VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
KOMPIUTERIJOS KATEDRA

Baigiamasis bakalauro darbas

Hibridinių mobiliųjų programėlių karkasų analizė ir galimybės

Hybrid Mobile Application Frameworks: Analysis and Facilities

Atliko: 4 kurso, 2 grupės studentas

Ugnius Petrokas (parašas)

Darbo vadovas:

lektorius Kazimieras Mickus

Vilnius
2017

Turinys

Sutartinių terminų sąrašas

WebView – speciali aplinka, per kurią veikia hibridinė programėlė. Tai tarsi naršyklės langas, be grafinių elementų, paliktas rodyti programėlės turinį per visą ekraną (plg. [Bri15]).

XSS (Cross-Site Scripting) – pažeidžiamumas, kuris leidžia įsilaužėliui įterpti savo kodą į internetinį puslapį. Savavališkai įterptas kodas yra vykdomas kito vartotojo naršyklėje. Naudojamas tokiems tikslams, kaip užgrobti naudotojų sesijas, įterpti kenkėjišką turinį, nukreipti vartotojus į kitus puslapius. Išskiriami 3 pagrindiniai tipai: DOM tipo XSS (angl. Document Object Model Based XSS), išsaugotas (angl. stored, persistent), atspindintis (angl. reflected) (plg. [Ver16]).

Anotacija

Pagrindinis baigiamojo darbo tikslas – išanalizuoti mobiliųjų programėlių hibridinius karkasus, atskleisti jų privalumus ir trūkumus. Šiam tikslui įgyvendinti buvo iškelti konkretūs uždaviniai: išanalizavus teorinę medžiagą, sukurti mobiliąsias gimtąsias ir hibridines programėles, kuriomis bus bandoma pasiekti mobiliųjų telefonų techninę įrangą. Galutinis darbo rezultatas – naudojant pasirinktus karkasus ir įrankius sukurtos veikiančios gimtosios ir hibridinės programėlės. Hibridinės programėlės kurtos naudojant hibridinius karkasus PhoneGap, Framework7, Ionic. Pagal sukurtas programėles buvo atlikta mobilaus telefono naudojamų resursų analizė. Darbe taip pat aprašomas hibridinių programėlių kūrimo procesas ir iškilusios problemos.

Summary

Hybrid Mobile Application Frameworks: Analysis and Facilities

The main aim of the bachelor thesis is to analyse hybrid frameworks, to reveal their strengths and weaknesses. To achieve this goal, specific tasks, such as to create native and hybrid applications and to try to access the phone hardware, were set. The final results of the work are functioning native and hybrid applications which were created by using selected frameworks and tools. The hybrid applications were created by using the frameworks PhoneGap, Framework7, Ionic. Following this, analysis of the mobile resources usage was conducted. The work contains description of hybrid application development process and the problems that I have faced.

Įvadas

Su augančiu žmonių poreikiu ir sparčiai besivystančiomis technologijomis programėlių paklausa taip pat augo. Todėl buvo pradėta ieškoti būdo, kaip galima spręsti susidariusią problemą, nes tiek gimtųjų programėlių, tiek žiniatinklio programų nebeužteko. Egzistavo ir tarpplatforminės kliūtys – norint sukurti programėles, kurios veiktų skirtingose platformose, reikėjo skirtingų programavimo kalbų žinių. Pats programėlių kūrimo procesas buvo brangus ir ilgas. Šioje situacijoje HTML5 buvo ta technologija, kuri pasiūlė sprendimą (pgl. [Gle16]). Ji panaikino tarpplatformines kliūtis – programos kodas buvo rašomas vienas ir paleidžiamas skirtingose operacinėse sistemose. Todėl sutrumpėjo programinio kodo rašymo laikas, sumažėjo kaštai.

Baigiamojo darbo pasirinkta tema – hibridinių mobiliųjų programėlių karkasų analizė ir galimybės. Darbe aprašomi programėlių tipai, karkasai bei įrankiai, kurie padeda kuriant ir testuojant programėles.

Baigiamojo darbo tikslas – išanalizuoti hibridinius karkasus, atskleisti jų privalumus ir trūkumus.

Baigiamojo darbo uždaviniai:

1. išanalizuoti teorinę medžiagą siekiant nustatyti, kas yra hibridinė programėlė, kuo ji skiriasi nuo gimtųjų programėlių ir žiniatinklio programų, kokie yra šių programėlių tipų privalumai ir trūkumai;
2. atlikti pasirinktų hibridinių karkasų analizę ir palyginimą;
3. sukurti mobiliąsias gimtąsias (angl. native) programėles Android telefonui;
4. naudojant pasirinktus hibridinius karkasus sukurti hibridines programėles, kurios veiktų skirtingose operacinėse sistemose;
5. ištirti, ar tikrai hibridinėmis programėlėmis galima pasiekti mobiliųjų telefonų techninę įrangą: kamerą, akselerometrą, GPS, kompasą;
6. atlikti analizę, kiek mobilaus telefono resursų naudoja sukurtos programėlės.

Ši baigiamojo darbo tema buvo pasirinkta todėl, kad norėjosi ištirti šiuo metu egzistuojančias hibridinių programėlių galimybes, sužinoti skirtumą tarp skirtingų mobiliųjų programėlių tipų, išsiaiškinti, ar tikrai hibridinės programėlės gali veikti taip pat gerai, kaip ir gimtosios, kokie yra hibridinių programėlių karkasai, įrankiai ir kaip jomis yra pasiekiama telefono techninė įranga. Taip pat buvo įdomu išsiaiškinti, ar hibridiniai karkasai yra tinkami kurti sudėtingas mobiliąsias programėles, ar hibridinės programėlės gali būti vertinamos, kaip tinkama alternatyva gimtosioms ir į kokius jų ypatumus programuotojai turėtų atkreipti dėmesį.

1. Mobiliųjų programėlių tipai

Mobilioji programėlė yra skirta prietaisams, tokiems kaip išmanieji telefonai, planšetės. Kiekvieną dieną dėl augančių žmonių poreikio jų yra sukuriama vis daugiau. Galima išskirti 3 pagrindinius tipus (žr. 1 lentelę):

- gimtosios (angl. native) programėlės;
- žiniatinklio (angl. web) programos;
- hibridinės programėlės.

Platforma	Gimtoji programėlė	Žiniatinklio programa	Hibridinė programėlė
Android	Java	Žiniatinklio technologijos	HTML5/CSS/JS (JavaScript)
iOS	Objective C, Swift	Žiniatinklio technologijos	HTML5/CSS/JS
Windows	C#, .NET	Žiniatinklio technologijos	HTML5/CSS/JS

1 lentelė. Programavimo kalbos, naudojamos kiekvienam programėlės tipui pagal operacinę sistemą.

Kiekvienai gimtajai programėlei sukurti reikia skirtingų programavimo kalbų žinių. Pavyzdžiui, gimtosioms Android programėlėms yra reikalinga JAVA programavimo kalba, iOS – Objective C arba Swift, o Windows – C# arba .NET. Žiniatinklio programoms kurti galima naudoti visas žiniatinklio technologijas (pavyzdžiui: HTML, CSS, JavaScript, PHP), o hibridinėms programėlėms kurti reikalingos HTML5, CSS ir JavaScript.

1.1. Gimtosios programėlės

Gimtosios (angl. native) programėlės yra kuriamos specialiai vienai platformai. Programėlė yra kompiliuojama į mašininį kodą ir paleidžiama tiesiogiai platformoje. Kadangi šio tipo programėlės yra sukurtos specialiai vienai platformai, jos gali pasiekti visą mobilaus prietaiso techninę įrangą: kamerą, akselerometrą, GPS. Tačiau jos gali veikti tik vienoje platformoje ir yra brangios pinigų ir laiko atžvilgiu. Gimtosios programėlės yra dažniausiai naudojamos, kai kūrėjai nori uždirbti pinigų iš sukurto produkto, kuris pasižymi aukštu funkcionalumu, didele apsauga, privatumu ir mokamu turiniu (pgl. [Sta16]). Daug pavyzdžių būtų galima rasti mobiliųjų žaidimų srityje. Žaidimai su daug grafinių elementų reikalauja nemažai telefono resursų, ir būtent gimtosios programėlės leidžia juos geriau išnaudoti negu hibridinės.

1.2. Žiniatinklio programa

Žiniatinklio (angl. web) programos gali būti paleidžiamos kiekvienoje naršyklėje. Tai vyksta dėka HTML5 – interpretuojamos hiperteksto žymėjimo kalbos. Programos kodas nėra paleidžiamas tiesiogiai, bet pirmiausia paleidžiamas per interpretatorių, kuris šiuo atveju yra naršyklė (ji atvaizduoja hipertekstą). Žiniatinklio programa turėtų būti naudojama, kai reikia saugoti ir apdoroti didelį kiekį duomenų – pavyzdžiui, bet kokio internetinio puslapio informacija telefone nėra saugoma, o tik atvaizduojama, kas yra duomenų bazėje, ir visi sudėtingi skaičiavimai (jeigu jų reikia) vyksta serveryje. Šio tipo programų kūrėjai už jas negauna jokio atlygio tiesiogiai, tačiau gali pasipelnyti iš jose rodomų reklamų arba mokamo turinio. Šio tipo programos yra labai populiarios, nes naudoja naršyklę kaip klientą, kartu supaprastindamos naudojimąsi jomis, nes nereikia jokių papildomų įrašymų, atnaujinimų – naudotojai visada galės pasiekti naujausią programos versiją. Geriausi šio tipo programų pavyzdžiai būtų įvairūs socialiniai tinklai – Facebook, Twitter ir kiti.

1.3. Hibridinės programėlės

Hibridinė programėlė yra gimtųjų ir žiniatinklio programų sujungimas, kuriuo bandoma suderinti gerąsias kiekvienos programos savybes, t.y. kad hibridinė programėlė būtų pasiekiama, kaip žiniatinklio, ir turtinga funkcionalumu, kaip gimtoji programėlė. Nors jos pagrindas ir yra HTML5, ji paleidžiama ne tiesiogiai per paprastą naršyklę, o per specialią aplinką, kuri veikia kaip naršyklė – WebView (plg. [Sta16]). Hibridines programėles gana paprasta apmokestinti (įdedant jas į Google Play ir pan.), jos yra prieinamos plačiai naudotojų rinkai, gali saugoti didelį kiekį informacijos, veikti keliose platformose ir pasiekti telefono techninę įrangą. Programos kūrėjai, kurdami bet kokio tipo programėles, susiduria su daugybe problemų, kurias lemia programėlės funkcionalumas, reikalavimai, kūrimo įrankiai, naudojamos technologijos. Tačiau yra keletas problemų rūšių, kurios gana dažnos ir būdingos kuriant hibridines programėles. Šios problemos būtų:

- informacijos pasiekiamumas;
- prieiga prie interneto ryšio ir galimybė dirbti be jo;
- sąveika tarp gimtųjų programėlių ir žiniatinklio programų turinio.

Informacijos pasiekiamumas yra svarbus, nes programos naudotojas neturėtų prarasti jokios informacijos, turėtų turėti galimybę naudotis programa tiek esant, tiek dingus interneto ryšiui. Todėl naudojamas kelių tipų informacijos kaupimas (pgl. [Dha11]). Pirmiausia yra saugoma sesijos informacija. Ji trunka, kol programos langas yra atidarytas. Žinoma, ši informacija yra riboto dydžio, nėra siunčiama automatiškai į serverį ir todėl jo neapkrauna. Lokali talpykla naudojama ne vien saugoti laikiną informaciją, bet ir kaupti ją ilgesniam laikui. Todėl informacija bus pasiekiama visą

programos naudojimo laikotarpį. O naudojant interneto ryšį būtų galima tiek išsiųsti reikiamą informaciją į serverį, kad šis duomenų bazėje ją saugotų, tiek atvirkščiai – parsiųsti reikiamą informaciją ir ją atvaizduoti.

Hibridinė programėlė gali būti ir labai paprasta, ir sudėtinga. Viskas priklauso nuo daugybės dalykų – programėlės pagrindinio tikslo, funkcionalumo, efektyvumo, turinio, saugumo.

Šiomis dienomis internetas suteikia galimybę manipuliuoti dideliu kiekiu informacijos ir paslaugomis, pavyzdžiui, Facebook, Google, Twitter, Instagram, Google Maps, Amazon ir daugybė kitų. Visos šios paslaugos nėra lengvai pasiekiamos per mobiliuosius telefonus dėl pelės, klaviatūros trūkumo, mažo ekrano. Todėl matyti akivaizdus mobiliųjų telefonų naršyklių trūkumas. Būtent čia gali padėti hibridinės programėlės. Jos gali turėti gimtųjų funkcionalumą, kuris gali išplėsti tam tikras programos galimybes, suteikti gražią naudotojo sąsają ir padaryti reikiamą informaciją lengvai pasiekiamą. Todėl dauguma populiarių paslaugų gali būti pasiekiamos per telefoną daug lengviau, nei naudojant telefone įdiegtą naršyklę (žinoma, viskas priklauso nuo to, kaip buvo realizuotas programėlės funkcionalumas).

Dalis hibridinių programėlių funkcionalumo yra saugoma serveryje. Tai suteikia papildomą privalumą – mažesnė užimama vieta telefone, greitesnis įdiegimo procesas, lengviau atnaujinti programą. Pati programėlė, įdiegta mobiliajame telefone, gali naudoti pagrindinį funkcionalumą, tačiau sudėtingi skaičiavimai gali būti atliekami serveryje ir gauti rezultatai atsiunčiami atgal į programėlę. Tokiu būdu tausojami resursai.

Kita vertus, hibridinių programėlių galimas trūkumas būtų reikalavimas, kad jos veiktų ir be interneto ryšio (jeigu toks funkcionalumas turi būti realizuotas). Tokiu atveju dalis programos funkcionalumo bus prarandama. Tačiau tokios problemos yra išsprendžiamos. Pavyzdžiui, programėle Twitter galima parašyti viešai prieinamą žinutę ir tuo metu nesant prisijungus prie interneto. Žinutė bus saugoma laikinai, kol bus atnaujintas interneto ryšis ir šiam atsiradus žinutė bus išsiųsta. Kita didelė problema yra saugumas. Kadangi hibridinė programėlė turi žiniatinklio bruožų, šias programėles lengva nulaužti, jų veikimas gali būti sustabdomas priverstinai dėl kibernetinių nusikaltėlių kaltės. Gali būti įvykdytos dažnos žiniatinklio technologijų atakos, tokios kaip XSS.

1.4. Kokį programėlių tipą pasirinkti?

Kaip jau minėta, hibridinė programėlė yra gimtųjų ir žiniatinklio programų kombinacija, su abiejų ypatybėmis ir funkcionalumu. Tačiau hibridinės programėlės negali visiškai pakeisti vieno ar kito. Yra situacijų, kai labiau apsimoka kurti gimtąsias ar žiniatinklio programas nei jų kombinaciją.

