KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Mantas Ramoška

MS .Net žiniatinklio programų pritaikymas bevielio ryšio įrenginiams

Magistro darbas

Vadovė
Prof. Dr. L. Nemuraitė

KAUNAS, 2008

KAUNO TECHNOLOGIJOS UNIVERSITETAS
INFORMATIKOS FAKULTETAS
INFORMACIJOS SISTEMŲ KATEDRA

Mantas Ramoška

MS .Net žiniatinklio programų pritaikymas bevielio ryšio įrenginiams

Magistro darbas

Recenzentas
doc. dr. A. Riškus
2008-01-14

Vadovas
Prof. Dr. L. Nemuraitė
2008-01-14

Atliko IFM-2/4 gr. stud.
Mantas Ramoška
2008-01-14

Kaunas, 2008

Adaptation of MS.Net internet applications for wireless connection devices

Summary

These days information technology is growing very fast, especially mobile technology. Almost every company has it‘s web site and uses other technologies. But now it is time to adapt services for mobile phones and other mobile devices. That is why adaptation of MS.Net internet applications for wireless connection devices is actual.

The goal of master thesys is to find technological solution to integrate internet systems, developed using Microdoft ASP.Net and WAP standarts. And to develope a practical solution for „Restaurant orders information system“.

Analyzed information from many sources, there are three main problems when adapting information system for mobile devices: screen size, navigation and input methods. All these problems describe system usability. And the goal of work is to reach maximum level of usability. To develope that solution Agile software development method eXtreem Programming and usability oriented method were chosen.

Finally users asked to rate the sistem have been evaluating a table with main quality points to the solution during many iterations until the average score went to 4 (5 - maximum). System implementation test has shown that requirements are met, and goal is achieved.

1 Įvadas

Šiuo metu informacinės technologijos ne tik vystomos žaibišku greičiu, bet stengiamasi jas padaryti kuo mobilesnes. Įmonės, kurios gali tikėtis sėkmės privalo būti matomos internete ir pasiekiamos tiek įprastais kompiuteriais, tiek mobiliais įrenginiais. Yra aibės elektroninių paslaugų, kuriomis naudojamės kasdieną ir be kurių tikriausiai neįsivaizduotume eilinės dienos. Tačiau būtų dar paprasčiau, jei šias paslaugas galėtume atlikti tarkim mobiliuoju telefonu. Tai elektroninis paštas, maisto užsakymas, banko operacijų atlikimas ir daugelis kitų paslaugų.

Atsižvelgiant į sistemos poreikį, buvo nuspręsta sukurti vietų restorane rezervavimo paslaugą, kai restoranų tinklas pasiskirstęs po visą Lietuvą. Įsivaizduokite, važiuojate su draugais į kitą miestą, o draugai rezervuoja staliuką Jūsų mėgstame restorane. Užsakymas naudojant bevielio ryšio protokolą WAP yra labai patogus, greitas, nereikia žinoti restorano telefono kiekviename mieste. Tereikia turėti mobilųjį telefoną, kuriame veiktų WAP paslauga. Šiandien tai jau nėra problema, nes daugelis naudojasi WAP paslaugomis, ypač jaunimo tarpe.

Tačiau norint kad sistema vartotojai naudotųsi, būtina užtikrinti sistemos panaudojamumą (usability). Kadangi mobilūs telefonai yra maži, jais nepatogu rinkti tekstą ar atlikti navigaciją, jie nepalaiko aukšto lygio grafinio vartotojo interfeiso galimybių, daugeliui šis naudojimosi būdas gali būti nepatogus ir vartotojai paprasčiausiai ja nesinaudos. Kad išvengti šios problemos, yra labai svarbu išsiaiškinti panaudojamumo problemas bei būdus, kaip galima padidinti panaudojamumą bevieliuose mobiliuose įrenginiuose.

Darbo tikslas – surasti technologinį sprendimą, kuris įgalintų integruoti interneto sistemas, kuriamas naudojant Microsoft „.NET" ir WAP standartą, bei pritaikyti šį sprendimą praktikoje sukuriant restorano užsakymų aptarnavimo sistemą.

Išanalizavus informaciją iš daugelio šaltinių [21], [22], [23] buvo išsiaiškinta trys pagrindinės panaudojamumo problemos bevielio ryšio mobiliems įrenginiams, tai: ekrano dydis, navigacija ir įvesties metodai. Sistema buvo projektuota kad sumažinti šias problemas. Kiekvienoje kūrimo iteracijoje buvo atliekamas sistemos įvertinimas. Jei šis įvertinimas buvo mažesnis nei 3, sistema buvo tobulinama kad pasiekti aukštesnį įvertinimą. Galutiniame sistemos įvertinime buvo pasiektas vidurkis 4.

