ŠIAULIŲ UNIVERSITETAS
INFORMATIKOS, MATEMATIKOS IR E. STUDIJŲ INSTITUTAS
INFORMATIKOS KATEDRA

Gediminas Dauskurdis
Informatikos specialybės IV kurso nuolatinių studijų studentas

MOBILI APLIKACIJA NETOLIESE ESANČIŲ LANKYTINŲ VIETŲ PAIEŠKAI

MOBILE APP FOR SEARCHING FOR NEARBY PLACES OF INTEREST

BAKALAURO DARBAS

Darbo vadovas:
Lekt. M. Stoncelis

Recenzentas:
Doc. Dr. S. Turskienė

Šiauliai 2013/2014 m.m.

„Tvirtinu, jog darbe pateikta medžiaga nėra plagijuota ir paruošta naudojant literatūros sąraše pateiktus informacinius šaltinius bei savo tyrimų duomenis“

Darbo autoriaus _____

_____

(vardas, pavardė, parašas)

Darbo tikslai ir uždaviniai

Tikslas

Sukurti mobilią aplikaciją netoliese esančių lankytinų vietų paieškai.

Uždaviniai

1. Ištirti vartotojų, kurie ieško informacijos apie lankytinas vietas, poreikius.
2. Ištirti aplikacijai, kuri pateikia informaciją apie lankytinas vietas, būdingus reikalavimus.
3. Išanalizuoti panašaus tipo produktus.
4. Suprojektuoti DB, kurioje būtų saugomi duomenys apie lankytinas vietas.
5. Suprojektuoti mobilią aplikaciją.
6. Realizuoti DB.
7. Realizuoti mobilią aplikaciją.
8. Testuoti mobilią aplikaciją.

Darbo vadovo _____

(vardas, pavardė, parašas)

TURINYS

ĮVADAS

Nuo pat pirmo telefono išradimo, kartu su technologijų vystymusi, vystėsi ir patys telefonai. Iš pradžių keitėsi jų dizainas, numerio rinkimo būdas. Po to, atsiradus pirmiesiems kompiuteriams, atsirado poreikis ir nešiojamam telefonui. XX a. antroje pusėje atsirado pirmasis mobilusis telefonas. Jis tuo metu neturėjo ekrano, tik skaičius numerio rinkimui. Po truputį tobulėjo, ir tuo pačiu metu jie mažėjo iki lengvai telpančio į delną, mobilieji telefonai, atsirado ekranai, vėliau atsirado spalvos, vis didėjo mobiliųjų telefonų funkcionalumas, galimybės. Tuo pačiu tobulėjo ir mažėjo kompiuteriai. Po truputį vis daugiau kompiuterio galimybių galėjo atlikti telefonas. Galiausiai, atsirado išmanieji telefonai, turintys panašią į kompiuterių operacinę sistemą, daugybę galimybių, kurios yra kompiuteriuose. Galima sakyti jie tapo mažais kompiuteriais, kurių pagalba galima paskambinti kitiems naudojant mobilųjį ryšį. Šiuo metu, beveik kiekvienas žmogus turi išmanųjį telefoną, moka juo naudotis, todėl programos jiems yra labai populiarios.

Tuo pačiu, žmonės mėgsta keliauti, pamatyti kažką, daugiau sužinoti. Ištobulėjus telefonams, paprasti popieriniai žemėlapiai, ar informacinės knygos, pakeisti žemėlapiais telefone, o informaciją galima lengvai surasti internete. Tačiau tam, kad nereikėtų visko atskirai daryti, prasminga sukurti mobilią aplikaciją, kuri pateiktų informaciją apie vietas, esančias netoli vartotojo buvimo vietos, kurias verta aplankyti, bei padėtų ten nukeliauti. Lietuvoje, tokių aplikacijų nėra daug, todėl prasminga kurti aplikaciją, kuri skirta Lietuvos vietoms.

Tikslas

Sukurti mobilią aplikaciją netoliese esančių lankytinų vietų paieškai.

Uždaviniai

1. Ištirti vartotojų, kurie ieško informacijos apie lankytinas vietas, poreikius.
2. Ištirti aplikacijai, kuri pateikia informaciją apie lankytinas vietas, būdingus reikalavimus.
3. Išanalizuoti panašaus tipo produktus.
4. Suprojektuoti DB, kurioje būtų saugomi duomenys apie lankytinas vietas.
5. Suprojektuoti mobilią aplikaciją.
6. Realizuoti DB.
7. Realizuoti mobilią aplikaciją.
8. Testuoti mobilią aplikaciją.

1. ANALITINĖ DALIS

1.1. Temos analizė

1.1.1. Android OS

Android operacinė sistema yra atviro kodo operacinė sistema, vystoma Google kompanijos. 2005 metais Google nusipirko Android Inc. kompaniją ir pradėjo jos vystymą. 2007 metais atsirado Atviro telefono sąjunga (angl. Open Handset Alliance) ir Android oficialiai tapo atviro kodo projektu. Google nesiekė uždirbti iš pačios operacinės sistemos, pagrindinis siekis buvo išpopuliarinti Android operacinę sistemą, kad kiek įmanoma daugiau mobiliųjų įrenginių naudotų šią platformą. Tokiu būdu, Google gali uždirbti teikdama reklamavimo paslaugas dar didesniu mastu. Be to, Google sukūrė programinės įrangos vystymo įrankių rinkinį (angl. Software development kit, dažniausiai naudojamas trumpinys SDK), kuris platinamas taip pat nemokamai, tam, kad būtų lengviau kurti aplikacijas, skirtas Android platformai. Daugėjant aplikacijų kiekiui, didėjo ir Android OS populiarumas.

Android operacinė sistema sukurta Linux pagrindu. Kuriant panaudotas Linux branduolys, prie kurio prijungtos bibliotekos, Android vykdymo bibliotekos, aplikacijų karkasas (angl. application framework) ir pačios aplikacijos. Viena iš Linux pasirinkimo priežasčių buvo jos saugumas. Ši Linux savybė išnaudojama vykdant Android programas kaip atskirus Linux procesus.

1.1.2. iOS

iOS yra Apple kompanijos sukurta operacinė sistema, skirta Apple kompanijos išmaniesiems telefonams iPhone bei planšetiniams kompiuteriams iPad. iOS pirmą kartą buvo pristatyta 2007 metais kartu su pirmuoju Apple iPhone. Tuomet iOS buvo pavadinta iPhone OS. 2008 metais buvo išleista iOS 2 versija. Kartu su ja, Apple sukūrė aplikacijų parduotuvę (angl. App store) bei išleido programinės įrangos vystymo įrankių rinkinį iOS SDK. 2010 metais, išleidusi iOS 4, Apple operacinę sistemą iš iPhone OS pervadino į iOS. Šiuo metu naujausia iOS versija yra 7, išleista 2013 metų rudenį.

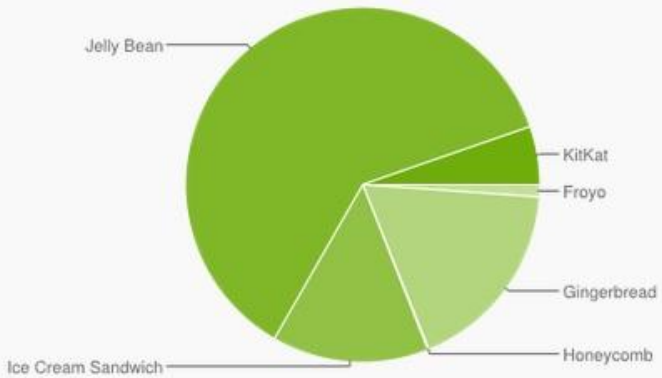
iOS operacinė sistema sukurta naudojant tą patį Unix branduolį, kokį Apple naudojo Mac OS X operacinėje sistemoje. Taip pat, naudojami BSD prievadai (angl. Sockets) interneto prieigai, Objective-C, C/C++ kompiliatoriai pagrindiniam vykdymui (angl. Native performance). Vartotojo sąveikai naudojamas Cacao Touch programavimo karkasas. Jo veikimas paremtas modelio – vaizdo – valdiklio (angl. Model-View-Controller) paradigma.

1.1.3. API lygiai

Android operacinė sistema vystoma išleidžiant naujas Android OS versijas. Pavyzdžiui: Android 1.1. Versijos keičiasi atlikus įvairius pakeitimus, ištaisius nedideles klaidas, pagerinant vykdymą (angl. performance) ar atlikus didesnius pokyčius. Tačiau mobiliųjų aplikacijų kūrėjams svarbu yra ne Android versija, o API lygis. API lygis – tai sveikasis skaičius, reiškiantis aplikacijų karkaso versiją. Šis karkasas taip pat atnaujinamas kartas nuo karto, pridedant daugiau funkcionalumo, bibliotekų, tuomet jo versija ir pasikeičia. Tačiau API lygis nesikeičia kartu su Android versija. Ne visada atnaujinant pačią operacinę sistemą yra keičiamas aplikacijų karkasas. Pavyzdžiui, 2 lygio karkasas (API 2) pradėtas naudoti Android 1.1 versijoje, o 3 lygio karkasas (API 3) pradėtas naudoti tik Android 1.5 versijoje. Naujesnis karkasas yra papildomas naujomis galimybėmis, bibliotekomis, tačiau palaiko ir senesnių karkasų turėtas bibliotekas. Todėl bendrai galimybės didėja su kiekviena karkaso versija. Kita vertus, programa sukurta aukštesnio lygio karkasui, neveiks ant žemesnio lygio karkasą turinčios Android versijos. Dėl to, programų kūrėjui svarbu pasirinkti kuo žemesnį API tam, kad kuo daugiau vartotojų galėtų naudotis programa.

Šiuo metu naujausia Android versija yra 4.4, pavadinta KitKat, su API 19. Šiuo metu populiariausios Android versijos pateiktos 1 pav.

Versija	Pavadinimas	API	Paplitimas
2.2	Froyo	8	1.1%
2.3.3 - 2.3.7	Gingerbread	10	17.8%
3.2	Honeycomb	13	0.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	14.3%
4.1.x	Jelly Bean	16	34.4%
4.2.x		17	18.1%
4.3		18	8.9%
4.4	KitKat	19	5.3%



1 pav. Populiariausios Android versijos 2014 metų balandžio 1 d. duomenimis
Šaltinis: http://developer.android.com/about/dashboards/index.html

Kaip matosi paveikslėlyje, šiuo metu populiariausia yra Android operacinė sistema, kodiniu pavadinimu Jelly Bean, apimanti versijas 4.1 – 4.3. Populiariausias API yra 16, naudojamas Android 4.1 versijoje.

