "Contacts Test #1");
        }

        void searchCompleted(object sender, ContactsSearchEventArgs e) {
            System.Text.StringBuilder sb = new System.Text.StringBuilder();
            foreach (Contact con in e.Results) {
                sb.AppendLine(con.DisplayName);
            }
```

```
            textBlock.Text = sb.ToString();
        }
    }
}
```

MainPage.xaml.cs

5.15. FR-1-15

FR-1-15 reikalavimo aprašymas pateiktas lentelėje – Lent. 5.15.

Lent. 5.15 - FR-1-15 reikalavimo aprašymas

Reikalavimo numeris	Testuojama sritis	Aprašymas
FR-1-15	Duomenų išsaugojimas	Sukurti formą, kurioje būtų teksto įvedimo laukas, teksto atvaizdavimo laukas bei mygtukas. Nuspaudus mygtuką vartotojo įvestas tekstas išvalomas ir įrašomas į duomenų bazės lentelę, teksto atvaizdavimo laukas išvalomas ir atvaizduojamas visas duomenų bazės lentelės turinys. Aplikacija turi turėti priemones duomenų bazės sukūrimui jos pirmojo startavimo metu.

5.15.1. Android

Android priemonėmis realizuotas reikalavimas:

```
public class MainActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button button = (Button) findViewById(R.id.button);
        button.setOnClickListener(new ButtonListener());

        SQLiteDatabase db = openOrCreateDatabase("NamesDatabase", MODE_PRIVATE, null);
        db.execSQL("CREATE TABLE IF NOT EXISTS names(name VARCHAR);");
        db.close();
    }

    class ButtonListener implements OnClickListener {

        @Override
        public void onClick(View view) {
            TextView textView = (TextView) findViewById(R.id.textView);
            textView.setText("");
            EditText editText = (EditText) findViewById(R.id.editText);
            String text = editText.getText().toString();

            SQLiteDatabase db = openOrCreateDatabase("NamesDatabase", MODE_PRIVATE, null);
            ContentValues values = new ContentValues();
            values.put("name", text);
            db.insert("names", null, values);

            Cursor cursor = db.rawQuery("SELECT name FROM names ORDER BY name", null);
            cursor.moveToFirst();
            while (cursor.isAfterLast() == false) {
                textView.append(cursor.getString(0) + " ");
                cursor.moveToNext();
            }
            db.close();
        }
    }
}
```

MainActivity.java

5.15.2. iOS

iOS priemonėmis realizuotas reikalavimas:

```
@interface ViewController : UIViewController {
    NSString *databasePath;
    sqlite3 *db;
```

```
}

@property (strong, nonatomic) IBOutlet UITextField *textField;
@property (strong, nonatomic) IBOutlet UILabel *label;

- (IBAction)buttonPressed:(id)sender;

@end
```

ViewController.h

```
@implementation ViewController

@synthesize textField;
@synthesize label;

- (IBAction)buttonPressed:(id)sender {
    sqlite3_stmt *statement;
    const char *dbpath = [databasePath UTF8String];

    if (sqlite3_open(dbpath, &db) == SQLITE_OK) {
        NSString *insertSQL = [NSString stringWithFormat:
                               @"INSERT INTO names (name) VALUES (\"%@\")", textField.text];
        const char *insert stmt = [insertSQL UTF8String];
        sqlite3_prepare_v2(db, insert_stmt, -1, &statement, NULL);

        if (sqlite3_step(statement) == SQLITE_DONE) {
            textField.text = @"";
        }
        sqlite3 finalize(statement);
        sqlite3 close(db);
    }

    if (sqlite3_open(dbpath, &db) == SQLITE_OK) {
        NSString *querySQL = [NSString stringWithFormat: @"SELECT name FROM names"];
        const char *query stmt = [querySQL UTF8String];
        sqlite3_prepare_v2(db, query_stmt, -1, &statement, NULL);

        while (sqlite3_step(statement) == SQLITE_ROW) {
            NSString *field = [[NSString alloc]
                               initWithUTF8String:(const char *) sqlite3 column text(statement, 0)];
            label.text = [label.text stringByAppendingString: field];
            label.text = [label.text stringByAppendingString:@" "];
        }
        sqlite3 finalize(statement);
        sqlite3 close(db);
    }
    [textField resignFirstResponder];
}