Kokiu atveju kokį programėlių tipą pasirinkti, kad jis atitiktų visus jų kūrėjų ir naudotojų lūkesčius, žr. 2 lentelę (pgl. [Sta16], [Cal16]).

	Gimtosios programėlės	Žiniatinklio programos	Hibridinės programėlės
Funkcionalumas (prieinama telefono techninė įranga, ištekliai)	Didelis	Ribotas	Didelis
Sukūrimo kaštai (pinigai)	Dideli	Palyginti maži (priklauso nuo turinio, funkcionalumo)	Palyginti maži (1 programėlė – daug platformų)
Sukūrimo laikas	Didelis	Vidutinis	Mažas
Palaikymo kaštai	Dideli	Vidutiniai	Vidutiniai
Privatumas, saugumas	Gana saugios	Nėra saugios dėl sparčiai besivystančių technologijų	Nėra saugios dėl sparčiai besivystančių technologijų
Našumas	Didelis (galima pilnai išnaudoti telefono resursus)	Vidutinis	Vidutinis
Rinka	Apribota vartotojų rinka (tai priklausančiai platformai, kuriai buvo kurta)	Plati vartotojų rinka (visi kas turi naršyklę telefone)	Plati vartotojų rinka (skirtingų platformų rinkos)
Apmokestinamumas (iš kūrėjų pusės)	Lengvas	Vidutinis (gali gauti pajamų iš reklamų rodymo, mokamo turinio rodymo naudotojams)	Lengvas

Interneto prieiga	Nėra privaloma (priklauso nuo reikalingo funkcionalumo)	Privaloma	Nėra privaloma
Įdiegimas (iš naudotojo pusės)	Privalomas	Nereikalingas (svarbu, kad būtų įdiegta naršyklė)	Privalomas
Atnaujinimai (iš naudotojų pusės)	Nesudėtinga	Nieko naudotojui nereikia pačiam daryti	Nesudėtinga

2 lentelė. Programėlių tipų atitikimas kriterijams.

Gimtosios programėlės yra brangios laiko ir pinigų atžvilgiu, nes, norint sukurti programėlę, kuri veiktų skirtingose platformose, gali prireikti skirtingų programuotojų komandų. Tačiau jos pasižymi našumu, gali labiau išnaudoti procesoriaus ir operatyviosios atminties resursus, lyginant su žiniatinklio ir hibridinėmis programėlėmis. Todėl jos labiau tinka kurti žaidimus ar tokiu atveju, kai naudotojui reikalingi grafiniai elementai, kurie naudoja daug resursų.

Žiniatinklio programa išsiskiria tuo, kad iš naudotojo pusės nereikalingi jokie papildomi įdiegimai (išskyrus tada, kai nėra pačios naršyklės telefone) ir atnaujinimai. Kartais gali būti sunku atvaizduoti visą turinį mažame telefono ekrane, todėl naudojami papildomi karkasai (pavyzdžiui, Bootstrap), kad turinys prisitaikytų prie ekrano dydžio. Tačiau šios programos nėra saugios, nes kartu su sparčiai besivystančiomis technologijomis vystosi ir įsilaužimo technikos, saugumo spragų išnaudojimo būdai. Taip pat verta paminėti, kad jos negali pasiekti visos telefono techninės įrangos.

Hibridinė programėlė turi žiniatinklio ir gimtųjų programėlių savybių. Ji gali būti ne tik lengvai prieinama, kaip žiniatinklio (įdiegta skirtingose platformose), bet ir turtinga funkcionalumu, kaip gimtoji. Sukūrimo kaštai yra maži (lyginant su gimtosiomis), nes vieną parašytą programos kodą galima paleisti skirtingose platformose (žinoma, reikia sugeneruoti skirtingus įdiegimo failus kiekvienai platformai atskirai).

Tačiau renkantis pagal reikiamus naujos programėlės kriterijus, funkcionalumą, naudotojų ratą, vieni programėlių tipai gali būti tinkamesni nei kiti (žr. 1 priedo 1 diagramą).

2. Hibridinių mobiliųjų karkasų analizė, galimybės, palyginimas

Kuriant hibridines programėles, svarbu skirti laiko karkasų paieškai, analizei. Norint sukurti iš tiesų gerą hibridinę programėlę, reiktų stengtis naudoti naujausius įrankius, technologijas. Todėl didelį dėmesį reikia kreipti ir į pačius karkasus – jų naujoviškumą, stabilumą, prieinamumą, galimybes. Juk niekam nesinorėtų įpusėjus programėlių kūrimo procesui suvokti, kad vieno ar kito planuoto funkcionalumo dėl ribotų karkaso galimybių nepavyks įgyvendinti.

Šiai analizei karkasai pasirinkti pagal šiuos kriterijus: populiarumą (pgl. [G2C16]), suteikiamus įrankius, grafinius komponentus (plg. [Fal16]).

2.1. Ionic

Ionic karkasas dažniausiai laikomas vienu populiariausių ir labiausiai mėgstamų tarp programėlių kūrėjų (plg. [G2C16]). Galima pasitelkti CSS įgūdžius, kurie padėtų sukurti gimtosios programėlės įvaizdį, tačiau, norint išnaudoti visas Ionic galimybes, rekomenduojama kartu naudoti ir AngularJS. Todėl Ionic naudoja MVC (angl. model – view – controller) būdą. Didžiausias dėmesys yra skiriamas našumui. Ionic turi CLI (angl. Command Line Interface), kuris skirtas kurti, testuoti, paleisti programėles su gerai atrodančiais komponentais. Oficialiame Ionic puslapyje (plg. [Ion16]) galima rasti gerai sutvarkytą, aiškią dokumentaciją su sukurtu forumu, kuris skirtas programėlių kūrėjams, susiduriantiems su įvairiais sunkumais kūrybos procese. Ionic palaiko Android, iOS, (Ionic2 patobulinta versija, po didelio pasisekimo palaiko ir Windows Phone).

2.2. Onsen UI

Onsen UI yra atvirojo kodo karkasas, kuris leidžia kurti programėles, kurios atrodo kaip gimtosios. Sprendžiant iš atsiliepimų (plg. [G2C16b]) apie šį karkasą, jis yra lengvai naudojamas, jį galima naudoti tiek su AngularJS, tiek be. Oficialiame puslapyje (plg. [Mot16]) galima rasti aiškią dokumentaciją su daugybe dažniausiai kuriamų programėlių pavyzdžių. Didelis dėmesys skiriamas našumui. Šis karkasas padeda atvaizduoti gimtųjų programėlių išvaizdą – Android, iOS, Windows. Jis suteikia CLI (angl. Command Line Interface), kuris vadinamas Monaca CLI ir skirtas lengvesniam programėlių kūrimui. Taip pat žmonės, kurie labiau mėgta grafinę naudotojo sąsają, gali parsisiųsti Monaca Localkit (lokaliai darbo aplinkai). Šis karkasas turi papildomų įrankių, kurie palengvina kūrimo ir testavimo procesus. Be to, programėlių kūrėjams jis leidžia naudoti savo mėgstamus IDE (angl. Integrated Development Enviroment), tokius kaip: Eclipse, Visual Studio, Dreamweaver, Sublime. Jis suteikia ir prieigą prie versijų kontrolės sistemų, pavyzdžiui, Git.

2.3. Framework 7

Nemokamas atvirojo kodo karkasas, specialiai skirtas iOS ir Android. Jis leidžia hibridinėms programėlėms suteikti gimtųjų išvaizdą ir naudojimosi jausmą. Oficialiame puslapyje (plg. [Kha16]) galima rasti dokumentaciją, forumą, programėlių pavyzdžius, maketus, jau paruoštus komponentus. Jis suteikia visišką laisvę kūrėjams. Tačiau jis, deja, nesuteikia jokių papildomų įrankių (pavyzdžiui, programinio kodo rašymo aplinkos, programėlių komponentų tvarkymo, įskiepių įdėjimo). Todėl siūloma šį karkasą naudoti su kitais įrankiais, karkasais.

2.4. PhoneGap

PhoneGap leidžia programų kūrėjams naudoti visą telefono techninę įrangą (Cordova įskiepių (angl. plug–in) dėka). Jis leidžia kurti hibridinio tipo programėles naudojant HTML, CSS, JavaScript ir tada kviečia gimtąjį API (angl. Application Programming Interface), pavyzdžiui: kamerą, kompasą ir kita. Taip pat jis suteikia daug visokių įskiepių ir įrankių. Šie įskiepiai yra sukurti pačios programuotojų bendruomenės ir yra laisvai prieinami per GitHub, o patys įrankiai yra prieinami oficialiame PhoneGap internetiniame puslapyje (plg. [Ado16]).

Naudojant žiniatinklio technologijų žinias, galima greitai kurti programėles. PhoneGap suteikia tarpplatformines galimybes – leidžia sukurti programėles kelioms platformoms remiantis vienu programos kodu. Taip pat jis leidžia pasiekti visus telefono API, tokius kaip kamerą, GPS, akselerometrą ir kt. Taigi, programėlė sukurta remiantis žiniatinklio technologijomis, bet gali funkcionuoti kaip gimtoji programėlė.

PhoneGap suteikia įrankius: PhoneGap CLI, PhoneGap Dekstop App, PhoneGap Developer App ir PhoneGap Build.

PhoneGap CLI – labai galingas ir lankstus įrankis. Jis leidžia kurti, kompiliuoti, paleisti programėles per komandinės eilutės terminalą. Jis veikia kiekvienoje platformoje ir leidžia dirbti tiek lokaliuoju būdu, tiek serveryje (plg. [Ado16]).

Dekstop App – lengviausias būdas naudotis PhoneGap teikiamomis paslaugomis. Juo galima kurti programėles ir lengvai perkelti jas į mobilųjį telefoną. Tai yra alternatyva PhoneGap CLI, nors abu naudoja tas pačias bibliotekas. Tiesiog nereikia įsiminti komandų ar rankiniu būdu diegti priklausomybių (angl. dependencies) – viskas yra iškarto padaroma įdiegimo metu. Palaikomas MAC ir Windows operacinėse sistemose.

PhoneGap Developer App – mobili programėlė, kuri leidžia greitai pamatyti, kaip sukurta programėlė atrodo ir veikia telefone. Programos kodą galima rašyti pas save kompiuteryje ir iš karto matyti pokyčius telefone. Norint atlikti parašyto programos kodo testavimą, nereikia jokio

papildomo kompiliavimo, įdiegimo pačiame telefone. Taip pat tokiu būdu galima prieiti prie mobilaus prietaiso API.

PhoneGap Build leidžia nusiųsti projekto failus į sukurtą debesų kompiuteriją, pasirinkti norimas operacines sistemas, ir tada jis automatiškai sugeneruoja programėlės įdiegimo failus (plg. [Ado16]).

2.5. jQuery mobile

jQuery mobile – paprastas, nemokamas karkasas, juo remiantis sukurtos programėlės paprastai nepanašios į gimtąsias. Tačiau pagrindinis karkaso tikslas yra suteikti galimybę kurti programėles, kurios veiktų vienodai gerai bet kokioje platformoje – tiek populiariausiose (Android, iOS, Windows (plg. [Idc16])), tiek mažiau naudojamose (Symbian, Blackberry, Bada). Labai daug papildomų įrankių nesuteikiama, tačiau su ThemeRoller įrankiu galima keisti komponentų spalvas, temas (angl. theme), šablonus (angl. template) ir matyti pokyčius realiu laiku (plg. [jQu16]). Taip pat leidžiama parsisiųsti tik tuos komponentus, kurių iš tikrųjų reikia, taip neapkraunant programėlės. Tiesa, našumo atžvilgiu šis karkasas negali labai išsiskirti iš kitų (pgl. [G2C16c]). Dokumentacija, palyginti su kitais karkasais, yra gana skurdi. Tiesa, yra programėlių kūrimo forumas.

2.6. Famous

Famous karkasas išsiskiria savo unikaliu programėlių kūrimo būdu. Jis jungia DOM (angl. Document Object Model) medį su WebGL (angl. Web Graphics Library), kuris yra JavaScript API, skirtas atvaizduoti 2D, 3D grafinius elementus bet kurioje naršyklėje be jokių papildomų įskiepių (plg. [Fam15]). Todėl šis karkasas leidžia veikti programėlių 60fps (angl. frames per second) dažniu, o tai labai primena gimtųjų programėlių naudojimosi jausmą. Dokumentacija – skurdi, ne iki galo aprašyta, maža bendruomenė. Nesuteikiami jokie papildomi įrankiai, kurie palengvintų programėlių kūrimą.

2.7. Sencha Touch

Sencha Touch yra MVC pagrindu paremtas JavaScript karkasas, skirtas kurti tarpplatformines programėles. Jis palaiko daug platformų, tokių kaip Android, iOS, BlackBerry, Windows ir kt. Teigiama, kad Sencha Touch karkasas yra tinkamiausias organizacijoms, įmonėms, norinčioms sukurti aukšto lygio programėles (plg. [Sen16]). Didelis dėmesys skiriamas jų našumui, kartu suteikiama ir gimtųjų išvaizda bei jausmas. Tačiau reikia daug laiko, kad būtų galima pritaikyti visą karkaso teikiamą funkcionalumą ir galimybes. Todėl laiko atžvilgiu tai nėra pats geriausias variantas. Sencha Touch suteikia šiuos įrankius: Sencha Touch Ext JS, Sencha Test, Sencha GXT, Architect.

Sencha Touch Ext JS – visapusiškas JavaScript karkasas, turtingas funkcionalumu, tarpplatforminėmis galimybėmis. Leidžia kurti programėles telefonams, planšetėms (plg. [Sen16]).

Sencha Test – programėlių testavimo įrankis, kuris užtikrina testavimo kokybę ir sumažina laiko sąnaudas (plg. [Sen16]).

Sencha GXT – karkasas, paremtas Java ir skirtas kurti turtingo turinio programėles. Turi GWT kompiliatorių, kuris leidžia kūrėjui rašyti programėlės kodą Java programavimo kalba ir kompiliuoti jį į HTML5 kodą (plg. [Sen16]).

Architect leidžia lengviau kurti programėles naudojantis „paimti“ ir „įdėti“ galimybe. Yra speciali naudotojo sąsaja, kurioje naudotojas gali manipuliuoti programėlės komponentais (plg. [Sen16]). Todėl programavimas užima nedaug laiko, bet programėlė vis vien atitinka aukščiausio našumo reikalavimus.

Deja, viskas yra mokama. Tačiau yra suteikiama galimybė išbandyti visus įrankius nemokamai 30 dienų laikotarpiu.