1.2. Darbinės srities analizė

Kuriant aplikaciją netoliese esančių lankytinų vietų paieškai, reikia išsiaiškinti ko tokiai programai reikia. Visų pirma, lankytinos vietos, tai žymios tam tikros vietovės vietos, turinčios istoriją, galbūt tam tikrą prasmę gyventojams. Tai gali būti įvairūs paminklai, statulos, statiniai,

bažnyčios, muziejai ar kitokie reikšmingi pastatai. Norint rasti netoliese esančius, reikia nustatyti savo buvimo vietą. Tam naudojamas GPS, kuris susisiekia su netoliese esančiais palydovais ir pagal jų duomenis apskaičiuoja vietą. Nustačius lankytinas vietas, svarbu pateikti apie juos informaciją, nurodyti kur jie yra, kuo jie reikšmingi. Yra sukurta nemažai mobilių aplikacijų nustatančių ir atvaizduojančių vietą, ar pateikiančių informaciją apie lankytinas vietas. Todėl būtina jas atidžiau apžiūrėti ir paanalizuoti.

1.2.1. Google maps ir kitos navigacijų programos ar įrenginiai

Šio grupės programos labiau pritaikytos vietos nustatymui bei maršrutų atvaizdavimui. Jose svarbu žemėlapio atvaizdavimas, kuris būna labai įvairius, detalus, tikslus. Kai kurie iš jų, geba žemėlapyje atvaizduoti ir lankytinas vietas, tačiau nepateikia pakankamai informacijos apie jas.

1.2.2. Šiauliai info

Tai aplikacija iš aplikacijų rinkinio City Info, kurį sudaro programos, skirtos Lietuvos miestams. Šios programos, skirtos tiek turistams, tiek vietiniams gyventojams. Šiauliai info aplikacijoje galima rasti įvairios informacijos apie turizmą, renginius, lankytinas vietas, muziejus, paslaugas, paskaityti naujienas, išsikviesti taksi ar pagalbą, užsisakyti maisto į namus, peržiūrėti darbo skelbimus. Viskas susiję su Šiaulių miestu. Programa tikrai daug visko pateikia, tačiau nepateikia žymių vietų istorijos, ar platesnio aprašymo. Nors žemėlapyje atvaizduojamos vietos, neskaitant muziejų ar kitokių pastatų, nėra galimybės susidaryti maršruto joms apžiūrėti.

1.2.3. VisitIN.lt

Nors „Google Play“ parduotuvėje ši aplikacija vadinasi „VisitIN.lt“, internete ši programa vadinama „Lietuvos turizmas tavo telefone“. Ši programa skirta informacijai apie įvairias vietas, pastatus, viešbučius. Pateikiamas pakankamai platus kiekvienos vietos aprašymas, taip pat adresas, jei yra darbo laikas, oficialus tinklalapis. Galima matyti sąrašą vietų, esančių netoli, taip pat galima įsijungti žemėlapį, kuriame matomos pasirinktos Lietuvos dalies žymios vietos, pastatai ar viešbučiai, ant jų paspaudus galima pamatyti informaciją apie juos. Tačiau, kaip ir anksčiau minėta programa, ji neturi galimybės sudaryti maršruto pasirinktoms vietoms aplankyti.

1.2.4. Palyginimas

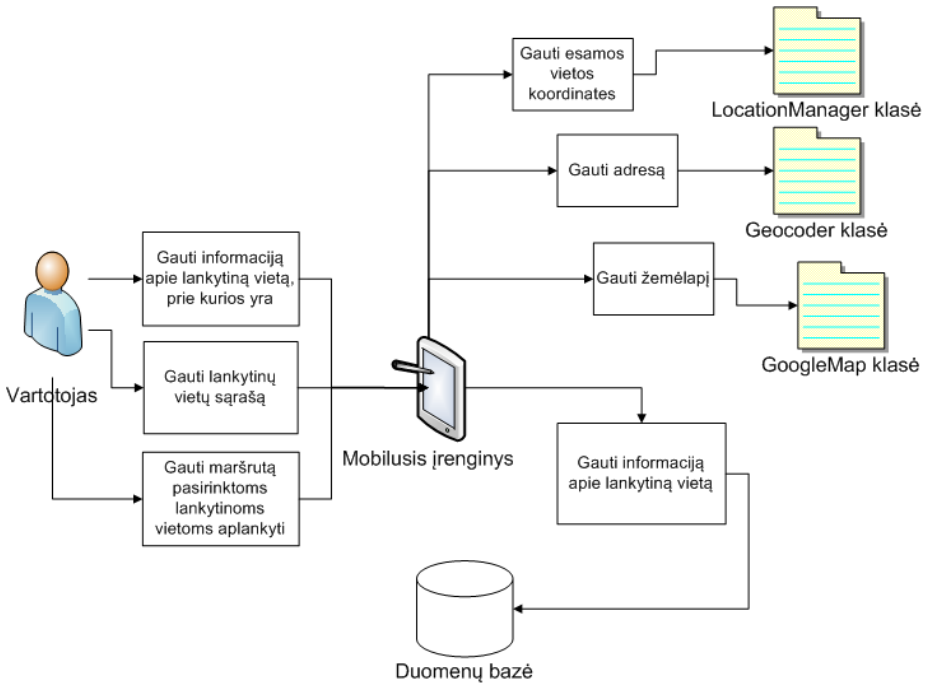
Išbandžius programas ir išanalizavus jų galimybes, jos palygintos tarpusavyje. Įvertintos tokios programų savybės: detalaus žemėlapio pateikimas, informacijos apie lankytinas vietas kiekis, ar pateikiamas sąrašas vietų, esančių netoliese, ar pateikiama informaciją apie objektą, prie kurio yra vartotojas, naudojantis programą, ar yra galimybė sudaryti maršrutą, pasirinktoms lankytinoms vietoms aplankyti. Programų palyginimas pateiktas 1 lentelėje.

1 lentelė. Panašaus tipo produktų palyginimas

Savybė	Google map ir kitos navigacijų programos ar įrenginiai	VisitIN.lt	City info
Detalus žemėlapis	Dažniausiai taip	Gana paprastas	Gana paprastas
Informacija apie lankytinas vietas	Dažniausiai nėra arba minimali informacija	Labai mažai	Pakankamai daug
Vietų, esančių netoli, sąrašas ir informacija	Nepateikiama	Pateikiama detaliai	Programos sukurtos kiekvienam miestui, todėl juose esančios vietos gali būti vadinamos netoli esančiomis
Informacijos pateikimas apie vietą, prie kurios yra vartotojas	Gali būti pateikiamas tik pavadinimas. Dažniausiai, nieko nepateikiama	Nepateikiama	Nepateikiama
Maršruto sudarymas lankytinoms vietoms aplankyti	Nėra galimybės	Nėra galimybės	Nėra galimybės

Palyginus programas galima teigti, jog nei viena aprašyta programa neatlieka visų reikalingų funkcijų.

1.3. Darbinės srities modelis



2 pav. Mobilios aplikacijos modelis

Mobili aplikacija netoliese esančių lankytinų vietų paieškai, visų pirma, turi apimti lankytinų vietų, kurios yra netoli, nustatymui. Kadangi, Lietuvoje atstumai yra pakankamai nedideli, palyginus su didžiosiomis Europos ar kitų kontinentų šalimis, sąvoką „netoliese esančios lankytinos vietos“ galime suprasti kaip tame pačiame mieste esančias vietas. Taigi, programėlė turi sugebėti

rasti tame pačiame mieste, kaip ir asmuo, kuris naudojasi aplikacija, esančias vietas. Tam reikia nustatyti programos buvimo vietą. Svarbu, jog vartotojas žinotų, kur jis yra, kur yra lankytinos vietos. Taip pat, vartotojui gali būti naudinga sužinoti apie pastatą, ar vietą, prie kurio jis stovi. Vartotojas turėtų gauti informaciją apie tą vietą, t.y. sužinoti pavadinimą, aprašymą, kuris gali būti tos vietos istorija ar reikšmė.

Kadangi, tokia mobili aplikacija neapsiriboja vienu veiksmu, ji turi turėti kelis vartotojo sąsajos langus. Bent vienas vartotojo sąsajos langas reikalingas vienai programos funkcijai atlikti. Be to, reikalingas meniu langas, skirtas vartotojui pasirinkti norimą veiksmą. Vartotojo langai turėtų būti suformuoti taip, jog vartotojas lengvai suprastų ką reikia daryti norint gauti rezultatus. Turi būti pateikti aiškūs nurodymai, ką reikia nurodyti tam, kad būtų galima atlikti užduotį.

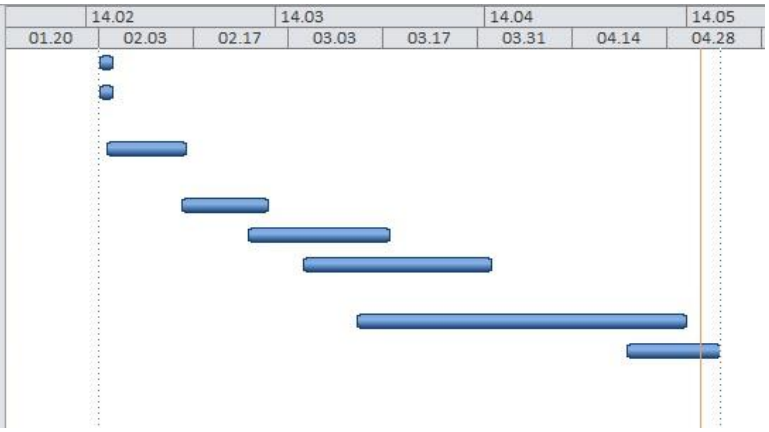
Duomenų bazė, kurioje saugoma informacija apie vietas, turi apimti tris pagrindinius dalykus: miesto informaciją, gatvės informaciją, pastato/lankytinos vietos informaciją. Miesto informacija neturėtų apsiriboti vien miesto pavadinimu. Žmogui, atvykusiam iš kito miesto, gali būti įdomi ir pačio miesto istorija, aprašymas. Todėl duomenų bazėje turi būti numatyta saugoti šią informaciją. Taip pat, turėtų būti ir su gatvės informacija. Ji turėtų apimti ne tik gatvės pavadinimą, ir miesto, kuriame ji yra, pavadinimą, bet ir aprašymą, jei jis yra prasmingas ir vertas paminėti. Informacija apie lankytiną vietą turi apimti miesto pavadinimą, kuriame ta vieta yra, gatvės pavadinimą, kurioje ji yra, bei numerį. Kitaip sakant, lankytinos vietos viena iš informacijos dalių turi būti adresas. Kita dalis, be abejonės, yra aprašymas, kuris gali būti istorija arba reikšmingumas, ar kažkokia svarbi, naudinga informacija.