- (void)viewDidLoad {
    [super viewDidLoad];

    NSString *docsDir = [dirPaths objectAtIndex:0];
    NSArray *dirPaths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,
                                                            NSUserDomainMask, YES);
    databasePath = [[NSString alloc] initWithString:
                                        [docsDir stringByAppendingPathComponent: @"my.db"]];
    NSFileManager *filemgr = [NSFileManager defaultManager];

    if ([filemgr fileExistsAtPath: databasePath ] == NO) {
        const char *dbpath = [databasePath UTF8String];
        if (sqlite3_open(dbpath, &db) == SQLITE_OK) {
            char *errMsg;
            const char *sql_stmt = "CREATE TABLE IF NOT EXISTS names (name TEXT)";
            sqlite3_exec(db, sql_stmt, NULL, NULL, &errMsg);
            sqlite3_close(db);
        }
    }
}

- (void)viewDidUnload {
    [self setLabel:nil];
    [self setTextField:nil];
    [super viewDidUnload];
}

@end
```

ViewController.m

5.15.3. Windows Phone

Windows Phone priemonėmis realizuotas reikalavimas:

```
namespace MasterWork {

    public partial class MainPage : PhoneApplicationPage {

        private const string strConnectionString = @"isostore:/NamesDB.sdf";

        public MainPage() {
            InitializeComponent();
            using (NameDataContext Empdb = new NameDataContext(strConnectionString)) {
                if (Empdb.DatabaseExists() == false) {
                    Empdb.CreateDatabase();
                }
            }
        }

        private void buttonClick(object sender, RoutedEventArgs e) {
            using (NameDataContext ndb = new NameDataContext(strConnectionString)) {
                Name newName = new Name {
                    text = textBox.Text.ToString()
                };
                ndb.names.InsertOnSubmit(newName);
                ndb.SubmitChanges();

                IQueryable<Name> namesQuery = from Names in ndb.names select Names;
                IList<Name> NamesList = namesQuery.ToList();
                StringBuilder strBuilder = new StringBuilder();
                foreach (Name name in NamesList) {
                    strBuilder.AppendLine(name.text);
                }
                textBlock.Text = strBuilder.ToString();
                textBox.Text = "";
            }
        }
    }
}
```

MainPage.xaml.cs

```
namespace MasterWork {

    [Table]
    public class Name {

        [Column (IsPrimaryKey = true, IsDbGenerated = true, CanBeNull = false,
                 AutoSync = AutoSync.OnInsert)]
        public int nameID {
            get;
            set;
        }

        [Column(CanBeNull = false)]
        public string text {
            get;
            set;
        }
    }
}
```

Name.cs

```
namespace MasterWork {

    public class NameDataContext : DataContext {

        public NameDataContext(string connectionString) : base(connectionString) {
        }