2.8. Appcelerator Titanium

Šis karkasas leidžia kurti hibridines programėles greitai ir lengvai. Jis suteikia prieigą prie telefono techninės įrangos. Taip pat siūloma nemažai papildomų paslaugų, tokių kaip žinučių siuntimas naudotojams, debesų kompiuterijos paslaugos (programėlių valdymas ir kita). Verta paminėti, kad pats Titanium Appcelerator naudoja Alloy (MVC karkasas, kuris palengvina programinio kodo rašymą, taupo laiką). Taip pat Appcelerator Titanium suteikia papildomus įrankius, paslaugas (pavyzdžiui, siųsti pranešimus programėlių naudotojams), naudoti jau parašytus modulius (pavyzdžiui, naudotojų autentifikacija, reklamų rodymas ir kt.) programėlių kūrėjams (plg. [App16]). Tiesa, jis yra mokamas.

2.9. Karkasų palyginimas

Hibridinių programėlių karkasai yra skirtingi, kiekvienas turi savų privalumų ir trūkumų (žr. 3 lentelę). Pavyzdžiui, Ionic, PhoneGap, Appcelerator Titanium, Sencha Touch palaiko daug platformų, tokių kaip Windows, iOS, Android, BlackBerry, kai tuo tarpu Framework7 tik 2: iOS ir Android. Famous ir Framework7 nesuteikia jokių papildomų įrankių, paslaugų, palengvinančias programėlių kūrimą. Reikėtų paminėti, kad beveik visi karkasai turi aiškias, išsamias dokumentacijas, kūrėjų bendruomenes, kur žmonės gali prašyti pagalbos vienu ar kitu klausimu. Kelis karkasus galima naudoti kartu, geras pavyzdys būtų PhoneGap su Framework7, nes PhoneGap suteikia prieigą prie skirtingų platformų, o Framework7 pačius šablonus, dizainą, komponentus. Žinoma, reikia atkreipti dėmesį į programėlių paskirtį ir funkcionalumą. Jeigu reikia didelio našumo, dirbama didelėje komandoje ir reikalingi papildomi įrankiai, tai Sencha Touch, Appcelerator Titanium būtų geri pasirinkimai. Nors jie ir yra mokami, tačiau suteikiami privalumai būtų to verti.

Karkasas	Trūkumai	Privalumai
Ionic	Ionic 1 palaiko tik Android ir iOS platformas.	Naudojamas kartu su AngularJS, kuris praplečia galimybes. Suteikia stilius, komponentų dizainą. Aiški dokumentacija. Didelė kūrėjų bendruomenė.
Onsen UI	Maža kūrėjų bendruomenė.	Naudojamas su AngularJS. Didelis dėmesys skiriamas našumui. Aiški dokumentacija. Siūlomi populiariausi programėlių šablonai.
Framework7	Palaiko tik Android ir iOS platformas. Neturi jokių papildomų įrankių.	Suteikia hibridinėms programėlėms gimtųjų išvaizdą, naudojimosi pojūtį.
PhoneGap	Nesiūlo jokių šablonų ar gražiai atrodančių gimtųjų komponentų.	Turi daug papildomų įrankių, kurie palengvina hibridinių programėlių testavimą, kūrimą. Galima naudoti su karkasais, tokiais kaip Framework7, Ionic ir kitais. Suteikia prieigą prie WebView, kuriame ir veikia hibridinės programėlės.
JQuery mobile	Skurdi dokumentacija. Neskiria dėmesio našumui.	Suteikia daug gražiai atrodančių komponentų, kurie atrodys vienodai gerai visose platformose.
Famous	Skurdi dokumentacija. Trūksta palaikymo. Maža bendruomenė. Nesuteikia jokių papildomų įrankių.	Padės atvaizduoti 2D, 3D elementus. Palaiko 60 kadrų per sekundę.
Sencha Touch	Mokamas.	Didelis dėmesys skiriamas našumui, kartu suteikiama ir gimtųjų programėlių išvaizda ir naudojimosi jausmas. Palaiko daug platformų. Suteikia daug įrankių. Remiasi MVC.
Appcelerator Titanium	Mokamas	Suteikia prieiga prie telefono techninės įrangos. Palaiko daug skirtingų platformų. Remiasi MVC. Suteikia daug papildomų įrankių, paslaugų, modulių.

3 lentelė. Karkasų palyginimas.

Remdamasis atlikta karkasų analize ir palyginimu, savo tolesniam darbui pasirinkau šiuos karkasus: PhoneGap, Framework7 ir Ionic. Visi trys karkasai yra nemokami.

PhoneGap buvo pasirinktas todėl, kad jis neturi jokių nereikalingų komponentų, leidžia viską kurti nuo pat pradžių, lengva naudoti įskiepius, kuriais yra pasiekiama telefono techninė įranga. Todėl šis karkasas yra tinkamas resursų analizei atlikti. Framework7 ir Ionic pasirinkti todėl, kad turi daug gražiai atrodančių grafinių elementų, pritaikytų skirtingoms platformoms (jei to reikia), yra lengvi naudoti ir turi aiškias dokumentacijas.

3. Gimtosios Android programėlės

Norinti suprasti hibridines programėles, jų veikimą, kūrimo procesą, reikia pirmiausia išbandyti ir gimtąsias programėles. Vienas iš iškeltų uždavinių buvo sukurti gimtąsias programėles, veikiančias Android platformoje, ir jomis pasiekti telefono techninę įrangą: kamerą, GPS, kompasą ir akselerometrą. Android platforma buvo pasirinkta todėl, kad ji yra pati populiariausia (remiantis 2016 m. trečio ketvirčio duomenimis (plg. [Idc16]). Taip pat todėl, kad neturėjau patirties ir žinių kurti gimtąsias programėles kitoms platformoms, nes jos reikalauja specifinių programavimo kalbų žinojimo (pavyzdžiui, iOS reikia Objective C arba Swift (ir Mac kompiuterio), o Windows reikia C# arba .NET). Be to, norint sukurti įdiegimo failus, reikėtų turėti programėlių kūrėjo sertifikatus, kurie yra mokami.

Pačios programos buvo kuriamos su Android siūlomu oficialiu kūrimo įrankiu Android Studio (plg. [And16a]). Dalis sukurtų programėlių buvo iš pradžių testuojamos emuliatoriuose. Nors Android Studio ir suteikia prieigą prie emuliatorių, tačiau programėlės paleidimas (įskaitant ir emuliatoriaus sukūrimą, paleidimą) užėmė daug laiko (~5 minutes). Todėl teko ieškoti greitesnio sprendimo: pradėta naudoti GenyMotion, kuris skirtas Android programėlių testavimui ir siūlo populiariausių telefonų modelių emuliatorius. Pats testavimo procesas sutrumpėjo iki 1 minutės.

Pats programos kodas buvo rašomas JAVA programavimo kalba.

Pirmoje programėlėje buvo pasiekiamas akselerometro sensorius ir realiu laiko atvaizduojamos gautos reikšmės. Iš sensoriaus grąžinamas masyvas, kurio pirmoji reikšmė X ašis, antroji Y ašis, trečioji Z ašis (žr. 2 priedo 1 paveiksliuką).

Antroje programėlėje buvo naudojamas GPS. Pirmiausia reikėjo nustatyti prieigos teises su naudotojo susitikimu. Pridėtas mygtukas, kurį paspaudus bus kviečiamos reikalingos funkcijos. Taip pat reikia pabrėžti, kad gauti GPS duomenis gali užtrukti iki 3 minučių, viskas priklauso nuo oro sąlygų (debesuotumas ir kita), todėl, norint gauti informaciją sparčiau, rekomenduojama naudoti programėlę su interneto prieiga. Galima gauti ilgumos, platumos koordinates, gautų koordinačių tikslumą, judėjimo greitį, aukštį virš jūros lygio ir kt. Šioje programėlėje pasirinkau pademonstruoti tik gautas ilgumos ir platumos reikšmes (žr. 3 priedo 1 paveiksliuką).

Trečioje programėlėje buvo naudojamas kameros kvietimas. Kaip ir antrojoje programėlėje, pirmiausia reikėjo suteikti prieigos teises. Antra, reikėjo nurodyti, kur nuotrauka bus išsaugota (šiuo atveju buvo sukuriamas naujas katalogas, kuriame bus išsaugota padaryta nuotrauka) ir parodyti nufotografuotą vaizdą ekrane (žr. 4 priedo 1 paveiksliuką). Reikia paminėti, kad Samsung Galaxy

S6 modelyje programėlė nevisiškai veikia. Nėra rodomas padarytos nuotraukos vaizdas ekrane. Tačiau ji puikiai veikia Samsung Galaxy S5 mini modelyje. Kodėl taip yra, išsiaiškinti nepavyko.

Ketvirtoje programėlėje reikėjo gauti kompaso duomenis. Visu pirma reikėjo kviesti sensorių, kaip ir antroje programoje. Sensorius grąžina skaičius nuo 0 iki 359.99, įskaitant ir kelis skaitmenis po kablelio. Todėl teko suapvalinti gautus skaičius. Taip pat buvo pridėti du paveiksliukai – kompaso galvutė (pats pagrindas, kur sugraduoti laipsniai) ir adata. Kviečiama sukimo animacija, kuri pagal gautus sensoriaus duomenis pasuka kompaso adatą (žr. 5 priedo 1 paveiksliuką).

Kaip jau minėta, visos parašytos programėlės iš pradžių buvo bandomos emuliatoriuose. Vėliau tos programėlės buvo išbandomos ir su realiu prietaisu – Samsung Galaxy S6 telefonu. Norint sugeneruoti įdiegimo failus, reikėjo iš pradžių sugeneruoti raktų saugyklą (angl. key store), o to reikėjo pasirašyti sukurtą programėlę (pgl. [And16b]). Visi šitie reikalavimai yra dėl saugumo sumetimų. Kitu atveju leidžiama sugeneruoti instaliavimo failus, tačiau kai bus bandoma įrašyti programėlę į telefoną, bus metamas klaidos pranešimas. Android Studio leidžia greitai sugeneruoti raktų saugyklą, pasirašyti sukurtą programėlę ir gauti įdiegimo failą (Android platformoje .apk).

4. Hibridinės PhoneGap programėlės

Hibridinės programėlės yra kuriamos HTML5, CSS, JS kalbomis. Pati programėlė veikia per specialią aplinką, kuri veikia kaip naršyklė – WebView (plg. [Bri15], žr. 6 priedo 1 paveiksliuką). Geriausias pavyzdys būtų Chrome ar Safari naršyklės langas atmetus naudotojo sąsają ir paliekant rodyti per visą ekraną programėlės turinį.

Iš pradžių įdiegiau į kompiuterį PhoneGap Dekstop app iš oficialaus PhoneGap puslapio. Ši programa leidžia sugeneruoti pagrindinius failus (www katalogą, index.html, style.css, js katalogą, config.xml), reikalingus hibridinei programėlei. Testavimo procesui naudojau PhoneGap Developer app programėlę, kurią įdiegiau į mobilųjį telefoną. Keičiant programinį kodą kompiuteryje, realiu laiku galima stebėti pokyčius telefone.

Visose hibridinėse programėlėse buvo naudojami Cordova įskiepiai tam, kad būtų galima pasiekti telefonų techninę įrangą (plg. [Cor15]).

Pirmoje programėlėje (žr. 7 priedo 1 paveiksliuką) turėjo būti pasiekiamas telefono akselerometro sensorius. Javascript funkcijoje buvo kviečiamas navigtor.accelerometer, kuris gauna informaciją iš akselerometro sensoriaus. Jis grąžina X, Y ir Z ašių reikšmes, laiką (kada gautos reikšmės). Taip pat priima papildomą parametrą, kas kiek laiko kreiptis į patį akselerometro sensorių. Klaidos atveju, t.y. nepavykus gauti informacijos iš akselerometro sensoriaus, metamas klaidos kodas.

Antroji programėlė turi rodyti dabartinę padėtį naudojant GPS (žr. 8 priedo 1 paveiksliuką). Buvo kviečiamas navigator.geolocation, kuris gali grąžinti ilgumos, platumos koordinates, duomenų tikslumą, aukštį virš jūros lygio, judėjimo greitį ir kt. Jis gali priimti kelis parametrus, tokius kaip tikslumas, kreipimosi dažnio į GPS ir gauto atsakymo laikus. Klaidos atveju metamas pranešimas su klaidos kodu.

Trečioje programoje buvo naudojamas kameros kvietimas. Kviečiamas navigator.camera, kuris paleidžia kamerą ir grąžina nuotrauką (Base64 koduote). Taip pat priima keletą papildomų parametrų, tokių kaip aukštis, plotis, kokybė, leidimas redaguoti, išsaugojimo vieta ir kita. Įvykus klaidai grąžinamas klaidos pranešimas. Sukurtas paprastas HTML mygtukas, kuris kviečia JavaScript funkciją. Pridėtas paveiksliuko elementas, kuriame bus rodomas sumažintas nufotografuotas vaizdas (žr. 9 priedo 1 paveiksliuką).

Ketvirtoje programėlėje reikėjo gauti kompaso duomenis. Kviečiamas navigator.compass, kuris grąžina laipsnius. Priima du parametrus: dažnį milisekundėmis (kas kiek laiko atnaujinti duomenis) ir laipsnių pokytį (kuris parodo, kas kiek pasikeitusių laipsnių atnaujinti duomenis). Įvykus klaidai

bus metamas klaidos pranešimas, jos kodas. Taip pat sukurti keli HTML elementai, naudojamas JQuery įvykdyti kompaso adatos posūkį (žr. 10 priedo 1 paveiksliuką).

Norint įrašyti hibridines programėles į mobilųjį telefoną naudojant PhoneGap, reikia atsižvelgti į kiekvienos platformos ypatumus. Pavyzdžiui, visos platformos reikalauja pasirašyti sukurtą programėlę (norint įrašyti ją į telefoną), bet pats procesas skiriasi. Android platformai tai galima padaryti sugeneravus raktų saugyklą: „keytool -genkey -v -keystore <KEYSTORE_NAME> alias <ALIAS_NAME> keyalg RSA -keysize 2048 validity 10000“, sukurti release-signing.properties failą (nurodyti storeFile, keyAlias, storePassword, keyPassword), įkelti į Android katalogą ir generuoti pasirašytą .apk įdiegimo failą. Kitose platformose, tokiose kaip Windows, iOS, reikia turėti nusipirkus vadinamuosius kūrėjų sertifikatus (jie yra mokami), kurių reikia norint pasirašyti programėles, ir be šito nebus leidžiama įdiegti programėlių į telefoną. Reiktų paminėti, kad didelis dėmesys skiriamas saugumui ir į App Store ar Google Play nėra leidžiama įkelti programėlių, kurios yra nepasirašytos ar neatitinka saugumo reikalavimų. Taip pat App Store didelį dėmesį skiria programų kokybei. Lėtai veikiančios programėlės yra pašalinamos (pgl. [App16]).