2. PROJEKTINĖ DALIS

2.1. Projekto (darbo) vykdymo planas.

Darbui atlikti ir nustatytiems uždaviniams bei tikslui įvykdyti išsikeltos užduotys ir sudarytas jų vykdymo planas. Planas pateiktas 2 paveikslėlyje.

Nr.	Užduotis	Pradžia	Pabaiga
1	Ištirti vartotojų poreikius	2014.02.03	2014.02.04
2	Ištirti aplikacijai būdingus reikalavimus	2014.02.03	2014.02.04
3	Suprojektuoti duomenų bazę	2014.02.04	2014.02.15
4	Suprojektuoti aplikaciją	2014.02.15	2014.02.27
5	Realizuoti duomenų bazę	2014.02.25	2014.03.17
6	Užpildyti duomenų bazę duomenimis	2014.03.05	2014.04.01
7	Realizuoti aplikaciją	2014.03.13	2014.04.30
8	Testuoti aplikaciją	2014.04.22	2014.05.05

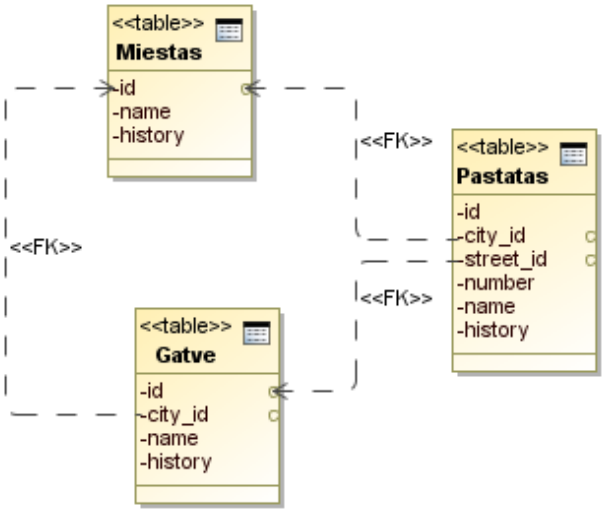


3 pav. Projekto vykdymo planas

Kaip matosi 2 paveikslėlyje, norint atlikti darbą reikia: ištirti vartotojų poreikius, ištirti aplikacijai būdingus reikalavimus, suprojektuoti duomenų bazę, suprojektuoti aplikaciją, realizuoti duomenų bazę, užpildyti duomenų bazę duomenimis, realizuoti aplikaciją, testuoti aplikaciją.

2.2. Pradinis projekto aprašymas.

Remiantis temos analize, darbinės srities analize ir darbinės srities modeliu, suprojektuota duomenų bazė, skirta informacijai apie lankytinas vietas saugoti. Duomenų bazės struktūra matoma 3 paveikslėlyje.



4 pav. Duomenų bazės struktūra

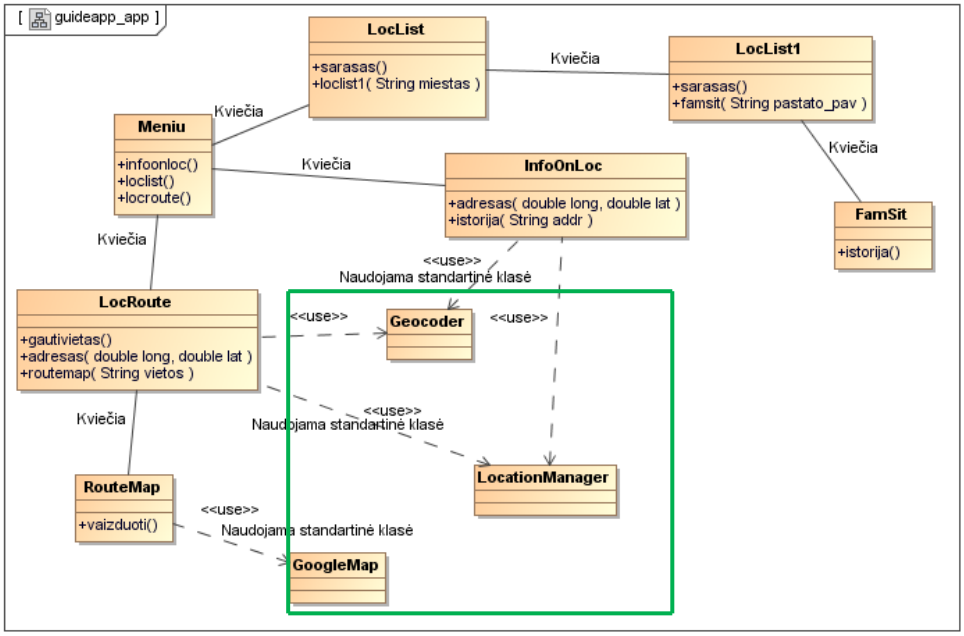
Kaip matosi iš paveikslėlio, duomenų bazę sudaro trys lentelės: miestas, gatvė ir pastatas. Miesto lentelė sudaryta iš trijų laukų: id, name ir history. „Id“ laukas skirtas miesto identifikatoriui,

kuris bus naudojamas kitose lentelėse. „Name“ laukas skirtas miesto pavadinimui saugoti. „History“ laukas skirtas informacijai apie konkretų miestą saugoti. Čia saugomas miesto aprašymas.

Gatvės lentelė sudaryta iš keturių laukų: id, city_id, name, history. „Id“ laukas skirtas gatvės identifikatoriui, kuris bus naudojamas pastato lentelėje. „City_id“ laukas skirtas susiejimui su miesto lentelėje saugomu miestu, kuriame yra gatvės lentelės įraše saugoma gatvė, naudojant miesto identifikatorių. „Name“ laukas skirtas gatvės pavadinimui saugoti. „History“ laukas skirtas informacijai apie konkrečią gatvę saugoti. Čia saugomas gatvės aprašymas.

Pastato lentelėje yra šeši laukai: id, city_id, street_id, number, name, history. „Id“ laukas skirtas pastato identifikatoriui. „City_id“ laukas skirtas susiejimui su miesto lentelėje saugomu miestu, kuriame yra pastato lentelės įraše saugoma lankytina vieta, naudojant miesto identifikatorių. „Street_id“ laukas skirtas susiejimui su gatvės lentelėje saugoma gatve, kurioje yra pastato lentelės įraše saugoma lankytina vieta, naudojant gatvės identifikatorių. „Number“ laukas skirtas lankytinos vietos adreso numeriui saugoti. „Name“ laukas skirtas lankytinos vietos pavadinimui saugoti. „History“ laukas skirtas lankytinos vietos aprašymui saugoti.

Įvertinus vartotojų poreikius ir tokiai aplikacijai būdingus reikalavimus suprojektuotos klasės: Meniu, InfoOnLoc, LocList, LocList1, FamSit, LocRoute, RouteMap. Surastos ir pridėtos prie projekto standartinės klasės, reikalingos vietai nustatyti ir adresui gauti: LocationManager, Geocoder. Surasta ir pridėta prie projekto standartinė klasė, skirta žemėlapiui atvaizduoti: GoogleMap. Sudaryta klasių diagrama matoma 4 paveikslėlyje. Paveikslėlyje apibrėžtos klasės, kurios nebus sukurtos, jos yra standartinės klasės, sukurtos Android programinės įrangos vystymo įrankių rinkinio kūrėjų.



5 pav. Projektuojamos aplikacijos klasių diagrama

Klasės: Meniu, LocList, LocList1, Famsit, InfoOnLoc, LocRoute ir RouteMap, bus vartotojo sąsajos langus valdančios klasės. Meniu klasė turės tokias funkcijas: infoonloc(), kuri iškviečia klasę InfoOnLoc; loclist(), kuri iškviečia klasę LocList; locroute(), kuri iškviečia klasę LocRoute. InfoOnLoc klasė turės funkcijas: adresas(double long, double lat), kuri, pagal gautas koordinates iš LocationManager klasės objekto, gaus adresą iš Geocoder klasės objekto; istorija(String addr), kuri, pagal gautą adresą iš adresas funkcijos, gaus informaciją apie tą vietą ir atvaizduos ją vartotojui. LocList klasė turės funkcijas: sarasas(), kuri gaus iš duomenų bazės miestų sąrašą ir jį atvaizduos vartotojui; loclist1(String miestas), kuri iškvies klasę LocList1, perduodama miesto pavadinimą. Klasė LocList1 turės funkcijas: sarasas(), kuri gaus lankytinų vietų sąrašą nurodytame mieste ir atvaizduos jį vartotojui; famsit(String pastato_pav), kuri iškvies klasę FamSit, perduodama lankytinos vietos pavadinimą. Famsit klasė turės funkciją istorija(), kuri pagal lankytinos vietos pavadinimą gaus miesto, gatvės ir lankytinos vietos informaciją iš duomenų bazės ir ją atvaizduos vartotojui. LocRoute klasė turės funkcijas: gautivietas(), kuri gaus lankytinų vietų sąrašą, esančių mieste, kuriame yra vartotojas, ir atvaizduos jį vartotojui; adresas(double long, double lat), kuri, pagal gautas koordinates iš LocationManager klasės objekto, gaus adresą iš Geocoder klasės objekto; routemap(String vietos), kuri iškvies klasę RouteMap, perduodama pasirinktų lankytinų vietų pavadinimus ir esamos vietos adresą, sudėtus į vieną String tipo kintamąjį. RouteMap klasė turės funkciją vaizduoti(), kuri iš lankytinų vietų pavadinimų gaus jų adresus iš duomenų bazės, nusiųs užklausą su adresais Google kelio nurodymų paslaugai (angl. Google Directions API) ir gautas koordinates atvaizduos žemėlapyje vartotojui.

Aplikacija turėtų atlikti tris pagrindinius veiksmus: pateikti informaciją apie vietą, prie kurios yra vartotojas, pateikti visų lankytinų vietų sąrašą, sudaryti maršrutą, pasirinkus kelias, netoliese esančias vietas. Objektai pateikiami viename sąraše, neskirstant į kategorijas. Paieška realizuojama kaip sąrašo skaitymas, be galimybės įvesti.

