KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

Andrius Blažinskas

REST architektūros panaudojimo paskirstytos sistemos projektavime tyrimas

Magistro darbas

Darbo vadovas

prof. dr. Eduardas Bareiša

Kaunas, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

PROGRAMŲ INŽINERIJOS KATEDRA

TVIRTINU

Katedros vedėjas

prof. dr. E. Bareiša

2008-05-

Andrius Blažinskas

REST architektūros panaudojimo paskirstytos sistemos projektavime tyrimas

Magistro darbas

Recenzentas

doc. dr. A. Riškus

2008-05-

Vadovas

prof. dr. E. Bareiša

2008-05-

Atliko

IFM-2/2 gr. stud.

A. Blažinskas

2008-05-

Kaunas, 2008

Turinys

Paveikslai

Lentelės

REST architecture application analysis in distributed system design

Summary

One of the most known web service technologies today – SOAP Web services, is fairly complicated and inefficient. Other SOAP Web service alternatives become more and more popular. One of these alternatives is REST style architecture application in web service implementations. This type of web services is much simpler and efficient than SOAP Web services. Furthermore, it is based on well-established web concepts. This work describes analysis and implementation of REST style architecture in created multi-user position tracking system. Document provides detailed description of implemented position tracking system possibilities, structure and implementation peculiarity. Finally, experimental proof is given about SOAP and REST service implementations efficiency in this system.

Santrauka

Viena iš labiausiai išplitusių žiniatinklio paslaugų technologijų – SOAP (*Simple Object Access Protocol*) servisai, yra pakankamai sudėtingi ir neefektyvūs. Vis dažniau naudojamos įvairios šių servisų alternatyvos. Viena tokių alternatyvų yra REST (*Representational state transfer*) architektūros principų taikymas žiniatinklio paslaugų kūrimui. Šio tipo paslaugas yra paprasta realizuoti ir jos yra efektyvesnės nei SOAP variantas. Be to, jų kūrimo metodas remiasi gerai žinomomis ir senai žiniatinklyje nusistovėjusiomis koncepcijomis. Šiame magistriniame darbe atliktas REST architektūros principų tyrimas ir taikymas, darbo metu sukurtos, daugiavartotojiškos pozicionavimo sistemos kontekste. Aprašytos šios sistemos teikiamos galimybės, struktūra ir įgyvendinimo ypatumai. Tai pat, pateikti šios sistemos kontekste atlikto SOAP ir REST servisų efektyvumo eksperimento rezultatai.

1. Įvadas

Pasaulinis žiniatinklis sparčiai plečiasi. Jame prieinamos pačios įvairiausios mėgėjiškos ir verslo pobūdžio svetainės bei teikiamos žiniatinklio paslaugos. Dažniausiai pasitaikančios žiniatinklio paslaugos yra grįstos SOAP (*Simple Object Access Protocol*, http://en.wikipedia.org/wiki/SOAP) pagrindu, kurios dar vadinamos SOAP servisais. Per santykinai trumpą laiką, šio tipo paslaugos ir jų kūrimo technologija greitai išpopuliarėjo ir yra vystoma toliau. Tačiau dabar, SOAP paslaugoms pasiekus savo klestėjimo viršūne, pradedami kelti klausimai dėl šios technologijos efektyvumo [10]. Pradedama ieškoti alternatyvių sprendimų, kurie būtų paprastesni ir efektyvesni. Vienas tokių sprendimų yra REST (*Representational state transfer*) architektūros principų taikymas atitinkamo tipo žiniatinklio paslaugoms kurti. REST – yra tam tikra programinė architektūra, skirta paskirstytoms hiperžiniasklaidos sistemoms, tokioms kaip žiniatinklis. Terminas pirmą kartą paminėtas 2000 metais vieno iš pagrindinių HTTP (*Hypertext Transfer Protocol*) specifikacijos autorių, Roy Fielding disertacijoje [9]. REST idėją jis buvo pradėjęs vystyti dar 1994 metais. Tuo pat metu šis autorius stipriai prisidėjo prie HTTP 1.0 protokolo specifikavimo, o vėliau tapo vienas pagrindinių HTTP 1.1 specifikacijos autorių.

Svarbu pastebėti, jog visas žiniatinklis veikia REST architektūros pagrindu. Tačiau šiandieninės praktinės REST servisų realizacijos pilnai nepaiso visų šių apribojimų. Todėl, šiame darbe atlikę teorinių šaltinių ir egzistuojančių sprendimų analizę, apibendrinsime ir pateiksime esminius akcentus į kuriuos derėtų atsižvelgti kuriant tokio pobūdžio servisus.

Šiame magistriniame darbe bus orientuojamasi į REST architektūros principų pritaikymą konkrečioje, projekto metu realizuotoje, daugiavartotojiškoje pozicionavimo sistemoje. Šios sistemos pagrindinis tikslas yra suteikti vartotojams galimybę stebėti vieniems kitų poziciją realiu laiku, pasinaudojant mobiliuoju įrenginiu. Detaliau apie sistemos teikiamas paslaugas ir galimybes aprašyta projektinėje dalyje (skyrius **3. Daugiavartotojiška pozicionavimo sistema**).

Magistrinio darbo dokumente trumpai pristatysime REST architektūros bazines koncepcijas ir akcentuosime šios architektūros taikymo sritis realizuotoje sistemoje. Pateiksime detalų daugiavartotojiškos pozicionavimo sistemos projekto aprašą ir realizavimo ypatumus. Eksperimento dalyje, palyginsime šios sistemos kontekste realizuotų, SOAP ir REST pagrindu sudarytų paslaugų efektyvumus ir nurodysime optimalų, šiai sistemai labiausiai tinkantį, variantą.

2. REST analizė

2.1 REST architektūra

REST architektūroje yra išskirtos esminės esybės ir suformuluoti pagrindiniai principai, kuriais turėtų remtis šią architektūrą įgyvendinančios programinės realizacijos. Šiuos principus ir esybes paanalizuosime sekančiuose skyriuose.

2.1.1 REST resursai ir jų reprezentacijos

Resursai ir jų reprezentacijos yra labai svarbus aspektas REST kontekste. Pačiame REST pavadinime atsispindi, jog klientai komunikuodami su serveriais dalinasi resursų reprezentacijomis. Resursas – tai bet koks naudingas ir konkrečiu atveju aktualus elementas arba esybė (angl.: *item of interest*) žiniatinklyje [5]. Bet kokia informacija, kuriai gali būti suteiktas vardas, gali būti resursas [9]. Resursų pavyzdžiai: bet koks dokumentas, paveikslėlis, servisas, kitų resursų kolekcija ir pan. Visas žiniatinklis susideda iš tokių resursų.

Resurso reprezentacija – resurso esamos arba pageidaujamos būsenos vaizdas. Tai yra baitų seka, kuria paprastai lydį aprašomieji metaduomenys [9]. Resursų ir jų reprezentacijos paaiškinimas pavyzdžiu galėtų būti toks [17]: naršyklė siunčia užklausą į tam tikrą HTTP serverį HTTP GET metodu, kad užmegzti ryšį su tam tikru resursu ir kaip atsaką gauna resurso būsenos HTML reprezentaciją. Bendruoju atveju reprezentacija nebūtinai turi būti HTML formatu, galimi ir kiti variantai kaip kad XML (*Extensible Markup Language*) ar paprasčiausias tekstas.

REST servisai yra grįsti resursais, kai tuo tarpu SOAP servisai – veiklomis [14].

2.1.2 Unikalūs resursų identifikatoriai

Vienas svarbiausių REST architektūros reikalavimų yra tai, kad kiekvienam resursui turi būti suteikiami unikalūs identifikatoriai. REST yra orientuotas į HTTP protokolą, o resursų identifikatoriai yra URI nuorodos [4][17]. Tokius identifikatorius patogu naudoti, nes jie yra aiškūs ir labai plačiai išplitę (pavyzdys yra visas pasaulinis nuorodomis grįstas žiniatinklis). Duomenų nuskaitymo atveju, šiuos identifikatorius lydinčius parametrus rekomenduotina realizuoti panaudojant HTTP GET metodą, t.y. visos užklausos neturėtų būti slepiamos HTTP POST duomenų perdavimo metodika. Toks atviras identifikatorių pateikimas suteikia galimybę lengvai dalintis konkretaus resurso nuoroda, naršyklėje jis gali būti įtraukiamas į mėgstamų nuorodų sąrašus ir pan.

Identifikuojant resursą, svarbu yra naudoti daiktavardžius, o ne veiksmažodžius [5].

Korektiškų resursų identifikatorių pavyzdžiai:

- `http://pavyzdys.lt/klientai/1234`
- `http://pavyzdys.lt/uzsakymai/2007/10/776654`
- `http://pavyzdys.lt/produktai/4554`

Taip pat, kartu su resurso vardu, teisinga yra naudoti HTTP GET parametrus:

- `http://pavyzdys.lt/uzsakymai?metai=2007&menuo=11`
- `http://pavyzdys.lt/produktai?spalva=juoda`

Duomenų nuskaitymo atveju, nekorektiška būtų šiuos parametrus pateikti HTTP POST metodu, arba vietoje pavyzdžiui daiktavardžio *produktai* įvesti metodo pavadinimą *gautiProduktus*.

2.1.3 Nuorodomis apjungtos resursų reprezentacijos

Resursai savo reprezentacijose taip pat gali pateikti kitų, kaip nors susijusių, resursų identifikatorius. Savybę iliustruosime XML tipo REST serviso pavyzdžiu, traktuodami, jog komunikacijoje dalyvauja tik programinė įranga, t.y. vartotojas tiesiogiai nesinaudoja paslauga per naršyklę.

Tarkime, jog tam tikras REST servisas, teikiantis informaciją apie automobilio dalis, pateikia automobilio dalių sąrašą – resursą, kuris pasiekiamas tokia nuoroda:

- `http://www.daliu-saugykla.lt/dalys`

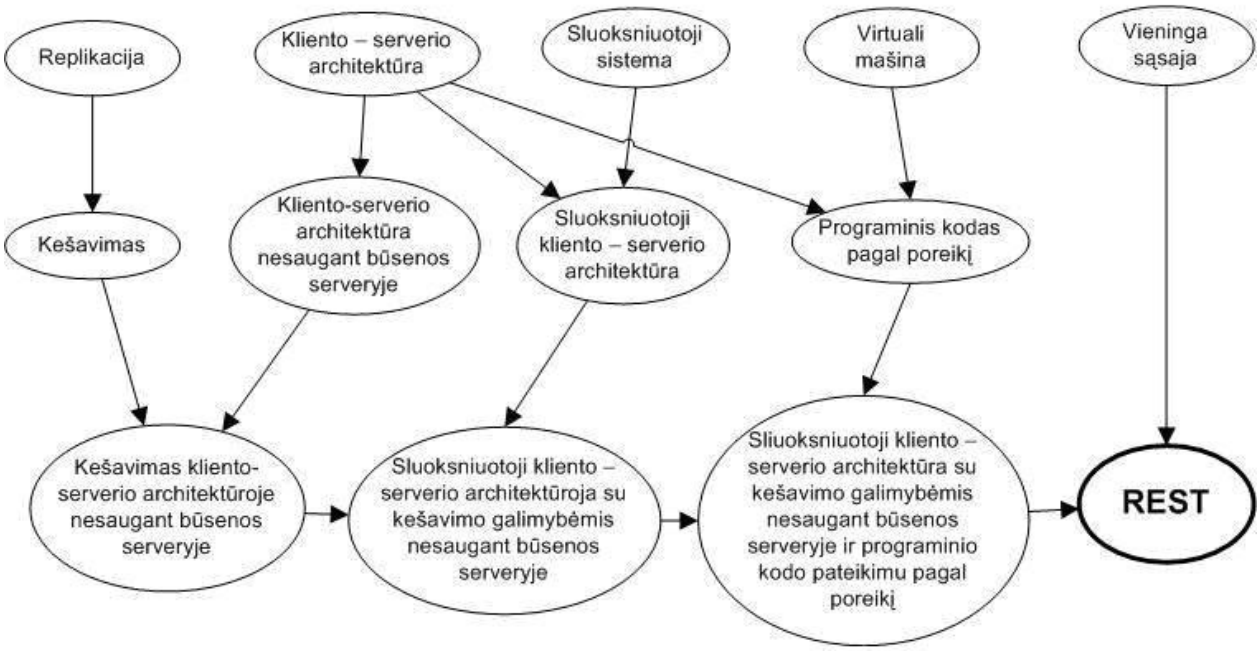
Galimas užklausos pagal šią nuorodą atsakas XML formatu:

```
<?xml version="1.0"?>
<p:Dalys xmlns:p="http://www.daliu-saugykla.lt"
         xmlns:xlink="http://www.w3.org/1999/xlink">
      <Dalis id="00345"
             xlink:href="http://www.daliu-saugykla.lt/dalys/00345"/>
      <Dalis id="00346"
             xlink:href="http://www.daliu-saugykla.lt/dalys/00346"/>
      <Dalis id="00347"
             xlink:href="http://www.daliu-saugykla.lt/dalys/00347"/>
</p:Parts>
```

Vartotojas, naudodamasis klientu, gali eiti gilyn ir peržiūrėti informaciją apie konkrečią jį dominančią dalį. Ši savo ruoštu, gali pateikti dar daugiau nuorodų į kitus resursus, pavyzdžiui gamintojo pateikiamą konkrečios dalies specifikaciją ir pan.

2.2 REST išvedimas

Yra apibrėžta aibė architektūrinių apribojimų, kurių pagrindu REST architektūra yra sudaryta. Pagal REST autorių Roy Fielding, REST architektūra yra išvedama remiantis **1 pav.** pateikta schema.



1 pav. REST išvedimas pagal Roy Fielding

Toliau, aiškindami REST, aptarsime kiekvieną **1 pav.** pateikiamą elementą atskirai.

Replikacija. Replikacijos savybę tenkinantys resursai tinkle turi daugiau negu vieną savo paties kopiją. Šios kopijos dažniausiai būna susietos tam tikra infrastruktūra, kuri leidžia imituoti lyg egzistuotų tik vienas resurso egzempliorius. Tokiu būdu, gali būti išlygiagretinta prieiga prie resurso ir tuo pačiu paskirstytos apkrovos tarp šią prieigą teikiančių serverių [9].

Kešavimas. REST atveju, kešavimas gali būti vykdomas kiekvienam HTTP užklausų atsakymui. Nevisais atvejais kešavimas yra pageidautinas, todėl sugeneruotas atsakas, kokiu nors būdu, turėtų būti pažymėtas kaip galimas arba negalimas kešuoti. Tais atvejais kai tai neįneša jokio šalutinio poveikio, atsakas turėtų būti pažymėtas kaip kešuojamas ir klientui turėtų būti suteikta galimybė pakartotinai šį atsaką panaudoti.

Kešavimo procedūra gali būti dviejų tipų: pagal užklaustus duomenis ir nuspėjančioji. Užklaustų duomenų kešavimas vykdomas iškarto po kreipties, kešuojant šios kreipties rezultatą.

Nuspėjančioji kešavimo procedūra vykdoma pagal esamą duomenų užklausą mėginant nuspėti sekančią duomenų porciją [9]. Kešavimo pagrindinis tikslas padidinti našumą ir kuo greičiau pateikti kliento užklausos rezultatą. Kad kešavimą įgyvendinančios sistemos veiktų efektyviai, reikia, kad resursų identifikatoriai ir nuorodos į jų reprezentacijas nekistų.

HTTP GET parametrų panaudojimas nuorodoje yra korektiškas, tačiau jų vieta nuorodoje gali kisti. Pavyzdžiui, abi žemiau pateikiamos nuorodos HTTP protokolo atžvilgiu yra identiškos:

- `http://pavyzdys.lt/uzsakymai?metai=2007&menuo=11`
- `http://pavyzdys.lt/uzsakymai?menuo=11&metai=2007`

Tačiau kešavimo procedūras vykdančių sistemų atžvilgiu, šios dvi nuorodos būtų traktuojamos kaip skirtingos. Taigi, kad efektyviai pasinaudoti teikiamomis kešavimo galimybėmis, rekomenduotina naudoti alternatyvų nuorodų formatą, išvengiant parametrų panaudojimo:

- `http://pavyzdys.lt/uzsakymai/2007/11`

Žinoma, lygiai tokio paties rezultato galima pasiekti ir naudojant parametrus, tačiau tokiu atveju reikėtų užtikrinti, jog kreipiniuose parametrai visada būtų pateikiami ta pačia eilės tvarka.

Kliento – serverio architektūra. Tai bene dažniausiai pasitaikantis tinkle veikiančių programinių sistemų architektūros tipas, todėl akcentuosime tik REST atžvilgiu svarbias savybes.

Kliento ir serverio sąveikos formoje duomenys iš serverio yra paimami tuomet, kai jie yra reikalingi, t.y. pats klientas juos pasiima (angl.: *pull-based*). Duomenų paėmimą gali vykdyti tiesiogiai vartotojas naudodamasis naršykle arba programinis komponentas – klientas.

Bazinėje kliento – serverio architektūroje nėra griežtai nurodyta kaip turėtų būti saugoma komunikavimo būsena tarp kliento ir serverio, todėl tai suteikia galimybę įgyvendinti sekantį esminį apribojimą, kuris teigia, jog ši būsena neturi būti saugoma serveryje.

Kliento – serverio architektūra nesaugant būsenos serveryje. Šis apribojimas suformuojamas į kliento – serverio architektūrą įvedant papildomą reikalavimą, jog su klientu susijusi sesijos būsena negali būti saugoma serverio pusėje. Ši būsena turi būti saugoma tik kliento dalyje, o kiekviena nauja užklausa į serverį turi saugoti visą reikalingą informaciją, kad būtų įmanoma ją tinkamai interpretuoti.

Įgyvendinto reikalavimo teikiami privalumai [9]:

- Pagerintas užklausos duomenų matomumas – kiekviena užklausa pateikia visą reikalingą informaciją.
- Padidintas patikimumas – įvykus klaidai sistemai paprasta atsistatyti, nes klaidą sukėlė tik viena, paprastai nesudėtinga, užklausa.
- Optimizuojamas serverių darbas sumažinant sunaudojamos atminties kiekį – pateikus užklausos rezultatą, serveris grįžta į savo pradinę būseną ir atlaisvina užimtą atmintį.

Kešavimas kliento – serverio architektūroje nesaugant būsenos serveryje. Apribojimas suformuojamas įkomponuojant kešavimą realizuojančius komponentus į kliento – serverio architektūrą, kurioje komunikavimo būsena saugoma tik kliento dalyje. Kešavimas įgyvendinamas specialiuose komponentuose kurie dalyvauja komunikavimo tarp kliento ir serverio grandinėje. Tai gali būti tokie tarpininkai kaip specializuoti kešavimo serveriai, arba paprasčiausiu atveju, programiniai komponentai, susieti su serveriu arba klientu.

Sluoksniuotosios sistemos ir sluoksniuotoji kliento – serverio architektūra. Sluoksniuotosios sistemos yra vienas dažniausiai pasitaikančiu sistemos tipų, kuomet sluoksniai organizuojami hierarchiškai. Kiekvienas sluoksnis teikia paslaugas aukščiau esančiam sluoksniui ir naudojasi žemiau esančio sluoksnio teikiamomis paslaugomis. Šios sluoksniuotosios sistemos veikiančios tinkle, praktiškai yra neatsiejamos nuo kliento – serverio architektūros, todėl joms apibūdinti įvestas tarpinis sluoksniuotos kliento – serverio architektūros pavadinimas. Šio tipo sistemos gali būti dvisluoksnės, trisluoksnės arba sudėtingesniais atvejais vadinamos tiesiog daugiasluoksnėmis [9].

Sluoksniuotoji kliento – serverio architektūra su kešavimo galimybėmis nesaugant būsenos serveryje. Apribojimas suformuojamas apjungiant sluoksniuotos kliento – serverio architektūros apribojimus su būsenos nesaugančios ir kešavimo savybę išnaudojančios tos pačios architektūros apribojimais. Kliento – serverio architektūra apjungime nedubliuojama [9].

Virtuali mašina. Virtuali mašina naudojama programiniam kodui vykdyti. Tai gali būti įvairūs konteineriai, kaip kad gerai žinoma Tomcat sistema, paremta *Servlet*

(http://java.sun.com/products/servlet/) technologijos pagrindu, arba interpretatoriai: Python, Perl, PHP ir pan. Panaudojant virtualias mašinas tinklo programų vykdymui, šios tampa kliento – serverio architektūros dalimi [9].

Programinis kodas pagal poreikį. Serveris klientui suteikia prieigą prie tam tikro resursų rinkinio, tačiau klientas ne visuomet gali žinoti kaip šiais resursais naudotis. Tokiu atveju serveris papildomai suteikia prieiga prie programinio kodo, kuris realizuoja tokį apdorojimą. Klientas parsisiuntęs šį programinį kodą vykdo jį lokaliai. Tokio principo pritaikymo praktikoje pavyzdys yra *JavaScript* (http://en.wikipedia.org/wiki/JavaScript) technologija. Pagrindinis teikiamas privalumas yra tam tikros programos dalies vykdymo perleidimas kliento pusei, tačiau tokiu atveju sistema tampa sudėtingesnė [9].

Sluoksniuotoji kliento – serverio architektūra su kešavimo galimybėmis nesaugant būsenos serveryje ir programinio kodo pateikimu pagal poreikį. Tai priešpaskutintas suformuotas kompleksinis apribojimas prieš įvedant REST sąvoką. Kadangi bendruoju atveju kodas pateikiamas pagal poreikį yra duomenų resursas, jis tinkamai įsikomponuoja į apibrėžtos sluoksniuotosios kliento – serverio architektūros su kešavimo elementais nesaugant būsenos serveryje sąvokos kontekstą [9].

Vieninga sąsaja. Vienas esminių REST architektūros reikalavimų yra vieninga sąsaja (angl.: *uniform interface*). Jis teigia, jog visi žiniatinklio resursai turi turėti vieningą prieigos sąsają.

Įprastai įvairių servisų ar kitokių programinių sistemų komunikavimo sąsaja yra skirtinga. REST koncepcijoje sąsaja yra vieninga. Tokia savybė neretai įneša komplikacijų sąsajos sąryšyje su vidinės logikos realizacija, tačiau REST atveju tai nesukelia ypatingų sunkumų.

Dažniausiai servisai turi sąsajas ne vien tik operacijoms iškviesti, bet ir apibrėžti duomenų struktūras, kurios naudojamos perduodant duomenis tarp kliento ir serverio. SOAP servisų atveju, WSDL (*Web Services Description Language*) kalba apibrėžia metodus ir susietas duomenų struktūras. Taigi, norint pasinaudoti SOAP servisu, reikia žinoti tiek serviso metodų iškvietimo tiek ir duomenų struktūrų specifiką. Be to, tokio tipo sąsaja yra dalinai specializuota ir įgyja priklausomybę nuo vidinės logikos realizacijos. Dėl tokios priklausomybės, keičiantis vidinei logikai, neretai tenka keisti sąsają. Tai labai komplikuoja padėtį jau esamiems vartotojams, nes tolesniam serviso panaudojimui reikia atitinkamai atnaujinti klientinę

programinę įrangą. Šiuo aspektu REST įgyja pranašumą prieš SOAP servisus: REST klientui pakanka žinoti tik specifines naudojamas duomenų struktūras, nes iškvietimo būdas yra vieningas ir nekintantis [17][9].