5. Resursų analizė

Nors hibridinės programėlės gali veikti skirtingose platformose, tačiau ar iš tiesų jos naudoja tiek pat resursų, kaip ir gimtosios? Kiek vietos jos užima telefone, koks įdiegimo failų dydis? Norint tai išsiaiškinti, buvo lyginamos jau sukurtos programėlės, kuriose buvo bandoma pasiekti telefono techninę įrangą (kamerą, GPS, kompasą, akselerometrą). Matuoti realiu laiku naudojamus resursus buvo naudojama programėlė Resource Monitor. Mobilusis telefonas – Samsung Galaxy S6. Analizės metu buvo išjungti visi nereikalingi nustatymai, kurie nėra būtini, tokie kaip internetas, geografinės vietos nustatymo sistema ir panašiai. Taip pat pasirinktas vienodas ekrano ryškumas.

Visi gauti rezultatai buvo dedami į lenteles. Tam tikru laiko momentu naudojamos operatyviosios atminties rezultatai pateikiami 4 lentelėje, procesoriaus – 5 lentelėje. Įdiegimo failo, užimamos vietos telefone ir baterijos naudojimo – 6 lentelėje.

	Bandymai										
Programėlė	1	2	3	4	5	6	7	8	9	10	Vidurkis
Gimtoji akselerometras	804	810	791	811	796	807	801	804	805	806	803.5
Hibridinė akselerometras	783	776	777	774	803	782	767	766	730	815	777.3
Gimtoji GPS	873	861	848	864	859	850	840	851	839	840	852.5
Hibridinė GPS	693	691	688	701	698	686	686	699	682	670	689.4
Gimtoji kamera	752	761	760	748	753	758	758	738	726	800	755.4
Hibridinė kamera	745	731	732	713	727	713	715	722	759	799	735.6
Gimtoji kompasas	563	569	594	583	555	572	573	551	576	583	571.9
Hibridinė kompasas	727	718	735	724	721	717	702	717	705	705	717.1

4 lentelė. Operatyvios atminties naudojimas. Matavimo vienetai – KB.

	Bandymai										
Programėlė	1	2	3	4	5	6	7	8	9	10	Vidurkis
Gimtoji akselerometras	25	26	28	28	27	27	31	26	30	31	27.9
Hibridinė akselerometras	29	30	26	27	28	29	24	32	33	28	28.6
Gimtoji GPS	22	16	25	22	27	28	26	28	23	14	23.1
Hibridinė GPS	24	19	25	23	27	29	24	24	27	29	25.1
Gimtoji kamera	2	13	8	11	20	9	3	23	30	2	12.1
Hibridinė kamera	23	7	11	25	20	14	23	3	22	15	16.3

Gimtoji kompasas	31	33	38	13	35	31	14	41	31	18	28.5
Hibridinė kompasas	24	21	24	6	31	11	19	37	25	5	20.3

5 lentelė. Procesoriaus naudojimas. Matavimo vienetai – proc. (procesoriaus naudojimas procentais).

Programėlė	Įdiegimo failas (MB)	Vieta telefone (MB)	Baterijos naudojimas (mA)
Gimtoji akselerometras	1.25	7.08	300
Hibridinė akselerometras	4.74	5.22	288
Gimtoji GPS	1.25	7.12	179
Hibridinė GPS	4.74	5.22	198
Gimtoji kamera	1.25	7.12	277
Hibridinė kamera	4.75	5.31	211
Gimtoji kompasas	1.93	7.77	360
Hibridinė kompasas	4.91	5.43	182

6 lentelė. Įdiegimo failas, užimama vieta telefone, baterijos naudojimas.

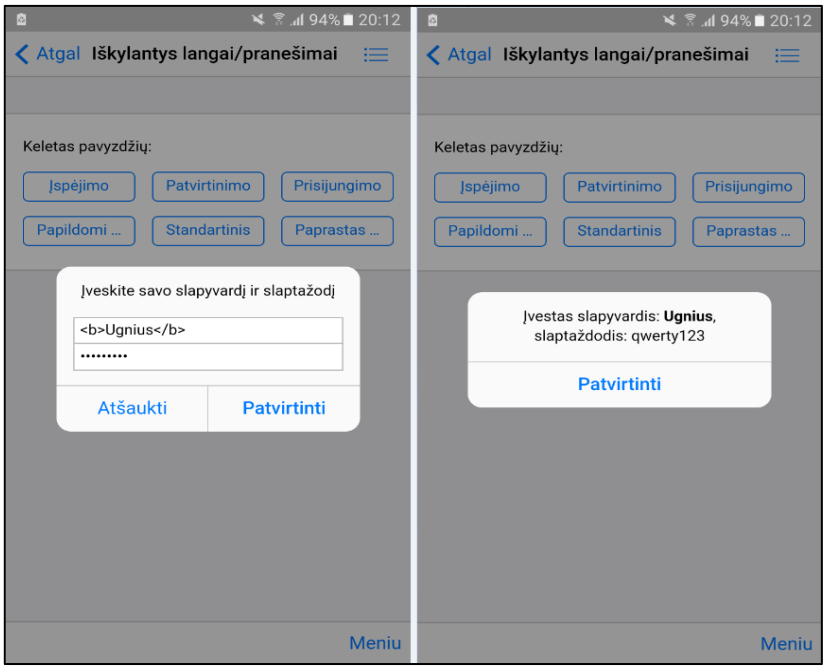
Baterijos naudojimo rezultatai (žr. 6 lentelę) buvo fiksuojami, naudojant programėlę 3C Battery Monitor. Buvo pasirinktas 10 minučių laiko tarpas, kurio metu buvo naudojamos sukurtos hibridinės, gimtosios programėlės ir praėjus tam laiko tarpui, gauti rezultatai buvo fiksuojami. Įdiegimo failas ir programėlių užimama vieta tikrinta sukūrus jau galutines programėles.

Remiantis gautais rezultatais, galima daryti išvadą, kad tiek hibridinės, tiek gimtosios programėlės naudoja maždaug tiek pat telefono resursų. Hibridinės programėlės, kaip matyti iš gautų bandymų rezultatų, naudoja mažiau operatyviosios atminties, tačiau daugiau procesoriaus pajėgumo. Taip pat jų užimama vieta telefone mažesnė, jos naudoja mažiau telefono baterijos, tačiau jų įdiegimo failas yra didesnis lyginant su gimtosiomis programėlėmis. Žinoma, nereikia pamiršti, kad hibridinės ir gimtosios programėlės buvo kuriamos naudojant skirtingas technologijas. Programos kodas nepanašus, tačiau mygtukų išdėstymą, rodomą vaizdą ekrane (ką mato programėlės naudotojas) buvo stengiamasi atkurti kiek įmanoma panašiau – stengtasi nenaudoti nereikalingų stilių (pavyzdžiui, naudojami mygtukai buvo standartiniai kiekvienai naudotai technologijai) tam, kad būtų galima tiksliau įvertinti, ar hibridinių programėlių technologijos (HTML5, CSS, JavaScript) nenaudos daugiau telefono resursų negu programėlės su gimtąja technologija atveju (pavyzdžiui, Android gimtosios programėlės rašomos su JAVA technologija).

6. Framework7 programėlė

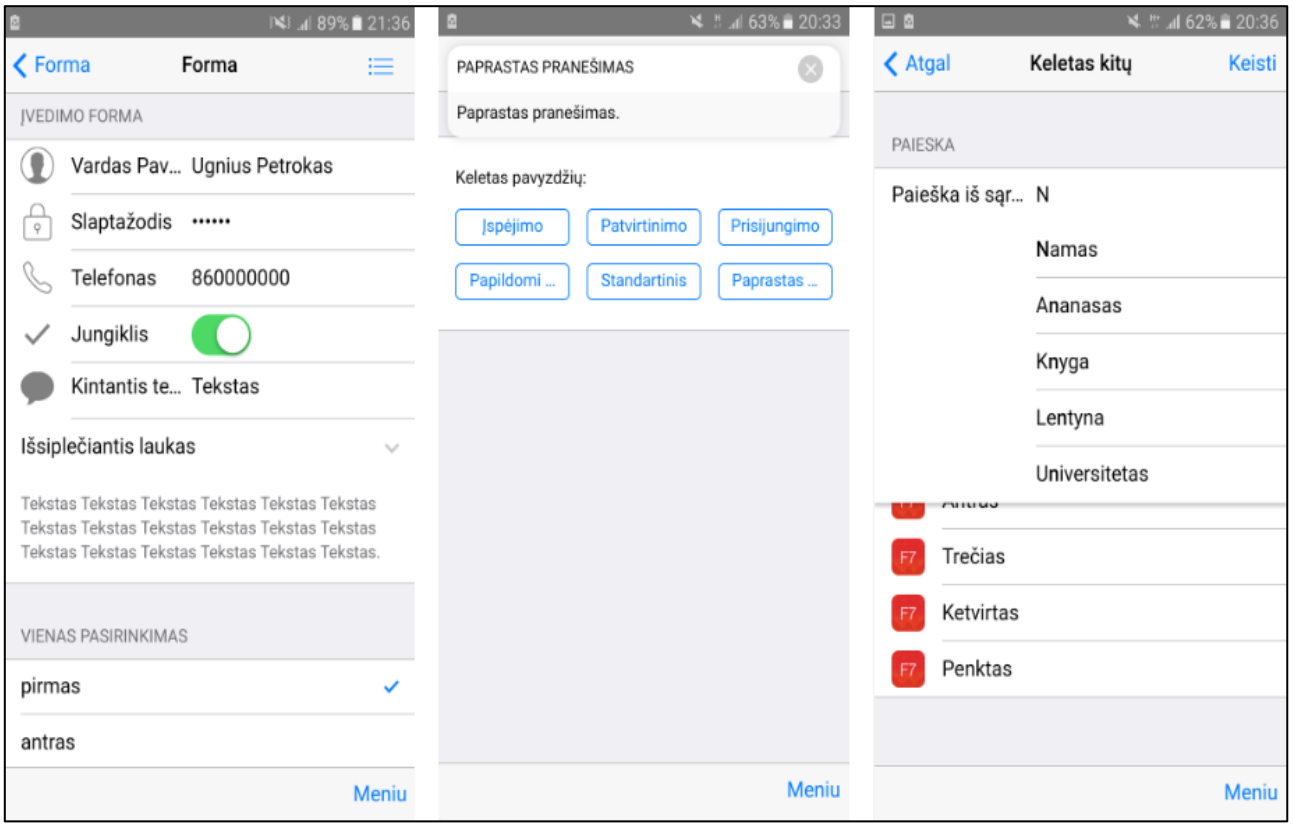
Framework7 yra tas karkasas, kurį siūloma naudoti kartu su Phonegap. Framework7 karkasas skirtas kurti hibridines programėles, kurios atrodytų, kaip gimtosios (iOS, Android). Parsisiuntus reikalingus failus (CSS ir JavaScript bibliotekas) galima pradėti rašyti savo HTML kodą. Šis karkasas bando atkartoti gimtųjų programėlių elgseną, stilius, animacijas. Buvo pasirinkta išbandyti iOS telefonui skirtą biblioteką ir stilius Android telefono modelyje – Samsung Galaxy S6. Buvo įdomu patikrinti, ar galima tą patį funkcionalumą (arba bent jau dalį), skirtą iOS platformai, pritaikyti Android telefone.

Kadangi pasirinkti CSS stiliai ir JavaScript bibliotekos yra skirtos iOS platformai, tai iš pirmo žvilgsnio tiek vaikščiojimas per langus, tiek įvedimo laukai identiški (arba labai panašūs) į tikruosius, ir tai suteikia gimtosios programėlės naudojimosi jausmą. Tačiau kai kurie specifiniai dalykai neveikia Android telefone. Vienas iš jų yra kalendorius, jo negalima iššaukti ir pasirinkti norimos datos. Nereikia stebėtis, nes ir oficialiame Framework7 puslapyje rašoma, kad skirtingi CSS failai ir JavaScript bibliotekos yra skirtos būtent tai platformai, kuriai jos yra kurtos. Taip pat pastebėta, kad yra pažeidžiamumų. Pavyzdžiui, įvedimo laukuose pabandžius įvesti tekstą su HTML žymėmis (angl. tags), tokiomis kaip <b>, <i> gaunamas toks rezultatas (žr. 1 paveiksliuką). Todėl galima daryti išvadą, kad karkasas neapsaugo nuo pagrindinių saugumo spragų ir tai yra atviras kelias įsilaužimo atakoms.



1 paveiksliukas. Įvedimo laukai, pranešimai.

Framework7 karkasas pasirūpina ir animacijomis, pavyzdžiui nuotraukų peržiūra. Nuotraukas galima peržiūrėti, jas padidinti, sumažinti (žr. 11 priedo 1 paveiksliuką). Taip pat Framework7 suteikia daug kitų iOS vaizdinių elementų tokių, kaip įvairūs iššokantys langai, paieškos laukai, sąrašo peržiūra, elementų keitimas vietomis (žr. 2 paveiksliuką). Visos karkaso teikiamos CSS animacijos yra sklandžios, pasižymi aukštu našumu.



2 paveiksliukas. Įvedimo formos, pranešimo, paieškos pavyzdžiai.

Taip pat reikėtų paminėti, kad Framework7 karkasas nenaudoja jokių vadinamųjų trečiųjų šalių bibliotekų (pavyzdžiui, turi savo DOM7 (angl. Document Object Model) biblioteką, kurioje naudojami didelio našumo metodai, skirti DOM valdymui), todėl programėlė neužima daug vietos, pasižymi dideliu našumu, lakstumu. Naudojant AJAX, programėlės langai yra užkraunami tik kartą (per 10 minučių, šis laiko tarpas nustatytas automatiškai), o visus kitus kartus yra rodomas iš atminties (pgl. [Kha16]). Todėl vaikščiojimas per skirtingus langus vyksta akimirksniu.

7. Ionic programėlė

Ši hibridinė programėlė, kurta remiantis Ionic karkasu, gali būti naudojama kaip prototipas agentūroms, kurios organizuoja ekskursijas, išvykas. Programėlė padėtų susiorientuoti naudojant GPS, o nesant ryšiui – naudojant kompasą (aktualu, kai yra blogos oro sąlygos, esama laukinėje gamtoje ar kalnuotoje vietovėje).

Kuriant šią hibridinę programėlę su Ionic karkasu, buvo nuspręsta naudoti dalį jau parašyto kodo (kameros, GPS, kompaso pasiekiamumas). Senas funkcionalumas šiek tiek praplėstas ir pridėtas naujas – naudotojo registracija, prisijungimas, žinučių rašymo galimybė (plačiau žiūrėti 12 priedo 1 paveiksliuką). Pirmiausia buvo pradėtas kurti meniu su pasirinkimais. Kiekvienas pasirinkimas turi savo būseną – aktyvi arba neaktyvi. Aktyvi bus tada, kai naudotojas pasirinks norimą pasirinkimą ir bus rodomas to pasirinkimo langas, o grįžus į meniu tas pasirinkimas bus paryškintas (žr. 3 paveiksliuką).