Remiantis temos analize, nuspręsta, jog minimali mobilios aplikacijos palaikoma Android versija bus 4.1, nes būtent ši versija šiuo metu yra paplitusi labiausiai.

2.3. Programavimo įrankių ir priemonių pasirinkimo analizė.

Mobilią aplikaciją galima kurti šioms operacinėms sistemoms: Android OS, iOS. Android OS yra atviro kodo, priklausanti Google kompanijai, diegiama kelių firmų išmaniuosiuose įrenginiuose. iOS operacinės sistema yra specialiai skirta Apple kompanijos produktams iPhone ir iPad.

Android OS pasirinkta dėl šių privalumų: plačiai aprašytos visos klasės, kurias galima naudoti Android aplikacijose, daug įvairių kodo pavyzdžių, patirtis realizuojant studijų metu užduotas praktines užduotis Android sistemai.

Mobiliai aplikacijai projektuoti galima rinktis bet kurį braižymo įrankį, kuris realizuoja UML diagramas. Galima rinktis SmartDraw, ArgoUML, MS Visio, Magic Draw UML ir kitus įrankius. SmartDraw – detalios diagramos, nelabai patogi, nelabai aiški, mokama, nėra patirties dirbant šia priemone. ArgoUml – nemokama, neišvaizdžios diagramos, nelabai patogi, nesudėtinga, nėra patirties naudojantis šia priemone. MS Visio – detalios diagramos, mokama, vidutinio sudėtingumo, patogi naudotis, nėra patirties dirbant šia priemone. Magic Draw UML – detalios diagramos, gana patogi naudotis, nėra sudėtinga, mokama, sukaupta šiek tiek patirties dirbant šia priemone.

Mobilios aplikacijos projektavimui pasirinkta „Magic Draw UML“ priemonė. Ši priemonė pasirinkta dėl šių privalumų: pritaikyta JAVA kalbos sintaksei, turima patirtis dirbant su šia priemone, studijų metu mokoma ir rekomenduojama dirbti su šia priemone.

Mobilios aplikacijos realizavimui galima rinktis iš šių programavimo įrankių: Eclipse su Android SDK, Android Studio. Eclipse įrankis skirtas programoms Java kalba programuoti. Įdiegus Android kūrimo įrankius (angl. Android development tools, ADT), galima kurti mobilias aplikacijas Android operacinei sistemai. Galingas įrankis, kadangi pritaikytas didelėms Java programoms. Todėl labiau pritaikytas Java programoms, nei Android aplikacijoms. Sukaupta patirtis studijų metu, naudojant Eclipse įrankį tiek programuojant Java programas, tiek Android aplikacijas. Internete dauguma randamų pavyzdžių, kurti naudojant Eclipse. Android Studio įrankis skirtas specialiai Android aplikacijų programavimui. Šiek tiek panašus į Eclipse. Labiau specializuotas. Nėra patirties dirbant šiuo įrankiu.

Mobilios aplikacijos realizavimui pasirinktas „Android SDK” įrankis ir „Eclipse“ programavimo aplinka. „Android SDK“ – įrankių rinkinys, leidžiantis kurti „Android“ operacinei sistemai skirtas programas.

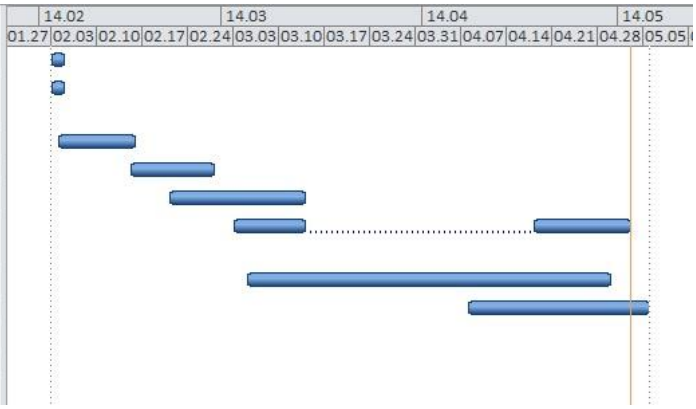
„Android SDK“ įrankis pasirinktas dėl šių privalumų: nemokamas ir nemokama dokumentacija, labai išsami dokumentacija, daug informacijos apie „Android“ platformą ir jos programų kūrimą. „Eclipse“ programavimo aplinka pasirinkta dėl šių kriterijų: suderinama su „Android SDK“, nemokama, patogi naudotis, patirtis realizuojant studijų metu užduotas praktines užduotis įvairiomis kalbomis, dauguma pavyzdžių kurti šioje aplinkoje.

Duomenims apie lankytinas vietas saugoti pasirinkta „MySQL“ duomenų bazė. Ši duomenų bazė pasirinkta dėl šių privalumų: nemokama, yra daug hostingų nemokamai talpinančių svetaines ir „MySQL“ duomenų bazes, sukaupta patirtis realizuojant studijų metu užduotas praktines užduotis naudojant šią duomenų bazę.

3. REALIZACINĖ DALIS

3.1. Darbų eigos grafas.

Nr.	Užduotis	Pradžia	Pabaiga
1	Ištirti vartotojų poreikiai	2014.02.03	2014.02.04
2	Ištirti aplikacijai būdingi reikalavimai	2014.02.03	2014.02.04
3	Suprojektuota duomenų bazė	2014.02.04	2014.02.15
4	Suprojektuota aplikacija	2014.02.15	2014.02.27
5	Realizuota duomenų bazė	2014.02.21	2014.03.13
6	Duomenų bazė užpildyta duomenimis	2014.03.03	2014.05.02
7	Realizuota aplikacija	2014.03.05	2014.04.29
8	Testuota aplikacija	2014.04.08	2014.05.05



6 pav. Darbų eigos grafas

Planuotos užduotys darbui atlikti nesikeitė. Šiek tiek keitėsi kai kurių užduočių laikas ir trukmė. Duomenų bazė pradėta realizuoti anksčiau, atsiradus poreikiui turėti duomenų bazę projektuojant aplikaciją tam, kad būtų galima tiksliau projektuoti duomenų perdavimą ir gavimą. Trukmė nesikeitė. Anksčiau pradėjus realizuoti duomenų bazę, anksčiau pradėti pildyti duomenys. Tačiau skirtingai nuo planavimo, ši užduotis nebuvo atlikta visa iki galo iš karto. Baigus realizuoti duomenų bazę, nutrauktas duomenų pildymas ir pradėtas aplikacijos realizavimas. Taip pasielgta, nes duomenų bazė turėjo pakankamai duomenų aplikacijai realizuoti. Trukmė šios užduoties nesikeitė. Aplikacijos realizavimas pradėtas anksčiau nei planuota, nes anksčiau atlikti darbai, kurie būtini aplikacijos realizavimo pradžiai. Aplikacijos realizavimas užtruko ilgiau nei planuota, nes ne visai tiksliai įvertintas darbo kiekis ir sudėtingumas. Mobilios aplikacijos testavimas pradėtas anksčiau, nes testavimas atliktas ne visos sukurtos aplikacijos, o kuriant atskiras dalis po vieną. Dėl tos pačios priežasties, pailgėjo testavimo atlikimo laikas.

3.2. Problemų ir jų sprendimų aprašymai ir pagrindimai.

Geocoder klasės kvietimas. Kartais Geocoder klasė grąžina klaidas, nepriklausomai nuo klasei perduodamų koordinačių. Net perdavus teisingas koordinatės (realios vietos), kartais gali grąžinti klaidą.

Funkcijoje kuri kviečia Geocoder klasę, klasės kvietimo vieta apgaubta try catch bloku, kuris atpažįsta klaidas ir neleidžia programai nustoti veikti. Blokui atpažinus klaidą, iš naujo kviečiama funkcija, kuri kviečia Geocoder klasę. Taip užtikrinama kad Geocoder klasė bus kviečiama tol, kol bus gautas adresas arba, kol bus nerasta galimo adreso.

Geocoder grąžinami adresai. Geocoder klasės funkcija getFromLocation grąžina galimų adresų masyvą. Tai gali būti šalia vienas kito esantys adresai, kurie pasaulio mastu yra visiškai arti, koordinatėse skiriasi gal 6 ar 7 skaičius po kablelio. Sunku išsirinkti vieną adresą iš masyvo.

Padaryta prielaida, jog pats pirmasis adresas masyve yra tiksliausias, paimamas ir naudojamas pirmas adresas grąžinamų adresų masyve.

Adreso formavimas Google užklausai. Nėra dokumentuota kaip turi būti rašomas konkretus adresas siunčiant užklausą Google kelio nurodymų paslaugai. Yra pavyzdžių tik siunčiant užklausas maršrutui tarp Jungtinėse Amerikos Valstijose esančių miestų gauti. Nėra aišku kaip turi būti formuojamas lietuviškas adresas.

Pasinaudojus Mozilla internetine naršykle asmeniniame kompiuteryje, išnagrinėtas https://maps.google.lt tinklalapyje esantis žemėlapis. Tyrinėtas Šiaulių žemėlapis, ieškant įvairių vietų ir tikrinant jų adresus. Tuomet, tame pačiame žemėlapyje bandyta gauti maršrutą tarp dviejų adresų Šiaulių mieste. Kai maršrutas buvo atvaizduotas, išanalizuotas naršyklės tinklalapio adreso laukelyje esantis adresas ir pagal jo šabloną suformuotos užklausų nuorodos mobilioje aplikacijoje.

3.3. Galutinio projekto būsenos aprašymas.

3.3.1. Duomenų bazė

Realizuota MySQL duomenų bazė lygiai tokia pati kaip buvo suprojektuota. Nepakeistas nei vienas laukas, nei vienas pavadinimas. Duomenų bazė patalpinta nemokamame 6te.net serveryje. Sukurti 7 php failai skirti paimti duomenis pagal užklausą iš duomenų bazės ir perduoti juos programai kaip Json objektus. Patys php failai pateikiami prieduose.

3.3.2. Mobili aplikacija

Realizuota mobili aplikacija. Minimali reikalaujama Android OS versija – 4.1. Programai veikti reikalinga nuolatinė interneto prieiga, taip pat turi būti įjungtas GPS modulis įrenginyje. Kitu atveju, nebus gaunama informacija.

Programa sukurta naudoti Lietuvoje. Aplikacija priderinta prie Lietuvos adresų. Jei iš Geocoder klasės būtų gaunamas adresas, kurį sudarytų gatvės pavadinimas, tada numeris, tada miesto pavadinimas, programa galėtų veikti ir užsienyje.