        public Table<Name> names {
            get {
                return this.GetTable<Name>();
            }
        }
    }
}
```

NameDataContext.cs

6. REZULTATAI

Visos nagrinėtos operacinės sistemos turi standartines programinės įrangos kūrimo priemones, kuriomis galima kurti programas, skirtas tik tai operacinei sistemai. Pagrindinės *Android, iOS* bei *Windows Phone* naudojamos technologijos yra pateiktos lentelėje – Lent. 6.1[[3]].

Lent. 6.1 - Pagrindinės *Android, iOS* ir *Windows Phone* technologijos

	Android	**iOS**	**WindowsPhone**
Programėlių parduotuvė	*Android Market*	*AppStore*	*App Marketplace*
Grafinė vartotojo sąsaja	*Java Application Framework*	*Cocoa*	*Silverlight*
Naršyklė	*Webkit*	*Webkit*	*Internet Explorer*
Žemėlapiai	-	*Apple MapKit*	*Bing Maps*
3D grafika	*OpenGL*	*OpenGL*	*DirectX*
Programavimo kalba	*Java*	*Objective-C*	*C#, Visual Basic*
Virtuali mašina	*Dalvik VM*	-	*CLR*

Android programoms kurti yra skirtas *Android ADT Bundle*, *iOS XCode*, o *Windows Phone Visual Studio* įrankis. Visos programavimo aplinkos turi virtualius įrenginius, kuriuose galima išbandyti kuriamą programinę įrangą. Testavimo metu pastebėta, kad *Android* virtualus įrenginys užsikraudavo maždaug 5 kartus lėčiau nei *iOS* ar *Windows Phone*. Be to nei vienas virtualus įrenginys nesugebėjo pilnai pakeisti tikrojo įrenginio. Su *Android* bei *iOS* telefono emuliatoriais buvo galima ištestuoti 13 iš 15, o *Windows Phone* 14 iš 15 programų. Visi virtualūs įrenginiai nesugebėjo imituoti kameros, *Android* įrenginys nepaleidžia foninių procesų, o *iOS* įrenginys neturi akselerometro funkcijos.

Geriausias derinimo *(angl. debugging)* galimybes turi *Visual Studio* įrankis. Dirbant su juo nepastebėta jokių nepatogumų. *Android ADT Bundle* įrankyje esantis *LogCat* klaidų auditavimo įskiepis neleidžia į atmintį nusikopijuoti klaidos steko, kas yra nepatogu kai norima internete paieškoti ar su panašiomis problemomis nėra susidūrę kiti vartotojai. Prasčiausias derinimo galimybes turėjo *XCode* įrankis. *Visual-C* klaba grąžinamos klaidos dažniausiai yra bendro pobūdžio ir jos neturi informacijos apie tai, kurioje kodo eilutėje ir dėl kokios priežasties įvyko programos klaida.

Vartotojo duomenų saugumui užtikrinti visos platformos naudoja praktiškai identiškas priemones. Programos autorius privalo užregistruoti kokius techninius ir programinius resursus naudos jo programa ir apie tai instaliavimo metu yra informuojamas vartotojas. *iOS* šiek tiek sušvelninęs šią saugumo politiką ir vartotoją informuoja tik kai instaliuojama programa kuri naudos pranešimų kūrimo ar vartotojo buvimo vietos nustatymo funkcijas, tačiau programuotojui vis tiek reikia suvesti visus naudojamus resursus. Vėliau ši informacija yra naudojama nustatyti ar vartotojo įrenginys yra pakankamas norint paleisti minėtą programą.

Visi nagrinėti įrankiai turi vizualias priemones grafinei vartotojo sąsajai kurti. *Windows Phone* ir *Android* operacinės sistemos leidžia nesinaudojant šiais įrankiais, ir rašyti programos kodą *XML* kalba. Grafinei vartotojo sąsajai kurti yra sukurta daugybė komponentų, kurie atitinka kiekvienos operacinės sistemos dizainą. Įvykių nustatymas *(angl. event handling)* yra patogus ir nesudėtingas visose platformose. *iOS* programinės įrangos kūrimo paketas turi papildomas sąsajas, kurios leidžia nustatyti tokius įvykius kaip įrenginio kratymas *(angl. shaking)*.