2.3 REST ir SOAP servisų palyginimas

Šiuo metu labiausiai išplitę žiniatinklio servisai – SOAP servisai. Šie servisai buvo pradėti vysti nuo 1998-ųjų, o pagrindiniai iniciatoriai buvo Microsoft ir IBM kompanijos, būtent todėl jų sėkmė buvo užtikrinta [6]. Bendruoju atveju, SOAP yra XML pagrindu sudarytas protokolas. Pirmoji SOAP versija buvo išimtinai paremta XML-RPC (http://www.xmlrpc.com/spec) technologija, kuri išliko kaip pagrindinė iki šių dienų. XML-RPC technologija yra specifikacija, kuri suteikia galimybę programinės įrangos sistemoms, veikiančioms skirtingose aplinkose, vykdyti nuotolines užklausas. Šios užklausos persiunčiamos kaip XML pranešimai, panaudojant HTTP protokolą transportavimui [8].

XML-RPC technologija yra taikoma ir kaip atskiras komunikavimo standartas. Jo panaudojimas servisams realizuoti yra efektyvesnis sprendimas nei SOAP, nes pastarasis yra sudėtingesnis ir persiunčia kur kas daugiau metaduomenų. Tačiau, nei viena iš šių technologijų efektyvumu neprilygsta REST.

Toliau trumpai palyginsime REST ir labiausiai išplitusios SOAP technologijos servisų savybes, akcentuodami naudojamų technologijų privalumus ir trūkumus.

1 lentelė. REST ir SOAP servisų palyginimas

Kriterijus	REST servisas	SOAP servisas
Nėra persiunčiama daug perteklinių duomenų	+	-
Paprasta realizacija	+	-
Nepriklausomybė nuo protokolo	-	+
Duomenų tipų tikrinimas	-	+
Yra serviso aprašymo kalba	+	+
Egzistuoja pagalbinės kūrimo priemonės	-	+

Kriterijus	REST servisas	SOAP servisas
Vieninga prieigos sąsaja	+	-
Kešavimo galimybės	+	-
Žmogui suprantamos užklausos ir jų rezultatai	+	-

Perteklinių duomenų persiuntimas. REST serviso atveju, nėra persiunčiama daug perteklinių duomenų. Vieninteliai pridėtiniai persiunčiami duomenys yra tik HTTP protokolo pranešimų tekstai, kurie yra mažos apimties. SOAP serviso atveju, kiekvienos užklausos ir atsakymo metu yra persiunčiami SOAP vokai, kurie tam tikrais atvejais užima daugiau vietos nei reikalingu persiųsti duomenų kiekis.

Paprasta realizacija, pagalbinės kūrimo priemonės. REST servisus yra paprasta realizuoti, be to, tam dažniausiai nėra reikalingos jokios papildomos bibliotekos ar kita programinė įranga. Tuo tarpu SOAP servisu kūrimas, be tokios papildomos programinės įrangos ir specialių SOAP servisus realizuojančių bibliotekų, yra komplikuotas.

Nepriklausomybė nuo protokolo. SOAP servisai yra nepriklausomi nuo protokolo [15]. Pranešimų persiuntimui gali būti naudojama daug kitų protokolų, kaip kad JMS (*Java Message Service*), SMTP *(Simple Mail Transfer Protocol*) ir pan. REST turi priklausomybę nuo HTTP protokolo. Iš tiesų, HTTP čia panaudojamas ne tik transportavimo tikslais, bet yra išnaudojamos kitos HTTP protokolo savybės, kaip kad GET, POST, PUT, DELETE užklausos [2]. Ši SOAP nepriklausomybė nuo protokolo yra traktuojama kaip trūkumas [15], nes atsiranda galimybė panaudoti protokolus kurie turi visiškai skirtingą paskirtį, dėl to gali stipriai nukentėti efektyvumas.

Duomenų tipų tikrinimas. SOAP protokolas sudaro galimybę atlikti tipų tikrinimą. REST servisuose tas nėra akcentuota, tačiau tai visada galima realizuoti *ad-hoc* principu.

Serviso aprašymo kalba. Tiek SOAP tiek REST servisai turi juos aprašančias kalbas. SOAP atveju įprastai yra naudojama WSDL (http://www.w3.org/TR/wsdl). REST servisu apibūdinimui kartais yra naudojama WADL (*Web Application Description Language*), kuri yra

gerokai paprastesnė už WSDL. Tai pat, visada gali būti panaudojamos kitos bendros paskirties kalbos, kaip kad XML schemos (http://www.w3.org/XML/Schema) [15].

Vieninga prieigos sąsaja. Vienas esminių REST servisų privalumų yra vieningą prieigos sąsaja (detaliau apie tai skyriuje **2.2 REST išvedimas**). SOAP servisų atveju, kiekvienos konkrečios realizacijos sąsaja gali skirtis, o nepriklausomumo elementai gali būti realizuojami tik panaudojant pagalbines priemones.

Kešavimo galimybės, žmogui suprantamos užklausos ir jų pateikiami rezultatai. Laikantis REST apribojimų ir prieigą prie resursų realizuojant HTTP GET metodu, šių servisų kūrime labai paprasta pasinaudoti kešavimogalimybėmis. Be to, visi persiunčiami duomenys yra vizualiai pateikiami nuorodoje ir suprantami jų vartotojams. Tuo tarpu SOAP servisu atveju, duomenys tipiškai yra persiunčiami HTTP POST metodu, todėl yra paslėpti ir dažnai tenka realizuoti specifinius komponentus, kurie galėtų pilnai išnaudoti kešavimo galimybes.

2.4 Egzistuojantys REST servisų sprendimai

REST architektūros principai yra įgyvendinti pasauliniame žiniatinklyje ir pakankamai senai ir gerai yra žinomi. Šios architektūros elementų funkcionalumą įgyvendinančios technologijos yra nusistovėjusios ir gerai išdirbtos, dėl to yra paprastos ir efektyvios. Servisai, sudaryti remiantis REST principais, paveldi šią efektyvumo ir paprastumo savybę.

REST serviso terminas dažnai yra naudojamas laisvesne prasme, kalbant apie paprastą sąsają, kuri skirta perduoti specifinius srities duomenis tiesiogiai panaudojant HTTP protokolą ir nenaudojant jokio papildomo pranešimų sluoksnio kaip SOAP. Sparčiai plintančius REST pagrindu sudarytus servisus galima laikyti tam tikra SOAP servisų alternatyva. Pastarieji laikui bėgant tapo sudėtingi ir neefektyvūs, o paprastumu ir efektyvumu pasižymintys REST servisai vis labiau plinta [10].

2.4.1 NewsGator REST sąsaja

NewsGator kompanija teikia informacijos saugojimo ir valdymo priemones. Ji siūlo įvairius RSS (*Really Simple Syndication*) pagrindu sudarytus produktus ir programinius žiniasklaidos įrankius. NewsGator internetinė sistema teikia jos vartotojams prieigą prie įvairių straipsnių, pranešimų ir kitokios žiniasklaidoje publikuojamos informacijos. Šios sistemos REST

sąsaja suteikia programų kūrėjams galimybę kurti nuosavas sistemas, galinčias naudotis šios sistemos duomenimis.

Pagrindiniai REST resursai prie kurių prieigą ir jų valdymą sistema užtikrina:

- Vietovės (locations) – vartotojas gali turėti keletą klientinių programų skirtingose vietovėse (namie, darbe ir pan.), todėl yra sudaroma galimybė sistemiškai atskirti šias vietoves, kurios gali talpinti skirtingas prenumeratas ir kitą informaciją.
- Katalogai (folders) – sistema vartotojams leidžia kurti nuosavus katalogus ir atlikti papildomus grupavimus.
- Prenumeratos (subscriptions) – yra esminiai sistemos duomenys. Vartotojas gali užsisakyti straipsnių ar kitokių periodinių leidinių prenumeratas.
- Informacijos rinkimo taškai (feeds) – tai užsiprenumeruotų straipsnių ar kitokios informacijos metaduomenų RSS 2.0 nurinkimo taškai.
- Straipsniai/pranešimai (arcticles/posts) – tai konkrečiam informacijos rinkimo taškui priskirti straipsniai ar kitokie pranešimai.

Pateikiame keletą REST serviso panaudos pavyzdžių:

- http://services.newsgator.com/ngws/svc/Location.aspx/<locationId>/unreadCount – HTTP GET metodo atveju, grąžina vartotojo sukurtos vietovės (identifikatorius *locationId*) susietų neperskaitytų straipsnių ar pranešimų kiekį.
- http://services.newsgator.com/ngws/svc/Location.aspx/<locationId> – HTTP GET metodo atveju, pateikia konkrečios vartotojo vietovės parametrus. POST metodo atveju šiuos parametrus atnaujina pagal pateiktus duomenis.
- http://services.newsgator.com/ngws/svc/Subscription.aspx/<locationname>/headlines – naudoja HTTP GET metodą. Pateikia vartotojo prenumeratų, priskirtų konkrečiai vartotojo vietovei, pirmąsias turinio eilutes.

Pateikiama internetinės sistemos REST sąsaja yra alternatyva egzistuojančiai SOAP sąsajai ir ją rekomenduotina naudoti tuomet kai SOAP variantas dėl kažkokių priežasčių neprieinamas arba yra per sudėtinga jį panaudoti.

Šioje REST sąsajoje yra išnaudojami HTTP GET, POST, PUT, DELETE metodai, tačiau jei PUT ir DELETE negali būti panaudoti, yra jų GET ir POST atitikmenys.

REST servisas naudoja bazinę (basic) arba santrumpomis (digest) grįsta HTTP autentifikacija. Bazinės autentifikacijos atveju turėtų būti naudojamas HTTPS (HTTP over Secure Socket Layer). Su autentifikacija susijusi informacija yra pridedama panaudojant HTTP antraštės.

Užklausos metu iškilus klaidai, klientui yra grąžinamas klaidos pranešimas panaudojant atitinkamus HTTP protokolo kodus:

2 lentelė. NewsGator REST sąsajos generuojami HTTP klaidų pranešimai

HTTP atsakymas	Paaiškinimas
200 - OK	Užklausa sėkmingai įvykdyta.
400 – Bad Request	Neteisingai suformuoti duomenys.
401 – Access Denied	Autentifikacija nepavyko.
404 – Not Found	Resursas nerastas.
405 – Method Not Allowed	Nurodytas HTTP metodas (GET, POST, PUT, DELETE) neįgyvendintas.

Duomenų persiuntimui yra naudojami NewsGator kompanijos praplėsti RSS 2.0 ir OPML (*Outline Processor Markup Language*) formatai.

Viena iš svarbiausių specifinių sistemos savybių yra sinchronizacijos simbolių eilutės, kurios naudojamos nustatyti pasikeitimams nuo paskutintos vartotojo pateiktos užklausos. Tokiu būdu yra sumažinamas persiunčiamų duomenų kiekis ir yra persiuntinėjami tik pasikeitę duomenys. Šis pasikeitimų saugojimas vyksta ne sesijos duomenyse, o tiesiogiai su vartotoju susijusiuose duomenyse. Tai serveryje nesukuria atskiros būsenos, todėl tai nepažeidžia serveryje vartotojo būsenos nesaugojimo reikalavimo.

Kaip matyti, ši sistema yra korektiškai realizuota REST reikalavimų požiūriu ir neturi jokių esminių trūkumų.

Informacija apie REST sąsaja imta iš: http://www.newsgator.com/ngs/api/NewsGatorRESTAPI.pdf

2.4.2 Fedora eksperimentinė REST sąsaja

Fedora yra skaitmeninių objektų saugykla. Jos modelis leidžia skirtingai atvaizduoti joje saugomus skaitmeninius objektus, o taip pat teikia plačias sąryšių tarp šių objektų realizavimo galimybes. Dinaminis objektų atvaizdavimas yra realizuojamas panaudojant išorinius žiniatinklio servisus. Be to, visos pagrindinės vidinės funkcijos joje yra realizuotos tokių servisu pagrindu. Fedora API-A ir API-M (API – *application programming interface*, A – *access*, M –

management) yra dvi SOAP paslaugomis grįstos sąsajos, teikiančios saugykloje saugomų skaitmeninių objektų prieigos ir valdymo galimybes. Pradedant nuo 3.0 versijos, kuri šiuo metu vis dar yra beta, minėtų sąsajų funkcionalumas taip pat numatytas pradėti teikti REST servisų pagrindu. Kadangi REST sąsaja yra eksperimentinė, todėl tik daugiau kaip pusė viso funkcionalumo kol kas yra prieinama.

Toliau pateikiame tik realizuotos REST sąsajos funkcionalumo aprašymą.

3 lentelė. Fedora REST sąsaja

SOAP metodas	HTTP Metodas	REST ekvivalentas
API-M sąsaja		
getObjectProfile	GET	/objects/demo:29
listMethods	GET	/objects/demo:29/methods
listDatastreams	GET	/objects/demo:29/datastreams
getDatastreamDissemination	GET	/objects/demo:29/datastreams/DC
findObjects	GET	/objects?pid=true&terms=demo:29
resumeFindObjects	GET	/objects?sessionToken=xyz
getObjectHistory	GET	/objects/demo:29/versions
API-A sąsaja		
Ingest	POST	/objects/new
modifyObject	PUT	/objects/demo:29?label=foo
getObjectXML	GET	/objects/demo:29/objectXML
exportObject	GET	/objects/demo:29/export
purgeObject	DELETE	/objects/demo:29
addDatastream	POST	/objects/demo:29/datastreams/DS99?controlGroup=X&dsLabel=foo
modifyDatastreamByReference	PUT	/objects/demo:29/datastreams/DS99
modifyDatastreamByValue	PUT	/objects/demo:29/datastreams/DS99
purgeDatastream	DELETE	/objects/demo:29/datastreams/RELS-EXT
getNextPID	GET	/objects/nextPID

Lentelės duomenys paimti iš http://www.fedora-commons.org/documentation/3.0b1/userdocs/server/webservices/rest/index.html

Kaip matyti iš lentelės, ši sąsaja išnaudoja HTTP protokolo galimybes: GET, POST, PUT ir DELETE metodus CRUD (*create, read, update, delete*) operacijoms realizuoti. Tai pat

pastebėsime, jog visos nuorodos suformuotos korektiškai, t.y. visos jos identifikuoja tam tikrą resursą. Pavyzdžiui nuoroda `/objects/demo:29/datastreams/DS99` nurodo, jog yra kreipiamasi į saugykloje saugomo objekto (*objects*) `demo:29` priskirtą duomenų srautą (*datastreams*) `DS99`. Tačiau yra aiškiai matomas vieno REST architektūros reikalavimo netenkinimas. Nuorodoje `/objects?sessionToken=xyz` yra nurodomas sesijos parametras, skirtas paimti sekančiai, prieš tai vykdytos paieškos metu rastų, rezultatų porcijai. Tai aiškiai nurodo apie paieškos rezultatų, susijusiu su konkrečiu vartotoju, saugojimu serverio pusėje. Taigi pažeidžiamas būsenos nesaugojimo serverio pusėje apribojimas.

Šią REST serviso sąsają numatyta tobulinti toliau ir planuojama, jog ji turėtų pakeisti paprastąsias (API-A-Lite, API-M-Lite) SOAP sąsajas.

2.4.3 Kitos realizacijos

Labai trumpai peržvelgsime keletą kitų sistemų, turinčių įgyvendintą REST sąsają.

FishEye yra SVN, CVS ir Perforce versijų valdymo sistemų, versijų peržiūros ir analizės sistema. Šios sistemos REST sąsaja (aprašymas pateikiamas adresu http://fisheye.cenqua.com/api/) yra sudaryta kombinuojant resursu savoka su metodais. Pavyzdžiui yra naudojami tokie identifikatoriai (pradžia nepateikiama): `api/rest/login`, `api/rest/logout`, `api/rest/repositories`, `api/rest/revision`. Kaip iš šių identifikatorių matyti yra realizuotas *ad-hoc* autentifikacijos mechanizmas. Šis mechanizmas prisijungus prie sistemos sugeneruoja simbolių eilutę, kurią reikia pridėti prie HTTP parametrų kiekvieną kartą siunčiant užklausą, kad serveris tinkamai identifikuotų vartotoją. Šioje sistemoje visos nuorodos naudoja HTTP GET metodą, tačiau taip ir turi būti, nes nėra nei vienos procedūros kuri leistų keisti kokius nors vidinius sistemos duomenis.

Kaip matyti iš sistemos REST sasajos aprašymo, ši netenkina REST rekomendacijų, nes naudoja prisijungimo ir atsijungimo metodų vardus kaip resursus, be to, serverio pusėje saugo su vartotoju susijusią būsena, identifikavimui realizuoti.

Internetinė sistema Flickr suteikia vartotojams galimybę talpinti ir saugoti nuotraukas ir paveikslus internetinėje paveikslų bazėje. Ji teikia REST sąsają (http://www.flickr.com/services/api/request.rest.html) šios paveikslų sistemos duomenų manipuliavimui. Sąveikavimas su šia sąsaja vyksta panaudojant kreipinius į metodus, pavyzdžiui:

- http://api.flickr.com/services/rest/?method=flickr.test.echo&name=value

Sąsaja yra pakankamai turtinga įvairais metodais, tačiau šis REST servisas netenkina REST rekomendacijos: kreipinius derėtų formuoti į resursus, o ne į metodus.

2.5 Apibendrinimas

Roy Fielding apibrėžta REST architektūra (**1 pav.**) iš tiesų suteikia didelę jos įgyvendinimo laisvę REST servisus realizuojantiems kūrėjams. Ne visi pateikti apribojimai būtinai turi būti įgyvendinami realizuojant konkretų REST servisą. Pavyzdžiui, programinis kodas pagal poreikį gali būti visiškai nereikalingas, o sistema gali turėti tik vieną sluoksnį, t.y. būti nesluoksniuota. Taip pat yra ir su kešavimu, kuris ne visada yra reikalingas ir naudingas. Svarbu tai, kad kiek įmanoma šie apribojimai nebūtų pažeidinėjami. To pasekoje, daugelis internete pateikiamų, REST pagrindu realizuotų sistemų, pilnai neįgyvendina visų REST charakteristikų, kas ir matyti iš atliktos egzistuojančių sprendimų analizės. Tai pat galima pastebėti, jog kai kurios tirtos realizacijos nesilaiko net tam tikrų bazinių reikalavimų, kaip kad būsenos nesaugojimas serveryje.

Trumpai apibendrinant teorines rekomendacijas ir ištirtas realizacijas, galima akcentuoti kelis labiausiai išsiskiriančius REST servisams būdingus bruožus, kurių turėtų būti paisoma realizuojant REST servisus:

- Pagrįsti resursais, o ne veiklomis, kaip kad SOAP atvejis.
- Sesijos būsenos nesaugojimas serveryje.
- HTTP protokolo GET, POST (kartais PUT ir DELETE) metodų išnaudojimas CRUD operacijoms atlikti.
- Įprastai grąžinamas tik *ad-hoc* XML formato rezultatas, nenaudojant jokio sudėtingo antstato. Jei konkrečiu atveju yra prasminga, šioje XML struktūroje turėtų būti pateikiamos nuorodos į susietus resursus.
- Taip pat kartais realizuojama papildoma savybė – galimybė klientui pasirinkti grąžinamą rezultatų formatą.

3. Daugiavartotojiška pozicionavimo sistema

Pozicijos nustatymo problema jau senai žmonėms yra labai aktuali. Ji pakankamai efektyviai išspręsta tik XX a. pabaigoje. Tą sėkmingiausiai padarė JAV Gynybos Departamentas, sukūręs palydovinę radijo navigacinę sistemą – GPS (*Global Positioning System*).

Kita pozicijos nustatymo technologija atsirado sparčiai išpopuliarėjus mobiliesiems telefonams. Ji remiasi mobilaus telefono pozicijos nustatymu GSM (*Global System for Mobile Communications*) tinkle, panaudojant tokius metodus kaip artimiausios bazinės stoties nustatymas (*Cell identification* – Cell-ID), signalo atėjimo kampo metodas (*Angle of Arrival* – AOA), radijo signalo atėjimo laiko metodas (*Time of Arrival* – TAO) ar patobulintas laikų skirtumo metodas (*Enhanced Observed Time Difference* – E-OTD) [12] [13].

GPS tapus visuotinai prieinamai ir atpigus GSM paslaugoms, buvo pradėtos kurti įvairios programinės įrangos (toliau - PĮ) sistemos, naudojančios šias technologijas ir skirtos eiliniams vartotojams [7]. Pavyzdžiui GPS naudojančios TomTom, Frotcom, GPSPursuit ir kt., GSM naudojanti Locator ir kt. Tokio pobūdžio sistemų taikymo sritis nuolat plečiasi ir jos plačiai pradedamos naudoti kasdieniniame gyvenime. Būtent paprastam vartotojui yra skirta realizuota daugiavartotojiška pozicionavimo sistema, kurią ir aptarsime toliau.

Šiame skyriuje pateikiamos esminės sukurtos sistemos specifikacijos dalys: sistemai keliami reikalavimai, sistemos architektūra, su sistemos realizacija susiję ypatumai, sistemos testavimas.

3.1 Reikalavimai sistemai

3.1.1 Sistemos kūrimo pagrindas ir tikslai

Pozicionavimo sistemos, kurios buvo kuriamos karinei pramonei, šiuo metu tampa vis plačiau naudojamos įvairiose kasdienėse gyvenimo srityse. Iš pradžių GPS paremtos sistemos buvo pradėtos naudoti laivyboje, aviacijoje, transporto logistikoje, o šiuo metu vis plačiau jos naudojamos ir privačiame sektoriuje. Vis labiau populiarėja automobilinės navigacijos sistemos, GPS naudojančios automobilinės signalizacijos bei kitos panašaus pobūdžio sistemos, kurios yra skirtos eiliniam vartotojui.

Kita sparčiai populiarėjanti ir tobulėjanti technologijų sritis – mobilūs įrenginiai: mobilūs telefonai, nešiojami kompiuteriai ir delniniai kompiuteriai (PDA – *personal digital assistant*). Nešiojami kompiuteriai jau beveik nesiskiria nuo standartinių stacionarių kompiuterių,

telefonuose diegiamos operacinės sistemos, o delniniais kompiuteriais galima atlikti daugumą vartotojams reikalingų funkcijų, kurias jie atlieka su stacionariais kompiuteriais. Tam nemažai įtakos turi ir vis plačiau prieinamos mobilaus interneto paslaugos. Tai leidžia praplėsti egzistuojančių sistemų galimybes ir padaryti jas prieinamas iš mobiliųjų įrenginių.

Šio projekto tikslas – sukurti nekomercinę sistemą privačiam sektoriui, panaudojant PDA ir GPS teikiamas galimybes. Sistemos tikslas – registruotiems vartotojams leisti stebėti savo ir kitų, sutikimą davusių, vartotojų poziciją realiu laiku. Taip pat, teikti maršrutų saugojimo ir peržiūros galimybes.

3.1.2 Sistemos vartotojai

Sistemoje numatytos dvi vartotojų grupės:

1) **Vartotojai** – asmenys, kurie naudojasi sistemos teikiamomis galimybėmis. Tai įvairių amžiaus grupių vartotojai, turintys PDA su integruotu GPS imtuvu ir prieigą prie interneto ir mokantys naudotis IT technologijomis. Šie vartotojai naudojasi sistema asmeniniais arba verslo tikslais.

2) **Administratoriai** – asmenys, administruojantys sistemą. Administratoriai prižiūri sistemos veikimą, administruoja vartotojus, sistemos programinę įrangą, prieigą prie sistemos. Tai turėtų būti patyrę darbuotojai, gerai nusimanantys apie informacines technologijas (toliau IT), sistemą, jos struktūrą, funkcijas ir galimybes. Šie žmonės turėtų išmanyti GPS ir kitų sistemoje naudojamų technologijų veikimą. Taip pat turėtų žinoti apie galimus sistemos veikimo sutrikimus ir jų pašalinimo būdus.