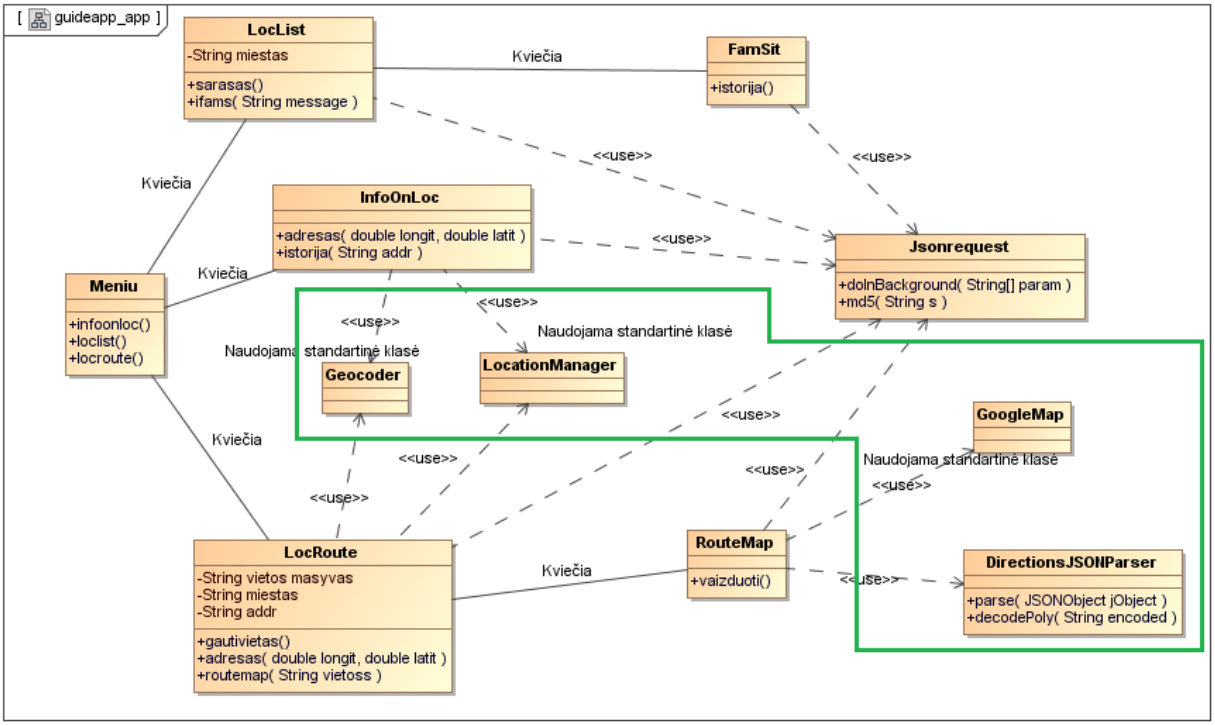
Klasės

Aplikacija naudoja tris standartines klases: LocationManager, Geocoder ir GoogleMap. LocationManager klasė skirta gauti įrenginio buvimo vietą naudojant GPS modulį, įmontuotą mobiliajame įrenginyje, kuris susisiekia su palydovais, arba naudojant interneto ryšį. Pasirinktas gavimo būdas naudojant GPS modulį, nes šiuo būdu nustatyta vieta gerokai tikslesnė. Geocoder klasė skirta gauti koordinates, perduodant adresą, tačiau gali atlikti ir atvirkštinį darbą, t.y. grąžinti adresą, perduodant koordinates. Šios klasės veikimui reikalingos Google paslaugos (angl. Google services), kurios turi būti įdiegtos išmaniajame įrenginyje. Jos įdiegiamos kartu su Google Play parduotuve. Taip pat, reikalingas interneto ryšys, kuriuo klasė siunčia užklausas adresui gauti.

GoogleMap klasė skirta Google žemėlapio atvaizdavimui įrenginio ekrane. Aplikacijos vartotojo sąsajos aprašo faile aprašomas fragment komponentas, kuriame bus vaizduojamas žemėlapis. Programos vykdymo metu žemėlapis užkraunamas interneto ryšiu tiesiai iš Google žemėlapių.

Prie programos projekto pridėta Google play paslaugų biblioteka, kuri reikalinga maršrutui gauti naudojamoms klasėms ir funkcijoms veikti.

Aplikacijos klasės šiek tiek skiriasi nuo projektuotos aplikacijos. Nesukurta LocList1 klasė (tapo nereikalinga, nes LocList klasėje ListView komponente duomenis galima keisti). Sukurta klasė Jsonrequest, skirta duomenų išrinkimui iš duomenų bazės. Pridėta papildoma klasė DirectionsJSONParser, kuri paimta iš George Mathew pavyzdžio, prieinamo šiuo adresu: http://wptrafficanalyzer.in/blog/route-between-two-locations-with-waypoints-in-google-map-android-api-v2/. To paties autoriaus, to paties pavyzdžio, dalis kodo įdėta klasėje RouteMap. Klasės išeities kode ši dalis pažymėta komentarais. Pakeisti kai kurie kintamųjų ar funkcijų pavadinimai, pridėti klasių kintamieji. Perbraižyta klasių diagrama pateikta prieduose ir 6 paveikslėlyje. Paveikslėlyje apibrėžtos klasės, kurios nebuvo sukurtos šiame darbe, o tik panaudotos.



7 pav. Realizuotos aplikacijos klasių diagrama

Meniu klasė išliko nepakitusi, lyginant su projektuota aplikacija. Ji turi funkcijas: infoonloc(), kuri iškviečia klasę InfoOnLoc; loclist(), kuri iškviečia klasę LocList; locroute(), kuri iškviečia klasę LocRoute. Tai pagrindinio meniu klasė iš kurios pereinama į kitas klases.

InfoOnLoc klasė liko tokia pati kaip projektuota, tik klasės funkcijos adresas parametrų pavadinimai pasikeitė. Ši klasė turi tokias funkcijas: adresas(double longit, double latit), kuri, pagal gautas koordinates iš LocationManager klasės objekto, gauna adresą iš Geocoder klasės objekto;

istorija(String addr), kuri, pagal gautą adresą iš adresas funkcijos, gauna informaciją apie tą vietą ir atvaizduoja ją vartotojui. Tai klasė, skirta gauti informaciją apie miestą, kuriame yra vartotojas, gatvę, kurioje yra vartotojas, lankytiną vietą, prie kurios yra vartotojas.

LocList klasėje pasikeitė viena funkcija ir atsirado klasės string tipo kintamasis miestas. LocList klasė turi tokias funkcijas: sarasas(), kuri gauna iš duomenų bazės miestų sąrašą ir jį atvaizduoja vartotojui; ifams(String message), kuri iškviečia klasę FamSit, perduodama lankytinos vietos pavadinimą. Ši klasė skirta atvaizduoti pasirinkto miesto lankytinų vietų sąrašui ir pasirinktą lankytiną vietą perduoti klasei FamSit.

Kaip minėta aukščiau, LocList1 klasė nesukurta, vietoj jos iš LocList klasės kviečiama FamSit klasė. FamSit klasė nepasikeitė, lyginant su aplikacijos projektu. Ji turi funkciją istorija(), kuri pagal lankytinos vietos pavadinimą gauna miesto, gatvės ir lankytinos vietos informaciją iš duomenų bazės ir ją atvaizduoja vartotojui. Ši klasė skirta atvaizduoti informaciją apie pasirinktą lankytiną vietą.

LocRoute klasėje pridėti trys kintamieji, du string tipo kintamieji: miestas, addr, ir vienas string tipo masyvas vietos. Pasikeitė ir funkcijų parametrų pavadinimai. Realizuotoje aplikacijoje LocRoute klasė turi tokias funkcijas: gautivietas(), kuri gauna lankytinų vietų sąrašą, esančių mieste, kuriame yra vartotojas, ir atvaizduoja jį vartotojui; adresas(double long, double lat), kuri, pagal gautas koordinates iš LocationManager klasės objekto, gauna adresą iš Geocoder klasės objekto; routemap(String vietos), kuri iškviečia klasę RouteMap, perduodama pasirinktų lankytinų vietų pavadinimus ir esamos vietos adresą, sudėtus į vieną String tipo kintamąjį. Ši klasė skirta vartotojui pasirinkti lankytinas vietas, kurioms jis nori sudaryti maršrutą, ir nustatyti vartotojo esamą vietą.

RouteMap klasėje pridėta papildoma klasė LoadViewTask, be to, joje prijungiama nauja klasė DirectionsJSONParser. Ši klasė turi funkciją vaizduoti(), kuri iš lankytinų vietų pavadinimų gauna jų adresus iš duomenų bazės, nusiunčia užklausą su adresais Google kelio nurodymų paslaugai (angl. Google Directions API) ir gautas koordinates atvaizduoja žemėlapyje vartotojui. Ši klasė skirta vartotojo pasirinktoms lankytinoms vietoms sudaryti maršrutą, naudojant Google kelio nurodymų paslaugą, bei atvaizduoti gautą maršrutą vartotojui. Pridėta papildoma klasė LoadViewTask, kuri paimta iš pavyzdžio, kurį pateikė vartotojas, pasivadinęs DimasTheDriver. Pavyzdys prieinamas šiuo adresu: http://www.41post.com/4588/programming/android-coding-a-loading-screen-part-1. Ši klasė skirta rodyti vaizdo įkrovimo užsklandą. Klasės išeities kode ši dalis pažymėta komentarais.