3 paveiksliukas. Meniu.

Po meniu punkto ir kelių pagrindinių puslapių sukūrimo, buvo pradėta kurti naudotojų autentifikacija. Tam naudota Ionic Cloud siūloma paslauga – Ionic Auth. Naudojant Ionic Auth buvo daromi registracijos ir prisijungimo langai (žr. 13 priedo 1 paveikliuką). Registracijos metu naudotojo yra prašoma suvesti vardą, pavardę, el. paštą ir slaptažodį. Prisijungimo metu naudotojui

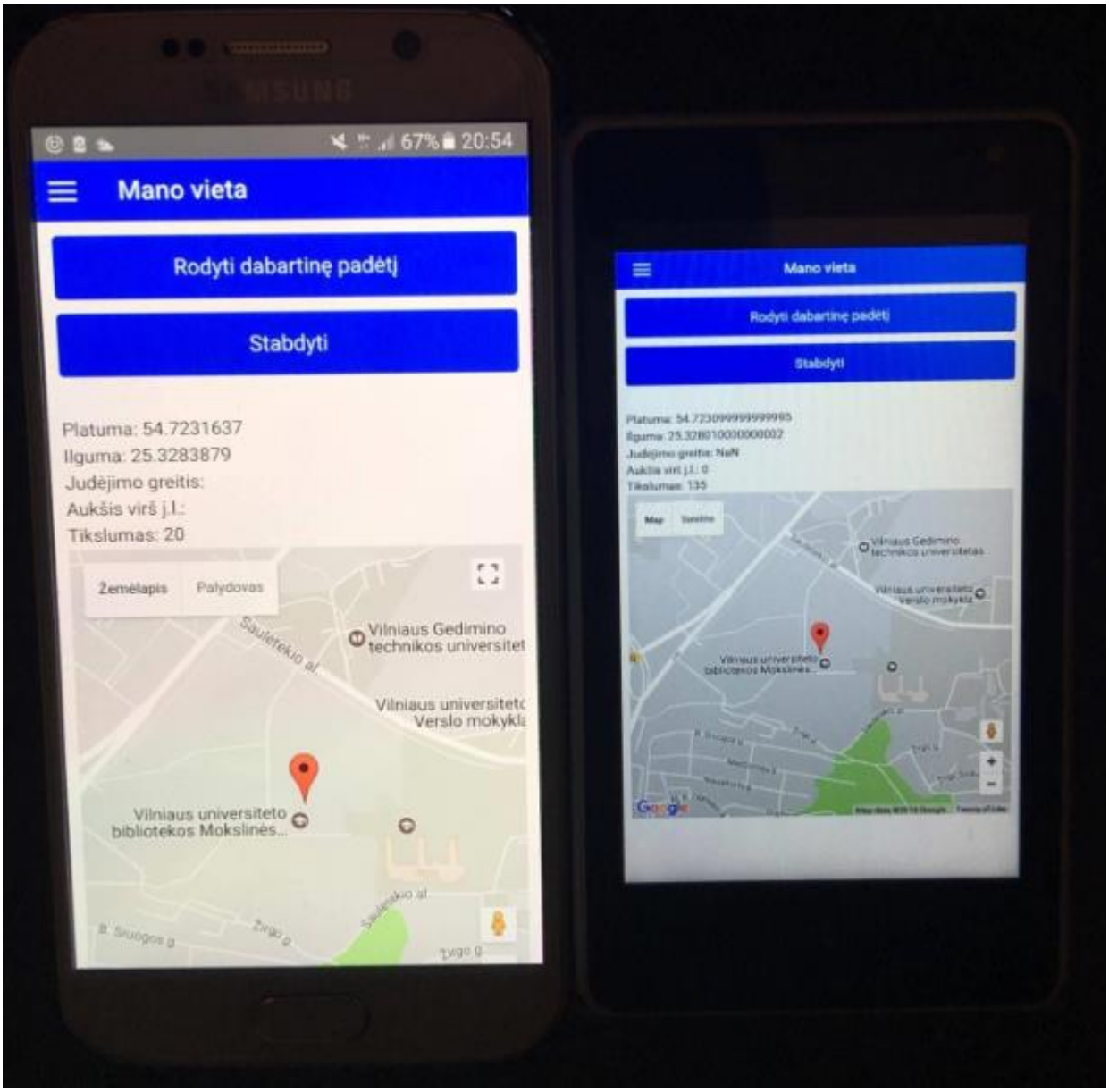
reikės suvesti el. paštą ir slaptažodį. Taip pat yra keletas validacijų. Jeigu registracijos metu toks naudotojas su tokiu pačiu el. paštu jau yra užsiregistravęs ar pats el. paštas neatitiks reikiamų kriterijų – bus metamas klaidos pranešimas su atitinkamu tekstu. Kai tik naudotojas prisijungs, jis iš karto bus perkeltas į pagrindinį langą. Deja, Ionic Auth didelėmis galimybėmis nepasižymi. Naudotojų informacijai saugoti yra skirti tik keli laukai, tokie kaip slapyvardis, el. paštas, slaptažodis, vardas ir pavardė, paveiksliukas (profiliui). Papildomais laukais, tokiais kaip telefonas, gimimo data, reikės rūpintis patiems – registracijos metu, kaip dalį registracijos duomenų, paduoti JSON objektą. Vėliau, atsiradus poreikiui, juos bus galima pasiekti. Pats autentifikavimo procesas yra paprastas. Nurodomas prisijungimo būdas (šiuo atveju Ionic Auth, kiti būdai galėtų būti Facebook, Twitter, Instagram, Google el. paštas ir kt.), paduodami prisijungimo nustatymai (pavyzdžiui, ar prisiminti naudotoją) ir prisijungimo duomenys (šiuo atveju el. paštas ir slaptažodis).

Lange „mano vieta“ buvo pridėtas naujas funkcionalumas – GPS duomenys: ilguma, platuma, judėjimo greitis, gautų duomenų tikslumas buvo atvaizduojami ekrane (žr. 4 paveikslių).

```
function rodytiDabartinePadeti(pozicija) {
    var platuma = pozicija.coords.latitude;
    var ilguma = pozicija.coords.longitude;
    document.getElementById('platuma').innerHTML = pozicija.coords.latitude;
    document.getElementById('ilguma').innerHTML = pozicija.coords.longitude;
    document.getElementById('greitis').innerHTML = pozicija.coords.speed;
    document.getElementById('aukstis').innerHTML = pozicija.coords.altitude;
    document.getElementById('tikslumas').innerHTML = pozicija.coords.accuracy;
    var manoVieta = new google.maps.LatLng(platuma, ilguma);
    var mapOptions = {center: manoVieta, zoom: 15, mapTypeId: google.maps.MapTypeId.ROADMAP};
    var zemelapis = new google.maps.Map(document.getElementById('map'), mapOptions);
    var marker = new google.maps.Marker({
        position: manoVieta,
        map: zemelapis
        });
}
```

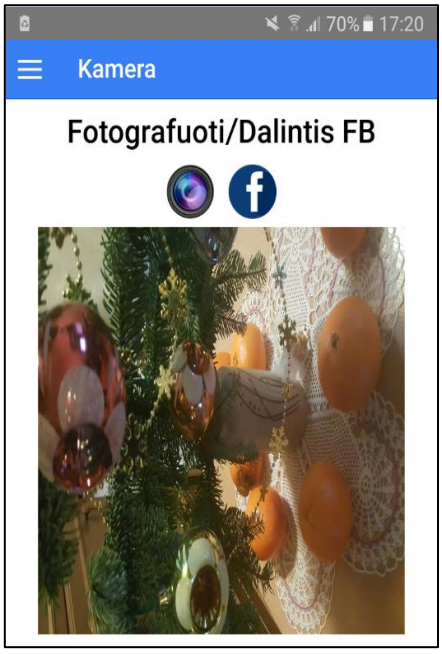
4 paveiksliukas. Dalis žemėlapio kūrimo kodo.

Taip pat buvo padaryta integracija su Google Maps. Pirmiausia programėlei reikėjo sugeneruoti raktą, sukurti JavaScript funkciją, kurioje bus kuriamas žemėlapio vaizdas, perduodamos koordinatės. Kadangi buvo naudojamas nemokamas Google Maps API planas, tai dalis funkcionalumo yra apribota, pavyzdžiui, naudotojo sekimas nuo pradžios taško iki galutinio kelionės tikslo nėra galimas. Naudotojo esama vieta ekrane rodoma žemėlapyje tik tuo laiko momentu, kai buvo išsiųstas pranešimas gauti dabartinės vietos informaciją. Sustabdžius GPS duomenų atnaujinimą, galima žemėlapį padidinti, sumažinti, pasileisti Street View ir naudoti kitus Google Map teikiamus privalumus (žr. 5 paveiksliuką).



5 paveiksliukas. Vietos nustatymas. Telefonų modeliai: Samsung Galaxy S6, Nokia Lumia 635.

Kamera yra iššaukiama pasirinkus „Kamera“ iš meniu punkto. Padarius nuotrauką ji bus rodoma ekrane ir jeigu fotografija netenkina naudotojo, pačiame lange yra greitas priėjimas prie kameros vėl (žr. 6 paveiksliuką). Tačiau jeigu ji tenkina naudotoją, paspaudus Facebook paveiksliuką bus galima greitai pasidalinti ta nuotrauka socialiniame tinkle Facebook (žr. 14 priedo 1 paveiksliuką).



6 paveiksliukas. Nuotraukos rodymas, dalijimasis socialiniame tinkle.

Žinutėms saugoti buvo naudojama Google Firebase duomenų bazė (pgl. [Goo16]). Sukurtos 3 lentelės pagal paskirtį („Maršrutai ir laikai", „Grupės pokalbis", „Rasti ir pamesti daiktai"), kuriose bus saugomos kiekvieno programėlės naudotojo įvestos žinutės. Kad būtų įvykdyta integracija su Firebase duomenų baze, reikėjo sugeneruoti unikalų URL (angl. Uniform Resource Locator) ir API raktą. Kadangi Ionic yra paremtas AngularJS, tai buvo naudojama atvirojo kodo biblioteka AngularFire, kuri sujungia Firebase su Ionic projektu. Žinutėms siųsti į duomenų bazę buvo skirta funkcija (žr. 7 paveiksliuką). Išsaugomas žinutės tekstas, išsiuntimo data, naudotojo el. paštas ir vardas, pavardė.

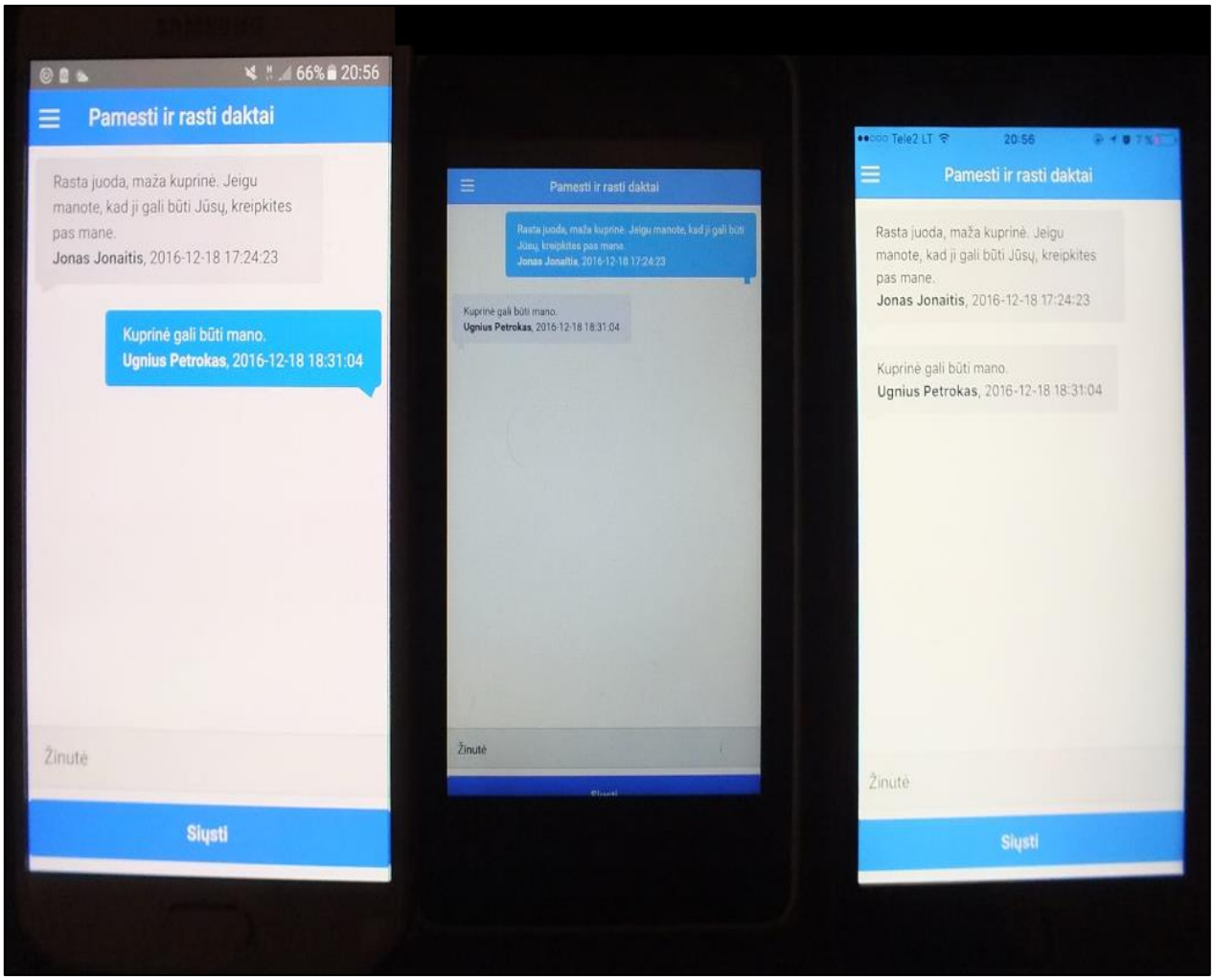
```
$scope.siustiZinute = function() {
  var zinutesData = new Date();
  $scope.zinutes.$add({
    tekstas: $scope.informacija.zinute,
    data: zinutesData.toJSON(),
    elPastas: $scope.vartotojas.elPastas,
    vardasPavarde: $scope.vartotojas.vardasPavarde
  });
  $scope.informacija.zinute = '';
};
```

7 paveiksliukas. Žinutės išsiuntimo kodo dalis.