2 Interneto informacinių sistemų ir bevielio ryšio įrenginių integravimo galimybių analizė

Analizės tikslas – išanalizuoti informacinių sistemų, sukurtų naudojant Microsoft ASP.Net technologiją [10], [11], [12], integraciją su bevielio ryšio mobiliais įrenginiais ir pasirinkti tinkamiausius sprendimus kuriant informacines sistemas, pasiekiamas per internetą ir bevielio ryšio įrenginius.

Kad pasirinkta tema yra aktuali, patvirtina tai, jog daugelis įmonių ir paslaugų tiekėjų savo paslaugas pritaiko mobiliems įrenginiams. Muzikiniai konkursai, Lietuvos krepšinio lygos žvaigždžių dienos žaidėjų rinkimai ir daug kitų renginių organizuojami taip, kad žiūrovai galėtų balsuoti per WAP, Galima surasti daug pavyzdžių kur naudojamos WAP technologijos. Štai keli iš jų:

- Autobusų ar traukinių maršrutų grafikų paieška.
- Bilietų įsigijimas.
- Skrydžių registravimas.
- Orų prognozės žiūrėjimas.
- Telefonų numerių paieška.
- Akcijų reikšmių ieškojimas.
- Adresų paieška.
- Naujienų skaitymas.

Šio darbo tyrimoji sritis apima tiek mobiliąsias WAP (*Wireless markup language*) technologijas, tiek Microsoft ASP.NET paketą [15], [16], [19], kurio pagalba kuriamos įprastos interneto svetainės. Tyrimo objektas – bevieliuose įrenginiuose veikiančių informacinių sistemų kūrimo procesas ir jo taikymas restorano užsakymų paslaugoms sukurti..

2.1 WAP technologijų taikymo problemų analizė

Kaip žinoma, ASP.NET yra ypač galingas įrankis [17], [18], kuriuo pagalba galima sukurti sudėtingus portalus. Tačiau prisitaikant prie WAP technologijos tenka susidurti su tam tikromis problemomis ir sunkumais (2.1 lentelė):

2.1 lentelė. WAP technologijų taikymo problemos

Problema	**Sprendimo būdas**
Panaudojamumo užtikrinimas, atsižvelgiant į	Interfeisas bevielio ryšio įrenginiams turi būti

mobilių įrenginių interfeiso savybes.	suprojektuotas taip, kad galiniam vartotojui neiškiltų sunkumų dirbant su sistema, atsižvelgiant į bevielio ryšio įrenginių galimybes apdoroti pateiktą informaciją.
WML (*Wireless markup language*) yra apribotas kai kurių žymių (tag) naudojimas, nes WML yra skirtas daugiau tekstinei informacijai pateikti.	Pasitelkti ASP.Net turimais komponentais, kurie yra skirti bevielio ryšio mobiliems įrenginiams. Tokiu būdu palengvinamas ir paspartinamas Web programų kūrimas bei WML kodo generavimas perkeliamas ASP.Net technologijai.
Kadangi bevielių įrenginių, lyginant su įprastiniais asmeniniais kompiuteriais resursai yra riboti, iškyla problemos apdorojant didelės apimties failus.	Puslapiai, parašyti WML, konstruojami kaip kortelių (*Cards*) rinkiniai. Kortelės elementas gali savyje turėti: tekstą, nuorodas, įvesties laukus, paveiksliukus, ir kitus komponentus. Kortelės vienos su kita gali būti susietos nuorodomis. Kai WML puslapis užklausiamas mobiliojo įrenginio, WAP serveris atsiunčia visas korteles į įrenginį. Navigacija tarp kortelių atliekama bevieliu įrenginiu.
WML yra ribotas JavaScript palaikymas. Jis vadinamas WMLScript ir turi būti sukompiliuotas serverio pusėje.	Pasitelkti ASP.Net technologija, kuri turi parašytas funkcijas (tokias kaip laukų validavimas, peradresavimas į kitą puslapį).

Darbo tikslas – surasti technologinį sprendimą, kuris įgalintų integruoti interneto sistemas, kuriamas naudojant Microsoft „.NET" ir WAP standartą, bei pritaikyti šį sprendimą praktikoje sukuriant restorano užsakymų aptarnavimo sistemą.

Tam tikslui bus išanalizuoti WML (*Wireless markup language*), bei WAP (*Wireless application protocol*) standartai [3].