Naujai sukurta klasė Jsonrequest turi tokias funkcijas: doInBackground(String[] param), kuri yra būtina visoms asinchroninių užduočių (angl. Async task) klasėms, joje vykdoma http užklausa ir grąžinamas atsakymas; md5(String s), kurioje tam tikra adreso dalis koduojama md5 koduote,

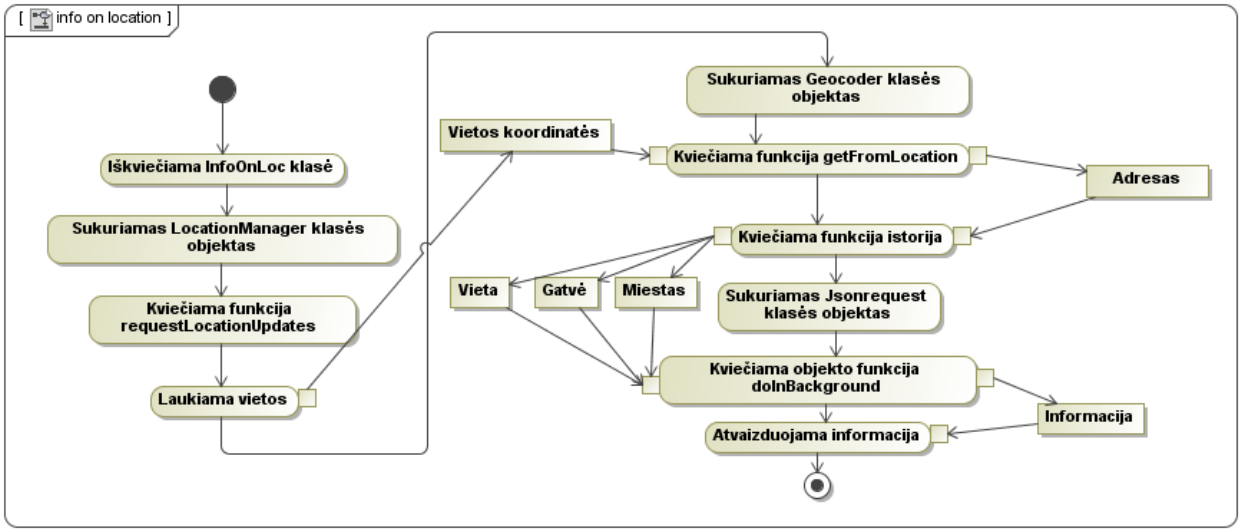
norint įsitikinti, jog gauta tinkama užklausa duomenų bazei. Ši klasė skirta siųsti užklausą serveryje, kuriame yra patalpinta duomenų bazė, esantiems php failams, kurie pagal užklausą išrenka duomenis iš duomenų bazės ir grąžina juos kaip json objektą. Gautą json objektą klasė paverčia string tipo kintamuoju ir jį grąžina. Tokiu būdu gaunama informacija apie lankytinas vietas, kuri saugoma duomenų bazėje.

Pridėta papildoma klasė DirectionsJSONParser. Kaip minėta anksčiau, ši klasė yra paimta iš George Mathew pavyzdžio, kurio atviras kodas patalpintas internete. Ši klasė apdoroja iš Google kelio nurodymų paslaugos gautą json objektą. Ji surenka reikalingas vietas (kuriose nurodomos koordinatės) iš didelio gauto json objekto, sudeda jas į sąrašų sąrašą kaip linijas ir jį grąžina. DirectionsJSONParser klasė turi tokias funkcijas: parse(JSONObject jObject), kuri apdoroja json objektą, sudeda viską į sąrašų sąrašą ir viską grąžina; decodePoly(String encoded), kuri iš gauto string tipo kintamojo, saugančio tam tikrą kiekį taškų, išrenka taškus ir juos sujungia į liniją, kurią grąžina.

Funkcijos

Realizuotoje aplikacijoje įgyvendinti trys veiksmai: miesto, gatvės ir lankytinos vietos, pagal vartotojo buvimo vietą, informacijos išrinkimas ir pateikimas vartotojui; informacijos pateikimas apie lankytiną vietą, kuri pasirenkama iš sąrašo visų lankytinų vietų bet kuriame mieste; maršruto sudarymas lankytinų vietų aplankymui, pradedant nuo vartotojo buvimo vietos ir baigiant toje pačioje vietoje, aplankant nuo vienos iki penkių lankytinų vietų.

Pirmos funkcijos, kurioje pateikiama informacija, pagal buvimo vietą, veikimas pateiktas 7 paveikslėlyje.

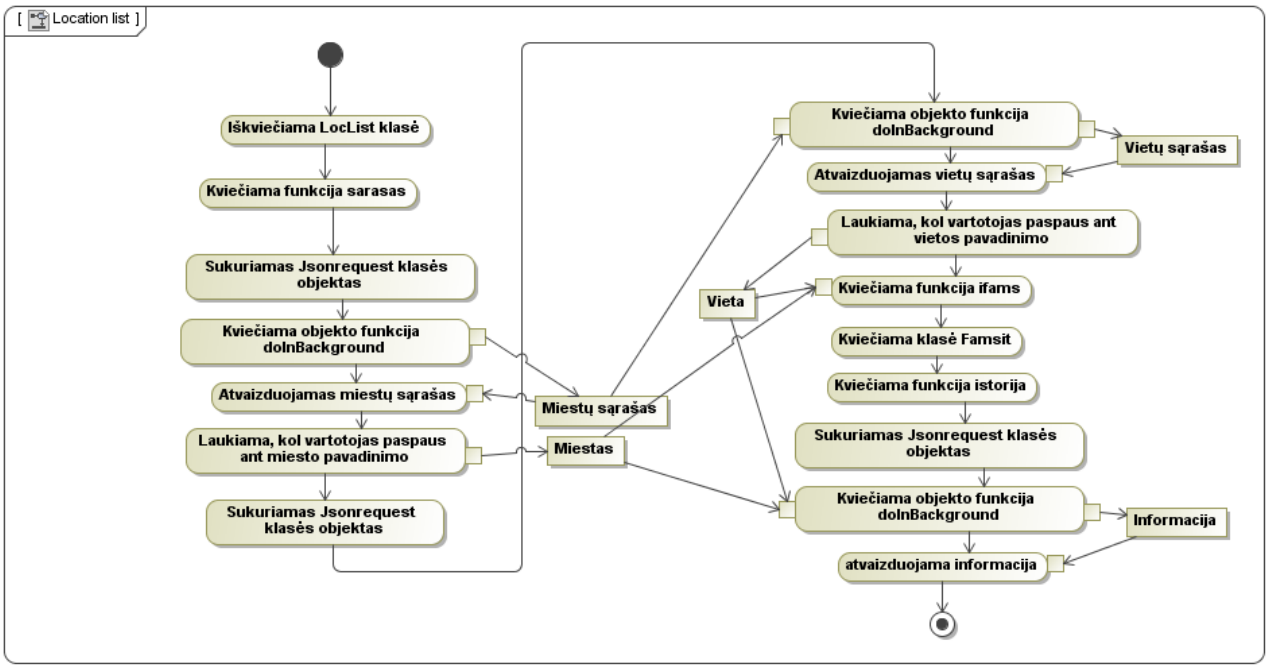


8 pav. Funkcijos informacijos pateikimas pagal buvimo vietą veikimo diagrama

Pasirinkus lankytinos vietos informacijos gavimą pagal buvimo vietą, pirmiausia, iškviečiama klasė InfoOnLoc, kuri kartu valdo ir vartotojo sąsają. Iškviestoje klasėje sukuriamas LocationManager klasės objektas ir kviečiama objekto funkcija requestLocationUpdates, kuri

kreipiasi į GPS modulį Android sistemoje ir laukia koordinačių. Gavus koordinates, ilguma ir platuma priskiriami kintamiesiems ir kviečiama funkcija adresas su parametrais ilguma ir platuma. Funkcijoje adresas sukuriamas Geocoder klasės objektas ir kviečiama objekto funkcija getFromLocation, kuri iš koordinačių gauna adresų sąrašą ir jį grąžina. Paimamas pirmas adresas (daroma prielaida, kad pirmas adresas tiksliausias), jis iš karto atvaizduojamas vartotojui pirmame TextView komponente, ir panaudojamas kaip parametras kviečiant funkciją istorija. Toje funkcijoje, pirmiausia, sukuriamas Jsonrequest klasės objektas. Gautas adresas padalinamas į tris dalis: gatvės pavadinimą, numerį, miesto pavadinimą. Su tomis trimis dalimis atliekami tokie patys veiksmai: kviečiama Jsonrequest objekto funkcija su dalimi kaip parametru; ta funkcija nusiunčia užklausą failams, kurie prisijungia prie duomenų bazės, išrenka duomenis ir juos grąžina funkcijai, kuri json objektą pavertusi string eilute, grąžiną ją pagrindinei klasei; pavadinimas ir informacija pateikiami vartotojui atskiruose TextView komponentuose.

Funkcijos, kurioje pateikiamas lankytinų vietų sąrašas ir informacija pagal pasirinktą vietą, veikimas pateiktas 8 paveikslėlyje.

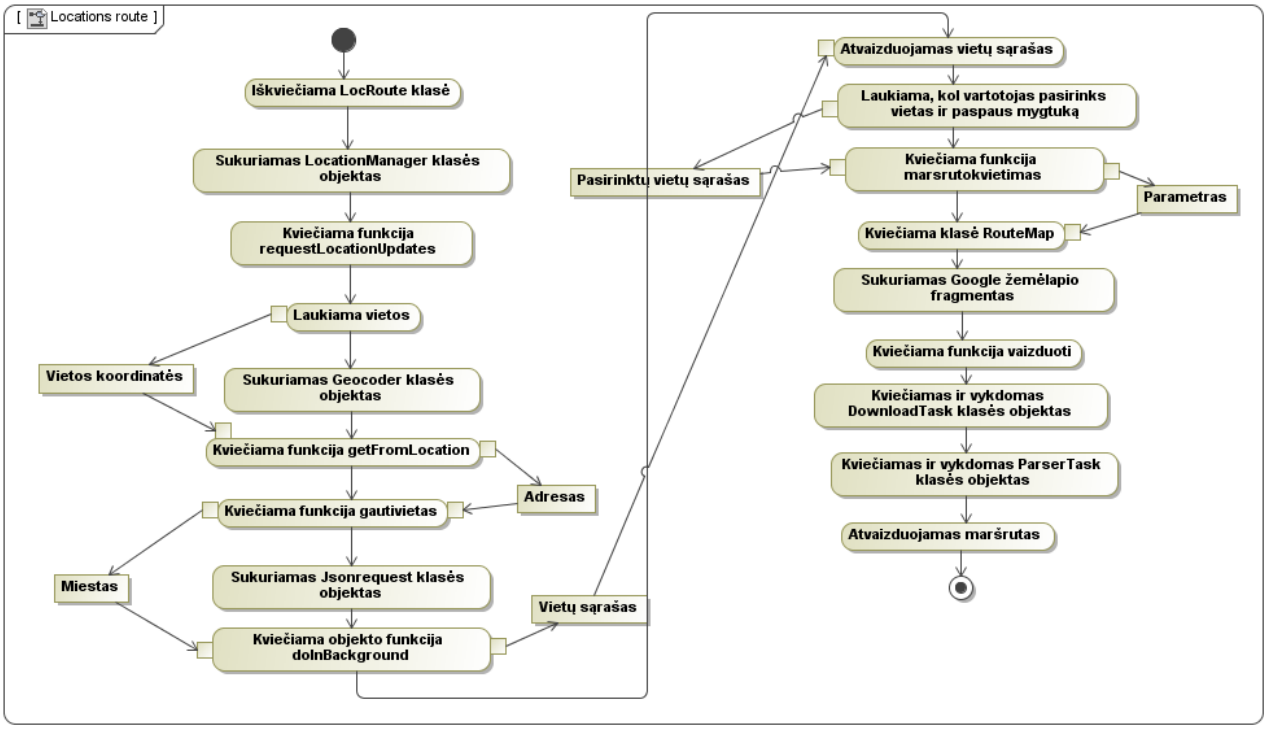


9 pav. Funkcijos lankytinų vietų sąrašo pateikimas veikimo diagrama

Pasirinkus informacijos pateikimą iš lankytinų vietų sąrašo pasirenkant lankytiną vietą, pirmiausia, iškviečiama klasė LocList, kuri kartu valdo ir vartotojo sąsają. Iškviestoje klasėje iš karto kviečiama funkcija sarasas. Joje sukuriamas Jsonrequest klasės objektas. Jei pagrindinės klasės kintamasis miestas neturi reikšmės, tuomet kviečiama Jsonrequest klasės objekto funkcija doInBackground (jos veikimas aprašytas prieš tai aprašytame veiksme), kuri grąžina miestų pavadinimus vienoje string tipo eilutėje. Pavadinimai išskaidomi ir sudedami į ListView komponentą, kuriame vartotojas mato miestų sąrašą. Laukiama kol vartotojas paspaus ant miesto