Kiekvienos žinutės išsiuntėjas yra lyginimas su dabartiniu prisijungusiu naudotoju (jo el. paštu). Jeigu naudotojas, kuris siuntė žinutę, yra tas pats, kaip ir prisijungęs naudotojas, tai žinutė bus atvaizduojama melsva spalva dešinėje ekrano pusėje, o visos kitos, gautos iš kitų naudotojų – pilka spalva kairėje ekrano pusėje (žr. 8, 9 paveiksliuką).

```
<div ng-repeat="zinute in zinutes" class="zinutes-konteineris">
  <div class="zinutes-pozicija" ng-class="{'desine': vartotojas.elPastas == zinute.elPastas,'kaire': vartotojas.elPastas != zinute.elPastas}">
    <div class="zinute">{{zinute.tekstas}}</div>
      <div class="zinutes-informacija">
          <span class="bold">{{zinute.vardasPavarde}}</span>
          <span>{{zinute.data | date: 'yyyy-MM-dd HH:mm:ss'}}</span>
      </div>
  </div>
<div class="cf"></div>
</div>
```

8 paveiksliukas. Dalis žinučių rodymo kodo.



9 paveiksliukas. Žinučių skirtingas atvaizdavimas. Modeliai: Samsung Galaxy S6, Nokia Lumia 635, Apple Iphone S6.

Ši hibridinė programėlė buvo bandoma su trimis skirtingais telefono modeliais: Samsung Galaxy S6, Nokia Lumia 635 ir Apple Iphone 6. Kadangi be sertifikatų pirkimo nebuvo galima sugeneruoti įdiegimo failo (kad būtų galima įrašyti į telefonus, neskaitant Android telefono, kuriam nereikia sertifikato), buvo ieškota alternatyvų, kaip būtų galima atlikti programėlės testavimą. PhoneGap

Desktop ir PhoneGap Developer app (ši įrašoma į testuojamą telefoną) dėka buvo galima realiu laiku keisti programinį kodą ir matyti pokyčius telefone. Pastebėtas vienas trūkumas – naudojant PhoneGap Developer app nebuvo galima pasiekti galerijos (nors ši problema nepasikartojo tada, kai buvo sugeneruojamas įdiegimo failas ir programėlė įrašoma į telefoną), todėl, padarius nuotrauką, vaizdas iš galerijos nebuvo rodomas ekrane. Taip pat reikalingas greitas interneto ryšys. Bandant programėlę Nokia Lumia 635 telefone, pastebėta, kad ne visi komponentai tilpdavo ekrane (lyginant su kitais testuotais telefonais). GPS duomenys Samsung Galaxy S6 telefone (aukštis virš jūros lygio, judėjimo greitis) nebuvo atvaizduojami ekrane.

Rezultatai ir rekomendacijos

Baigiamojo darbo esminis tikslas buvo išanalizuoti hibridinius karkasus, atskleisti jų privalumus ir trūkumus. Be kita ko, šiuo tikslu buvo sukurtos hibridinės programėlės, kurios galėtų pasiekti mobilaus telefono techninę įrangą ir veiktų taip pat gerai, kaip ir gimtosios. Taip pat pagal sukurtas programėles buvo atlikta naudojamų telefonų resursų analizė, iš kurios paaiškėjo, kad hibridinės programėlės užima mažiau vietos telefone, naudoja mažiau operatyviosios atminties ir telefono baterijos, tačiau daugiau procesoriaus pajėgumo ir jų įdiegimo failo dydis didesnis, lyginant su gimtosiomis programėlėmis.

Naudojant hibridinius karkasus Ionic, Framework7, buvo bandoma ištirti hibridinių karkasų teikiamus privalumus. Taip pat buvo naudojami tų karkasų rekomenduojami įrankiai (pavyzdžiui, PhoneGap Developer) bei paslaugos (pavyzdžiui, Ionic Auth).

Sukurtų programėlių rekomendacijos:

- Patobulinti grafinę sąsają, kuri būtų patrauklesnė naudotojų akiai.
- Hibridinėje programėlėje, kurioje buvo naudojamas Ionic karkasas, pirmiausia reikėtų Ionic Auth pakeisti į kitą sistemos naudotojų autentifikavimo sistemą, kad tiek naudotojų duomenys, tiek žinutės būtų saugomos vienoje duomenų bazėje. Antra, reikėtų sukurti papildomą funkcionalumą, kad sistemos naudotojas galėtų kitam naudotojui parašyti privačią žinutę. Trečia, patobulinti dabartinės padėties rodymą su Google žemėlapiu.
- Sukurti gimtąją (pavyzdžiui, Android) ir hibridinę programėlę (geriausiai tiktų žaidimas), kuri turėtų daug grafinių elementų ir reikalautų nemažai telefono resursų, tam, būtų galima patikslinti prieš tai darytą tyrimą ir atsakyti į klausimą, kokios programėlės naudoja daugiau resursų – gimtosios ar hibridinės. Kadangi programėlių sudėtingumas padidės, realizacijos būdai skirsis, tai gali būti, kad nauji tyrimo rezultatai skirsis nuo prieš tai atlikto tyrimo.

Remiantis išanalizuota teorine medžiaga, gautais resursų analizės rezultatais, sukurtomis programėlėmis, galima daryti išvadą, kad hibridiniai karkasai, siūlomi įrankiai ir paslaugos iš tiesų pagreitina programėlių kūrimą, suteikia prieigą prie platesnės naudotojų rinkos. Tačiau taip pat egzistuoja saugumo spragos ir gali prireikti papildomo laiko, norint įsitikinti, ar programėlė vienodai gerai veiks skirtinguose telefono modeliuose (ar nebus iškraipomi elementai, ar jie tilps į ekraną).

Kadangi technologijos, karkasai ir įrankiai toliau tobulėja, visus atliktus tyrimus reikėtų periodiškai kartoti, o dar geresniems tyrimams būtų galima atlikti palyginimą su atitinkamomis gimtosiomis Android, iOS ir Windows Phone programėlėmis.

Literatūros sąrašas

[Ado16] Adobe Systems Inc., PhoneGap. URL: http://www.phonegap.com, 2016.

[Cor15] Apache Software Foundation, Cordova. URL: https://cordova.apache.org, 2015.

[Ion16] Ionic. URL: https://ionicframework.com, 2016.

[Mot16] Monaca and Onsen team. URL: https://onsen.io, 2016.

[Kha16] Vladimir Kharlampidi, Framework7. URL: https://framework7.io/docs/, 2016.

[Jqu16] jQuery Foundation. URL: https://jquerymobile.com, 2016.

[Fam15] Famous Industries Inc, Documentation. URL: https://famo.us/docs, 2015.

[Sen16] Sencha Inc. URL: https://www.sencha.com, 2016.

[App16] Appcelerator Inc. URL: http://www.appcelerator.com, 2016.

[Goo16] Google, Firebase. URL: https://firebase.google.com/docs/database/web/start, 2016.

[Idc16] IDC. URL: www.idc.com/promo/smartphone-market-share/os, 2016.

[G2C16a] G2 Crowd, Inc. URL: https://www.g2crowd.com/categories/mobile-development-frameworks, 2016.

[G2C16b] G2 Crowd, Inc. URL: https://www.g2crowd.com/products/onsen-ui/reviews, 2016.

[Fal16] Markus Falk, Mobile frameworks comparison chart. URL: http://mobile-frameworks-comparison-chart.com/, 2016.

[Sta16] Joe Stangarone, The Mobile App Comparison Chart. URL: http://www.mrc-productivity.com/blog/2016/06/the-mobile-app-comparison-chart-hybrid-vs-native-vs-mobile-web/, 2016.

[Bri15] John Bristowe, What is a Hybrid Mobile App. URL: http://developer.telerik.com/featured/what-is-a-hybrid-mobile-app/, 2015.

[Ver16] Veracode, Cross-Site Scripting. URL: https://www.veracode.com/security/xss, 2016.

[And16a] Android Studio. URL: https://developer.android.com/studio/index.html, 2016.

[Cal16] Jack Calder, HTML5 vs Native Apps: What‘s best for 2016. URL: http://justcreative.com/2016/03/17/html5-vs-native-apps-whats-best-for-2016/, 2016.

[Tho14] Alina Thomas, PhoneGap App Development. URL: http://www.slideshare.net/Alinathomas1/phonegap-app-development-an-overview, 2014.

[Dha11] Arpan Dhandhania, HTML5: Client-side Storage. URL: http://www.webreference.com/authoring/languages/html/HTML5-Client-Side/index.html, 2016.

[Gle16] Tal Gleichger, Hybrid UI framework shootout. URL: https://www.airpair.com/ionic-framework/posts/hybrid-apps-ionic-famous-f7-onsen, 2016.

[And16b] Android Studio, Sign Your App. URL: https://developer.android.com/studio/publish/app-signing.html, 2016.

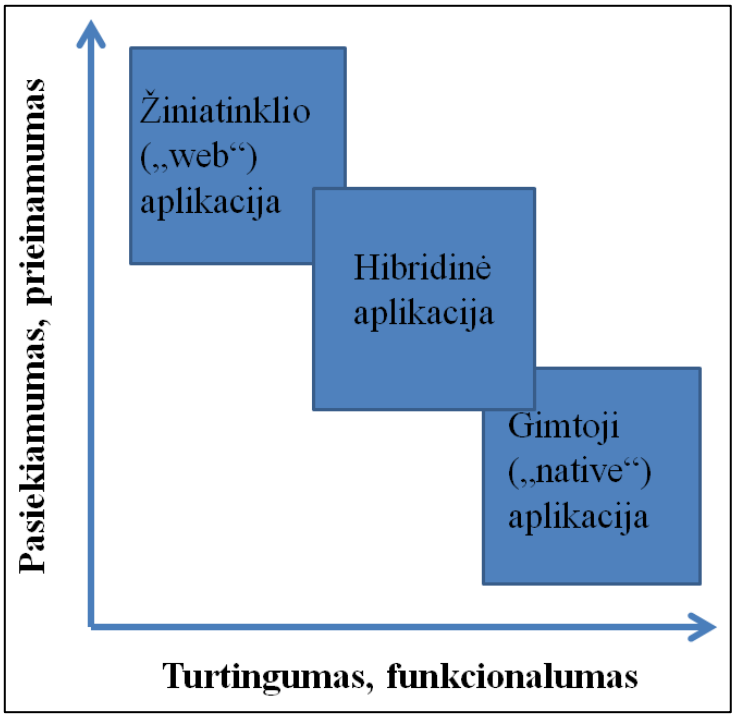
[G2C16c] G2C Crowd, Inc. URL: https://www.g2crowd.com/products/jquery-mobile/reviews, 2016.

[App16] Apple, Inc. App Store Review Guidelines. URL: https://developer.apple.com/app-store/review/guidelines/, 2016.

Paskutinį kartą visi šaltiniai žiūrėti ir patikrinti, ar veikia jų URL, 2017 m. sausio 10 d.

Priedas Nr. 1

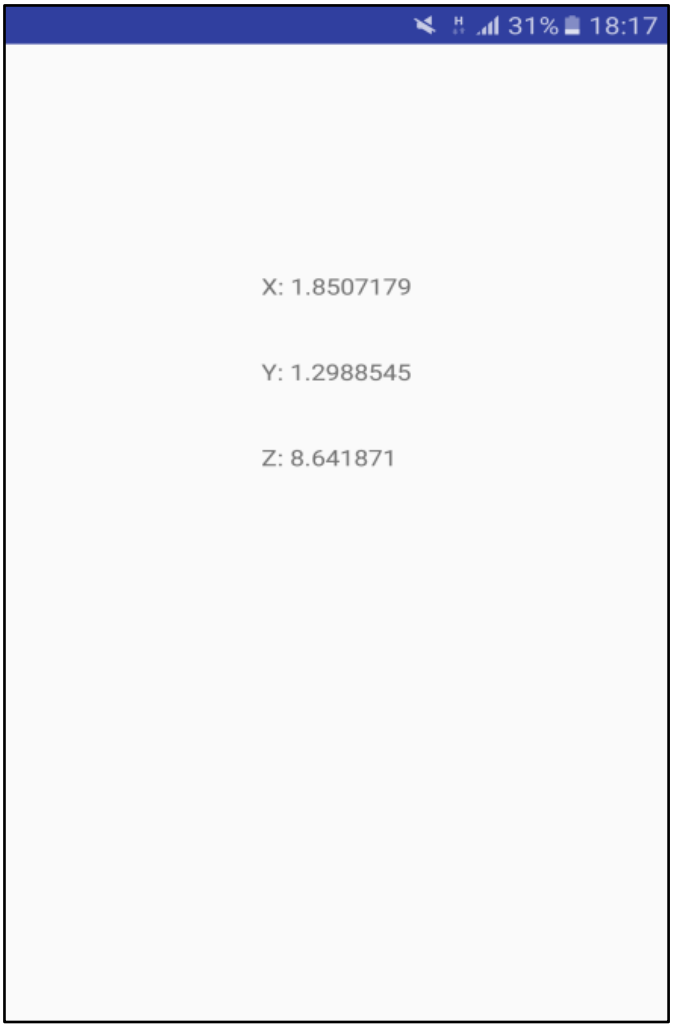
Žiniatinklio, hibridinių ir gimtųjų programėlių tinkamumas pagal pasiekiamumą, funkcionalumą.



1 diagrama. Programėlių tipų palyginimas.

Priedas Nr. 2

Gimtoji programėlė, akselerometras.



1 paveiksliukas. Programėlės pagrindinis langas.

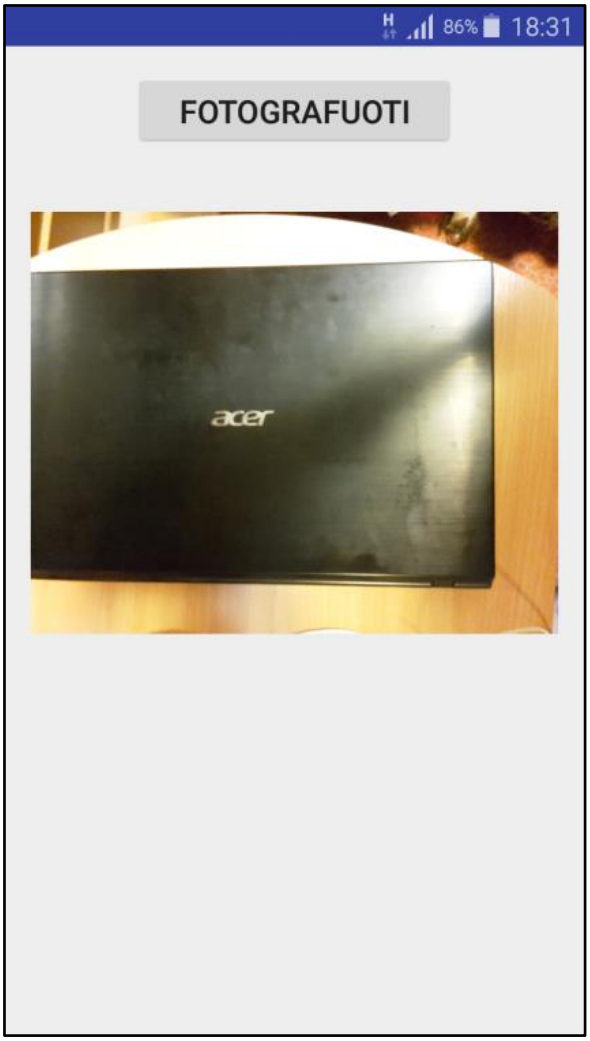
Priedas Nr. 3

Gimtoji programėlė, GPS.