Geriausias daugiakalbystės priemones turi *Android* ir *iOS* programinės įrangos kūrimo paketai. Programuojant šioms operacinėms sistemoms nereikia rašyti jokio papildomo kodo norint kurti programas keletu kalbų, tuo tarpu kuriant programas *Windows Phone* operacinei sistemai reikia realizuoti keletą klasių.

Visos operacinės sistemos turi panašias priemones animacijai kurti ir atvaizduoti. Šios priemonės leidžia animuoti tiek grafinės vartotojo sąsajos elementus, tiek ir kurti nesudėtingą animaciją sudarytą iš primityvių elementų (pvz. apskritimų, trikampių ir t.t.). *Android* operacinei sistemai kurtas programos kodas buvo didesnis, nes reikėjo sukurti atskirą giją, kuri užtikrintų animacijos tęstinumą, tuo tarpu *Windows Phone* bei *iOS* tam turėjo standartines priemones.

Patogiausias multimedijos priemones turi *Windows Phone* programinės įrangos kūrimo paketas. Kuriant eksperimentinę programinę įrangą šiai platformai programos kodas buvo 47% mažesnis nei *Android* ir 62% mažesnis nei *iOS*. Visų operacinių sistemų programų kūrimo sąsajos palaiko populiariausius multimedijos formatus.

Visos trys nagrinėtos programų kūrimo sąsajos turi integruotas naršykles, leidžiančias atvaizduoti *HTML* kalba parašytą programos turinį.

iOS bei *Windows Phone* programų kūrimo sąsajos turi labai patogias priemones darbui su žemėlapiais ir vartotojo vietos nustatymu. *Windows Phone* gamintojai reikalauja, kad prieš naudojant šią funkciją programuotojai užsiregistruotų *Microsoft Inc.* svetainėje, kur jiems būtų suteiktas šios paslaugos aktyvavimo raktas. *Apple MapKit* karkasas turi unikalias priemones, leidžiančias išsisaugoti *(angl. cache)* žemėlapius atmintyje ir vėliau juos panaudoti nenaudojant interneto tinklo. *Android SDK* neturi priemonių darbui su žemėlapiais, tačiau tam rekomenduoja naudoti *Google Play Services SDK* bibliotekas.

Darbui su kamera visose platformose rekomenduojama naudoti integruotas programėles ir tik *Android* operacinėje sistemoje yra priemonės tiesiogiai prieiti prie techninės įrangos funkcijų.

Mažiausiai kodo eilučių norint gauti informaciją iš sensorių, reikėjo *Windows Phone* operacinėje sistemoje, tačiau ši sąsaja leidžia gauti informaciją tik iš 3 tipų sensorių. *Android* palaiko net 14 tipų sensorius, *iOS* – 4.

Visos trys nagrinėtos programų kūrimo sąsajos turi priemones leidžiančias prisijungti ir atsisiųsti informaciją *HTTP, HTTPS* bei *FTP* protokolais.

Norint kurti programas kurios turi tam tikru periodiškumu atlikti užduotis, geriausiai tinka *Android* operacinė sistema. Ji vienintelė netaiko nei periodo, nei atliekamos užduoties trukmės apribojimų. Be to tiek *Windows Phone*, tiek ir *iOS* sąsajos nepaleidžia šių procesų jeigu įrenginio akumuliatorius yra netoli išsikrovimo ribos. Minimalus *Windows Phone* programos paleidimo periodas yra 30 minučių, kas gali netenkinti kuriamos programos reikalavimų, o *iOS* taisyklės leidžia kurti foninius procesus tik tam tikroms funkcijoms atlikti.