Sistemos vartotojai, naudodamiesi sistema, gali:

1) registruotis sistemoje;
2) realiu laiku stebėti savo bei kitų vartotojų poziciją žemėlapyje;
3) peržiūrėti užfiksuotus savo ir kitų vartotojų judėjimo maršrutus;
4) įrašyti ir išsaugoti serveryje savo judėjimo maršrutą;
5) peržiūrėti savo bei kitų vartotojų išsaugotus maršrutus;
6) administruoti savo išsaugotus maršrutus.

Sistemos administratoriai gali:

1) sukurti naujus vartotojus ir patvirtinti pageidaujančiųjų registraciją;
2) administruoti vartotojus ir su jais susijusią informaciją sistemoje;
3) administruoti vartotojų maršrutus.

3.1.3 Apribojimai sprendimui

Vartotojų realaus laiko poziciją atvaizduojama žemėlapiuose, kurie imami iš specialios internetinės žemėlapių saugyklos. Judant tam tikru maršrutu, norint stebėti savo padėtį, būtina turėti PDA ar kitą mobilųjį kompiuterį su integruotu GPS imtuvu savo padėties nustatymui ir atvaizdavimui. Bendruoju atveju, realizacija neriboja naudojamo mobilaus kompiuterio, tačiau šiame magistriniame darbe yra naudojamas konkretus, šiai sistemai skirtas PDA, todėl visi su klientine dalimi susiję bandymai buvo vykdyti Fujitsu-Siemens N560 Pocket Loox delninio kompiuterio aplinkoje. Tikėtina, jog realizuota sistema taip pat sėkmingai funkcionuos ir su kitais mobiliaisiais įrenginiais, kurie palaikys J2ME technologiją, nes būtent ši technologija naudojama sistemos realizacijoje.

Taip pat labai svarbu akcentuoti, jog mobilusis įrenginys būtinai turi turėti prieigą prie interneto, priešingu atveju bus neįmanomą parsisiųsti žemėlapių pozicijos atvaizdavimui, nebent šie bus patalpinti žemėlapių keše iš anksčiau. Be to, neturint interneto, negalima stebėti kitų vartotojų pozicijos.

Kadangi numatyti naudoti žemėlapiai yra rastrinio formato, didelės priklausomybės nuo naudojamų žemėlapių nėra ir juos nesunkiai galima pakeisti kitais tokio tipo žemėlapiais. Vienintelis reikalavimas – turi būti suderinta žemėlapių saugyklos sąsaja.

Serveris realizuotas *Servlet* technologijos pagrindu, neturi ypatingos specifikos, todėl gali funkcionuoti tiek Windows, tiek Linux operacinėse sistemose be griežtesnių apribojimų. Klientas su serveriu turi komunikuoti panaudojant realizuota SOAP arba REST sąsają.

Apibendrinus, turime sekančius pagrindinius apribojimus, kurių netenkinant nepavyks pilnavertiškai pasinaudoti sistemos teikiamomis paslaugomis:

- Būtinas delninis kompiuteris palaikantis J2ME. Maksimaliam suderinamumui pasiekti, rekomenduojama naudoti *Fujitsu-Siemens N560 Pocket Loox* su integruotu GPS.
- Įrenginyje turi būti įdiegta J2ME (CLDC-1.1, MIDP-2.0) specifikaciją atitinkanti JVM (Java Virtual Machine). Rekomenduojama naudoti IBM J9.
- Būtina pastovi prieiga prie interneto.

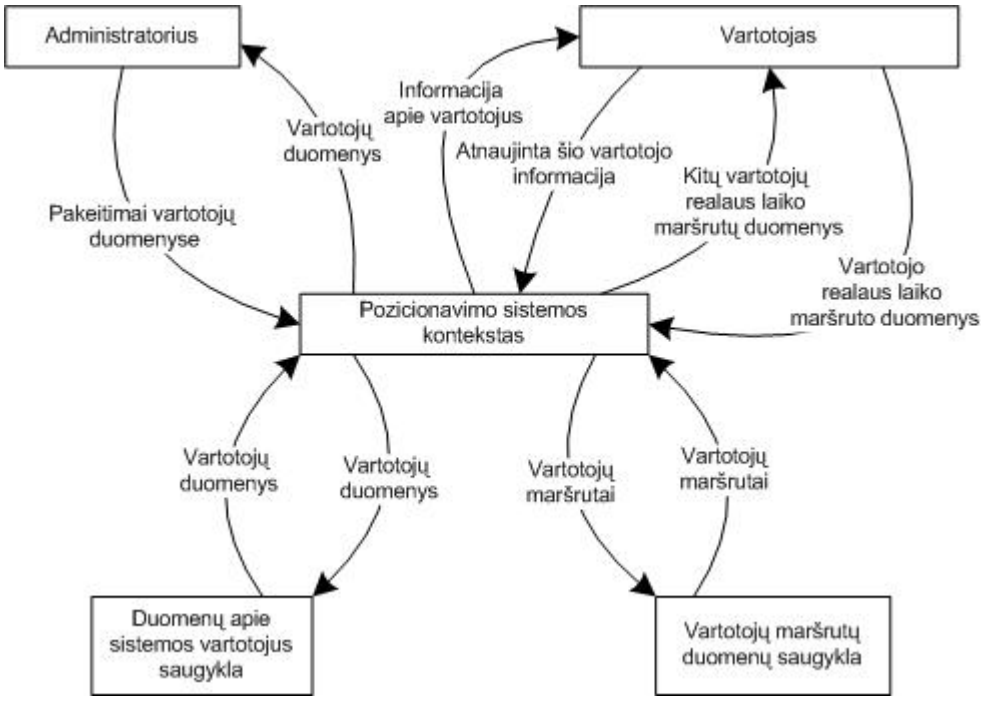
3.1.4 Diegimo aplinka

Kaip jau ir buvo minėta, viena pagrindinių sistemos dalių – klientinė, funkcionuoja delniniame kompiuteryje Fujitsu-Siemens N560 Pocket Loox su integruotu GPS įrenginiu (SirfStar3, GPS protokolas – NMEA). Šis delninis kompiuteris turi įdiegtą Windows Mobile 2005 operacinę sistemą. Naudojama J2ME specifikacijos realizacija yra IBM J9 JVM (MIDP 2.0, CLDC 1.1).

Serverio PĮ realizuota panaudojant Tapestry karkasą, kuris remiasi Java Servlet technologiją. Serveris ir portalas veikia Apache Tomcat 5.5 konteineryje.

3.1.5 Veiklos kontekstas

Šiame skyriuje pateikiama pozicionavimo sistemos veiklos konteksto diagrama (**2 pav.**), vaizduojanti pozicionavimo sistemos sąveiką su vartotojais ir duomenų saugyklomis.



2 pav. Veiklos konteksto diagrama

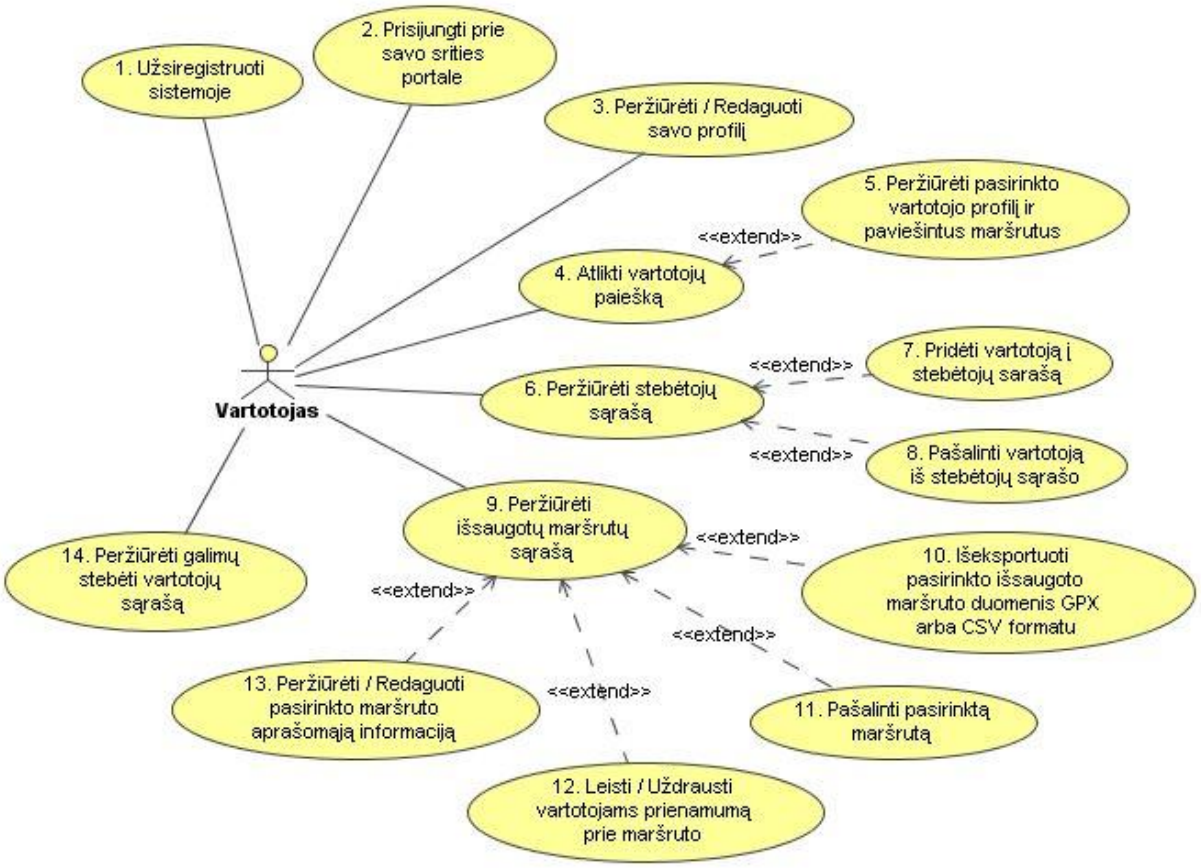
Sistemos duomenų srautai pateikiami **4 lentelėje**. Čia „in“ žymimas įeinantis srautas, o „out“ – išeinantis srautas. Pateikiami tik esminiai duomenų srautai.

4 lentelė. Sistemos duomenų srautai

Eil. Nr.	Įvykio pavadinimas	Įeinantys/Išeinantys informacijos srautai
1	Vartotojas pateikia pageidavimą naudotis sistema (registracija).	Vartotojo asmeniniai duomenys: vardas, pavarde, el. paštas ir kt. (in).
2	Vartotojas/Administratorius jungiasi prie WEB srities.	Vartotojo prisijungimo duomenys: vartotojo vardas ir slaptažodis (in).
3	Vartotojo GPS įrenginys realiu laiku perduoda vartotojo buvimo vietą.	Vartotojo realaus laiko ilgumos ir platumos GPS koordinatės (in).
4	Vartotojas stebi kitų sistemos vartotojų buvimo poziciją realiu laiku.	Stebimų vartotojų identifikatoriai (in). Vartotojų būvimo pozicija (out).
5	Vartotojas keičia savo asmeninius duomenis.	Atnaujinti vartotojo asmeniniai duomenys (in, out).
6	Vartotojas suteikia galimybę jį stebėti kitiems vartotojams.	Sistemos vartotojų, kuriems ši galimybė suteikiama, identifikatoriai (in).
7	Vartotojas atlieka vartotojų paiešką.	Ieškomų vartotojų daliniai preliminarūs duomenys (paieškos užklausa) (in). Atitikusių paieškos kriterijus vartotojų sąrašas bei informacija apie juos (out).
8	Vartotojas peržiūri savo ar kito vartotojo duomenis.	Pasirinkto vartotojo identifikatorius (in). Vartotoją apibūdinantys duomenys (out).
9	Vartotojas paviešina savo išsaugotą maršrutą.	Publikuojamo maršruto id (in).
10	Vartotojas išeksportuoja jam prieinamo pasirinkto maršruto duomenis.	Pasirinkto prieinamo maršruto identifikatorius (in). Maršruto duomenys pageidaujamo formato faile (out).
11	Vartotojas redaguoja maršrutą apibūdinančią informaciją.	Maršrutą apibūdinantys duomenys (in, out).
12	Vartotojas peržiūri maršrutą.	Pasirinktas maršruto identifikatorius (in). Pasirinktas maršrutas ekrane (out).
13	Vartotojas/Administratorius šalina iš sistemos išsaugotą maršrutą.	Pasirinkto maršruto identifikatorius (in).
14	Administratorius patvirtina/atmeta vartotojo pageidavimą užsiregistruoti sistemoje.	Vartotojas el. paštu gauna administratoriaus patvirtinimą/atmetimą (out).
15	Administratorius sukuria naują vartotoją.	Vartotoją apibūdinantys duomenys (in).
16	Administratorius pašalina egzistuojantį vartotoją.	Vartotojo identifikatorius (in).

3.1.6 Sistemos panaudos atvejai

Sistemos vartotojas prie sistemos portalo gali prisijungti pasinaudodamas standartine interneto naršykle. Ši prieiga yra skirta registracijai sistemoje, savo profilio ir saugomų maršrutų administravimui. Detaliau apie portalo teikiamas galimybes pateikiame portalo vartotojo panaudos atvejų diagramoje (**3 pav.**).



3 pav. Vartotojo panaudos atvejų diagrama (portalas)

Pakomentuosime kiekvieną iš panaudos atvejų.

1. **Užsiregistruoti sistemoje** – registracijos formos užpildymas ir jos pateikimas kaip prašymas prisiregistruoti, kad galima būtų naudotis sistemos teikiamomis paslaugomis.

2. **Prisijungti prie savo srities portale** – naudojantis standartine interneto naršykle, prisijungti prie sistemos portalo ir naudotis jo teikiamomis galimybėmis.

3. **Peržiūrėti / Redaguoti savo profilį** – vartotojas gali peržiūrėti/keisti jį apibūdinančią informaciją (profilį), kurią jis pateikė registracijos metu arba modifikavo vėliau. Ši informacija apima tokius duomenis kaip vardas, pavardė, el. pašto adresas, telefonas ir kt.

4. **Atlikti vartotojų paiešką** – vartotojas gali atlikti paiešką tarp kitų sistemoje registruotų vartotojų. Gavęs paieškos rezultatus, gali peržiūrėti pasirinktų vartotojų duomenis, paviešintus maršrutus bei pridėti vartotoją į stebėtojų sąrašą.

5. **Peržiūrėti pasirinkto vartotojo profilį ir paviešintus maršrutus** – atlikęs paiešką, vartotojas gali peržiūrėti pasirinkto vartotojo informaciją, jo paviešintų maršrutų sąrašą ir maršrutus apibūdinančią informaciją.

6. **Peržiūrėti stebėtojų sąrašą –** vartotojas gali peržiūrėti kitų sistemos vartotojų sąrašą, kuriems yra suteikęs teisę matyti savo poziciją realiu laiku.

7. **Pridėti vartotoją į stebėtojų sąrašą** – vartotojas gali pridėti bet kurį kitą sistemos vartotoją į savo stebėtojų sąrašą, t.y. leisti bet kuriam sistemos vartotojui stebėti savo poziciją realiu laiku.

8. **Pašalinti vartotoją iš stebėtojų sąrašo** – vartotojas, peržiūrėdamas savo stebėtojų sąrašą, gali pašalinti bet kurį stebėtoją. Pašalinus vartotoją iš stebėtojų sąrašo, pašalintas vartotojas netenka teisės stebėti šio vartotojo pozicijos realiu laiku.

9. **Peržiūrėti išsaugotų maršrutų sąrašą** – vartotojas gali peržiūrėti sistemoje išsaugotų savo maršrutų sąrašą.

10. **Išeksportuoti pasirinkto išsaugoto maršruto duomenis GPX arba CSV formatu** – vartotojas, peržiūrėdamas maršrutų sąrašą, gali pasirinkti norimą maršrutą ir išeksportuoti šį maršrutą GPX (http://www.topografix.com/gpx.asp) ir/arba CSV (http://en.wikipedia.org/wiki/Comma-separated_values) formatais.

11. **Pašalinti pasirinktą maršrutą** – vartotojas gali pašalinti iš sistemos bet kurį savo išsaugotą maršrutą.

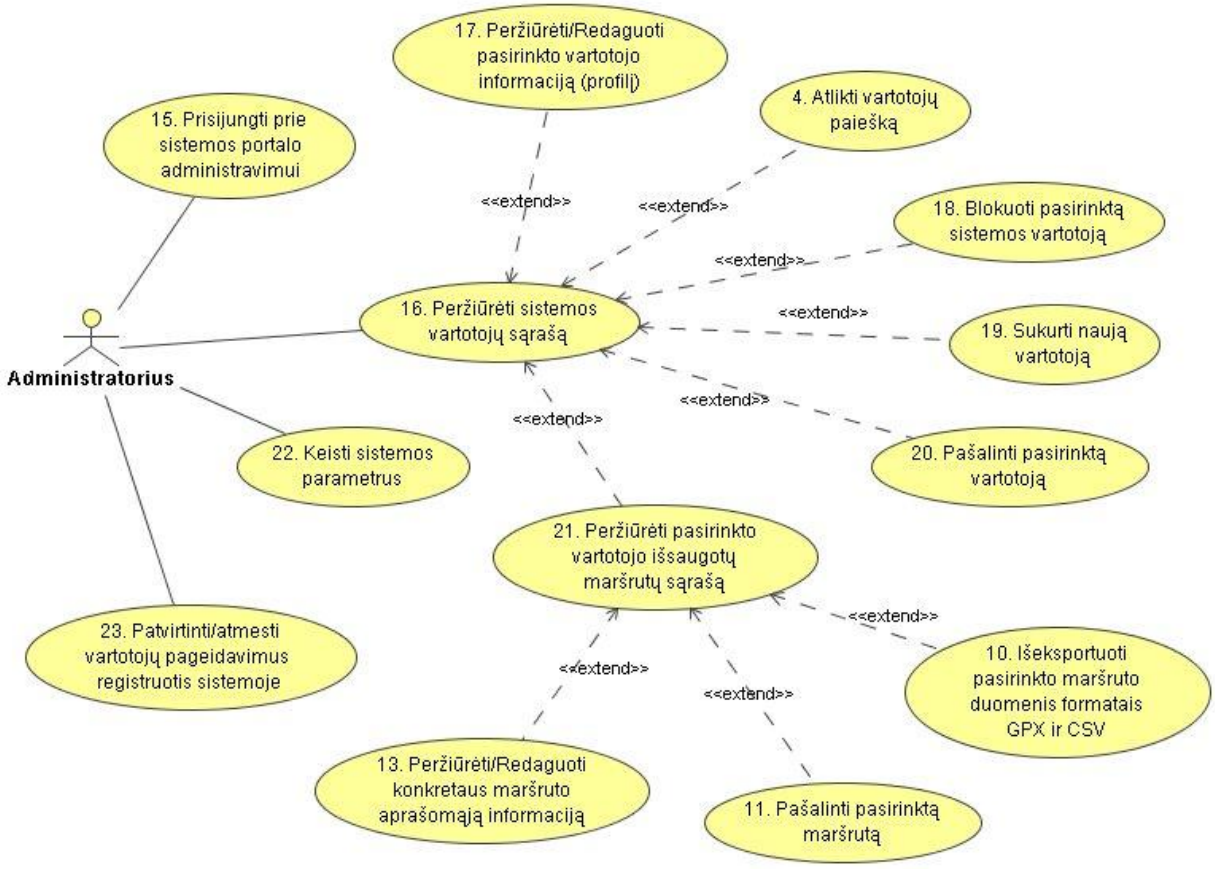
12. **Leisti / Uždrausti vartotojams prieinamumą prie maršruto** – vartotojas gali leisti kitiems sistemos vartotojams peržiūrėti pasirinktą maršrutą, t.y. gali paviešinti norimą maršrutą. Jei maršrutas nepaviešintas, jis yra nematomas ir neprieinamas kitiems sistemos vartotojams.

13. **Peržiūrėti / Redaguoti pasirinkto maršruto aprašomąją informaciją** – vartotojas gali peržiūrėti ir pakeisti maršrutą apibūdinančią informaciją, tokią kaip pavadinimas, aprašymas ir kt.

14. **Peržiūrėti galimų stebėti vartotojų sąrašą** – vartotojas gali peržiūrėti kitų vartotojų, kurie yra jam suteikę galimybę stebėti juos, sąrašą.

Sistemos serveryje saugomų duomenų administravimas atliekamas per portalo sąsają. Administratorius prie sistemos prisijungti gali su standartine interneto naršyklę. Be specifinių

administravimo veiksmų, jis sistemoje gali atlikti didžiąją dalį tipiniam vartotojui prieinamų veiksmų. Administratoriui suteikiamas galimybes pateikiame administratoriaus panaudos atvejų diagramoje (**4 pav.**).



4 pav. Sistemos administratoriaus panaudos atvejų diagrama (portalas)

Pastaba: kai kurie panaudos atvejai sutampa su prieš tai buvusia diagrama, todėl numeriai kartojasi.

15. **Prisijungti prie sistemos portalo administravimui** – naudodamasis standartine interneto naršykle gali prisijungti prie sistemos administravimo veiksmams atlikti.

16. **Peržiūrėti sistemos vartotojų sąrašą** – administratorius gali peržiūrėti visų sistemoje registruotų vartotojų sąrašą. Iš čia jis gali peržiūrėti pasirinktų vartotojų informaciją, maršrutus ir atlikti reikiamas vartotojų administravimo funkcijas.

17. **Peržiūrėti / Redaguoti pasirinkto vartotojo informaciją** – administratorius gali keisti kito vartotojo duomenis.

18. **Blokuoti pasirinktą sistemos vartotoją** – administratorius gali užblokuoti pasirinktą sistemos vartotoją. Užblokuotas vartotojas negali prisijungti ir naudotis sistemos teikiamomis paslaugomis.

19. **Sukurti naują vartotoją** – administratorius gali sukurti naują sistemos vartotoją.

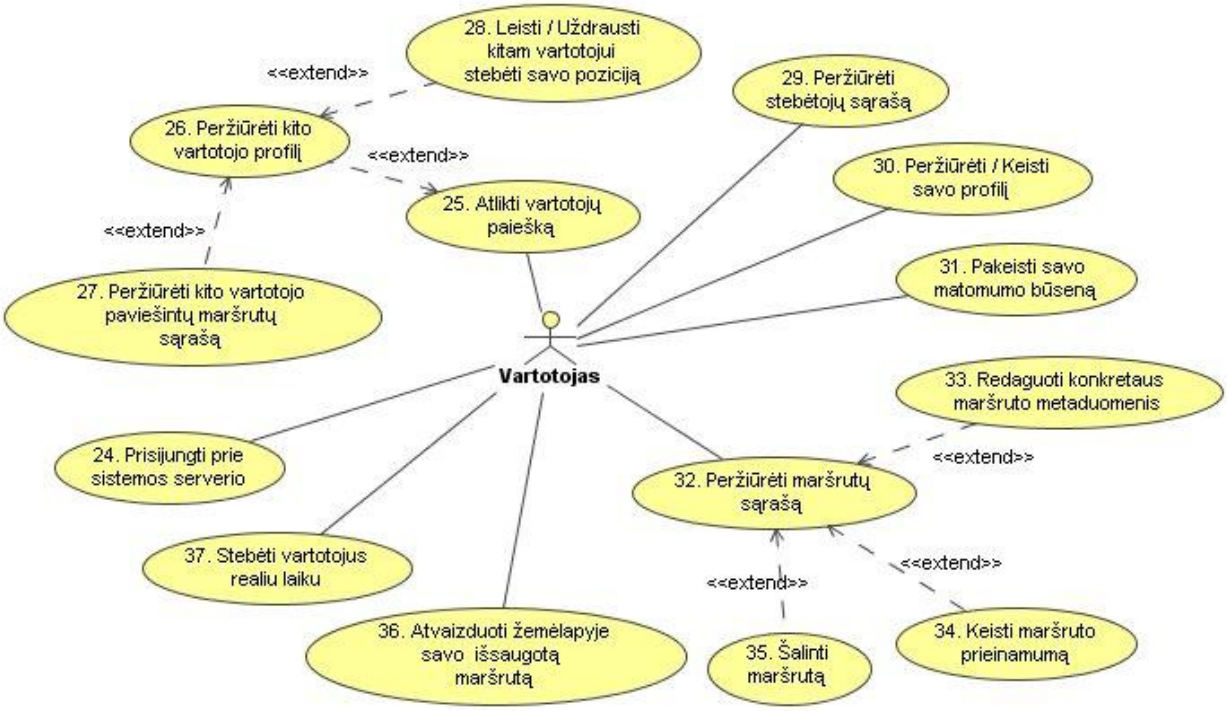
20. **Pašalinti pasirinktą vartotoją –** sistemos administratorius gali pašalinti sistemos vartotoją. Tokiu atveju, iš sistemos yra ištrinami visi su vartotoju susiję duomenys.