1 paveiksliukas. Programėlės pagrindinis langas.

Priedas Nr. 4

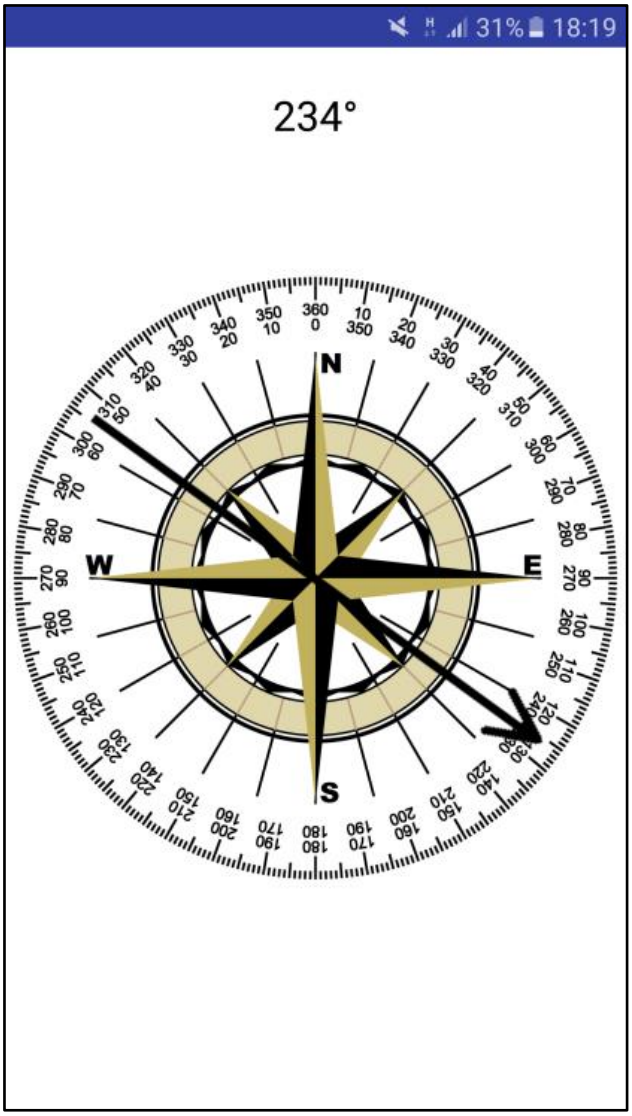
Gimtoji programėlė, kamera.



1 paveiksliukas. Programėlės pagrindinis langas.

Priedas Nr. 5

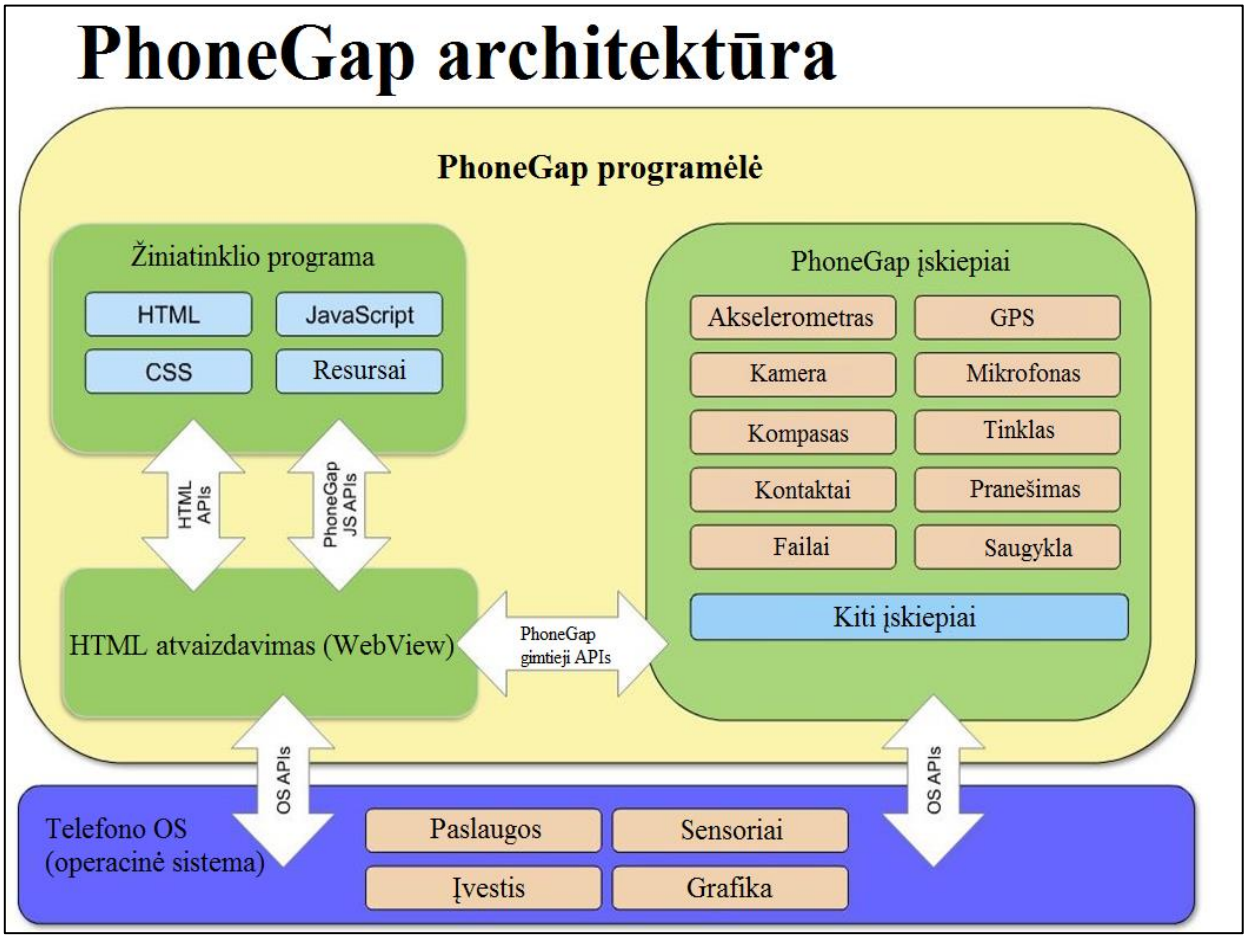
Gimtoji programėlė, kompasas.



1 paveiksliukas. Programėlės pagrindinis langas.

Priedas Nr. 6

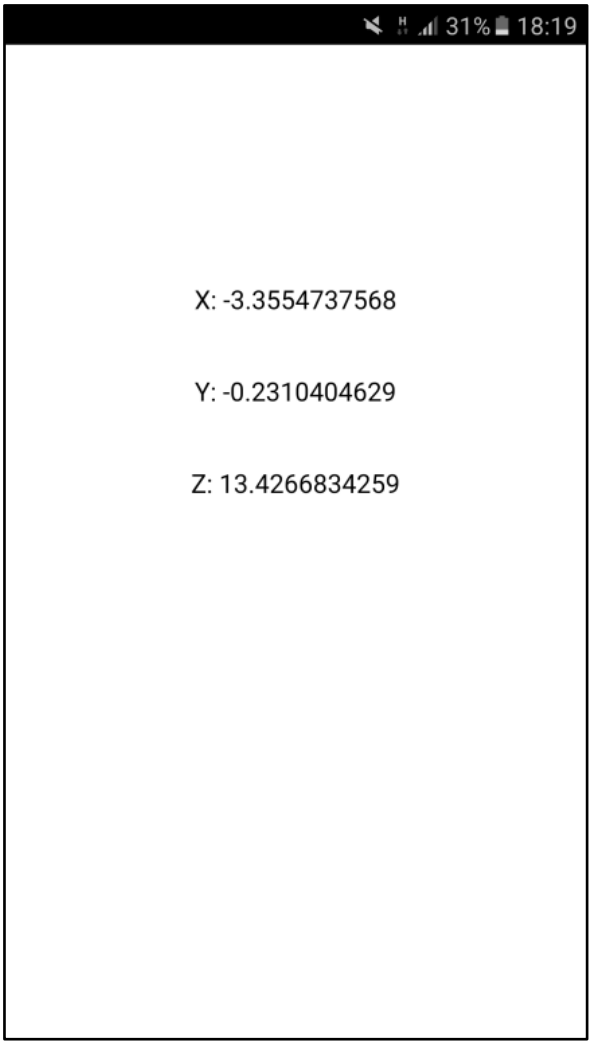
PhongeGap architektūra. Naudojamos žiniatinklio technologijos (HTML, CSS, JS), PhoneGap (Cordova) įskiepiai. Programėlė veikia specialioje aplinkoje – WebView. Tai yra tarsi naršyklės langas, išmetus grafinę sąsają ir palikus rodyti programėlės turinį per visą ekraną (plg. [Bri15]).



1 paveiksliukas. PhoneGap architektūra (pgl. [Tho14]).

Priedas Nr. 7

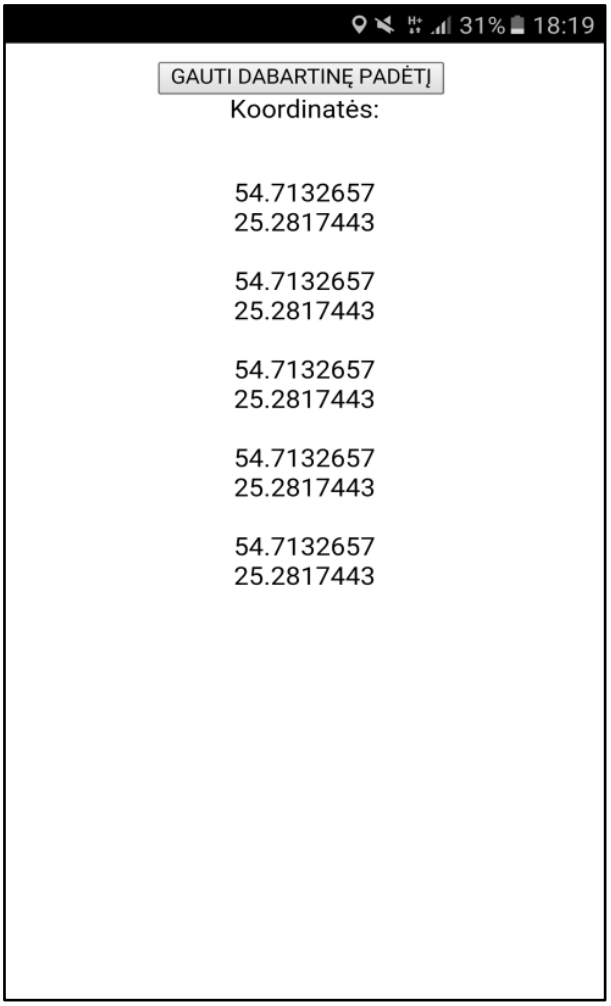
Hibridinė programėlė, akselerometras.



1 paveiksliukas. Programėlės pagrindinis langas.

Priedas Nr. 8

Hibridinė programėlė, GPS.



1 paveiksliukas. Programėlės pagrindinis langas.

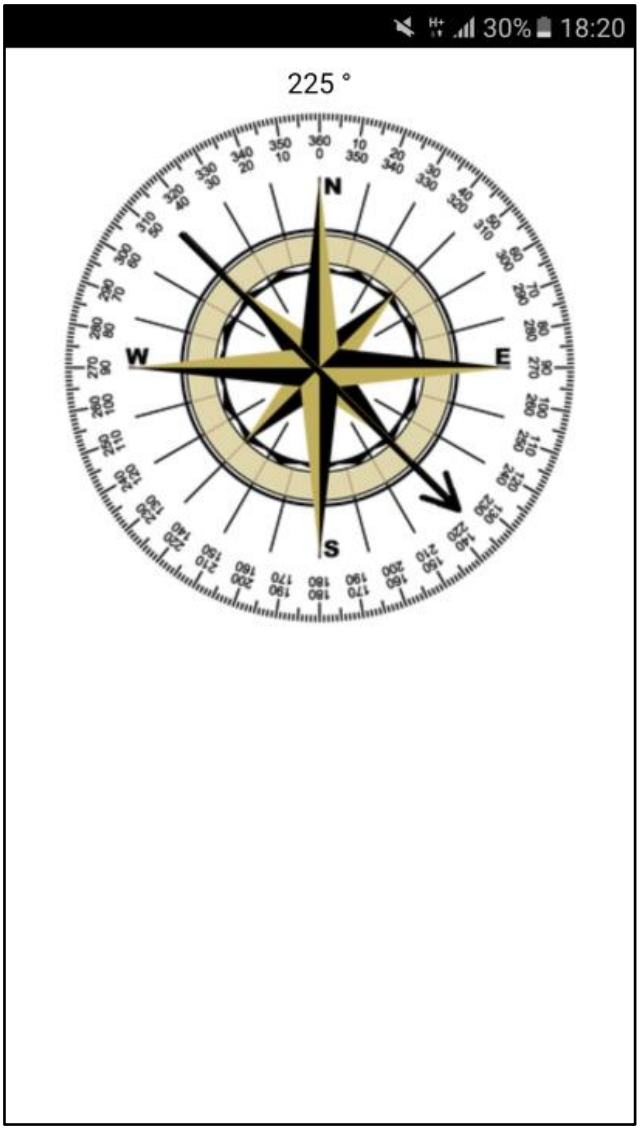
Priedas Nr. 9

Hibridinė programėlė, kamera.

1 paveiksliukas. Programėlės pagrindinis langas.