Saugumo sumetimais tiek *iOS* tiek *Windows Phone* paleidžia visas programas izoliuotoje failinėje sistemoje. Tai apriboja galimybę kelioms programoms keistis duomenimis. Šių operacinių sistemų programuotojai gali gauti duomenis tik iš sisteminių programėlių tokių kaip adresų knygutė, kai tuo tarpu *Android* leidžia vartotojo programoms turėti sąsajas, kuriomis kelios vartotojo programos gali apsikeisti duomenimis. Tai yra patogu kai norima sukurti kelias programas, kurios turėtų tą patį duomenų šaltinį.

Visose platformose duomenų išsaugojimui galima rinktis saugoti informaciją failinėje sistemoje arba reliacinėje duomenų bazėje. Duomenys *Android* ir *iOS* duomenų bazėse įterpiami ir gaunami naudojantis *SQL* sintakse, tuo tarpu *Windows Phone* duomenų bazė dirba su rečiau naudojama *LINQ* kalba.

Eksperimentinės programinės įrangos programos kodo eilučių skaičius skirtingose operacinėse sistemose yra palygintas lentelėje – Lent. 6.2.

Lent. 6.2 – Programos kodo eilučių skaičius skirtingose operacinėse sistemose

Reikalavimo numeris	**Android**		**iOS**		**WindowsPhone**	
	Grafinė vartotojo sąsaja	Programos kodas	Grafinė vartotojo sąsaja	Programos kodas	Grafinė vartotojo sąsaja	Programos kodas

Reikalavimo numeris	Android		iOS		WindowsPhone	
	Grafinė vartotojo sąsaja	Programos kodas	Grafinė vartotojo sąsaja	Programos kodas	Grafinė vartotojo sąsaja	Programos kodas
FR-1-1	22	19	Kurta su įrankiu	27	20	13
FR-1-2	8	23	X	X	12	13
FR-1-3	-	31	-	29	13	23
FR-1-4	17	8	3 formos kurtos su įrankiu	15	Kurta su įrankiu	22
FR-1-5	3	56	Kurta su įrankiu	34	11	34
FR-1-6	-	27	-	28	3	13
FR-1-7	-	22	-	39	4	13
FR-1-8	3	21	Kurta su įrankiu	32	4	14
FR-1-9	-	34 (nėra žemėlapių)	Kurta su įrankiu	44	5	31
FR-1-10	-	67	-	63 (naudojant integruotą programėlę)	-	22 (naudojant integruotą programėlę)
FR-1-11	-	31	-	34	-	23
FR-1-12	-	32	-	48	-	20
FR-1-13	-	76	-	29	-	46 (nepilnai išpildyti reikalavimai)
FR-1-14	-	28	-	35	-	24
FR-1-15	-	38	-	78	-	68

Pagal (2.1) ir (2.2) formules apskaičiuoti *Android* programinės įrangos kūrimo paketo rezultatai yra pateikti lentelėje – Lent. 6.3.

Lent. 6.3 – *Android* programinės įrangos kūrimo paketo rezultatai

Sritis	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	Viso
Architektūra			10	8	8	10	3	9	10	10	10	8,8
Įrankiai				9								9,0
Techninės įrangos emuliatorius	6		7	4								6,0
Suderinamumas su technine įranga	4											4,0
Derinimas				8								8,0

Sritis	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	Viso
Saugumas	10											10,0
Grafinė vartotojo sąsaja	10	8	10	10	10	10	10	10	10	10		9,4
Meniu	10	7	10	10	10	10	10	10	10	10		9,0
Dialogai	10	9	10	10	10	10	10	10	10	10		9,7
Daugiakalbystė	10	10	10	10	10	10	6	10	10	10		9,8
Animacija	10	7	10	10	10	10	10	10	10	10		9,0
Audio	10	7	10	10	10	10	10	10	10	10		9,0
Video	10	7	10	10	10	10	10	10	10	10		9,0
HTML	10	8	10	10	10	10	10	10	10	10		9,4
Žemėlapiai	0	0	0	0	0	0	0	0	0	0		0,0
Vartotojo buvimo vietos nustatymas	10	10	10	10	10	10	10	10	10	10		10,0
Kamera	10	10	10	10	10	10	5	10	10	10		9,7
Sensoriai	10	9	10	10	10	10	10	10	10	10		9,7
Tinklai	10	8	10	10	10	10	10	10	10	10		9,4
Foniniai procesai	10	10	10	10	10	10	5	10	10	10		9,7
Duomenų apsikeitimas	10	10	10	10	10	10	10	10	10	10	10	10,0
Duomenų išsaugojimas	10	10	10	10	10	10	10	10	10	10	10	10,0
												84,8