pavadinimo. Šiam paspaudus, klasės kintamajam miestas priskiriama reikšmė, ir dar kartą kviečiama funkcija sarasas. Vėl sukuriamas Jsonrequest klasės objektas. Šiuo atveju kintamasis miestas turi reikšmę, todėl Jsonrequest objekto funkcija doInBackground kviečiama su parametru, kuris yra miesto pavadinimas. Funkcija grąžina lankytinų vietų pavadinimus vienoje string tipo eilutėje. Pavadinimai išskaidomi ir sudedami į tą patį ListView komponentą. Laukiama, kol vartotojas pasirinks norimą lankytiną vietą. Vartotojui paspaudus ant pavadinimo, kviečiama funkcija ifams su parametru, kurį sudaro lankytinos vietos pavadinimas ir miesto pavadinimas. Ifams funkcija iškviečia klasę FamSit, kuri kartu valdo ir vartotojo sąsają, pridėdama gautą parametrą. Iškviestoje klasėje iš karto kviečiama funkcija istorija. Paimamas gautas parametras ir išskaidomas į du string tipo kintamuosius lankytinos vietos ir miesto pavadinimui. Sukuriamas Jsonrequest objektas, kviečiama jo funkcija doInBackground, perduodant miesto pavadinimą ir lankytinos vietos pavadinimą, kuri grąžina informaciją apie miestą, gatvę ir lankytiną vietą. Visa informacija pateikiama atskiruose TexView komponentuose vartotojui.

Funkcijos, kurioje sudaromas ir atvaizduojamas vartotojo pasirinktų vietų maršrutas, veikimas pateiktas 9 paveikslėlyje.



10 pav. Funkcijos pasirinktų vietų maršruto sudarymas veikimo diagrama

Pasirinkus maršruto sudarymo veiksmą, pirmiausia, kviečiama klasė LocRoute, kuri kartu valdo ir vartotojo sąsają. Iškviestoje klasėje sukuriamas LocationManager klasės objektas ir kviečiama objekto funkcija requestLocationUpdates, kuri kreipiasi į GPS modulį Android sistemoje ir laukia koordinačių. Gavus koordinates, ilguma ir platuma priskiriami kintamiesiems ir kviečiama funkcija adresas su parametrais ilguma ir platuma. Funkcijoje adresas sukuriamas Geocoder klasės

objektas ir kviečiama objekto funkcija getFromLocation, kuri iš koordinačių gauna adresų sąrašą ir jį grąžina. Paimamas pirmas adresas (daroma prielaida, kad pirmas adresas tiksliausias) ir priskiriamas pagrindinės klasės LocRoute string tipo kintamajam. Kviečiama funkcija gautivietas. Joje sukuriamas Jsonrequest klasės objektas ir kviečiama jo funkcija doInBackground, perduodant miesto pavadinimą. Gauta lankytinų vietų pavadinimų eilutė išskaidoma ir sudedama į ListView komponentą. Laukiama, kol vartotojas paspaus mygtuką, pasirinkęs bent vieną lankytiną vietą. Vartotojui nepasirinkus nei vienos vietos, mygtuko paspaudimas ignoruojamas. Vartotojui pasirinkus penkias vietas, šeštos rinktis neleidžiama. Vartotojui paspaudus ant nepažymėtos lankytinos vietos pavadinimo, ji įrašoma į pagrindinės klasės masyvą, ir pažymima varnele, o paspaudus ant pažymėtos lankytinos vietos, lankytina vieta ištrinama iš masyvo ir nuimamas pažymėjimas. Vartotojui paspaudus mygtuką, iškviečiama funkcija marsrutokvietimas. Joje vartotojo buvimo adresas ir lankytinos vietos iš masyvo sudedami į vieną string tipo eilutę ir kviečiama klasė RouteMap, kuri kartu valdo ir vartotojo sąsają, perduodant sudarytą eilutę. Iškviestoje klasėje sukuriamas Google žemėlapio fragmentas, ir kviečiama funkcija vaizduoti. Funkcijoje parametras, gautas iš LocRoute klasės, padalinamas į masyvą. Sukuriamas Jsonrequest klasės objektas ir kviečiama doInBackground funkcija, perduodant lankytinos vietos pavadinimą, tiek kartų, kiek masyve yra elementų. Funkciją grąžina tos vietos adresą. Tuomet suformuojamas užklausos Google kelio nurodymų paslaugai adresas, su visomis lankytinomis vietomis. Nuo šitos vietos vykdoma George Mathew programos kodo dalis. Ją sudaro dvi vidinės klasės (sukuriamos RouteMap klasės viduje), viena funkcija ir viena išorinė klasė (projekte sukuriama kaip atskira klasė). Pirmiausia funkcijoje vaizduoti sukuriamas vidinės klasės DownloadTask objektas ir kviečiama standartinė funkcija execute, kuri vykdo visą klasę, tai nėra klasėje aprašyta funkcija. Funkcijai execute perduodamas užklausos adresas. Pirmiausia, vykdoma funkcija doInBackground, kurioje kviečiama funkcija downloadUrl, perduodant užklausos adresą. Toje funkcijoje vykdoma url užklausa ir gautas atsakymas patalpinamas string tipo kintamajame, kuris grąžinamas kvietusiai funkcijai. DownloadTask klasės funkcija doInBackground gautą atsakymą grąžina klasę valdančiam objektui. Tuomet vykdoma funkcija onPostExecute, perduodant jai prieš tai gautą atsakymą. Toje funkcijoje sukuriamas kitos vidinės klasės ParserTask objektas ir kviečiama standartinė funkcija execute, kuri vykdo visą klasę. Funkcijai execute perduodamas gautas užklausos atsakymas. Vykdoma doInBackground funkcija. Gautas užklausos atsakymas patalpinamas kaip json objektas. Sukuriamas klasės DirectionsJSONParser objektas ir kviečiama funkcija parse, perduodant json objektą. Toje funkcijoje nagrinėjamas json objektas, randami taškai, kurie sukeliami į sąrašus, panaudojant decodePoly funkciją taškai suskirstomi pagal gaunamas linijas. Visas didelis sąrašų sąrašas grąžinamas funkciją kvietusiam objektui. Vidinė klasė ParserTask, kuri kvietė klasę DirectionsJSONParser, grąžina gautą sąrašų sąrašą. Vykdoma tos klasės funkcija onPostExecute.

Joje iš sąrašų sąrašo išrenkami taškai, sukuriami Point tipo objektai, jiems priskiriami taškai, sukuriamos linijos, joms parenkamas plotis, spalva. Sudėjus visas linijas, jos siunčiamos Google žemėlapio fragmentui, kuris jas atvaizduoja žemėlapyje. Vartotojas mato sudarytą maršrutą.

3.4. Darbo rezultatų analizė.

Pradinis projektas. Pradiniame projekte sumodeliuota duomenų bazė duomenims saugoti, sudaryta iš trijų lentelių: miestas, gatve, pastatas. Miesto lentelę sudaro trys laukai: id, name ir history. Gatvės lentelę sudaro keturi laukai: id, city_id, name, history. Pastato lentelę sudaro šeši laukai: id, city_id, street_id, number, name, history. Suprojektuotos mobilios aplikacijos klasės (Meniu, LocList, LocList1, Famsit, InfoOnLoc, LocRoute ir RouteMap), funkcijos, kurias galėtų turėti klasės. Rastos ir numatytos, kada bus kviečiamos, standartinės klasės LocationManager, Geocoder ir GoogleMap.

Galutinis projektas. Galutiniame projekte realizuota duomenų bazė duomenims saugoti, sudaryta iš trijų lentelių: miestas, gatve, pastatas. Miesto lentelę sudaro trys laukai: id, name ir history. Gatvės lentelę sudaro keturi laukai: id, city_id, name, history. Pastato lentelę sudaro šeši laukai: id, city_id, street_id, number, name, history. Realizuota mobili aplikacija, kurią sudaro klasės: Meniu, LocList, Famsit, InfoOnLoc, LocRoute ir RouteMap, Jsonrequest, DirectionsJSONParser. Mobilios aplikacijos klasės pasikeitė nedaug. Kai kurioms pasikeitė funkcijų ar jų parametrų vardai. Atsisakyta klasės LocList1, viską atliekant LocList klasėje. Taip atsisakyta papildomo vaizdo generavimo. Pridėta klasė Jsonrequest, kurioje atliekamos internetinės užklausos php failams, kurie išrenka informaciją iš duomenų bazės. Rasta internete ir pridėta George Mathew pavyzdyje naudotos klasė DirectionsJSONParser ir funkcija downloadUrl.

3.5. Patarimai, pastebėjimai, rekomendacijos

Tęsiant šį darbą, tobulinant šią mobilią aplikaciją, reikėtų labiau pasigilinti į Google žemėlapių galimybes, tokiu būdu būtų galima patobulinti atvaizduojamą maršrutą, bei pateikti realaus laiko keliavimo nuorodas. Tam atlikti reikėtų taip pat, labiau pasigilinti į Google kelio nurodymų paslaugos grąžinamus atsakymus, Google pateikia daug informacijos apie kelionę. Galima vartotojui atvaizduoti ne tik maršrutą, bet ir pateikti nuorodas, atstumą ar kelionės trukmę visam maršrutui ar visoms dalims atskirai. Turint daugiau laiko, daugiau dėmesio skirti informacijos atvaizdavimui vartotojui, kad būtų gražiau pateikiama informacija. Pasidomėti kitais vartotojo sąsajos komponentais, kurie padėtų tai atlikti.