Priedas Nr. 10

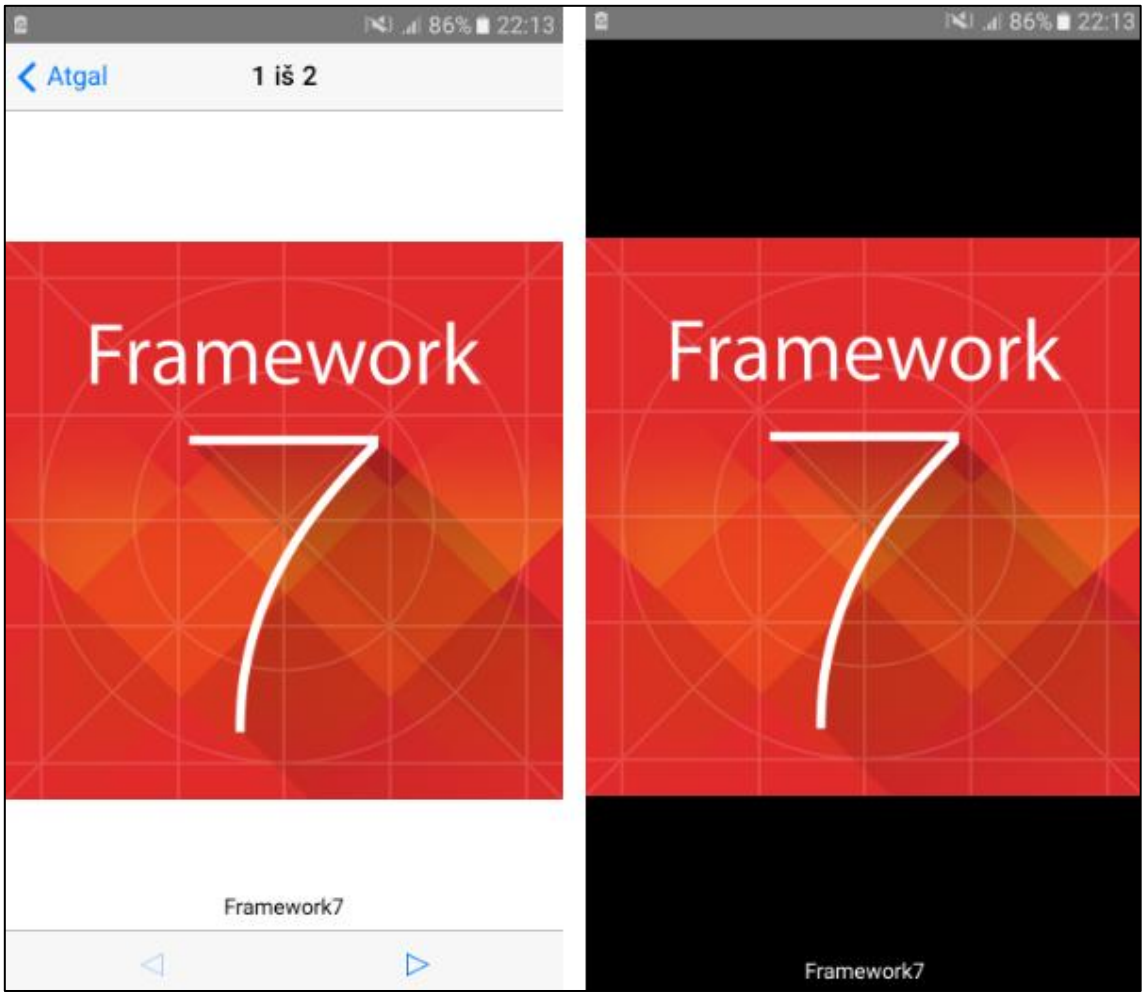
Hibridinė programėlė, kompasas.



1 paveiksliukas. Programėlės pagrindinis langas.

Priedas Nr. 11

Framework7 programėlė.



1 paveiksliukas. Nuotraukų peržiūra.

Priedas Nr. 12

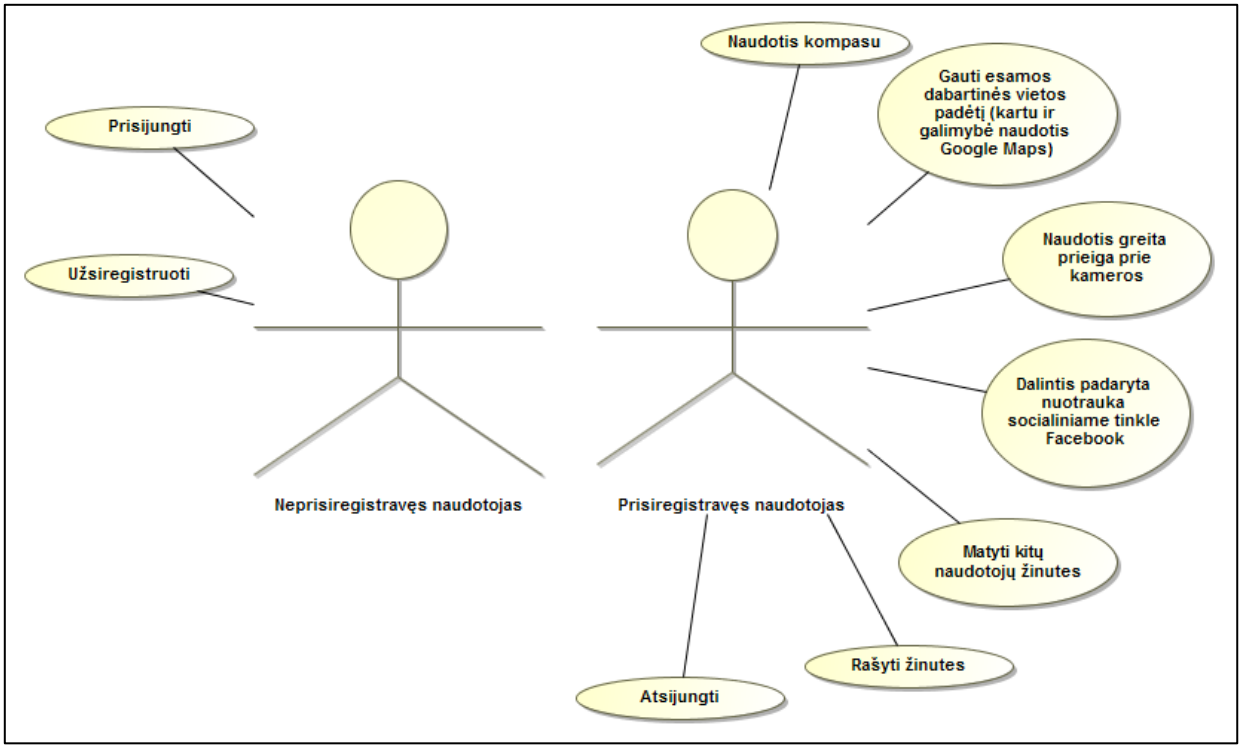
Hibridinės Ionic programėlės naudojimo atvejų diagrama (prisiregistravusio, neprisiregistravusio naudotojo).

Neprisiregistravęs naudotojas gali:

- užsiregistruoti;
- prisijungti.

Prisiregistravęs naudotojas gali:

- naudotis kompasu (jei telefone yra atitinkamas sensorius);
- gauti informaciją apie esamos vietos padėtį bei naudotis Google Maps paslauga;
- naudotis greita prieiga prie kameros;
- dalintis nuotrauka socialiniame tinkle;
- matyti kitų naudotojų žinutes;
- rašyti žinutes;
- atsijungti.

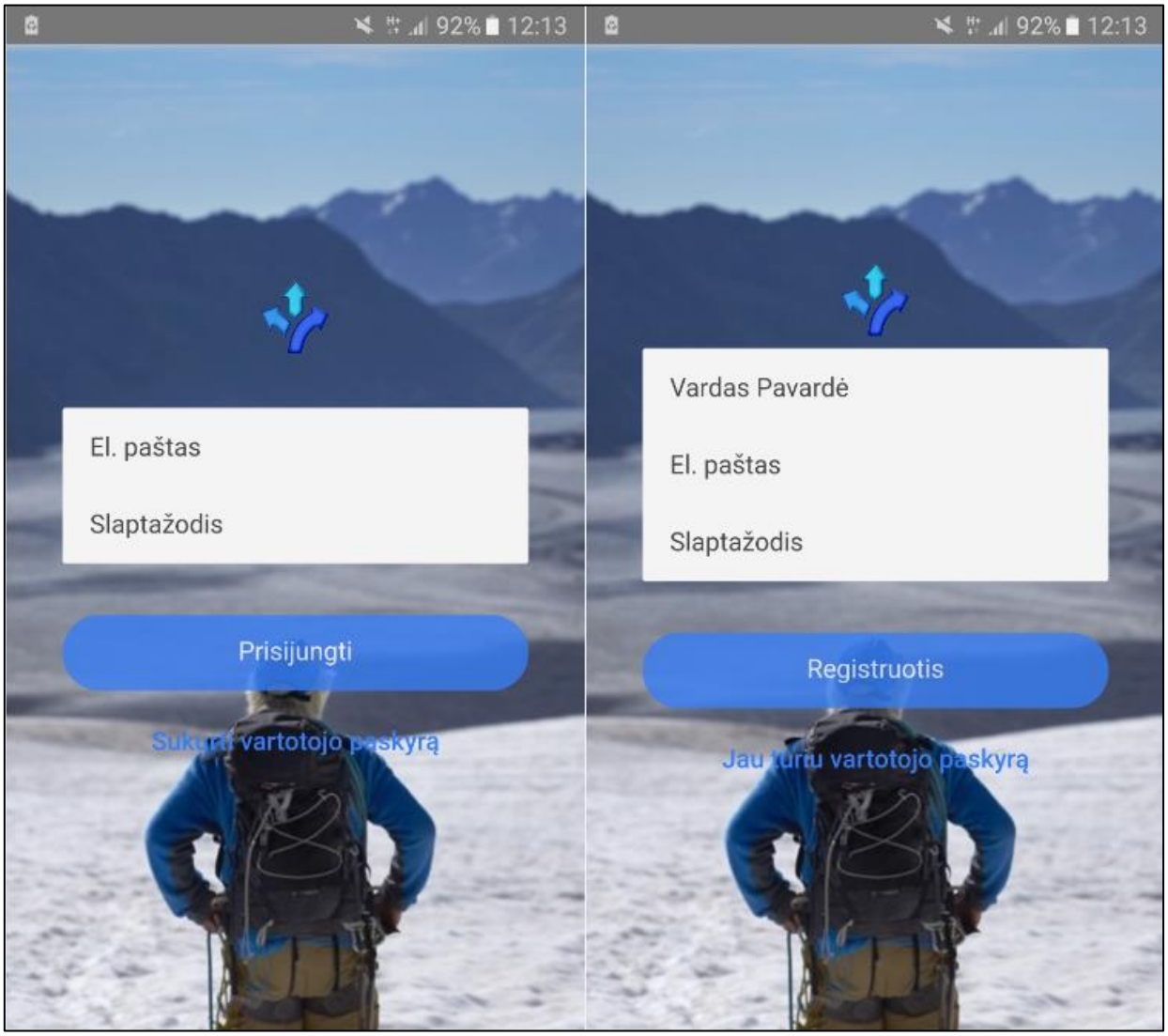


1 paveiksliukas. Naudojimo atvejų diagrama.

Priedas Nr. 13

Ionic programėlės naudotojo prisijungimo ir registracijos langai. Prisiregistravęs naudotojas bus prijungiamas automatiškai.

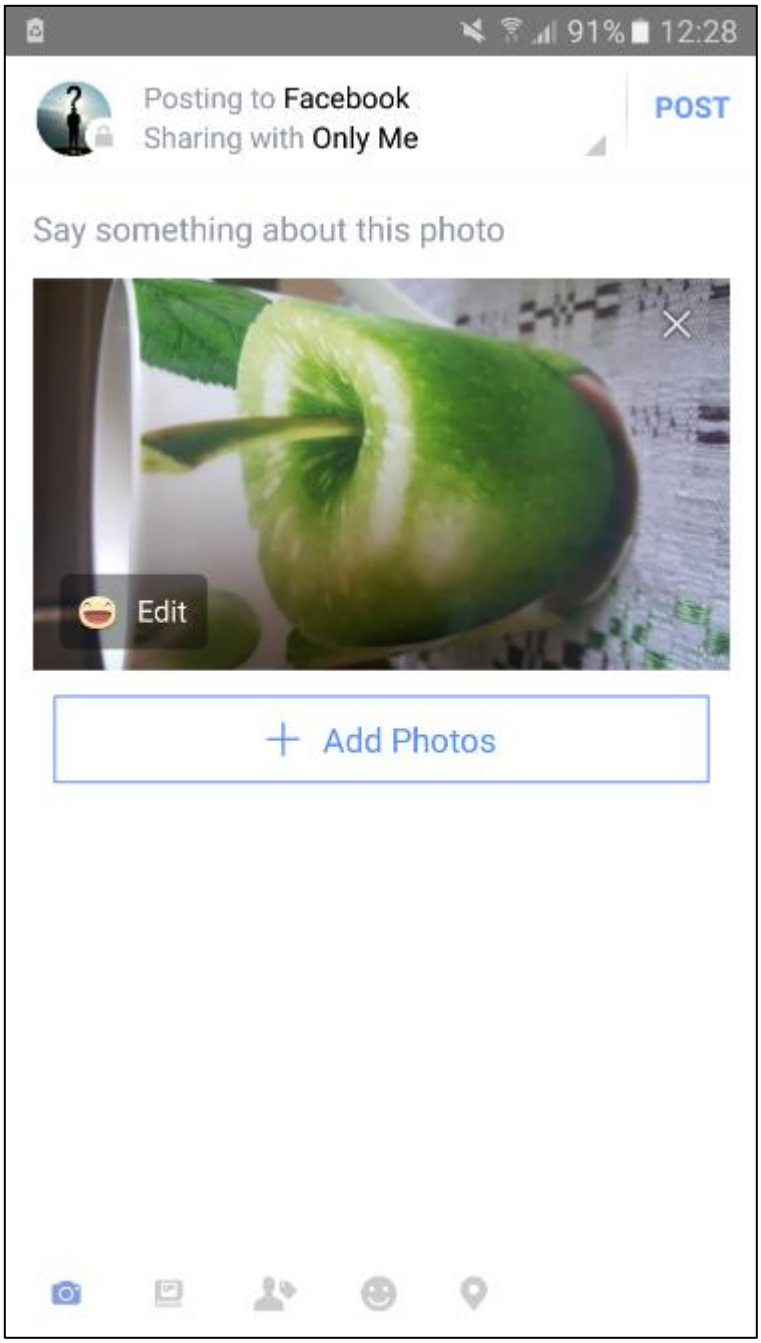
Jeigu nebus paspaudžiamas „Atsijungti“ mygtukas iš meniu, tai naudotojas vis vien bus autentifikuotas, t.y. galės naudoti kitas programėles ir, grįžęs prie šios, neturės įvesti savo prisijungimo duomenų iš naujo.



1 paveiksliukas. Naudotojų prisijungimo ir registracijos langai.

Priedas Nr. 14

Ionic programėlė. Nufotografuoto vaizdo dalinimasis socialiniame tinkle Facebook.



1 paveiksliukas. Nuotraukos dalinimasis.