Pagal (2.1) ir (2.2) formules apskaičiuoti *iOS* programinės įrangos kūrimo paketo rezultatai yra pateikti lentelėje – Lent. 6.4.

Lent. 6.4 – *iOS* programinės įrangos kūrimo paketo rezultatai

Sritis	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	Viso
Architektūra			10	9	10	10	7	8	5	10	6	8,5
Įrankiai				10								10,0
Techninės įrangos emuliatorius	6		9	8								7,1
Suderinamumas su technine įranga	10											10,0
Derinimas				5								5,0
Saugumas	10											10,0
Grafinė vartotojo sąsaja	10	7	10	10	10	10	10	10	10	10		9,0
Meniu	0	0	0	0	0	0	0	0	0	0		0,0

Sritis	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	Viso
Dialogai	10	10	10	10	10	10	10	10	10	10		10,0
Daugiakalbystė	10	10	10	10	10	8	10	10	10	10		9,9
Animacija	10	10	10	10	10	10	10	10	10	10		10,0
Audio	10	7	10	10	10	10	10	10	10	10		9,0
Video	10	5	10	10	10	10	10	10	10	10		8,4
HTML	10	6	10	10	10	10	10	10	10	10		8,7
Žemėlapiai	10	9	10	10	10	10	10	10	10	10		9,7
Vartotojo buvimo vietos nustatymas	10	10	10	10	10	10	10	10	10	10		10,0
Kamera	5	7	10	10	10	10	5	10	10	10		7,7
Sensoriai	8	8	10	10	10	10	5	10	10	10		8,7
Tinklai	10	6	10	10	10	10	10	10	10	10		8,7
Foniniai procesai	7	9	5	10	10	10	10	8	8	8		8,2
Duomenų apsikeitimas	5	9	10	10	10	10	10	10	10	10	5	8,4
Duomenų išsaugojimas	10	4	10	10	10	10	10	5	10	5	5	7,4
												86,0

Pagal (2.1) ir (2.2) formules apskaičiuoti *Windows Phone* programinės įrangos kūrimo paketo rezultatai yra pateikti lentelėje – Lent. 6.5.

Lent. 6.5 – *Windows Phone* programinės įrangos kūrimo paketo rezultatai

Sritis	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	Viso
Architektūra			10	9	10	10	9	9	8	10	6	9,1
Įrankiai				10								10,0
Techninės įrangos emuliatorius	8		9	8								8,3
Suderinamumas su technine įranga	8											8,0
Derinimas				10								10,0
Saugumas	10											10,0
Grafinė vartotojo sąsaja	10	10	10	10	10	10	10	10	10	10		10,0
Meniu	10	10	10	10	10	10	10	10	10	10		10,0
Dialogai	10	8	10	10	10	10	10	10	10	10		9,4
Daugiakalbystė	10	10	10	10	10	5	6	6	10	6		9,1
Animacija	10	10	10	10	10	10	10	10	10	10		10,0
Audio	10	10	10	10	10	10	10	10	10	10		10,0

Sritis	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	Viso
Video	10	10	10	10	10	10	10	10	10	10		10,0
HTML	10	10	10	10	10	10	10	10	10	10		10,0
Žemėlapiai	10	10	10	10	10	10	10	10	10	10		10,0
Vartotojo buvimo vietos nustatymas	10	10	10	10	10	10	10	10	10	10		10,0
Kamera	5	9	10	10	10	10	5	10	10	10		8,4
Sensoriai	6	10	10	10	10	10	10	10	10	10		9,2
Tinklai	10	10	10	10	10	10	10	10	10	10		10,0
Foniniai procesai	4	10	5	10	10	10	10	5	8	6		7,6
Duomenų apsikeitimas	5	10	10	10	10	10	10	10	10	10	5	8,7
Duomenų išsaugojimas	10	5	10	10	10	10	10	5	10	5	5	7,7
												92,9

Atlikus skaičiavimus nustatyta, kad *Android* programinės įrangos kūrimo paketo suminis vertinimas yra 84.8, *iOS* – 86.0, o *Windows Phone* – 95.9 balo.

7. IŠVADOS

1. *Android SDK 4.0.0* programinės įrangos kūrimo paketas turi geriausias priemones darbui su foniniais procesais, duomenų mainais tarp aplikacijų bei duomenų bazėmis. Pagrindinis šios platformos trūkumas yra techninės įrangos, į kurią diegiama ši operacinė sistema, įvairovė ir labai prastos priemonės leidžiančios išbandyti kuriamą programinę įrangą virtualiuose įrenginiuose.