21. **Peržiūrėti pasirinkto vartotojo išsaugotų maršrutų sąrašą** – administratorius gali tvarkyti vartotojų išsaugotus maršrutus, peržiūrėdamas šių maršrutų sąrašą.

22. **Keisti sistemos parametrus** – administratorius gali konfigūruoti sistemą nustatydamas tam tikrus sistemos veikimo parametrus.

23. **Patvirtinti / Atmesti vartotojų pageidavimus registruotis sistemoje** – sistemos vartotojams prisiregistravus sistemoje, t.y. pateikus prašymą, administratoriui pateikiamas atitinkamas pranešimas apie šiuos pageidavimus. Administratorius gali patvirtinti arba atmesti vartotojo registraciją.

Pasinaudodamas klientu, įdiegtu delniniame kompiuteryje, sistemos vartotojas gali atlikti veiksmus pateiktus sekančioje panaudos atvejų diagramoje (**5 pav.**).



5 pav. Vartotojo panaudos atvejų diagrama (klientinė dalis)

24. **Prisijungti prie sistemos serverio –** prisijungti prie sistemos serverio, kad pasinaudoti jo teikiamomis paslaugomis. Prisijungimas būtinas norint stebėti kitus vartotojus.

25. **Atlikti vartotojų paiešką** – vartotojas gali atlikti kitų sistemoje registruotų vartotojų paiešką. Gavęs paieškos rezultatus, gali peržiūrėti pasirinktų vartotojų duomenis, išpublikuotus maršrutus bei pridėti vartotoją į stebėtojų sąrašą.

26. **Peržiūrėti kito vartotojo profilį** – vartotojas gali peržiūrėti kito sistemos vartotojo duomenis.

27. **Peržiūrėti kito vartotojo paviešintų maršrutų sąrašą** – vartotojas gali peržiūrėti pasirinkto kito sistemos vartotojo paviešintų maršrutų sąrašą, atlikdamas jo duomenų peržiūrą.

28. **Leisti / Uždrausti kitam vartotojui stebėti savo poziciją** – vartotojas gali pasirinktam vartotojui leisti/uždrausti stebėti savo poziciją realiu laiku įtraukdamas/pašalindamas vartotoją į/iš stebėtojų sąrašo.

29. **Peržiūrėti stebėtojų sąrašą** – vartotojas gali peržiūrėti sąrašą kitų sistemos vartotojų, kuriems yra suteikęs teisę matyti savo poziciją realiu laiku.

30. **Peržiūrėti / Keisti savo profilį** – vartotojas gali peržiūrėti ir keisti savo profilio duomenis, t.y. vardą, pavardę ir kitą informaciją.

31. **Pakeisti savo matomumo būseną** – vartotojas gali keisti savo matomumo būseną. Jei vartotojas pasirenka būti nematomu, duomenys nėra siunčiami į serverį, net ir tada kai jis prisijungęs.

32. **Peržiūrėti maršrutų sąrašą** – vartotojas gali administruoti savo maršrutus, peržiūrėdamas jų sąrašą.

33. **Redaguoti maršruto metaduomenis** – vartotojas gali redaguoti pasirinkto maršruto duomenis, tokius kaip maršruto pavadinimas, aprašymas ir kt.

34. **Keisti maršruto prieinamumą** – vartotojas gali keisti maršruto prieinamumą kitiems sistemos vartotojams. Maršrutai gali būti išpublikuoti arba neišpublikuoti..

35. **Šalinti maršrutą** – vartotojas gali pašalinti pasirinktą išsaugotą maršrutą iš serverio.

36. **Atvaizduoti žemėlapyje savo išsaugotą maršrutą** – pasinaudodamas klientu, vartotojas gali atvaizduoti žemėlapyje pasirinktą maršrutą.

37. **Stebėti vartotojus realiu laiku** – vartotojas gali realiu laiku stebėti savo ir kitų, sutikimą davusių, sistemos vartotojų poziciją žemėlapyje.

3.1.7 Funkciniai reikalavimai

Funkciniai reikalavimai suformuluoti detalizuojant esamus panaudos atvejus. Kadangi tam tikros sistemos teikiamos funkcijos prieinamos tiek per portalą, tiek ir per klientą, tai dalis reikalavimų galioja abiems atvejams.

Toliau pateikiame tik esminius, sistemai taikomus, funkcinius reikalavimus:

- Vartotojo vardas sistemoje turi būti unikalus, todėl sistema turi patikrinti ir neleisti sukurti vartotojo su jau egzistuojančiu vartotojo vardu ir formuoti atitinkamą pranešimą.
- Registracijos ir profilio keitimo metu, vartotojo įvedama informaciją turi būti patikrinama, ar tenkina duomenų turinio ir apimties reikalavimus.
- Sistemos teikiamos galimybės turi būti prieinamos tik registruotiems ir prisijungusiems sistemos vartotojams. Bet kurie neprisijungę vartotojai, mėgindami atlikti veiksmus, leistinus tik prisijungusiems vartotojams, turi būti automatiškai nukreipiami į prisijungimo formos langą.
- Jei sistemos vartotojas yra užblokuotas, tai jam jungiantis prie sistemos jis turi būti apie tai informuojamas atitinkamu pranešimu.
- Atlikęs paiešką, vartotojas turi turėti galimybę viename puslapyje peržiūrėti jo nurodytą pageidaujamą rezultatų kiekį, o sekančius rezultatų puslapius pasiekti nuspausdamas atitinkamą nuorodą.
- Kiekvienos paieškos užklausos metu, turėtų būti atliekama dalinė paiešką, o paieškos rezultatai nesaugomi serverio atmintyje.
- Vartotojui peržiūrint savo maršrutų arba jį galinčių stebėti vartotojų sąrašą, turi būti pateikiamas puslapiuojamas sąrašas su navigacija.
- Administratoriui patenkinus registracijos prašymą ir vartotojui patvirtinus savo el. pašto adresą, turi būti automatiškai sukuriamas aktyvus sistemos vartotojas.
- Vartotojo pašalinimo atveju, visi su juo susiję sistemoje saugomi duomenys turi būti pašalinami.
- Su klientu neprisijungęs prie serverio vartotojas turi turėti galimybę stebėti savo poziciją žemėlapyje ir išsaugoti maršrutą lokaliai, tačiau visos kitos sistemos teikiamos galimybės jam turi būti neprieinamos.

- Naudodamasis klientu, vartotojas turi turėti galimybę pasirinkti, iš galimų stebėti vartotojų sąrašo, kokius vartotojus jis nori stebėti žemėlapyje.

3.1.8 Nefunkciniai reikalavimai

Nefunkciniai sistemos reikalavimai apima reikalavimus vartotojo sąsajai, sistemos saugumui, panaudojamumui, vykdymo charakteristikoms, duomenims, sistemos palaikymui ir pan.

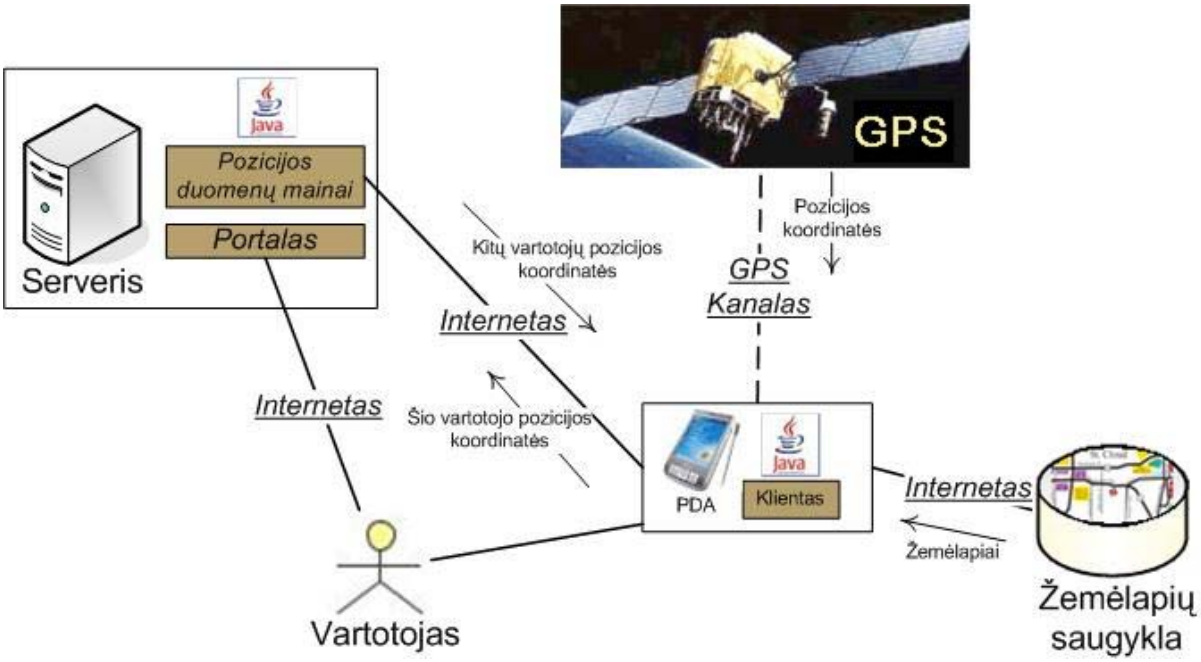
Toliau pateikiame tik esminius nefunkcinius, sistemai taikos reikalavimus:

- Sistema turi būti paprasta naudotis vartotojui susipažinusiam su bazine naudojimosi instrukcija.
- Sistema neturi perduoti asmeninių vartotojo duomenų, kitiems su sistemos administravimu nesusijusiems, asmenims;
- Delninis kompiuteris, lyginant su stacionariu, turi nedidelį ekraną, todėl kliento sąsajos elementai turi būti pakankamo dydžio ir aiškiai įžiūrimi, nes smulkų tekstą sunku įskaityti, o smulkius mygtukus – paspausti.
- Vartotojo sąsajos elementai, turi būti sugrupuoti pagal paskirtį ir, jei reikia, išdėlioti atskirose skiltyse. Sąsajos elementų grupavimas pagal paskirtį suteikia galimybę greičiau juos aptikti ir greičiau atlikti reikiamus pakeitimus.
- Sistemos kliento pilnas paleidimas turėtų būti įvykdomas paleidžiant vieną vykdomąjį failą, o ne atskirai paleidinėjant atskiras posistemes.

3.2 Sistemos struktūra

3.2.1 Sistemos veikimo principas

Sistemos veikimas yra paremtas kliento ir serverio principu. Vartotojas naudodamasis delniniame kompiuteryje įdiegta programine įranga – klientu, gali stebėti savo ir kitų vartotojų poziciją. Delninio kompiuterio GPS įrenginys vartotojo pozicijos duomenis gauna iš GPS palydovų ir perduoda klientui. Klientas šiuos duomenis siunčia į serverį, tokiu būdu sudarydamas galimybę juos pasiimti kitiems, tokiu pačiu klientu besinaudojantiems ir atitinkamą leidimą turintiems, vartotojams. Žemėlapius klientas tiesiogiai nuskaito iš lokalaus kešo, arba, jei reikiamo fragmento ten nėra, iš internetinės žemėlapių saugyklos. Serverio dalyje veikiantis portalas suteikia vartotojui galimybę tvarkyti savo duomenis, įskaitant ir maršrutus, kuriuos jis gali išsaugoti serveryje pasinaudodamas klientu.

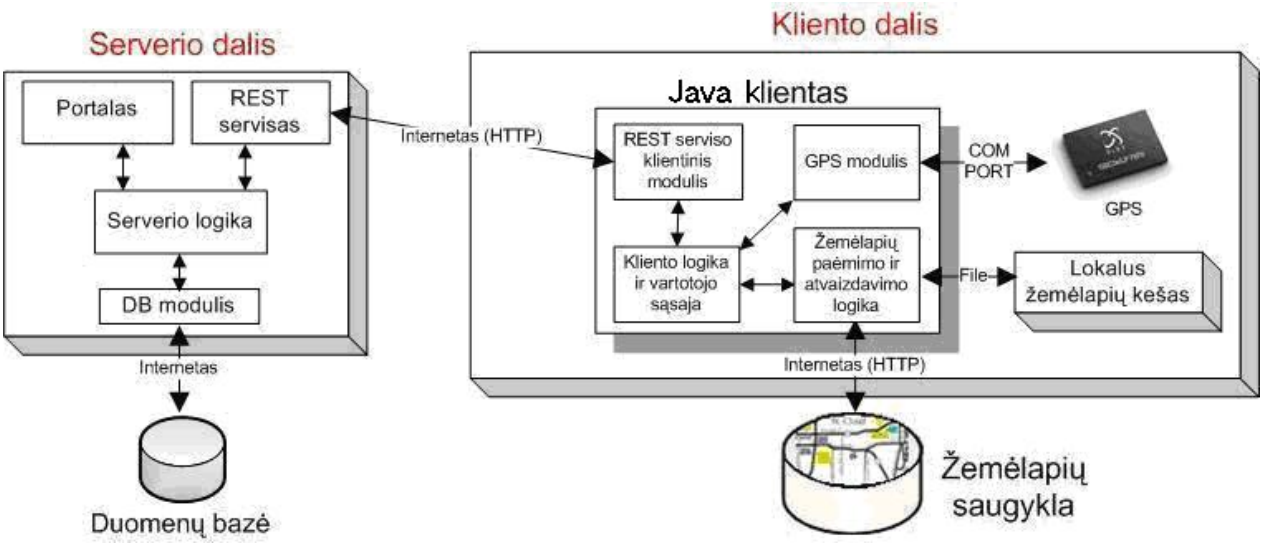


6 pav. Sistemos veikimo principas

Renkantis pozicionavimo technologiją buvo nuspręsta naudoti GPS, kadangi ji dengia visą Žemės paviršių, gali pateikti trimates koordinates, nustatyti objekto judėjimo kryptį ir greitį. Dauguma šių savybių sistemoje nėra pilnai išnaudojamos, tačiau tai suteikia sistemos plėtimo galimybes ateityje, papildant ją nauju funkcionalumu. Nors dauguma GPS imtuvų veikia tik lauko sąlygomis, tačiau yra tikslūs: standartinis GPS imtuvas gali pateikti pozicijos koordinates 3 – 15 m tikslumu, o aukštesnio lygio – 1 m tikslumu. GSM pozicionavimo technologija veikia ne tik lauke, bet ir pastatų patalpose, tačiau šių koordinačių tikslumas yra žymiai mažesnis ir priklauso nuo GSM tinklo struktūros, pozicijos nustatymui naudojamo metodo ir vietovės. GSM atveju paklaida svyruoja nuo 50 m iki 30 km.

3.2.2 Bendroji architektūra

Kaip ir buvo minėta, šios sistemos architektūra yra paremta kliento ir serverio principu. Serveris vartotojams teikia prieigą prie portalo ir prie REST serviso, naudodamas bendrą serverio logikos modulį. Bendrosios logikos modulis komunikuoja su duomenų baze per atitinkamą duomenų bazės modulį.



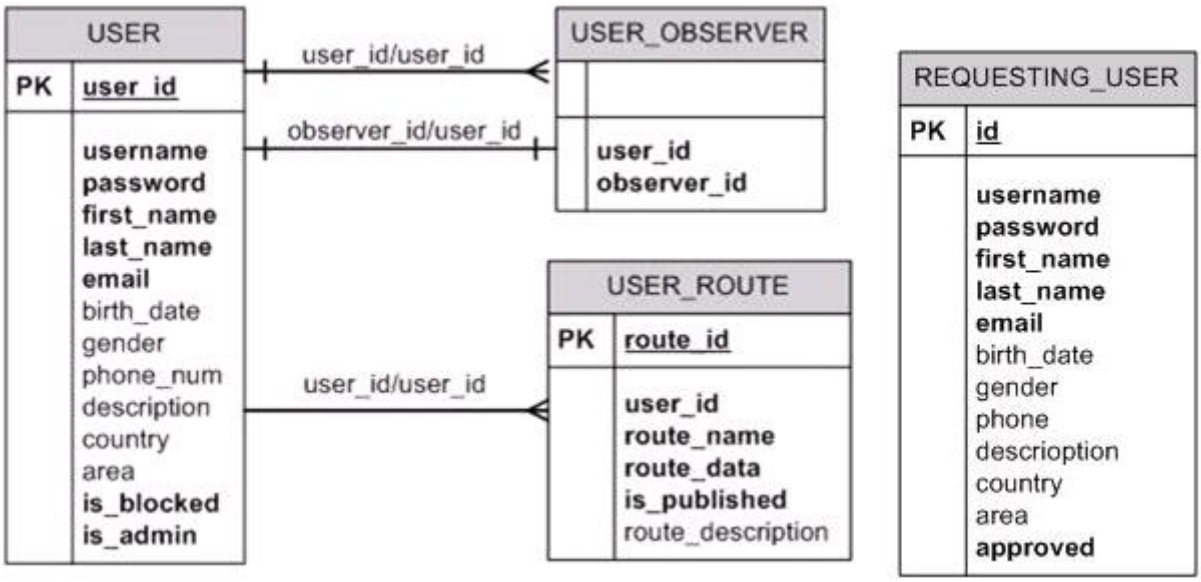
7 pav. Bendroji architektūra

Kliento dalyje pagrindinis vidinės logikos modulis yra glaudžiai susijęs su vartoto sąsaja. Jis naudoja GPS modulį duomenų iš GPS įrenginio nuskaitymui, žemėlapių nuskaitymo ir atvaizdavimo logikos modulį. Su serveriu jis komunikuoja naudodamas paprastą, bet efektyvų, REST serviso klientinį modulį. GPS modulis koordinates nuskaito iš GPS įrenginio per nuosekliąją sąsajos jungtį (COM port). Žemėlapių nuskaitymo ir atvaizdavimo modulis reikalingus žemėlapių fragmentus nuskaito iš failų sistemoje esančio lokalaus kešo arba iš internetinės žemėlapių saugyklos. Vartotojui nurodžius, naujai parsiųsti žemėlapių fragmentai gali būti kešuojami.

Viena pagrindinių sistemos dalių – klientas, veikia delniniame kompiuteryje. Tokio kompiuterio našumas lyginant su paprasto stalinio kompiuterio našumu yra labai mažas. Prieigai prie interneto dažniausiai naudojamas bendras paketinis radijo ryšys (*General Packet Radio Service* – GPRS), kur apmokestinamas kiekvienas išsiustas megabaitas. Būtent dėl šių priežasčių bendra sistemos architektūra yra sudaryta taupant resursus ir mažinant persiunčiamų bei naudojamų duomenų apimtis.

3.2.3 Duomenų bazės schema

Sistemoje duomenų saugojimui yra naudojama MySql duomenų bazė. Duomenų bazės schema pateikta **8 pav.**



8 pav. Duomenų bazės schema

Pastaba: Diagramoje naudojamas žymėjimas PK – pirminis raktas. Paryškinti laukai yra privalomi.

Duomenų bazės lentelių detalios specifikacijos su lentelių laukais, duomenų tipais ir laukų aprašymais pateikiamos žemiau.

Duomenų bazės lentelė „USER“ naudojama saugoti duomenims apie sistemos vartotojus.

5 lentelė. Duomenų bazės lentelės "USER" specifikacija

USER			
PK	**Laukas**	**Tipas**	**Aprašymas**
*	**user_id**	INTEGER	Vartotojo identifikatorius
	username	VARCHAR(32)	Vartotojo prisijungimo vardas
	password	CHAR(41)	Vartotojo prisijungimo slaptažodis
	first_name	VARCHAR(40)	Vartotojo vardas
	last_name	VARCHAR(40)	Vartotojo pavardė
	email	VARCHAR(90)	Vartotojo el. pašto adresas
	birth_date	BIGINT(20)	Vartotojo gimimo data
	gender	TINYINT(1)	Vartotojo lytis
	phone_num	VARCHAR(24)	Vartotojo telefono numeris
	description	TEXT	Vartotojo aprašymas
	country	VARCHAR(90)	Vartotojo šalis
	area	VARCHAR(90)	Vartotojo miestas
	is_blocked	TINYINT(1)	Laukas skirtas pažymėti blokuojamus vartotojus.
	is_admin	TINYINT(1)	Laukas nurodantis ar vartotojas yra administratorius.

Duomenų bazės lentelėje „REQUESTING_USER“ saugomi pageidavimai registruotis sistemoje. Čia saugoma informacija naudojama naujo sistemos vartotojo sukūrimui, kai administratorius patvirtina prašymą registruotis. Vartotojas papildomai turi patvirtinti savo el. paštą nuspausdamas į šį el. paštą atsiųstame pranešime esančia nuorodą.

6 lentelė. Duomenų bazės lentelės "REQUESTING_USER" specifikacija

REQUESTING_USER			
PK	**Laukas**	**Tipas**	**Aprašymas**
*	**id**	INTEGER	Vartotojo pageidavimo įrašo identifikatorius
	username	VARCHAR(32)	Vartotojo prisijungimo vardas
	password	CHAR(41)	Vartotojo prisijungimo slaptažodis
	first_name	VARCHAR(40)	Vartotojo vardas
	last_name	VARCHAR(40)	Vartotojo pavardė
	email	VARCHAR(90)	Vartotojo el. pašto adresas
	birth_date	BIGINT(20)	Vartotojo gimimo data
	gender	TINYINT(1)	Vartotojo lytis
	phone	VARCHAR(24)	Vartotojo telefono numeris
	description	TEXT	Vartotojo aprašymas
	country	VARCHAR(90)	Vartotojo šalis
	area	VARCHAR(90)	Vartotojo miestas
	approved	TINYINT(1)	Laukas nurodantis ar administratorius patvirtino registraciją.

Duomenų bazės lentelėje „USER_ROUTE“ yra saugomi vartotojo maršrutai ir su jais susiję metaduomenys, kuriuos jis išsaugo pasinaudodamas klientu.

7 lentelė. Duomenų bazės lentelės "USER_ROUTE" specifikacija

USER_ROUTE			
PK	**Laukas**	**Tipas**	**Aprašymas**
*	**id**	INTEGER	Identifikatorius
	user_id	INTEGER	Identifikatorius nurodantis kuriam vartotojui priklauso maršrutas.
	route_name	VARCHAR(90)	Maršruto pavadinimas
	route_data	LONGTEXT	Maršruto koordinatės
	is_published	TINYINT(1)	Laukas nurodantis ar maršrutas išpublikuotas.
	route_description	TEXT	Maršruto aprašymas

Duomenų bazės lentelė „USER_OBSERVER“ yra naudojama vartotojų stebėtojų sąrašui nusakyti.

8 lentelė. Duomenų bazės lentelės "USER_OBSERVER" specifikacija

USER_OBSERVER			
PK	**Laukas**	**Tipas**	**Aprašymas**
	user_id	INTEGER	Identifikatorius nurodantis vartotoją, kuris turi stebėtoją.
	observer_id	INTEGER	Stebėtojo identifikatorius, nurodo kuris vartotojas yra stebėtojas.