Prieš pradedant analizuoti WAP, buvo susipažinta su sekančiomis technologijomis:

- WWW, Html ir žiniatinklių kūrimo pagrindais
- JavaScript
- XML

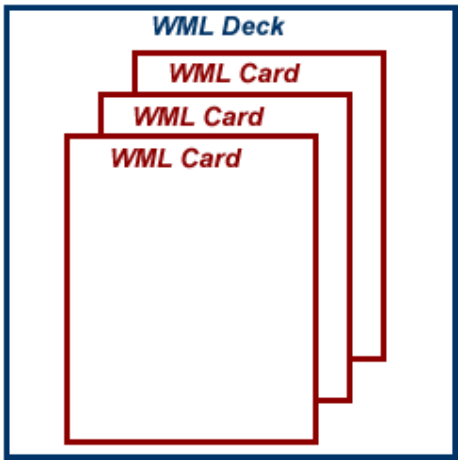
Belaidė industrija atėjo jau su WAP idėja. WAP protokolas buvo parašytas su tikslu rodyti interneto turinį bevieliuose įrenginiuose, tokiuose kaip mobilusis telefonas. WML yra kalba, naudojama kurti puslapiams, kurie bus atvaizduoti WAP naršyklėse.

- WAP laikomas bevieliu programų protokolu (*Wireless application protocol*)
- WAP yra programų komunikavimo protokolas
- WAP yra naudojamas pasiekti servisams ir protokolams
- WAP yra paveldėtas iš kitų interneto standartų
- WAP skirtas bevieliams įrenginiams, tokiems kaip mobilusis telefonas
- WAP yra protokolas, sukurtas mini naršyklėms
- WAP įgalina kurti interneto programas mobiliems įrenginiams
- WAP naudoja „mark-up" kalbą WML (ne html)
- WAP yra aprašoma kaip XML 1.0 programos

WAP protokolas yra pirmaujantis standartas bevieliams įrenginiams, tokiems kaip mobilusis telefonas. Šis standartas paremtas kitais interneto standartais: HTML, XML ir TCP/IP. Jis susideda iš WML kalbos specifikacijos, WMLScript specifikacijos ir Belaidės telefonijos programos sąsajos – WTAI (Wireless telephony aplication interface). WAP buvo pristatytas WAP forume, įkurtame 1997 metais tokių kompanijų kaip „Ericcson", „Motorolla", „Nokia" ir „Unwired planet".

Kad atitiktų nedidelį mobilių įrenginių dydį, WAP technologijai reikalingos mini naršyklės. Mini naršyklė yra nedidelė dalis programinės įrangos (*software*), kuri reikalauja minimalių aparatūrinės įrangos (*hardware*) resursų. Ši naršyklė gali atvaizduoti informaciją, parašytą griežta „mark-up" kalba, pavadinta WML. Mini naršyklė taip pat gali interpretuoti „apkarpytą" JavaScript versiją, pavadintą WMLScript.

WML [6] (*Wireless markup language*) paveldėta iš HTML, tačiau WML yra pagrįsta XML standartu, taigi ji yra žymiai griežtesnė nei HTML. WML naudojama kurti puslapiams, kurie bus atvaizduoti WAP naršyklėse. Puslapiai, parašyti WML yra vadinami „DECKS" (*liet.: denis*). Deniai konstruojami kaip kortelių (*Cards*) rinkiniai (2.1 pav.). Kortelės elementas gali savyje turėti: tekstą, nuorodas, įvesties laukus, paveiksliukus, ir kitus komponentus. Kortelės vienos su kita gali būti susietos nuorodomis. Kai WML puslapis užklausiamas mobiliojo įrenginio, WAP serveris atsiunčia visas korteles į įrenginį. Navigacija tarp kortelių atliekama bevieliu įrenginiu.



2.1 pav. WML kortelės

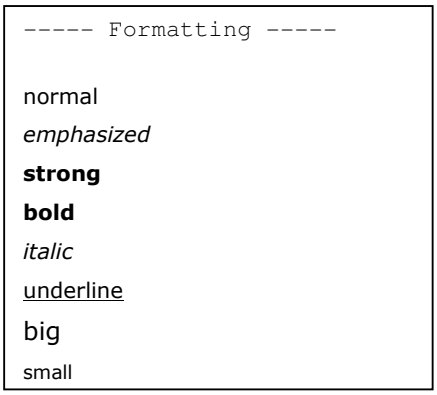
WML naudoja WMLScript kad vykdytų kai kuriuos veiksmus kliento pusėje. WMLScript yra lengva JavaScript kalba. Kaip ten bebūtų, WML skriptai nėra įmontuoti į WML kodą. WML kode aprašytos tik nuorodos į WMLScrip URL (inified resource location). Prieš vykdant WML skriptą, jis turi būti sukompiliuotas serverio pusėje, tik tada mobilieji įrenginiai galės jį vykdyti.