2. *iOS SDK 5.0.1* programinės įrangos kūrimo paketas turi pačius patogiausius įrankius kurti grafinę vartotojo sąsają, daugiakalbystę palaikančias programas, animaciją bei labai patogų žemėlapių karkasą. Tačiau multimedijos integravimas į kuriamą programinę įrangą reikalauja beveik dvigubai daugiau programos kodo nei analogiškose programose jas kuriant su *Windows Phone* ar *Android* priemonėmis. Be to ši platforma turi pačią sudėtingiausią ir nepatogiausią sąsają darbui su duomenų bazėmis.
3. *Windows Phone SDK 7.8* programinės įrangos kūrimo paketas turi patogiausias sąsajas darbui su multimedija, grafine vartotojo sąsaja, žemėlapiais bei vartotojo buvimo vietos nustatymu, o *Windows Phone* virtualus įrenginys geriausiai imitavo tikrojo įrenginio funkcijas. Pagrindinis šios operacinės sistemos priemonių trūkumas yra tai, kad nėra patikimų priemonių foninių procesų kūrimui.
4. Atlikę programinės įrangos suminį vertinimą nustatėme, kad patogiausias priemones turi *Windows Phone* operacinė sistema surinkus 92,9 iš 100 galimų balų, toliau sekė *iOS* surinkusi 86 balus, o paskutinė liko *Android* su 84,8 balais.

LITERATŪRA

[1] H. Blodget, *The Future of Mobile* [Tinkle]. Prieiga per internetą: http://articles.businessinsider.com/2012-03-28/research/31248281_1_ios-android-hard-drive. [kreiptasi 2013-01-20]

[2] *GlobalStats* [Tinkle]. Prieiga per internetą: http://gs.statcounter.com. [kreiptasi 2013-01-20]

[3] Sun Microsystems Inc., Java Code Conventions, California, 1997. 20 p.

[4] T. L. Levi. *iOS, Android, Windows Phone 7 And The Great Changes in The Operating Systems Market* [Tinkle]. Prieiga per internetą: http://blog.radvision.com/voipsurvivor/2010/10/07/ios-android-windows-phone-7-and-the-great-changes-in-the-operating-systems-market/. [kreiptasi 2013-05-04]

[5] *Android Developer's Guide* [Tinkle]. Prieiga per internetą: http://developer.android.com/guide/index.html. [kreiptasi 2012-02-22]

[6] B. Jacobs. *Exploring the iOS SDK* [Tinkle]. Prieiga per internetą: http://mobile.tutsplus.com/tutorials/iphone/exploring-the-ios-sdk/. [kreiptasi 2013-03-14]

[7] *iOS Developer Library* [Tinkle]. Prieiga per internetą: https://developer.apple.com. [kreiptasi 2012-03-17]

[8] M. Grothaus. *An In-Depth Comparison Between iOS Map Frameworks: Apple MapKit vs. Google Maps SDK* [Tinkle]. Prieiga per internetą: http://www.fastcolabs.com/3006725/open-company/depth-comparison-between-ios-map-frameworks-apple-mapkit-vs-google-maps-sdk. [kreiptasi 2012-03-17]

[9] A. Eckermann. *Beginning iOS Development: Data Persistence* [Tinkle]. Prieiga per internetą: http://mobile.tutsplus.com/tutorials/iphone/iphone-sdk_store-data/. [kreiptasi 2013-04-18]

[10] *Windows Phone 7 Series Minimum Hardware Requirements* [Tinkle]. Prieiga per internetą: http://www.techautos.com/2010/03/17/windows-phone-7-series-hardware-requirements/. [kreiptasi 2013-03-22]

[11] *Windows Phone 7 Platform introduced to iPhone application developers* [Tinkle]. Prieiga per internetą: http://windowsphone.interoperabilitybridges.com/articles/chapter-1-windows-phone-7-platform-introduced-to-iphone-application-developers. [kreiptasi 2013-03-25]

[12] *Microsoft Developer Network* [Tinkle]. Prieiga per internetą: http://msdn.microsoft.com/en-US/. [kreiptasi 2012-03-17]