3.3 Realizacijos ypatumai

Magistriniame darbe realizuotos sistemos klientas žemėlapių nuskaitymui ir vartotojo pozicijos taško atvaizdavimui žemėlapyje atlikti vykdo įvairius skaičiavimus ir transformacijas. Šiame skyriuje aptarsime šiems skaičiavimams ir transformacijoms atlikti naudojamus algoritmus, duomenų struktūras ir susijusias matematines funkcijas. Taip pat aprašysime pozicijos duomenų nuskaitymą iš GPS įrenginio.

3.3.1 Vartotojo pozicijos koordinačių nuskaitymas

Savo ir kitų vartotoju pozicijai stebėti, vartotojas naudojasi delniniu kompiuteriu, kuriame įdiegtas klientas. Šis klientas vartotojo pozicijos koordinates nuskaito iš kompiuteryje esančio GPS įrenginio, per nuosekliosios sąsajos jungtį (COM port). Šis GPS įrenginys, kaip ir daugelis kitų, naudoja labiausiai paplitusį NMEA 0183 (http://www.nmea.org/pub/0183/) komunikavimo standartą. Taigi, vartotojui keičiant savo buvimo vietą atviroje aplinkoje, GPS įrenginys generuoja NMEA standarto apibrėžtus, specialios struktūros sakinius, kuriuose yra pateikiamos pozicijos koordinatės ir kita susijusi informacija. Papildoma informacija apima įvairius duomenis apie įrenginio aptiktus palydovus, aukštį virš jūros lygio ir pan.

Keletas NMEA sakinių pavyzdžių:

- `$GPGLL,4916.45,N,12311.12,W,225444,A,*31`
- `$GPGSV,2,1,08,01,40,083,46,02,17,308,41,12,07,344,39,14,22,228,45*75`
- `$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A`
- `$GPVTG,054.7,T,034.4,M,005.5,N,010.2,K*33`
- `$GPAAM,A,A,0.10,N,WPTNME*43`
- `$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47`
- `$GPBOD,045.,T,023.,M,DEST,START*01`

Iš tiesų, vidutiniškai tik nedidelis kiekis įrenginio sugeneruotu NMEA sakinių saugo pozicijos koordinates. Klientas šias koordinates paima tik iš trijų tipų sakinių, kurių pavyzdžius pateikiame žemiau:

- `$GPGLL,4916.45,N,12311.12,W,225444,A,*31`
- `$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A`
- `$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47`

Pabraukti sakiniuose esantys fragmentai yra reikiamos ilgumos ir platumos koordinatės.

Internete yra nemažai NMEA 0183 sakinių analizės atvirojo kodo programinių bibliotekų, tačiau šiuo atveju reikalingų sakinių kiekis yra labai mažas, todėl sistemoje realizuotas efektyvus specializuotas šių sakinių programinis analizės modulis. Pastarasis ištraukia reikiamas koordinates ir atiduoda tolimesniems, šias koordinates naudojantiems moduliams.

3.3.2 Vartotojo pozicijos koordinačių transformavimas

Iš GPS įrenginio nuskaitytos ir iš NMEA sakinių paimtos vartotojo pozicijos ilgumos ir platumos koordinatės priklauso geografiniai koordinačių sistemai, kuri yra sferinės koordinačių sistemos atmaina [11]. Ilgumos ir platumos koordinatės yra apriboti geometriniai kampai: ilgumos koordinatės kitimo rėžiai yra nuo -180° iki 180°, o platumos nuo -90° iki 90°. Sistemoje šios koordinatės naudojamos reikiamam žemėlapio fragmentui išskaičiuoti ir ant šio fragmento atvaizduoti vartotojo pozicijos tašką. Naudojami žemėlapiai, pasiekiami internetinėje žemėlapių saugykloje, yra sudaryti cilindrinės Mercator projekcijos [19] pagrindu. Išskleidus cilindrą yra gaunamas viso pasaulio žemėlapis stačiakampėje koordinačių sistemoje. Natūraliai iškyla koordinačių sistemų nesuderinamumo problema, kurią galima išspręsti transformuojant geografines pozicijos koordinates į stačiakampę koordinačių sistemą. Tokią transformaciją galima atlikti panaudojant atvirkštinę Gudermano (*Gudermannian*) funkcija [18]:

$$y = gd^{-1}(\varphi) = \frac{1}{2}\ln\frac{1+\sin(\varphi)}{1-\sin(\varphi)}, \text{ kur } \varphi \text{ – platumos koordinatė.}$$

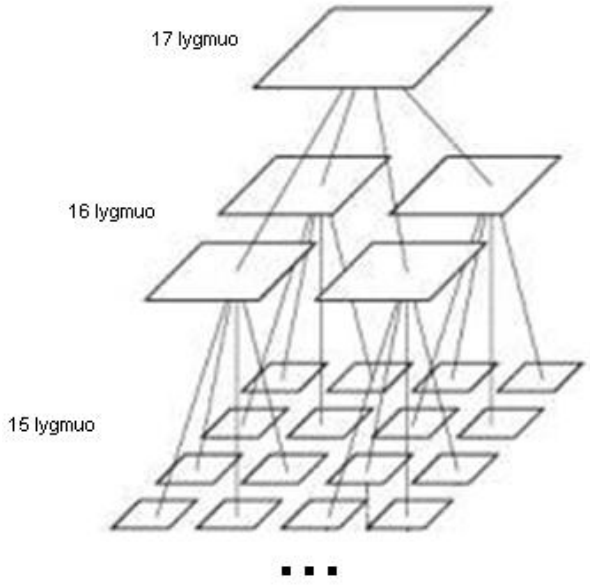
Kaip matyti, transformacija yra reikalinga tik platumos koordinatei. Po šios transformacijos pozicijos koordinatės yra suderinamos su žemėlapių koordinačių sistema.

3.3.3 Žemėlapių fragmentų koordinačių išskaičiavimas

Objekto pozicijos atvaizdavimui kliente reikalingi žemėlapiai. Turint objekto pozicijos koordinates, reikiamų žemėlapių fragmentų pirmiausiai ieškoma lokaliame žemėlapių keše. Jei reikiamo fragmento keše nėra, jis imamas iš internetu pasiekiamos žemėlapių saugyklos. Žemėlapių saugykla žemėlapių fragmentus pateikia pagal jų specifines koordinates, todėl klientas turi mokėti išskaičiuoti reikiamo fragmento koordinates.

Saugykloje žemėlapiai saugomi kvadratinių, 256x256 taškų dydžio, PNG (http://www.libpng.org/pub/png/) formato fragmentų pavidalu, o pilnas iš šių fragmentų sudarytas žemėlapis paremtas *Quadtree* struktūros pagrindu.

Quadtree struktūra yra medis kurio kiekviena tėvinė viršūnė turi iki keturių žemesnio lygmens viršūnių. Ši struktūra dažniausiai taikoma rekursiniam dviejų dimensijų plokštumų sudalinimui į keturis kvadrantus, kas konkrečiu atveju yra pritaikytą naudojamiems žemėlapiams. Pilnas žemėlapis turi 18 išdidinimo lygmenų (0..17), kas atsispindi **9 pav.** iliustruotoje *Quadtree* struktūroje. 17 lygmuo yra labiausiai nutolintas žemėlapio fragmentas ir šiuo atveju atitinka visą pasaulį.



9 pav. *Quadtree* žemėlapių struktūra

Pastebėsime, jog iliustracijoje pateikti fragmentai mažėja su kiekvienu lygmeniu gilyn, tačiau taip pavaizduota tik vaizdumo sumetimais ir realybėje visi jie yra vienodo – 256x256 pikselių dydžio.

Toliau trumpai nusakysime minėtos internetinės saugyklos žemėlapio fragmentų koordinačių išskaičiavimą.

Išskaičiavimas pradedamas nuo pradinio fragmento X_0, kuris atitinka visą pasaulį ir yra z_0=17 priartinimo lygmens. Šis fragmentas yra menamai dalinamas į keturias lygias dalis (**10 pav.**) ir pagal turimas pozicijos koordinates, parenkama viena reikiama dalis. Naujai parinktos dalies atžvilgiu veiksmai yra kartojami tol, kol pasiekiamas reikiamo priartinimo lygio reikiamas fragmentas. Kitaip tariant, einama gilyn per *Quadtree* struktūros lygmenis ir kaskart parenkamas reikiamas, vienas iš keturių, fragmentas.

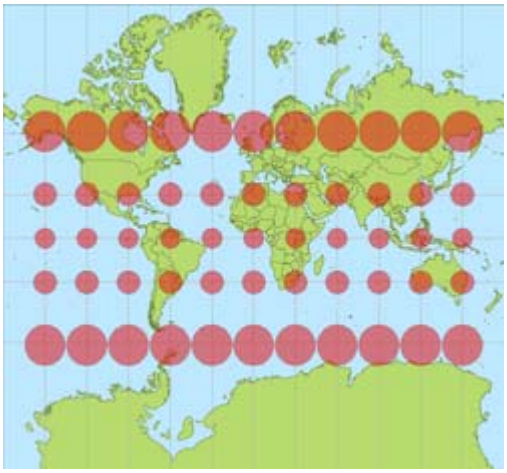
10 pav. Žemėlapio fragmentas

Matematiškai pirmos iteracijos kiekvieno galimo naujai parenkamo fragmento išskaičiavimą galima apibrėžti taip:

$$X_1(x_1;y_1;z_1) = X_0(2x_0;2y_0;z_0-1) \qquad X_3(x_3;y_3;z_3) = X_0(2x_0;2y_0+1;z_0-1)$$
$$X_2(x_2;y_2;z_2) = X_0(2x_0+1;2y_0;z_0-1) \qquad X_4(x_4;y_4;z_4) = X_0(2x_0+1;2y_0+1;z_0-1)$$

Kur $X_i(x_i;y_i;z_i)$, $i=\overline{1,4}$ yra **10 pav.** pateiktas i-tasis fragmentas: x, y koordinatės ir z – priartinimo lygis.

Kaip ir buvo minėta, saugykloje saugomi žemėlapiai yra sudaryti Mercator projekcijos pagrindu. Ši projekcija yra plačiai taikoma ir gerai žinoma geografijos srities specialistams, tačiau turi esminį trūkumą – stipriai iškraipo žemėlapio mastelį tolstant nuo pusiaujo link ašigalių [3]. Kaip pavyzdį galima palyginti Afrika su Grenlandija, kurios Mercator projekcijoje yra beveik tokio paties dydžio, tačiau realybėje Grenlandija yra beveik 14 kartų mažesnė už Afriką.



11 pav. Mercator projekcijos deformacijos

(Paveikslėlis paimtas iš http://en.wikipedia.org/wiki/Mercator_projection)

Nepaisant šio trūkumo, ši projekcija yra tinkama interaktyvių internetinių žemėlapių sudarymui, kuriais patogu manipuliuoti, todėl neretai yra taikoma geografinėse informacinėse sistemose (GIS) ir puikiai tinka realizuotos sistemos poreikiams.

3.4 Testavimas

Testavimo metu buvo siekiama aptikti galimas programavimo ir logikos klaidas, patikrinti atskirų sistemos dalių funkcionavimą bei kliento ir serverio dalių komunikavimą. Taip pat buvo vykdoma vartotojo sąsajos patikra.

3.4.1 Pagrindiniai apribojimai

- Įrangos trūkumas – kliento vartotojo sąsajos patikrai vykdyti buvo naudojamas tik vienas delninis kompiuteris (Fujitsu-Siemens N560 Pocket Loox), todėl nebuvo galima numatyti sistemos elgesio ir tai kaip jos vartotojo sąsaja bus atvaizduojama kituose, panašaus tipo įrenginiuose.
- Sistemos testavimą realiomis sąlygomis apsunkino komplikuotas interneto ryšio pajungimas per USB (*Universal Serial Bus*) jungtį į delninį kompiuterį.

3.4.2 Sąsajos testavimas

Portalo realizacijai, buvo naudojamas Tapestry karkasas. Visi portalo puslapiai šiuo atveju yra sudaryti iš atskirų Tapestry komponentų, kuriuos pagrindinai sudaro HTML šablono failas ir susieta Java klasė. Šie komponentai realizuoti tokiu būdu, kad juose būtų paliktas tik

grafinės vartotojo sąsajos kodas, siekiant kuo labiau atskirti sąsają nuo funkcionalumą realizuojančios dalies.

Automatizuotai testuoti grafinę vartotojo sąsaja nėra paprasta, todėl grafinių komponentų testavimas buvo atliekamas rankiniu būdu – paprasčiausiai atvaizduojant komponentą ir stebint jį ekrane prie įvairių įmanomų vaizduojamų duomenų.

Atskirta grafinių komponentų funkcionalumo dalis randasi atskirose Java klasėse, todėl joms pritaikyta vienetų testavimo (angl.: *unit test*) strategija komponentų funkcionalumui testuoti.

Kadangi rankiniu būdu testuojant grafinę vartotojo sąsają sistemai buvo paduodamos įvairios reikšmės ir stebimas sistemos elgesys, tai tuo pačiu, kai kur dalinai buvo testuojamas ir sistemos funkcionalumas. Tačiau, pagrindinis akcentas buvo vartotojo sąsajos komponentų languose pateikiamų klaidos ir kitokių pranešimų korektiškumas.

Testuojant vartotojo sąsają kliente, buvo naudojamas analogiškas testavimo būdas.

3.4.3 Testavimo strategijos

- **Vienetų testavimas** – strategija apimanti vienetų, kurie yra mažiausios sistemos dalys, testavimą. Konkrečiu atveju šie vienetai yra Java klasės ir jų metodai.
- **Integravimo testavimas** – strategija taikoma prieš bendrą sistemos testavimą. Konkrečiu atveju taikoma apjungiant atskirus iš klasių sudarytus modulius.
- **Aukšto lygio testavimas** – produkcinėje aplinkoje vykdomas testavimas, kurio metu siekiama nustatyti, ar korektiškai veikia sistema šioje specifinėje aplinkoje.

3.4.4 Vienetų testavimas

Kaip jau buvo minėta, portalo programavimas buvo vykdomas siekiant kuo labiau atskirti grafinę sąsają nuo susieto funkcionalumo, todėl automatizuotas vienetų testavimas buvo vykdomas atskirtoms, funkcionalumą realizuojančioms (vidinė serverio logika), Java klasėms. Taip pat buvo testuojamos ir duomenų bazės modulio klases.

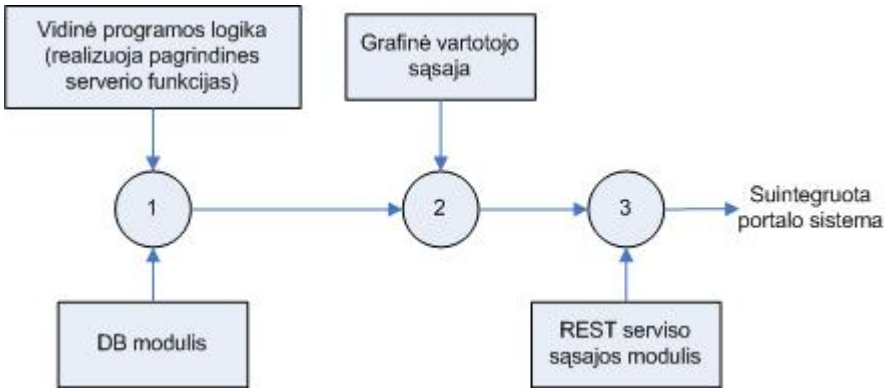
Klientinėje dalyje PĮ yra realizuota naudojant J2ME technologiją, todėl vienetų testavimas čia buvo taikomas J2ME klasėms.

Testuojant klases buvo testuojamos jų funkcijos. Programuojant buvo siekiama kur įmanoma ir prasminga bendrojo pobūdžio funkcionalumą realizuoti kaip statines funkcijas, tokiu būdu supaprastinant sistemos struktūrą ir palengvinant testavimo procedūrą. Tokios klasių funkcijos testuojamos paprasčiausiai jas iškviečiant ir lyginant jų gražinamus rezultatus. Funkcijų parametrams buvo suteikiamos įvairios ribinės ir kitos galimos reikšmės.

3.4.5 Integravimo testavimas

Portalo dalyje, integravimo testavimas buvo vykdomas grafinę vartotojo sąsają ir REST serviso sąsają apjungus su bendrai naudojamu vidinės logikos moduliu. Prieš tai, vidinės logikos modulis buvo apjungtas su nedideliu duomenų bazės moduliu. Vidinė logika realizuoja pagrindines sistemos funkcijas ir prieš tai buvo ištestuota naudojant vienetų testavimo strategiją. Tai pat, vienetų testavimas buvo pritaikytas ir duomenų bazės moduliui. Taigi integravimo testavimo metu buvo siekiama nustatyti sistemos klaidas, atsiradusias apjungiant diagramoje pavaizduotus modulius tarpusavyje.

Portalo komponentų integravimo proceso schema pateikiama žemiau.



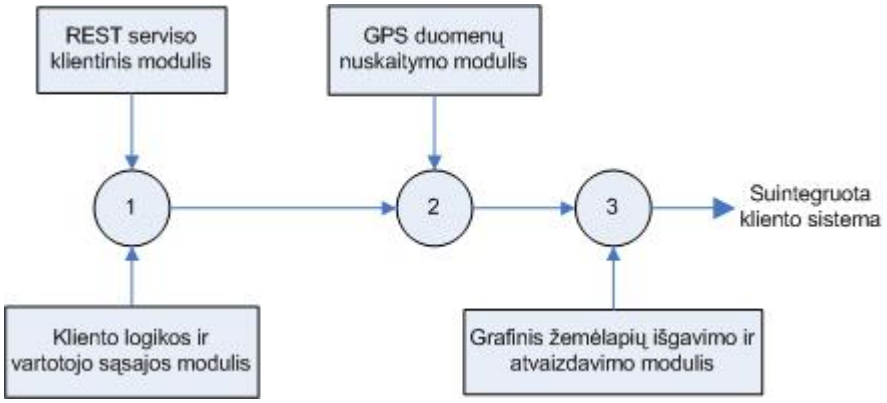
12 pav. Sistemos portalo integravimo testavimo schema

Grafike pavaizduoti sunumeruoti apskritimai atitinka atskirus integravimo testavimo etapus, kuomet palaipsniui yra suintegruojama sistema. Iš esmės, grafinės vartotojo sąsajos ir REST serviso modulio integravimo tvarka šioje schemoje jokio esminio skirtumo neturi, nes tiek grafinė portalo vartotojo sąsaja, tiek REST serviso modulis naudoja tą pačią vidinę logiką, o tarpusavyje jie jokio sąryšio neturi.

Schemoje pateiktą grafinę vartotojo sąsają sudaro sekantys pagrindiniai komponentai:

- Registracijos prie sistemos ir profilio redagavimo komponentas
- Prisijungimo prie sistemos komponentas
- Profilio peržiūros komponentas
- Profilio peržiūros/redagavimo komponentas
- Vartotojų paieškos vykdymo komponentas
- Galinčių stebėti vartotojų sąrašo komponentas
- Vartotojo stebimųjų vartotojų sąrašo komponentas
- Vartotojo išsaugotų maršrutų sąrašo komponentas
- Sistemos vartotojų sąrašo komponentas
- Besiregistruojančiųjų vartotojų sąrašo komponentas

Kliento dalyje integravimo testavimas buvo vykdomas apjungiant modulius sekančia tvarka:



13 pav. Klientinės PĮ integravimo testavimo schema

Iš tiesų, tokiam testavimui atlikti reikalingos specifinės sąlygos. Pavyzdžiui, kad iš GPS įrenginio nuskaityti duomenis, reikia, kad įrenginys būtų atviroje aplinkoje (lauke), nes uždarose patalpose GPS neveikia. REST serviso komunikacijų modulio testavimui reikalingas interneto ryšys bei veikiantis REST servisą teikiantis serveris. Šie du moduliai komplikuoja testavimo procesą, todėl jų testavimas atliekamas atskirai, o integravimo testavimo metu, juos pakeičiame atitinkamais kamščiais (angl. *stub*), kurie dirbtinai simuliuoja tikrųjų modulių veikimą.

3.4.6 Aukšto lygio testavimas

Aukšto lygio testavimo metu buvo atliekami saugumo, greitaveikos ir stresinis testavimas produkcinėje aplinkoje. Detaliau apie kiekvieną:

- Saugumo testavimas – šio testavimo metu buvo testuojamas sistemos saugumas. Buvo tikrinamos galimybės prisijungti prie sistemos su klaidingais prisijungimo duomenimis bei prieiti ir valdyti resursus, kai tam neturima reikiamų teisių.
- Greitaveika – buvo testuojama sistemos veikimo ir duomenų perdavimo sparta. Testavimo metu tikrinamas duomenų perdavimas tarp kliento ir serverio bei pozicijos duomenų atvaizdavimas žemėlapiuose.
- Stresinis testavimas – tai sistemos testavimas esant didelėms apkrovoms, pavyzdžiui, kai su sistema vienu metu dirba daug vartotojų. Sistema buvo testuota su daugiau nei 20 vienalaikių aktyvių vartotojų imituojančiais kamščiais, kurie siuntė ir priiminėjo pozicijos koordinates.

3.4.7 Testavimo rezultatų kaupimas

Testavimo metu momentiniai testavimo rezultatai buvo pateikiami ekrane bei registruojami testavimo pranešimų žurnale (angl.: *log*). Tokio žurnalo įrašo pavyzdys pateikiamas **9 lentelėje** (įrašas paimtas iš vienetų testavimo žurnalo).

9 lentelė. Testavimo rezultatų žurnalo pavyzdys

Data (YYYY-MM-DD)	Laikas (HH:MM:SS)	Testuojamo vieneto (klasės) pavadinimas	Testo pavadinimas	Sėkmingai įvykdytas	Testavimo pranešimas
2007-10-17	14:57	Database Access	DB prisijungimo testas	Taip	Testas įvykdytas sėkmingai.

3.4.8 Testavimo rezultatai

Šiame skyriuje pateikiami testavimo, vykdyto JUnit karkaso priemonėmis, rezultatai ir išvados. Pateikiame kai kurių testuotų klasių testavimo rezultatus **10 lentelėje**.

10 lentelė. Vienetų testavimo rezultatai

Testuojamo vieneto (klasės) pavadinimas	Testo pavadinimas	Testavimo atvejų kiekis	Atrasta klaidų	Testavimo pastabos
CoordinateTransformer	Ilgumos ir platumos koordinačių transformacija į planimetrinę koordinačių sistemą.	6	Ne	Testavimas vykdytas tikrinant leistinas paklaidas, susidariusias transformuojant tuos pačius koordinačių rinkinius į planimetrinę koordinačių sistemą ir atgal.
Deserializer	Transportavimo struktūrų deserializacija.	8	Taip	Testavimas vykdytas paduodant pilnus ir nepilnus XML duomenis.
Serializer	Transportavimo struktūrų serializacija.	5	Ne	Testavimas vykdytas serializuojant objektus su priskirtomis ir nepriskirtomis skirtingų atributų reikšmėmis.
PhoneNumberValidator	Telefono numerio patikra.	13	Ne	Testavimo atvejai buvo parinkti pagal nustatytus patikros apribojimus, parenkant skirtingo ilgio ir formato testavimo duomenis.

Testavimo metu buvo sudaryti automatizuoti testai, testuojantys tiek atskirus vienetus, tiek ir jau suintegruotas posistemes. Viena iš tokių posistemių yra serverio pusėje veikiantis REST servisas su susieta vidinę logiką. Šio serviso pagrindinių funkcijų testavimo rezultatai pateikti **11 lentelėje**. Pastebėsime, jog testavimas buvo vykdomas su pradinę REST serviso realizacija.