Mobili aplikacija pritaikyta Lietuvoje esančioms vietoms, nebuvo išsiaiškinta ar ji veiktų užsienyje. Norint patobulinti aplikaciją, reikėtų išsiaiškinti koks adresas gaunamas iš Geocoder klasės, esant užsienyje. Svarbiausia vieta yra gautame adreso sąraše, addressline elemente esantis adresas. Lietuvoje jį sudaro gatvės pavadinimas (pvz.: Vilniaus gatvė) ir namo numeris. Jei

užsienyje jis nėra taip pat sudarytas, tuomet reikia taisyti aplikaciją, kad būtų pritaikyta tokiam adresui.

3.6. Testavimo rezultatai

Programa išbandyta šiuose įrenginiuose: LG E460, LG P700, Sony Xperia S.

LG E460

Stovint prie PLC „Saulės miestas“, aplikacija grąžino tos vietos adresą, informaciją apie Šiaulių miestą, apie Tilžės gatvę, apie PLC „Saulės miestas“. Stovint adresu Vilniaus gatvė 220, Šiauliai, programa grąžino tos vietos adresą, informaciją apie Šiaulių miestą, informaciją apie Vilniaus gatvę, o prie vietos informacijos pranešė, jog nėra informacijos. Stovint adresu Medelyno gatvė 40, programa grąžino tos vietos adresą, informaciją apie Šiaulių miestą, kitur parašė, jog nėra informacijos. Stovint Kužių miestelyje, aplikacija grąžino tos vietos adresą, kurį sudaro miestelio pavadinimas ir pašto kodas, kitur parašė, jog nėra informacijos. Ši dalis veikia gerai.

Sąraše pasirinkus Šiaulių miestą, atvaizduotas sąrašas vietų, esančių Šiaulių mieste. Pasirinkus Ch. Frenkelio vilą, parašyta kuriame mieste ji yra, gauta informacija apie pačią vilą. Pasirinkus kelias kitas vietas, gautas toks pat atsakymas. Ši dalis veikia gerai.

Pasirinkus maršruto sudarymą, stovint adresu Vilniaus gatvė 225, Šiauliai, gautas sąrašas vietų, esančių Šiaulių mieste. Pasirinkus Ch. Frenkelio vilą, Šiaulių katedrą ir Dviračių muziejų, ekrane atvaizduotas žemėlapis, kuriame pavaizduotas maršrutas. Ši dalis veikia gerai.

Išvada: aplikacija veikia šiame įrenginyje.

LG P700

Tokie patys veiksmai atlikti ir naudojant LG P700 įrenginį. Gauti tokie patys rezultatai.

Išvada: aplikacija veikia šiame įrenginyje.

Sony Xperia S

Tokie patys veiksmai atlikti ir naudojant Sony Xperia S įrenginį. Pirmą veiksmą atlikus gauti visi aukščiau paminėti rezultatai. Atliekant antrą veiksmą, miestų ir vietų, esančių tame mieste, sąrašai gauti, tačiau pasirinkus konkrečią vietą, negaunama informacijos. Atliekant trečią veiksmą, programa, radusi buvimo vietą, pateikia sąrašą vietų, esančių Šiaulių mieste, bet pasirinkus vietas, atvaizduojamas tik žemėlapis, o maršrutas neatvaizduojamas. Problema gali būti dėl parametrų perdavimo tarp klasių, kurios atsakingos už vartotojo sąsajos langų atvaizdavimą, nes abiem atvejais yra perduodami parametrai kitai klasei. O vietose, kur nėra perduodami parametrai, programa veikia gerai.

Išvada: aplikacija šiame įrenginyje veikia ne taip kaip turėtų.

IŠVADOS

1. Ištyrus vartotojų poreikius išsiaiškinta, jog vartotojai nori žinoti miesto aprašymą ar informaciją, lankytinos vietos aprašymą ar istoriją. Taip pat, įdomu ir gatvės istorija, jei tokia yra. Vartotojams aktualios netoli esančios lankytinos vietos, taip pat, svarbu žinoti maršrutą joms aplankyti.

2. Ištyrus aplikacijoms, kurios pateikia informaciją apie lankytinas vietas, būdingus reikalavimus, nustatyta, jog užduotims, susijusioms su internetinėmis užklausomis, naudojamos asinchroninių užduočių (angl. Async task) klasės. Darbui su žemėlapiais naudojami mapfragment komponentai, kurie atvaizduoja Google žemėlapį. Nustatyta, jog aplikacijai bus reikalinga interneto prieiga ir aktyvus GPS modulis.

3. Projektuojant mobilią aplikaciją išsiaiškinta, jog aplikacijos projektavimas vėliau palengvina aplikacijos realizavimą, nes dauguma aspektų jau būna apgalvota. Turint planą, greičiau vykdomas realizavimas, rečiau tenka perrašyti kodą, nes vykdymas išsiaiškinamas prieš kuriant, o ne kūrimo metu.

4. Realizuojant duomenų bazę, išsiaiškinta, kaip svarbu teisingai formuoti užklausas duomenų bazei, nes svarbus kiekvienas simbolis. Netinkamai suformuota SQL užklausa duomenų bazei (pavyzdžiui, neapgaubtas lentelės pavadinimas ar lauko pavadinimas simboliais „`“) negrąžina atsakymo.

5. Realizuojant mobilią aplikaciją išsiaiškinta, jog dauguma Android operacinei sistemai skirtų programų kūrimui naudojamų funkcijų yra realizuotos klasėse, kurias reikia prisijungti prieš panaudojant funkcijas. Tokiu būdu, neužkraunamos funkcijos, kurios nebus reikalingos programos vykdymui, sumažinamas aplikacijos užimamas dydis. Taip pat išsiaiškinta, jog Android aplikacijoms kurti Android programų kūrimo įrankių rinkinyje realizuota daugybė įvairių klasių, skirtų atlikti įvairiausioms užduotims.

6. Testavimo metu išsiaiškinta, jog programa gerai veikia LG išmaniuosiuose įrenginiuose. Sony Xperia įrenginiuose negaunama visa informacija, todėl galima sakyti, jog šiuose įrenginiuose aplikacija veikia netinkamai. To priežastis gali būtų šių įrenginių techninis sprendimas informacijai tarp vartotojo langų perduoti.

LITERATŪROS IR INFORMACINIŲ ŠALTINIŲ SĄRAŠAS

1. Atviro telefono sąjungos (angl. Open Handset Alliance) oficialus tinklalapis. [Žiūrėta 2014-05-02]. Prieiga per internetą: http://www.openhandsetalliance.com/android_overview.html

2. GARGENTA, Marko. *Learning Android*. Jungtinės Amerikos valstijos, 2011. ISBN 978-1-449-39050-1.

3. Android characteristikų paplitimas, Android oficialus tinklalapis, programų kūrėjų skiltis. [Žiūrėta 2014-05-02]. Prieiga per internetą: http://developer.android.com/about/dashboards/index.html?utm_source=ausdroid.net

4. API gidas, Android oficialus tinklalapis, programų kūrėjų skiltis. [Žiūrėta 2014-05-02]. Prieiga per internetą: http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels

5. City info aplikacijų rinkinio oficialus tinklalapis. [Žiūrėta 2014-05-02]. Prieiga per internetą: http://www.cityinfo.lt/

6. Alytaus krašto turizmui skirtas tinklalapis. [Žiūrėta 2014-05-01]. Prieiga per internetą: http://www.alytus-tourism.lt/lt/right_side-nekeisti/mobili-br-aplikacija-br-

7. Staff, Verge. *iOS: A visual history*. 2013. [Žiūrėta 2014-05-07]. Prieiga per internetą: http://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad

8. iOS technologija. Apple oficialus tinklalapis, programų kūrėjų skiltis. [Žiūrėta 2014-05-07]. Prieiga per internetą: https://developer.apple.com/technologies/ios/

SANTRAUKA

Autorius: Gediminas Dauskurdis
Tema: Mobili aplikacija netoliese esančių lankytinų vietų paieškai.
Šiaulių universitetas 2014.

Darbe ištirti vartotojų poreikiai tokiai programai, reikalavimai programai. Išnagrinėtos mobiliems įrenginiams skirtos operacinės sistemos Android ir iOS. Palyginti panašūs produktai. Remiantis surinkta informacija, suprojektuota ir sukurta trijų lentelių duomenų bazė informacijai apie lankytinas vietas saugoti. Suprojektuota ir sukurta mobili aplikacija, kurioje yra galimybė rasti netoliese esančias lankytinas vietas, gauti informaciją apie jas, sudėlioti maršrutą pasirinktoms vietoms aplankyti.

SUMMARY

Author: Gediminas Dauskurdis
Subject: Mobile App for searching for nearby places of interest
Šiauliai University 2014.

During this work, consumer needs and app requirements were examined. Operating systems Android and iOS were investigated. Similar products were compared. Focusing on collected data, the database for saving information about places to visit was designed and developed. The database consists of three tables. Also, an app which helps finding nearby places of interest, gives information about those places, makes route that would help visiting those places, was designed and developed.

PRIEDAI

1 Priedas. Kompaktinis diskas. Turinys:

1.1. Bakalauro darbo aprašymas.
1.2. Programos projektas, skirtas „Eclipse“ programavimo aplinkai su įdiegtu „Android SDK“ papildiniu.
1.3. Google play paslaugų bibliotekos projektas, skirtas „Eclipse“ programavimo aplinkai su įdiegtu „Android SDK“ papildiniu.
1.4. Trumpas aplikacijos aprašymas.
1.5. Duomenų bazės struktūros diagrama.
1.6. Aplikacijos klasių diagrama.
1.7. Aplikacijos pavyzdiniai paveikslėliai.
1.8. Aplikacijos apk failas.
1.9. Informacijos pateikimo pagal vietą funkcijos veikimo diagrama.
1.10. Vietų sąrašą pateikiančios funkcijos veikimo diagrama.
1.11. Maršruto sudarymo veikimo diagrama.
1.12. PHP failai, esantys prie duomenų bazės.

2 Priedas. Mobilios aplikacijos vartotojo sąsajos langų vaizdai