11 lentelė. REST serviso testavimo rezultatai

Testuojamo vieneto (metodo) pavadinimas	Testo pavadinimas	Testavimo atvejų kiekis	Atrasta klaidų	Testavimo pastabos
`login( username, password )`	Prisijungimo metodo testavimas.	7	Taip	Bandyta prisijungti su tais pačiais, jau prisijungusio vartotojo, ir kitais prisijungimo duomenimis.
`logout()`	Atsijungimo metodo testavimas.	2	Ne	Testuota iškviečiant šį metodą prisijungus ir neprisijungus prie sistemos.
`getMyProfile()`	Einamojo vartotojo profilio paėmimo metodo testavimas.	2	Ne	-
`getUserProfile( username )`	Vartotojo profilio paėmimo metodo testavimas.	3	Ne	Bandyta su egzistuojančių ir neegzistuojančių vartotojų vardais.

Testuojamo vieneto (metodo) pavadinimas	Testo pavadinimas	Testa-vimo atvejų kiekis	Atrasta klaidų	Testavimo pastabos
`updateMyProfile( userProfileData )`	Einamojo vartotojo profilio duomenų atnaujinimas	4	Taip	Testuota parenkant skirtingas reikšmes ir nepriskiriant būtinų laukų.
`getMyObservers()`	Einamojo vartotojo stebėtojų sąrašo paėmimo metodo testavimas.	2	Ne	-
`getMyObservable Users()`	Einamajam vartotojui galimų stebėti vartotojų sąrašo paėmimo metodo testavimas.	2	Ne	-
`addMyObserver( username )`	Stebėtojo priskyrimo einamajam vartotojui metodo testavimas.	5	Ne	Bandyta su egzistuojančių ir neegzistuojančių vartotojų vardais bei jau priskirtais.
`removeMyObserver( username )`	Einamojo vartotojo stebėtojo pašalinimo metodo testavimas.	5	Taip	Bandyta su egzistuojančių ir neegzistuojančių vartotojo stebėtojų vartotojų vardais.
`sendCoordinates( lon, lat )`	Einamojo vartotojo pozicijos duomenų metodo testavimas.	2	Ne	-
`getUsers Coordinates( usernames[] )`	Einamojo vartotojo stebimų vartotojų pozicijos duomenų paėmimo metodo testavimas.	5	Taip	Testuota paduodant vartotojui leistinų ir neleistinų stebėti vartotojų vardai.
`searchUsers( query, from, Count )`	Vartotojų paieškos metodo testavimas.	7	Ne	Testuota su įvairiomis paieškos užklausomis bei leistinomis ir neleistinomis parametrų *from* ir *count* reikšmėmis.
`saveMyRoute( routeData )`	Einamojo vartotojo maršruto išsaugojimo metodo testavimas.	5	Taip	Testuota su maršrutais, turinčiais daugiau ir mažiau nei 2 poros maršruto koordinačių.
`getMyRouteList()`	Einamojo vartotojo maršrutų sąrašo paėmimo metodo testavimas.	2	Ne	-
`getMyRoute( routeId )`	Einamojo vartotojo konkretaus maršruto paėmimo metodo testavimas.	4	Ne	Testuota su vartotojui priklausančių ir nepriklausančių maršrutų identifikatoriais.
updateMyRoute(datXML)	Maršruto metaduomenų atnaujinimo metodo testavimas.	4	Ne	Testuota parenkant skirtingas reikšmes ir nepriskiriant būtinų laukų.
removeMyRoute(routeId)	Maršruto pašalinimo metodo testavimas.	5	Ne	Bandyta su egzistuojančių ir neegzistuojančių maršruto identifikatoriais.
getUserPublishedRouteList (username)	Paviešinto vartotojo maršrutų sąrašo nuskaitymo metodo	2	Ne	Bandyta su egzistuojančiais ir neegzistuojančiais vartotojų vardais.

Testuojamo vieneto (metodo) pavadinimas	Testo pavadinimas	Testa-vimo atvejų kiekis	Atrasta klaidų	Testavimo pastabos
	testavimas.			
getUserPublishedRoute(username, routeId)	Paviešinto vartotojo maršruto duomenų paėmimas.	6	Ne	Bandyta su egzistuojančiais ir neegzistuojančiais vartotojų vardais ir maršruto identifikatoriais.
publishMyRoute(routeId, flag)	Savo maršruto paviešinimo metodo testavimas.	5	Ne	Bandyta su neegzistuojančiais maršruto identifikatoriais.

Visi aukščiau pateikti REST serviso metodai taip pat buvo testuoti prisijungus ir neprisijungus prie sistemos. Tai atsispindi testavimo atvejų kiekyje – pridėtas papildomas testavimo atvejis.

Šios pagrindinės REST funkcijos taip pat nusako funkcionalumą, prieinamą portalo vartotojo srityje. Svarbu pastebėti, jog testuojant REST servisą, taip pat dalinai buvo ištestuotas portale teikiamas funkcionalumas, nes tiek servisas, tiek ir portalas naudoja bendrą logiką.

Portalo testavimo metu paaiškėjo vienas svarbus sistemos trūkumas, kai paspaudus naršyklės atnaujinimo (angl.: *refresh*) mygtuką, tam tikrais atvejais yra atkartojama prieš tai vykdyta operacija. Dėl to atitinkamai duomenų bazėje buvo gaunami duomenų dublikatai. Šis ir visi kiti išaiškėję sistemos trūkumai buvo sėkmingai ištaisyti vėlesniuose sistemos tobulinimo etapuose.

4. REST serviso įgyvendinimo tyrimas

4.1 Galimos įgyvendinimo alternatyvos

Pradinis sistemos serviso variantas yra realizuotas klasikiniu būdu – panaudojant SOAP servisą. Šio tipo serviso panaudojimas yra priimtinas paprastoms funkcijoms įgyvendinti. Paprastomis funkcijomis čia laikome funkcijas, kurioms pilnai įvykdyti pakanka vieno ar vos kelių kreipinių į serverį ir kurios vykdomos santykinai retai. Tokių funkcijų realizuotos sistemos kontekste pavyzdžiai: vartotojo profilio atnaujinimas, maršruto išsaugojimas, vartotojų paieškos atlikimas ir pan. Tačiau realizuotoje sistemoje, be šio tipinio funkcionalumo, yra tik šiai aplinkai būdingas funkcionalumas, kuriam SOAP serviso panaudojimas yra neefektyvus sprendimas. Specifinis funkcionalumas yra vartotojų pozicijos koordinačių mainai su serveriu realiu laiku. Šis pozicijos duomenų perdavimas ir paėmimas yra vykdomas periodiškai ir santykinai dažnai, pavyzdžiui kas 1 ar 2 sekundes. Tai yra būtina, jei pageidaujama matyti vartotojų poziciją realiu laiku. Šiuo aspektu SOAP servisai tampa neefektyvūs dėl dviejų pagrindinių priežasčių: santykinai lėto veikimo ir didelio persiunčiamų duomenų kiekio.

Klientas veikia delniniame kompiuteryje, kuris kaip taisyklė dažniausiai turi GPRS interneto prieigą, todėl persiunčiamų duomenų kiekis yra labai svarbus akcentas. Reikalingas alternatyvus sprendimas, galintis išspręsti šią problemą. Vienas iš tokių sprendimų yra naudoti paprasčiausius, programavimo aplinkos teikiamus, žemiausio lygmens – TCP/IP protokolo *Socket* (http://www.devarticles.com/c/a/Java/Socket-Programming-in-Java) tipo susijungimus. Tačiau, tai komplikuotu realizaciją, nes šiuo atveju reikėtų sukurti specifinį žinučių protokolą pozicijos koordinačių perdavimui. Iš tiesų, toks *ad-hoc* protokolas taptų dar sudėtingesnis kai reikėtų įgyvendinti visą likusį sistemos teikiamą funkcionalumą. Taigi *Socket* panaudojimas tampa neparankus ir šiek tiek komplikuotas sprendimas.

Palankiausias variantas yra naudoti REST metodiką. Ši metodika yra tinkamiausia, nes jos pagrindu sudaryti servisai yra labai paprasti ir efektyvūs. Be to, tokio serviso įgyvendinimo procesas yra paprastas ir nereikalauja daug laiko ir pastangų. Lyginant REST su *Socket*, pastarasis variantas yra efektyvesnis, nes REST panaudoja HTTP protokolą ir jo savybes, kuris įveda papildomą žinučių sluoksnį. Tačiau šis HTTP sluoksnis tuo pačiu yra ir privalumas, nes jo pagrindu yra įgyvendintas visas pasaulinis žiniatinklis ir nebereikia šio sluoksnio realizuoti specifiniam atvejui. Be to, efektyvumo požiūriu REST nedaug skiriasi nuo *Socket* atvejo, tuo tarpu skirtumas lyginant su SOAP servisais yra didelis. REST servisas gali būti realizuotas taip,

kad jis pilnai pakeistų SOAP servisą neįvesdamas jokių esminių trūkumų. Būtent REST principais grįstą sprendimą ir pasirinkome sistemos patobulinimui atlikti.

4.2 Pradinė REST serviso realizacija

Pradinė sistemos REST serviso realizacija, veikianti sistemos serverio dalyje, pagrįsta labai paprastu principu. Ji naudoja HTTP protokolą ir neturi jokio papildomo pranešimų perdavimo sluoksnio. Pranešimai yra perduodami tiesiog per HTTP GET parametrus. Kliento siunčiamas pranešimas serveriui, arba tiesiog HTTP užklausa, iškviečia tam tikrą REST serviso metodą, kuris gali turėti parametrus arba ne. Tokios užklausos rezultatas yra HTTP atsakas – struktūrizuotas XML srautas.

Komunikuojant klientui ir serveriui yra būtina identifikuoti, kuris vartotojas vykdo užklausą, todėl yra būtinas identifikacijos mechanizmas. Šiuo atveju jis realizuotas *ad-hoc* principu. Pirmą kartą jungiantis vartotojui yra vykdoma autentifikacija ir grąžinamas SHA-1 (http://www.itl.nist.gov/fipspubs/fip180-1.htm) algoritmu sugeneruotas kodas. Šis kodas naudojamas kaip sesijos identifikatorius prisijungusiam vartotojui identifikuoti ir yra pridedamas prie užklausos parametrų kiekvieną kartą klientui kviečiant kitus REST serviso metodus.

Prisijungimo nuorodos, kuria vykdoma užklausa, pavyzdys:

- http://myserver.com/RESTService?method=login&username=myUsername&password=myPassword

Tokios užklausos galimas atsakas XML formatu atrodo taip:

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
  <data>true</data>
  <sessionToken>465bf07ec6e9d588ecfdfff0e5e175b266f45a00</sessionToken>
</root>
```

4.3 Patobulinta REST serviso realizacija

Pradinė REST serviso realizacija yra pilnai funkcionali, efektyvi ir puikiai pritaikoma realizuotoje sistemoje. Tačiau ji netenkina pagrindinių, REST architektūros reikalavimų (skyrius **2.5 Apibendrinimas**). Todėl, sistemoje buvo realizuota alternatyvi REST serviso realizacija, kuri atitinka pateiktus reikalavimus. Toliau pateiksime šios alternatyvos aprašymą.

REST architektūroje negali būti saugoma jokia su klientu susijusi būsena. Pradinėje realizacijoje yra realizuotas autentifikacijos mechanizmas, kuris kiekvieną kartą prisijungus

vartotojui, sugeneruoja SHA-1 identifikatorių ir naudoja prisijungusių vartotojų identifikavimui, kiekvienos atėjusios užklausos metu. Natūralu, jog šis identifikatorius turi būti saugomas atmintyje, o tai jau yra sesijos būsena serverio pusėje. Kad pašalinti šį neatitikimą, autentifikacijos procesą turime vykdyti kiekvienos užklausos metu iš naujo. Todėl, prie kiekvienos užklausos parametrų pridedame du papildomus – vartotojo vardo ir slaptažodžio parametrus: *username* ir *password*. Šis sprendimas akivaizdžiai sumažina sistemos efektyvumą, nes kiekvienos užklausos metu vartotojo prisijungimo duomenys turi būti imami iš duomenų bazės. Todėl, kad išlaikyti REST reikalavimą ir neprarasti efektyvumo, realizacija pakeičiama taip, kad visų vartotojų prisijungimo duomenys būtų saugomi ne tik duomenų bazėje, bet ir užkraunami operatyviojoje atmintyje. Žinoma, jei tokių vartotojų kiekis bus didelis, tai reikalaus nemažai atminties, todėl tokiu atveju tektų grįžti prie pradinės REST serviso realizacijos.

Dar vienas patobulinimas yra redagavimo operacijų realizavimas išnaudojant HTTP POST metodą. Pradinėje realizacijoje, kiekviena operacija įgyvendinta HTTP GET metodu. Tačiau tai nėra geras sprendimas, nes operacijos, kurios redaguoja duomenis, gali būti įvykdomos paprasčiausiai iškviečiant paprastą nuorodą su parametrais. Jei tokia nuoroda taptų prieinama viešai, atsivertų galimybė bet kam redaguoti ar kaip nors kitaip keisti sistemoje saugomus duomenis, paprasčiausiai pasinaudojant naršykle. Todėl, HTTP GET metodo taikymas paliekamas tik duomenų nuskaitymo operacijai vykdyti, o visos duomenis keičiančios operacijos, realizuojamos HTTP POST metodu.

REST architektūroje teigiama, jog ji yra grįsta resursais, o ne veiklomis. Todėl, pradinėje realizacijoje pakeičiame kreipinio formatą į serverį taip, kad neliktų metodų pavadinimų, o atsirastų resursų pavadinimai. Žemiau pateikiame lentelę, kurioje matyti kaip įvedamas REST resursų panaudojimas vietoje metodų pavadinimų.

12 lentelė. REST serviso patobulinimas

Metodas	Resursas ir parametrai	HTTP metodas	Pastabos
`login(username, password)`	-	GET ir POST	Autentifikacija realizuojama panaudojant papildomus *username* ir *password* parametrus kiekvienos užklausos metu. Parametrai pridedami tiek GET tiek ir POST atvejais.
`logout()`	-	-	Kadangi būsena nesaugoma, atsijungimo metodas nebetenka prasmės.
`sendCoordinates( longitude, latitude)`	`/coordinates? longitude=val1 &laitutde=val2`	POST	Savo pozicijos koordinačių siuntimo atveju naudojame POST.
`getUsersCoordinates( usernames[])`	`/coordinates? users=val1`	GET	Kitų vartotojų koordinačių paėmimo atveju naudojame GET. Parametras

			users nurodo stebimų vartotojų vardų sąrašą.
`getMyProfile()`	`/profile?user=val1`	GET	Nuskaitant savo profilio informacija *user* parametro nurodyti nereikia, nes naudojamas *username* parametras pateikiamas kartu su autentifikacijos duomenimis.
`getUserProfile(username)`			
`updateMyProfile(profileDataXML)`	`/profile?profileData=val1`	POST	Atnaujinant savo profilio informaciją, paduodamas atnaujinamų duomenų profileData parametras.
`searchUsers(searchQuery, from, count)`	`/profile?searchQuery=val1&from=val2&count=val3`	GET	Vartotojų paieška. *searchQuery* – paieškos užklausa, *from* – rezultatai nuo, *count* – rezultatų kiekis.
`getMyObservers()`	`/observers`	GET	Vartotojo stebėtojų sąrašas.
`addMyObserver(username)`	`/observers?addUser=val1`	POST	Stebėtojo pridėjimas. *addUser* nurodo pridedamo vartotojo vardą.
`removeMyObserver(username)`	`/observers?removeUser=val1`	POST	Stebėtojo pašalinimas. *removeUser* nurodo šalinamo vartotojo vardą.
`getMyObservableUsers()`	`/observables`	GET	Vartotojui galimų stebėti vartotojų sąrašas.
`getMyRouteList()`	`/routes`	GET	Vartotojo maršrutų sąrašas.
`getMyRoute(routeId)`	`/routes?routeId=val1`	GET	Pilnas konkretaus vartotojo maršruto duomenų nuskaitymas.
`saveMyRoute(datXML)`	`/routes?saveRoute=val1`	POST	Vartotojo maršruto išsaugojimas. Parametras *saveRoute* pateikia XML maršruto duomenis.
`updateMyRoute(datXML)`	`/routes?routeId=val1&routeMetadata=val2`	POST	Konkretaus maršruto metaduomenų atnaujinimas.
`removeMyRoute(routeId)`	`/routes?removeRouteId=va1`	POST	Konkretaus vartotojo maršruto pašalinimas.
`publishMyRoute(routeId, flag)`	`/routes?routeId=val1&publish=val2`	POST	Vartotojo maršruto paviešinimas. *publish* nurodo ar paviešinamas maršrutas ar uždraudžiama prieiga prie jo.
`getUserPublishedRouteList(username)`	`/routes?user=val1`	GET	Kito vartotojo paviešinti maršrutai.
`getUserPublishedRoute(username, routeId)`	`/routes?user=val1&routeId=val2`	GET	Kito vartotojo pilnas paviešinto maršruto duomenų nuskaitymas.

Pastaba: kiekvienos užklausos į resursą metu, pridedami *username* ir *password* parametrai, kurie lentelėje atskirai neakcentuojami.

4.4 REST panaudojimas žemėlapiams nuskaityti

REST principų panaudojimas geografinių žemėlapių pateikimui internetu yra viena iš naujai besiformuojančių metodikų [1]. Realizuotos sistemos atveju, klientas žemėlapius nuskaito iš žemėlapių saugyklos, prie kurios prieiga teikiama per REST servisą. Kiekvienas žemėlapio

fragmentas išrenkamas pagal jį apibrėžiančias x, y koordinates ir z priartinimo lygį. Šis parametrų trejetas paprasčiausiai nurodomas nuorodoje kreipiantis į servisą.

Tokios nuorodos pavyzdys:

- http://myserver.com/MAPService?x=1160&y=648&z=6

Užklausos pagal šią nuorodą rezultatas yra reikiamas žemėlapio fragmentas.

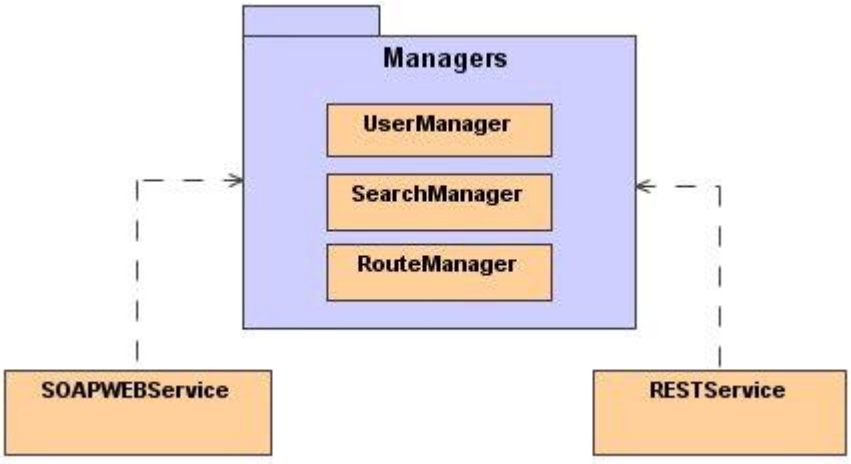
5. REST ir SOAP paslaugų efektyvumas

5.1 Eksperimento objektas sistemoje

Realizuotoje pozicionavimo sistemoje svarbiausia dalis yra realaus laiko vartotojo pozicijos koordinačių perdavimas iš kliento į serverį ir kitų vartotojų analogiškų koordinačių paėmimas iš serverio. Pozicijos duomenų išsiuntimą ir paėmimą visada inicijuoja klientas. Šių duomenų mainų procedūrų efektyvumas stipriai priklauso nuo tarp kliento ir serverio naudojamos duomenų apsikeitimo metodikos. Tokiam duomenų perdavimui yra realizuotos dvi skirtingos komunikavimo sąsajos – REST servisas ir SOAP servisas. Abi sąsajos teikia analogišką funkcionalumą ir galimybes, tačiau jų efektyvumas skiriasi. Kad pagrįsti šiuos skirtumus ir nustatyti optimaliausią, šiai sistemai labiausiai tinkamą variantą, eksperimentinėje dalyje palyginsime šiuos realizuotus servisus greitaveikos ir persiunčiamų duomenų kiekio požiūriu ir apibendrinsime išvadomis.

5.2 Eksperimento procedūra ir metrikos

Pirmiausia pastebėsime, jog tiek SOAP servisas tiek ir REST servisas naudojasi viena ir ta pačia vidine programinę logiką ir neturi jokio specifinio pridėtinio kodo, kuris galėtų vienaip ar kitaip įtakoti šių servisų efektyvumą. Kitaip tariant, abu servisai naudojasi tais pačiais programiniais komponentais tam pačiam funkcionalumui realizuoti. Šių servisų priklausomybę nuo vidinės logikos galima pavaizduoti **14 pav.** pateikiamoje diagramoje.



14 pav. Bendros vidinės logikos panaudojimas servisų realizacijoje

Tai pat, labai svarbu akcentuoti, jog SOAP servisas yra realizuotas panaudojant Axis 1.4v programinę biblioteką, kai tuo tarpu REST serviso realizacijai nereikalingos jokios papildomos programinės priemonės. Dar kartą pabrėšime, jog abu servisai realizuoti Java programavimo kalba. Abiejų servisų efektyvumo nustatymo eksperimentą atliksime panaudodami papildomus, šiems servisams Java kalba realizuotus klientus.

Eksperimento metu, bus vykdomos sekančios veiksmų sekos:

1. Pradinių pozicijos koordinačių išsiuntimas sulaukiant atsakymo.
2. Kito vartotojo pozicijos koordinačių užklausimas ir paėmimas.

Kadangi SOAP serviso atveju, būtina vykdyti prisijungimą, jis bus įvykdomas tik vieną kartą. Abiejų servisų atveju, pozicijos koordinačių išsiuntimo ir paėmimo užklausos bus kartojamos n = 1000 kartų (ciklų). Šis skaičius parinktas didesnis, kad išryškinti naudojamų metodų efektyvumo skirtumus.

Efektyvumo tyrimus atliksime vykdydami realizuotus servisų klientus ant to pačio lokalaus kompiuterio, kur yra įdiegti minėti servisai. Taip pat tyrimą pakartosime dviejuose, skirtingus parametrus turinčiuose, kompiuteriuose. Tai atliksime tam, kad parodyti efektyvumo priklausomybę nuo techninių resursų ir akcentuoti išlaikytą arba neišlaikytą efektyvumo santykį tarp skirtingų servisų.

Kompiuterių, su kuriais bus atliekamas tyrimas, konfigūracijos pateikiamos **13 lentelėje**.

13 lentelė. Eksperimentui naudojamų kompiuterių konfigūracijos

	1 konfigūracija	**2 konfigūracija**
Procesorius	Pentium 4, 1.7 GHz, FSB 400 MHz	Pentium 4, 3.0 GHz, FSB 800 MHz
Operatyvioji atmintis	1GB, DDR-400	4 GB, DDR2-800

Tyrimo metu bus naudojamos šios pagrindinės metrikos:

- laikas sekundėmis, per kurį bus įvykdytas pozicijos koordinačių išsiuntimas ir kito vartotojo koordinačių paėmimas (kartojama n = 1000 ciklų);
- perduotų duomenų kiekis megabaitais (kartojama n = 1000 ciklų).

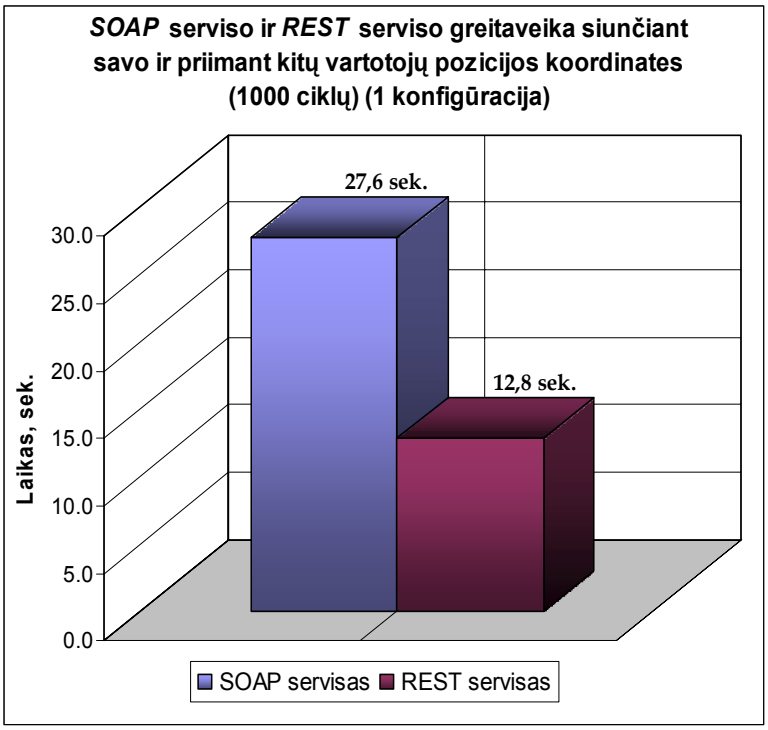
Iš tiesų, realiomis sistemos veikimo sąlygomis, vartotojų pozicijos duomenys yra siunčiami ir nuskaitomi kas tam tikrą, laisvai pasirenkamą laiko intervalą, pavyzdžiui 1 sekundę.

Tačiau efektyvumo parametrams apskaičiuoti šis „miegojimo“ laiko intervalas yra panaikintas, kad išryškinti naudojamų servisų greitaveikos skirtumus.

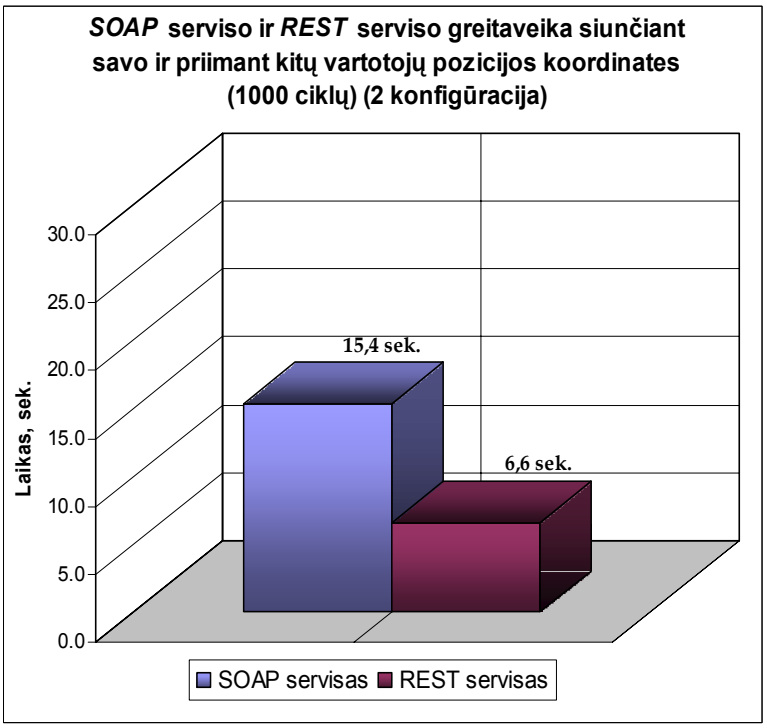
Šiame eksperimente taip pat apytikriai pamatuosime minėtų servisų persiunčiamų duomenų kiekius. Persiunčiamu duomenų kiekis preliminariai bus skaičiuojamas neįvertinant HTTP protokolo pranešimų. Tačiau šių pranešimų užimamas duomenų kiekis yra labai nedidelis ir apytikriai lygus tiek SOAP tiek ir REST serviso atveju, todėl galutiniam įvertinimui didelės įtakos neturi.

5.3 Eksperimento rezultatai

Toliau pateikiame servisų greitaveikos eksperimentų rezultatus.



15 pav. SOAP ir REST servisų greitaveika siunčiant savo ir priimant kitų vartotojų pozicijos koordinates (1000 ciklų) (1 konfigūracija)



16 pav. SOAP ir REST serviso greitaveika siunčiant savo ir priimant kitų vartotojų pozicijos koordinates (1000 ciklų) (2 konfigūracija)

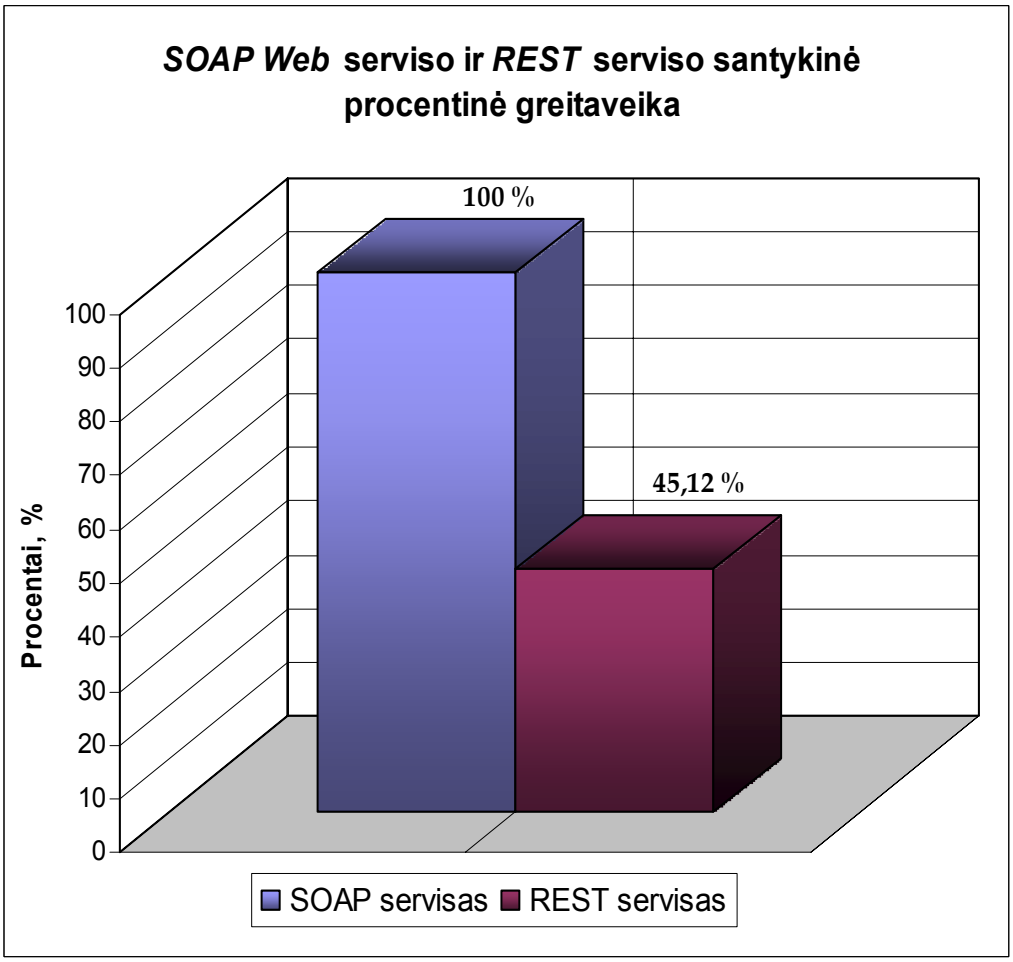
Kaip matyti iš diagramų, kompiuterio parametrai turi didžiulę įtaką servisų greitaveikai. Eksperimentą atlikus ant galingesnio kompiuterio, servisų greitaveika pagerėjo bene du kartus. Tačiau svarbiausia pastebėti, jog pasikeitus greitaveikai, santykis tarp servisų praktiškai išliko toks pat:

$$\frac{27,6}{12,8} \approx \frac{15,4}{6,6} \Rightarrow 2,156 \approx 2,333$$

Paskaičiuosime šių servisų greitaveikos procentinį santykį, tačiau prieš tai, priklausomybei nuo techninės įrangos sumažinti, išvesime iš šių dviejų skirtingų konfigūracijų gautų rezultatų vidurkį:

$$\frac{27,6+15,4}{2} = 21.5 \qquad \frac{12,8+6,6}{2} = 9,7$$

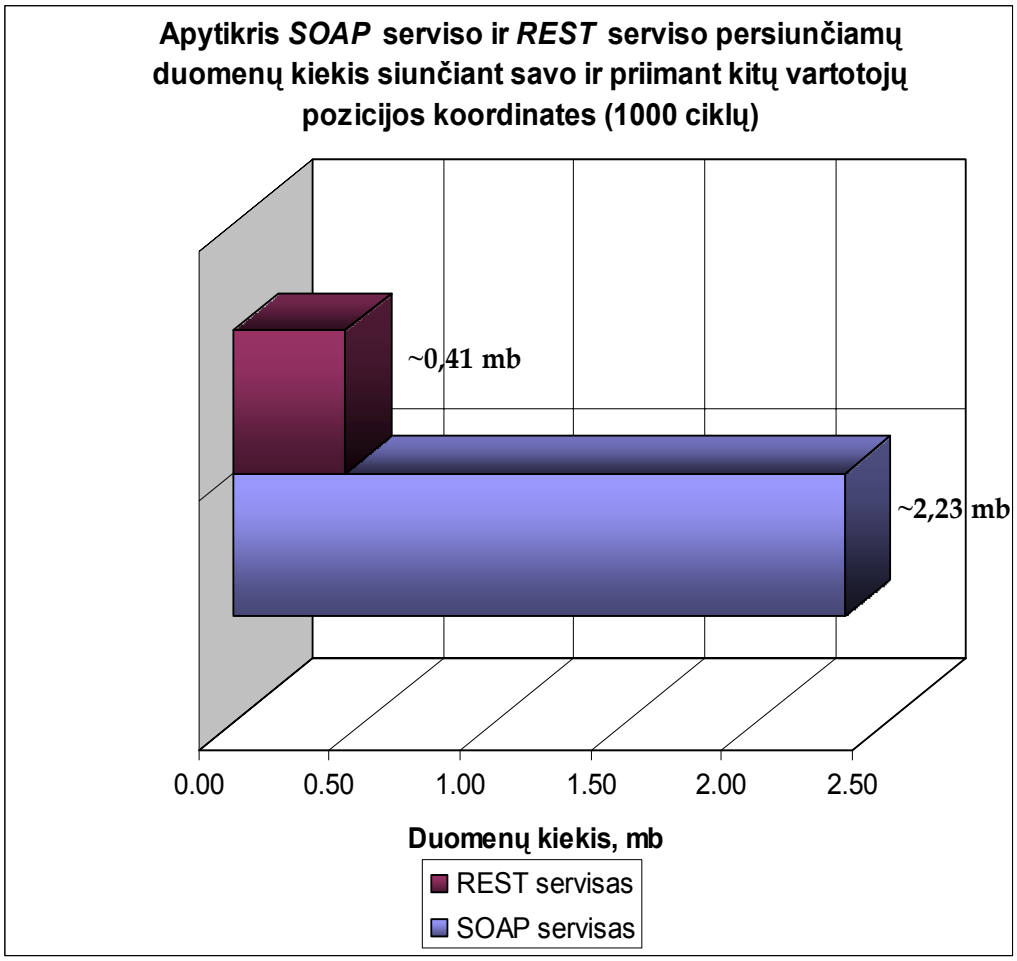
Pateikiame procentinį servisų efektyvumo santykį:



17 pav. SOAP ir REST serviso santykinė procentinė greitaveika

Kaip matyti iš diagramos, REST servisas veikia daugiau nei per pus greičiau, lyginant su analogišką funkcionalumą realizuojančiu SOAP servisu.

Toliau pateikiame šių servisų palyginimą persiunčiamų duomenų kiekio atžvilgiu.



18 pav. SOAP ir REST servisų persiunčiamų duomenų kiekis siunčiant savo ir priimant kitų vartotojų pozicijos koordinates (1000 ciklų)

Kaip matyti, REST serviso atveju išsiunčiant ir paimant pozicijos koordinates, iš viso persiunčiama net penkis kartus mažiau duomenų lyginant su SOAP servisu variantu.

5.4 Išvados

Atlikus eksperimentą, tampa akivaizdus REST serviso pranašumas. Greitaveikos požiūriu jis pralenkia SOAP servisą du kartus, t.y. veikia du kartus greičiau. Persiunčiamų REST serviso duomenų kiekis, išsiunčiant ir paimant vartotojų pozicijos koordinates, yra daugiau nei penkis kartus mažesnis. Toks mažas persiunčiamų duomenų kiekis šios sistemos kontekste yra labai svarbus akcentas, nes delniniuose kompiuteriuose dažniausiai yra naudojamas GPRS (*General Packet Radio Service*) internetas, kuomet yra apmokestinamas kiekvienas persiųstas megabaitas.

6. Išvados

- Literatūros šaltinių ir egzistuojančių sprendimų analizės metu ištirti galimi problemos sprendimo variantai ir išskirti esminiai reikalavimai praktinėms REST servisų realizacijoms.
    - SOAP pagrindu grįsti servisai šiandieną yra populiariausi, tačiau yra pakankamai sudėtingi ir neefektyvūs.
    - REST servisai yra paprastesnė ir efektyvesnė SOAP servisų alternatyva, be to ji remiasi senai žiniatinklyje nusistovėjusiomis koncepcijomis ir siekia išnaudoti HTTP protokolo galimybes.
    - REST nėra standartas ir neapibrėžia griežtų reikalavimų šią architektūrą įgyvendinančioms realizacijoms, todėl kūrėjai turi nemažai laisvės įgyvendinant tokio tipo servisus.
- Realizuota daugiavartotojiška pozicionavimo sistema teikia jos vartotojams galimybę stebėti savo ir kitų, sutikimą davusių, vartotojų poziciją realiame laike, panaudojant mobilųjį įrenginį su įmontuotu GPS imtuvu.
    - GSM pozicijos nustatymo metodai labai praverčia stebimiems objektams esant uždarose patalpose, tačiau realizuotoje sistemoje naudojama didelį tikslumą turinti GPS technologija yra nepakeičiama lauko sąlygomis.
    - Iš GPS įrenginio gaunamos pozicijos koordinatės priklauso sferinės koordinačių sistemos atmainai, o žemėlapiai – stačiakampei, todėl, pozicijos taško atvaizdavimui ant žemėlapio atlikti, būtina pozicijos koordinates transformuoti į stačiakampę koordinačių sistemą panaudojant atvirkštinę Gudermano (*Gudermannian*) funkciją.
    - Mercator projekcijos pagrindu sudaryto žemėlapio mastelis yra stipriai iškraipytas, ypač tolstant nuo pusiaujo link ašigalių. Tačiau ši projekcija puikiai tinkama interaktyvių internetinių žemėlapių sudarymui, kuriais patogu manipuliuoti, todėl neretai yra taikoma geografinėse informacinėse sistemose.
    - Taupant mažus delninio kompiuterio resursus ir siekiant sumažinti persiunčiamų duomenų kiekį, tikslinga naudoti REST servisus, o naujai atsiųstus žemėlapių fragmentus – kešuoti.

- Pradinis realizuotas REST serviso variantas yra funkcionalus ir tinkamas panaudoti realizuotos pozicionavimo sistemos kontekste. Tačiau, jis pilnai netenkina pagrindinių REST architektūros reikalavimų, todėl buvo patobulintas, kad atitiktų šiuos reikalavimus.
- Eksperimento metu buvo įrodyta, jog REST serviso panaudojimas daugiavartotojiškos pozicionavimo sistemos kontekste yra žymiai efektyvesnis sprendimas nei SOAP serviso variantas.

7. Literatūra

[1]. **C. J. Andrews**. Emerging Technology: Geospatial Web Services and REST. *Autodesk, Inc.*, August 02, 2007. [žiūrėta 2008-05-10]. Prieiga per internetą: http://www.directionsmag.com/article.php?article_id=2515&trv=1.

[2]. **R. Battle, E. Benson**. Bridging the semantic Web and Web 2.0 with Representational State Transfer (REST). *BBN Technologies*, published on ScienceDirect, June 02, 2007.

[3]. **J. Brainerd, A. Pang.** Floating ring: a new tool for visualizing distortion in map projections. *Computer Graphics International*. Proceedings 22-26 June, 1998, p. 466 – 480. [žiūrėta 2008-02-20]. Prieiga per internetą: http://ieeexplore.ieee.org/iel4/5647/15133/00694300.pdf?tp=&isnumber=15133&arnumber=694300.

[4]. **T. Berners-Lee, R. Fielding, L. Masinter.** Uniform Resource Identifiers (URI): Generic Syntax. *MIT/LCS, U.C. Irvine, Xerox Corporation*, August 1998. [žiūrėta 2008-05-15]. Prieiga per internetą: http://www.ietf.org/rfc/rfc2396.txt.

[5]. **R. L. Costello.** Building Web Services the REST Way. *xFront*. [žiūrėta 2008-04-23]. Prieiga per internetą: http://www.xfront.com/REST-Web-Services.html.

[6]. **F. Curbera, M. Duftler, R. Khalaf, W. Nagy, N. Mukhi, S. Weerawarana**. Unraveling the Web Services Web. *IBM T.J.Watson Research Center*. March/April 2002. [žiūrėta 2008-04-15]. Prieiga per internetą: http://ieeexplore.ieee.org/iel5/4236/21386/00991449.pdf?tp=&isnumber=21386&arnumber=991449.

[7]. **D. Dao, C. Rizos, G. Wang**. Location-based service: Technical and business issues. *GPS solutions,* 2002, Vol. 6(3), p. 169-178.

[8]. **S. Dissanaike, P. Wijkman, M. Wijkman**. Utilizing XML-RPC or SOAP on an Embedded System. *Stockholm University/Royal Institute of Technology*, 2004. [žiūrėta 2008-04-15]. Prieiga per internetą: http://ieeexplore.ieee.org/iel5/9027/28651/01284068.pdf?tp=&isnumber=28651&arnumber=1284068.

[9]. **R.T. Fielding**. Architectural Styles and the Design of Network-Based Software Architectures. Doctoral dissertation, *Dept. of Computer Science, Univ. of Calif.*, Irvine, 2000. [žiūrėta 2008-04-12]. Prieiga per internetą: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

[10]. **G. Goth.** Critics Say Web Services Need a REST. *IEEE Distributed systems online, IEEE Computing Society*, Dec. 2004, Vol. 5(12). [žiūrėta 2008-05-16]. Prieiga per internetą: http://ieeexplore.ieee.org/iel5/8968/30113/01381275.pdf?tp=&isnumber=&arnumber=1381275.

[11]. **M. Kennedy, S. Kopp.** Understanding Map Projections. *Environmental Systems Research Institute, Inc.*, 2000. [žiūrėta 2008-02-20]. Prieiga per internetą: http://www.kartografie.nl/geometrics/Map%20projections/Understanding%20Map%20Projections.pdf.

[12]. **V. Liutkauskas, I. Lagzdinytė**. GSM tinklo mobilios vietos patikslinimo būdai. *Elektronika ir elektrotechnika*, *KTU,* 2004, Nr. 5, p. 141-146.

[13]. **V. Palilionis**. GIS ir mobiliųjų technologijų integravimo lokalizuotųjų paslaugų sistemose. *Geodezija ir kartografija*, *VGTU,* 2004, XXX t., Nr. 4, p. 117-122.

[14]. **J. Snell**. Resource-oriented vs. activity-oriented Web services. *IBM*, October 12, 2004. [žiūrėta 2008-03-15]. Prieiga per internetą: http://www.ibm.com/developerworks/xml/library/ws-restvsoap/.

[15]. **S. Tilkov.** Addressing Doubts about REST. *InfoQ's SOA community*. March 13, 2008. [žiūrėta 2008-04-15]. Prieiga per internetą: http://www.infoq.com/articles/tilkov-rest-doubts.

[16]. **S. Vinoski**. Demystifying RESTful Data Coupling. *IEEE Internet Computing online*, *IEEE Computing Society,* Mar.-Apr. 2008, Vol. 12(2), p 87-90. . [žiūrėta 2008-05-15]. Prieiga per internetą: http://ieeexplore.ieee.org/iel5/4236/4463372/04463391.pdf?tp=&isnumber=4463372&arnumber=4463391.

[17]. **S. Vinoski**. REST eye for the SOA Guy. *IEEE Internet Computing online, IEEE Computing Society*, Jan.-Feb. 2007, Vol. 11(1), p. 82-84. [žiūrėta 2008-04-20]. Prieiga per internetą: http://ieeexplore.ieee.org/iel5/4236/4061105/04061127.pdf?tp=&isnumber=4061105&arnumber=4061127.

[18]. **E. W. Weisstein.** Inverse Gudermannian. *MathWorld-A Wolfram Web Resource*. [žiūrėta 2008-03-11]. Prieiga per internetą: http://mathworld.wolfram.com/InverseGudermannian.html.

[19]. **E. W. Weisstein.** Mercator Projection. *MathWorld-A Wolfram Web Resource*. [žiūrėta 2008-03-11]. Prieiga per internetą: http://mathworld.wolfram.com/MercatorProjection.html.

8. Terminų ir santrumpų žodynas

Terminas	Paaiškinimas
JVM	Java programavimo kalbos virtualioji mašina, kuri interpretuoja ir vykdo Java kompiliatoriumi sukompiliuota kodą (*Java Virtual Machine*).
J2ME	Tai specifikacija, apibrėžianti programavimo priemonių rinkinį, paremtą Java programavimo kalba ir skirtą programinės įrangos, orientuotos į mobiliuosius įrenginius, kūrimui (*Java 2 Platform Micro Edition*). Ji apima CLDC (Connected Limited Device Configuration) ir MIDP (Mobile Information Device Profile) specifikacijas.
PDA	Tai portabilus, dažniausiai delne telpantis, kompiuteris (*Personal digital assistant*).
GPS	Globali pozicionavimo sistema sudaryta iš 24 palydovų tinklo, kurį įsteigė JAV gynybos departamentas (*Global Positioning System*).
GSM	Globalus mobilių telefonų ryšio standartas (*Global System for Mobile communications*). Yra labiausiai paplitęs mobiliųjų telefonų ryšio standartas pasaulyje.
ad-hoc	Žodis nusakantis specifinį problemos sprendimą sudarytą konkrečiam uždaviniui spręsti.
XML	Išplėstinė žymių kalba – duomenų formatas struktūrizuotų dokumentų keitimuisi tinkle (*eXtensible Markup Language*).
SOAP	SOAP yra XML pagrindu sudarytas protokolas, skirtas informacijos apsikeitimui tarp programinės įrangos sistemų (*Simple Object Access Protocol*).
REST	Tai tam tikra programinė architektūra, skirta paskirstytoms hiperžiniasklaidos sistemoms, tokioms kaip žiniatinklis (*Representational state transfer*).
URI	Tai kompaktiška simbolių eilutė – identifikatorius (*Uniform Resource Identifier*). Žiniatinklyje tai yra paprasčiausia nuoroda nurodanti tam tikrą resursą.
XML-RPC	Tai programinėms sistemoms skirtas, nuotolinių procedūrų vykdymo protokolas. Dažnai naudojamas kartu su SOAP protokolu.
WSDL	Tai žiniatinklio servisų apibūdinimo kalba, nusakanti kaip reikia naudotis konkrečiu žiniatinklio servisu.(*Web Services Description Language*). Dažniausiai naudojama SOAP servisams apibūdinti.
WADL	Tai taip pat žiniatinklio servisų apibūdinimo kalba (*Web Application Description Language*), tačiau mažiau populiari ir paprastesnė nei WSDL.
RSS	Tai XML duomenų formatas, naudojamas dažnai besikeičiančiai informacijai viešinti (*Really Simple Syndication*).
API	Tai programinės sistemos teikiama programinė sąsaja, kuri skirta komunikacijai su kitomis išorinėmis programomis vykdyti (*application programming interface*).
CRUD	Operacijų rinkinys: sukurti, nuskaityti, atnaujinti ir pašalinti (*create, read, update, delete*).
COM port	Nuoseklioji kompiuterio sąsajos jungtis, skirta duomenų mainams su

	išoriniais įrenginiais realizuoti,
GPRS	Specifinis mobiliojo interneto ryšio tipas mobiliajame įrenginyje (*General Packet Radio Service*).
NMEA 0183	GPS įrenginiuose naudojamas komunikavimo standartas – specifinis komunikavimo protokolas.
Quadtree struktūra	Tai medis (duomenų struktūra) kurio kiekviena tėvinė viršūnė turi iki keturių žemesnio lygmens viršūnių.
SHA-1	160-bitų criptografinės santrumpos generavimo algoritmas. Vienkryptis algoritmas.
Mercator projekcija	Labai senas ir gerai žinomas cilindrinės žemėlapių projekcijos tipas.
Atvirkštinė Gudermano funkcija	Matematinė funckija, konkrečiu atveju naudojama sferinių koordinačių transformavimui į planimetrinę koordinačių sistemą.
USB	Standartinė kompiuterinių sistemų jungtis, skirta prijungti išorinius įrenginius prie kompiuterio ar kito įrenginio (*Universal Serial Bus*).
JUnit	Java programavimo kalbos testavimo karkasas skirtas vienetų testavimui įgyvendinti.

9. Priedai

1 Priedas. Tarpuniversitetinės magistrantų ir doktorantų konferencijos „Informacinės technologijos'08“ straipsnis: Daugiavartotojiška pozicionavimo sistema

Daugiavartotojiška pozicionavimo sistema

Andrius Blažinskas, Rytis Rudelis
KTU Informatikos fakultetas, Studentų g. 50, LT-51368 Kaunas, Lietuva

Šiame straipsnyje aptarsime realizuotą daugiavartotojišką pozicionavimo sistemą, naudojančią GPS technologiją. Detaliau paaiškinsime jos veikimo principus ir architektūrą. Išdėstysime pozicijos koordinačių nuskaitymo iš GPS įrenginio ir žemėlapių fragmentų nuskaitymo iš internetinės žemėlapių saugyklos metodus. Aptarsime naudojamą WEB servisų alternatyvą – REST servisus. Taip pat paminėsime keletą kitų panašių programinės įrangos sistemų ir jų naudojamų GPS ir GSM technologijų privalumus ir trūkumus.

Įvadas

Pozicijos nustatymo problema jau nuo senų laikų žmonėms buvo labai aktuali. Ji pakankamai efektyviai išspręsta tik XX a. pabaigoje. Tą sėkmingiausiai padarė JAV Gynybos Departamentas, sukūręs palydovinę radijo navigacinę sistemą – GPS (*Global Positioning System*).

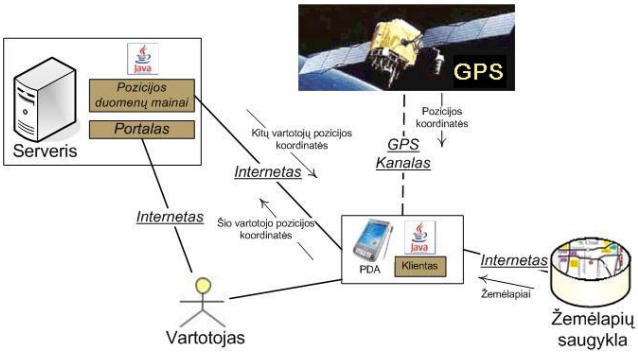
Kita pozicijos nustatymo technologija atsirado sparčiai išpopuliarėjus mobiliesiems telefonams. Ji remiasi mobilaus telefono pozicijos nustatymu GSM (*Global System for Mobile Communications*) tinkle, panaudojant tokius metodus kaip artimiausios bazinės stoties nustatymas (*Cell identification* – Cell-ID), signalo atėjimo kampo metodas (*Angle of Arrival* – AOA), radijo signalo atėjimo laiko metodas (*Time of Arrival* – TAO) ar patobulintas laikų skirtumo metodas (*Enhanced Observed Time Difference* – E-OTD) [4][5].

GPS tapus visuotinai prieinamai ir atpigus GSM paslaugoms, buvo pradėtos kurti įvairios programinės įrangos (toliau - *PĮ*) sistemos, naudojančios šias technologijas ir skirtos eiliniams vartotojams [1]. Pavyzdžiui GPS naudojančios *TomTom*, *Frotcom*, *GPSPursuit* ir kt., GSM naudojanti *Locator* ir kt. Tokio pobūdžio sistemų taikymo sritis nuolat plečiasi ir jos plačiai pradedamos naudoti kasdieniniame gyvenime. Būtent paprastam vartotojui yra skirta *Daugiavartotojiška pozicionavimo sistema*, kurios prototipą aptarsime toliau.

Sistemos prototipas

Veikimo principas

Sistemos prototipo veikimas yra paremtas kliento ir serverio principu. Vartotojas naudodamasis delniniame kompiuteryje įdiegta programine įranga (toliau - *klientas*) gali stebėti savo ir kitų vartotojų poziciją. Delninio kompiuterio GPS įrenginys vartotojo pozicijos duomenis gauna iš GPS palydovų ir perduoda klientui. Klientas šiuos duomenis siunčia į serverį, tokiu būdu sudarydamas galimybę juos pasiimti kitiems, tokiu pačiu klientu besinaudojantiems ir atitinkamą leidimą turintiems, vartotojams. Žemėlapius klientas tiesiogiai nuskaito iš lokalaus kešo, arba, jei reikiamo fragmento ten nėra, iš internetinės žemėlapių saugyklos. Serverio dalyje veikiantis portalas suteikia vartotojui galimybę tvarkyti savo duomenis, įskaitant ir maršrutus, kuriuos jis gali išsaugoti serveryje pasinaudodamas klientu.



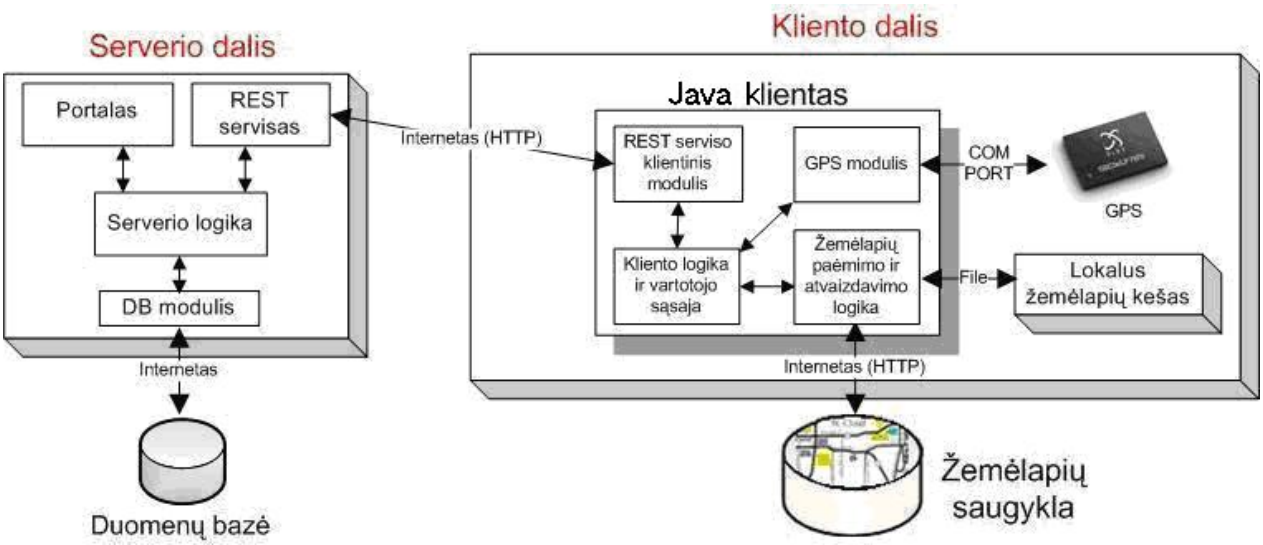
1 pav. – veikimo principas

Renkantis pozicionavimo technologiją buvo nuspręsta naudoti GPS, kadangi ji dengia visą Žemės paviršių, gali pateikti trimates koordinates, nustatyti objekto judėjimo kryptį ir greitį. Nors dauguma GPS imtuvų veikia tik lauko sąlygomis, tačiau yra ganetinai tikslūs: standartinis GPS imtuvas gali pateikti pozicijos koordinates 3 – 15 m

tikslumu, o aukštesnio lygio – 1 m tikslumu. GSM pozicionavimo technologija veikia ne tik lauke, bet ir pastatų patalpose, tačiau ji pateikia tik dvimates koordinates. Šių koordinačių tikslumas yra žymiai mažesnis ir priklauso nuo GSM tinklo struktūros, pozicijos nustatymui naudojamo metodo ir vietovės. GSM atveju paklaida svyruoja nuo 50 m iki 30 km.

Bendroji architektūra

Kaip ir buvo minėta, šios sistemos architektūra yra paremta kliento ir serverio principu. Serveris vartotojams teikia prieigą prie portalo ir prie REST (žiūrėti 3 skyrių „**REST panaudojimas**“) serviso, naudodamas bendrą serverio logikos modulį, kuris komunikuoja su duomenų baze.



2 pav. – bendroji architektūra

Kliento dalyje pagrindinis vidinės logikos modulis yra glaudžiai susijęs su vartoto sąsaja. Jis naudoja GPS modulį duomenų iš GPS įrenginio nuskaitymui, žemėlapių nuskaitymo ir atvaizdavimo logikos modulį. Su serveriu komunikuoja naudodamas paprastą, bet efektyvų, REST serviso klientinį modulį. GPS modulis koordinates nuskaito iš GPS įrenginio per nuosekliąją sąsajos jungtį (COM port). Žemėlapių atvaizdavimo ir valdymo modulis reikalingus žemėlapių fragmentus nuskaito iš failų sistemoje esančio kešo arba iš internetinės žemėlapių saugyklos. Vartotojui nurodžius, naujai parsiūsti žemėlapių fragmentai gali būti kešuojami.

Viena pagrindinių sistemos dalių – klientas, veikia delniniame kompiuteryje. Tokio kompiuterio našumas lyginant su paprasto stalinio kompiuterio našumu yra labai mažas. Prieigai prie interneto dažniausiai naudojamas bendras paketinis radijo ryšys (*General Packet Radio Service* – GPRS), kur apmokestinamas kiekvienas išsiustas megabaitas. Būtent dėl šių priežasčių bendra sistemos architektūra yra sudaryta taupant resursus ir mažinant persiunčiamų bei naudojamų duomenų apimtis.

Objekto pozicijos koordinatės ir jų transformacija

Objekto pozicijos duomenis klientas nuskaito iš delninio kompiuterio GPS įrenginio. Šis įrenginys, kaip ir daugelis kitų, naudoja labiausiai paplitusį NMEA 0183 (http://www.nmea.org/pub/0183/) komunikavimo standartą. Šis standartas apibrėžia įrenginio per nuoseklaus duomenų perdavimo jungti siunčiamus sakinius ir jų struktūrą. GPS įrenginys formuoja įvairių tipų NMEA 0183 sakinius, tačiau tik dalis iš jų saugo informacija apie objekto poziciją. Tokių pozicijos koordinates saugančių sakinių pavyzdžiai:

- `$GPGLL,4916.45,N,12311.12,W,225444,A,*31`
- `$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A`
- `$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47`

Internete yra nemažai NMEA 0183 sakinių analizės atvirojo kodo bibliotekų, tačiau šiuo atveju reikalingų sakinių kiekis yra labai mažas, todėl realizuotas efektyvus specializuotas šių sakinių analizės modulis.

NMEA 0183 sakiniuose, kartu su visa kita informacija, yra pateikiamos geografinės ilgumos ir platumos koordinatės, kurios priklauso sferinei koordinačių sistemai. Naudojami žemėlapiai, pasiekiami internetinėje žemėlapių saugykloje, yra sudaryti cilindrinės Mercator projekcijos (http://mathworld.wolfram.com/MercatorProjection.html) pagrindu. Taigi, naudojamos skirtingos koordinačių

sistemos sukelia problemą išrenkant reikiamą žemėlapio fragmentą ir atvaizduojant pozicijos tašką žemėlapyje. Šiai problemai išspręsti reikalingas koordinačių pervedimas iš sferinės koordinačių sistemos į stačiakampę. Tokį pervedimą galima atlikti panaudojant atvirkštinę Gudermano (Gudermannian) funkciją:

$$y = gd^{-1}(\varphi) = \frac{1}{2}\ln\frac{1+\sin(\varphi)}{1-\sin(\varphi)}, \text{ kur } \varphi \text{ – platumos koordinatė.}$$

Kaip matyti, transformacija yra reikalinga tik platumos koordinatei. Po šios transformacijos pozicijos koordinatės yra suderinamos su žemėlapių koordinačių sistema ir gali būti panaudojamos tiek reikiamo žemėlapio fragmento koordinačių išskaičiavimui, tiek ir paties pozicijos taško žemėlapyje atvaizdavimui.

Žemėlapiai

Objekto pozicijos atvaizdavimui kliente reikalingi žemėlapiai. Turint objekto pozicijos koordinates, reikiamų žemėlapių fragmentų pirmiausiai ieškoma lokaliame žemėlapių keše. Jei reikiamo fragmento keše nėra, jis imamas iš internetu pasiekiamos žemėlapių saugyklos. Žemėlapių saugykla žemėlapių fragmentus pateikia pagal jų specifines koordinates, todėl klientas turi mokėti išskaičiuoti reikiamo fragmento koordinates.

3 pav. – žemėlapio fragmentas

Saugykloje žemėlapiai saugomi kvadratinių, 256x256 taškų dydžio, fragmentų pavidalu, o pilnas iš šių fragmentų sudarytas žemėlapis paremtas Quadtree struktūros pagrindu. Trumpai nusakysime tokios saugyklos žemėlapio fragmentų koordinačių išskaičiavimą.

Išskaičiavimas pradedamas nuo pradinio fragmento X_0, kuris paprastai atitinka visą pasaulį. Šis fragmentas yra dalinamas į keturias lygias dalis (3 pav.) ir parenkama viena reikiama dalis. Naujai parinktos dalies atžvilgiu veiksmai yra kartojami tol, kol pasiekiamas reikiamo priartinimo lygio reikiamas fragmentas.

Matematiškai kiekvieno fragmento išskaičiavimą galima apibrėžti taip:

$$X_1(x_1;y_1;z_1) = X_0(2x_0;2y_0;z_0-1) \qquad X_3(x_3;y_3;z_3) = X_0(2x_0;2y_0+1;z_0-1)$$
$$X_2(x_2;y_2;z_2) = X_0(2x_0+1;2y_0;z_0-1) \qquad X_4(x_4;y_4;z_4) = X_0(2x_0+1;2y_0+1;z_0-1)$$

Kur $X_i(x_i;y_i;z_i)$, $i = \overline{1,4}$ yra 3 pav. pateiktas i-tasis fragmentas: x, y koordinatės ir z – priartinimo lygis.

Kaip ir buvo minėta, saugykloje saugomi žemėlapiai yra sudaryti Mercator projekcijos pagrindu. Ši projekcija yra plačiai taikoma ir gerai žinoma geografijos srities specialistams, tačiau turi esminį trūkumą – stipriai iškraipo žemėlapio mastelį tolstant nuo pusiaujo link ašigalių. Kaip pavyzdį galima palyginti Afrika su Grenlandija, kurios Mercator projekcijoje yra beveik tokio paties dydžio, tačiau realybėje Grenlandija yra beveik 14 kartų mažesnė už Afriką. Nepaisant šio trūkumo, ši projekcija yra tinkama interaktyvių internetinių žemėlapių sudarymui, kuriais patogu manipuliuoti, todėl neretai yra taikoma geografinėse informacinėse sistemose (GIS).

REST panaudojimas

REST (RE*presentational State Transfer*) – yra tam tikra programinė architektūra, skirta paskirstytoms hiperžiniasklaidos sistemoms, tokioms kaip žiniatinklis. Terminas pirmą kartą paminėtas 2000 metais vieno iš pagrindinių HTTP (*Hypertext Transfer Protocol*) specifikacijos autorių Roy Fielding disertacijoje [2]. Yra suformuluoti pagrindiniai principai, kuriais turėtų remtis REST architektūrą įgyvendinančios programinės realizacijos [6][7].

Sparčiai plintančius REST pagrindu sudarytus servisus galima laikyti tam tikra SOAP (*Simple Object Access Protocol*) Web servisų alternatyva. Pastarieji laikui bėgant tapo sudėtingi ir neefektyvūs, o paprastumu ir efektyvumu pasižymintys REST servisai vis labiau plinta [3]. Tačiau šiandien REST ideologija dar nėra apibrėžta kokio nors standarto ir neretai kyla debatai dėl šios architektūros ir termino panaudojimo. REST terminas dažnai yra naudojamas laisvesne prasme, kalbant apie paprastą sąsają, kuri skirta perduoti specifinius srities duomenis tiesiogiai panaudojant HTTP protokolą ir nenaudojant jokio papildomo pranešimų sluoksnio kaip SOAP. Internete yra daug šaltinių, kurie skelbiasi savo sistemose naudojantys REST architektūrą, tačiau daugelis iš jų yra tik tam tikros jos atmainos, įgyvendinančios tik dalį Roy Fielding suformuluotų REST principų.

Prototipo sistemos REST servisas pagrįstas labai paprastu principu. Jis naudoja HTTP protokolą ir neturi jokio papildomo pranešimų perdavimo sluoksnio. Pranešimai yra perduodami tiesiog per HTTP parametrus. Kliento siunčiamas pranešimas, arba tiesiog HTTP užklausa, iškviečia tam tikrą REST serviso metodą, kuris gali turėti parametrus arba ne. Tokios užklausos rezultatas yra HTTP atsakas – struktūrizuotas XML (*eXtensible Markup*

Language) srautas. Komunikuojant klientui ir serveriui yra būtina identifikuoti, kuris vartotojas vykdo užklausą, todėl yra būtinas identifikacijos mechanizmas. Šiuo atveju jis realizuotas ad-hoc principu. Pirmą kartą jungiantis vartotojui yra vykdoma autentifikacija ir grąžinamas SHA-1 (http://www.itl.nist.gov/fipspubs/fip180-1.htm) algoritmu sugeneruotas kodas. Šis kodas naudojamas kaip sesijos identifikatorius prisijungusiam vartotojui identifikuoti ir yra pridedamas prie užklausos parametrų kiekvieną kartą klientui kviečiant kitus REST serviso metodus.

Prisijungimo užklausos pavyzdys panaudojant HTTP GET metodą:

- http://myserver.com/RESTService?method=login&username=myUsername&password=myPassword

Tokios užklausos galimas atsakas XML formatu atrodo taip:

```
<?xml version="1.0" encoding="utf-8" ?>
<root>
  <data>true</data>
  <sessionToken>465bf07ec6e9d588ecfdfff0e5e175b266f45a00</sessionToken>
</root>
```

Roy Fielding REST ideologijoje serveryje neturėtų būti saugoma jokia būsena įskaitant ir sesijos duomenis. Šiuo atveju autentifikacija yra būtina, todėl realizuotas REST servisas yra tik minėto autoriaus architektūros atmaina.

REST principų panaudojimas geografinių žemėlapių pateikimui internetu yra viena iš naujai besiformuojančių metodikų (http://www.directionsmag.com/article.php?article_id=2515&trv=1). Realizuoto sistemos prototipo atveju, klientas žemėlapius nuskaito iš žemėlapių saugyklos, prie kurios prieiga teikiama per REST servisą. Kiekvienas žemėlapio fragmentas išrenkamas pagal jį apibrėžiančias x, y koordinates ir z priartinimo lygį. Šis parametrų trejetas paprasčiausiai nurodomas nuorodoje kreipiantis į servisą. Tokios nuorodos pavyzdys:

- http://myserver.com/MAPService?x=1160&y=648&z=6

Šios užklausos rezultatas yra reikiamas žemėlapio fragmentas.

Išvados

- GSM pozicijos nustatymo metodai labai praverčia stebimiems objektams esant uždarose patalpose, tačiau didelį tikslumą turinti GPS technologija yra nepakeičiama lauko sąlygomis.
- Taupant kuklius delninio kompiuterio resursus ir mažinant persiunčiamų duomenų kiekį, tikslinga naudoti plintančią Web servisų atmainą – REST servisus, o naujai atsiųstus žemėlapių fragmentus – kešuoti.
- Pozicijos koordinatės gaunamos iš GPS įrenginio priklauso sferinei koordinačių sistemai, o žemėlapiai – stačiakampei, todėl, pozicijos taško atvaizdavimui ant žemėlapio atlikti, būtina pozicijos koordinates transformuoti į stačiakampę koordinačių sistemą panaudojant atvirkštinę Gudermano (Gudermannian) funkciją.
- Mercator projekcijos pagrindu sudaryto žemėlapio mastelis yra stipriai iškraipytas, ypač tolstant nuo pusiaujo link ašigalių. Tačiau ši projekcija puikiai tinkama interaktyvių internetinių žemėlapių sudarymui, kuriais patogu manipuliuoti, todėl neretai yra taikoma geografinėse informacinėse sistemose.

Literatūros sąrašas

[1] **D. Dao, C. Rizos, G. Wang**. Location-based service: Technical and business issues. *GPS solutions,* 2002, Vol. 6(3), p. 169-178.

[2] **R.T. Fielding**. Architectural Styles and the Design of Network-Based Software Architectures. doctoral dissertation, Dept. of Computer Science, Univ. of Calif., Irvine, 2000.

[3] **G**. **Goth**. Critics Say Web Services Need a REST. *IEEE Distributed systems online*, *IEEE Computing Society*, Dec. 2004, Vol. 5(12).

[4] **V. Liutkauskas, I. Lagzdinytė**. GSM tinklo mobilios vietos patikslinimo būdai. *Elektronika ir elektrotechnika*, *KTU,* 2004, Nr. 5, p. 141-146.

[5] **V. Palilionis**. GIS ir mobiliųjų technologijų integravimo lokalizuotųjų paslaugų sistemose. *Geodezija ir kartografija*, *VGTU,* 2004, XXX t., Nr. 4, p. 117-122.

[6] **S. Vinoski**. Demystifying RESTful Data Coupling. *IEEE Internet Computing online*, *IEEE Computing Society,* Mar.-Apr. 2008, Vol. 12(2), p 87-90.

[7] **S**. **Vinoski**. REST eye for the SOA Guy. *IEEE Internet Computing online, IEEE Computing Society*, Jan.-Feb. 2007, Vol. 11(1), p. 82-84.

Multi-user position tracking system

Different kinds of positioning systems are being used around the world. Very often they are specialized for a particular problem domain and named differently, but usually based on the same well known technologies. Most of these systems can be classified as GPS or GSM based systems, although some of them combine these two technologies to achieve optimal results. In most of these systems it is very important their operational range, position data accuracy and timing. Prevailing systems use mostly GPS technology. However both GPS and GSM based positioning systems have their advantages and drawbacks. Because mobile internet connection is improving quickly and becoming widely available, it opens wider possibilities of using such systems to ordinary consumers in every day life. In this article we briefly describe concept and main functional principles of implemented multi-user positioning system prototype based on GPS technology.