Žymės (*tags*), kurios sulėtintų komunikavimą su bevieliu mobiliu įrenginiu, griežtai apribotos. Šios žymės būtų: lentelės (*tables*), paveiksliukai (*images*) ir kitos. WML yra daugiausiai orientuotas į tekstą. Kadangi WML yra XML programa, žymių didžiosios bei mažosios raidės skiriasi. Pavyzdžiui, <WML> nėra tas pats kas <wml> ir visos žymės privalo būti tinkamai uždarytos </wml>.

WML kodo pavyzdys:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>
<card title="Formatting">
<p>
normal<br/>
<em>emphasized</em><br/>
<strong>strong</strong><br/>
<b>bold</b><br/>
<i>italic</i><br/>
<u>underline</u><br/>
<big>big</big><br/>
<small>small</small>
</p>
</card>
</wml>
```



WML palaiko laikmatį (Timer), kurio pagalba galima uždelsti prieš atliekant kažkurį veiksmą. Laiko vienetas trunka 1/10 sekundės. WML kalba taip pat palaiko kintamuosius (*variables*). Kai vartotojas pereina nuo vienos kortelės prie kitos, duomenis reikia išsaugoti kintamuosiuose. Šie kintamieji yra taip pat jautrūs didžiosioms ir mažosioms raidėms. Kai vartotojas duoda komandą (Pvz.: *go*, *prev* ir *refresh*), „*setvar*" elementas gali būti naudojamas išsaugoti reikšmę. Sekantis pavyzdys parodo kaip išsaugoti reikšmę („name" ir „value" atributai būtini):

```
<setvar name="i" value="500"/>
```

Kintamieji gali būti išsaugomi ir naudojant įvesties (input) elementus: *input*, *select*, *option*, bei kitus.

```
<card id="card1">
<select name="schoolname">
<option value="HTML">HTML kintamasis</option>
<option value="XML">XML kintamasis</option>
</select>
</card>
```

Panaudoti išsaugotą kintamąjį galima sekančiai:

```
<card id="card2">
<p>Jūs pasirinkote: $(schoolname)</p>
</card>
```

2.2 Sistemos panaudojamumas (usability)

Išanalizavus informaciją iš daugelio šaltinių [21], [22], [23] buvo išsiaiškinta trys pagrindinės panaudojamumo problemos bevielio ryšio mobiliems įrenginiams, tai: ekrano dydis, navigacija ir įvesties metodai.

2.2.1 Ekrano dydis

Be jokios abejonės, didžiausia problema bevieliuose mobiliuose įrenginiuose yra ekrano dydis. Lyginant daugelį kitų vartotojo interfeiso platformų, matome, kad kuo didesnis ekrano dydis – tuo geresnį panaudojamumą galima išgauti. Grafinis interfeisas taip pat pagerina sistemos panaudojamumą. Kaip matome, šiuos veiksnius riboja patys bevieliai mobilūs įrenginiai – jie sukurti tam kad būtų mobilūs – maži, bei neturi techninių galimybių naudoti sudėtingesnę grafinę informaciją. Manoma, kad tai netgi gali sumažinti vartotojų entuziazmą naudotis WAP.

Tačiau mažas ekrano dydis nebūtinai reiškia prastą aiškumą, suprantamumą ar neefektyvų naudojimąsi. WAP bevielio ryšio įrenginiai labai efektyviai gali atvaizduoti trumpą, aiškią ir apibendrintą informaciją.

2.2.2 Navigacija

Kaip žinia, WAP svetainės susideda iš eilės puslapių (*angl. Deck - denis*), kurie savo ruožtu gali turėti vieną ar daugiau kortelių (cards). Navigacija gali būti atliekama tarp kortelių bei tarp puslapių. Kortelės apima atskirus informacijos blokus ir gali būti naudojamos jei WML puslapis yra per didelis atvaizduoti mobiliajame įrenginyje – nuorodos daromos tarp kortelių.

Navigacijos problema yra tame, kad vartotojui norint atlikti tam tikrą funkciją reikia atlikti daug veiksmų, kas gali būti šiek tiek nepatogu ir užimti papildomai laiko. Judėdamas svetainės hierarchija vis tolyn: wml-> wml vaikas -> wml vaiko vaikas-> … vartotojas labai lengvai atitrūksta. Kad judėjimas hierarchijos lygiais būtų efektyvus, reikia naudoti nuorodas į aukščiausio hierarchinio lygio puslapius, kad vienu paspaudimu būtų galima grįžti į pradinį puslapį.

2.2.3 Įvesties metodai

Šiuolaikiniuose bevieliuose mobiliuose įrenginiuose kiekvienas įvesties laukas reikalauja bent menkiausių pastangų iš vartotojo. Kad palengvinti šį darbą, mobiliuose įrenginiuose naudojami automatinio nuspėjamumo žodynai. Taip pat reikalinga suprojektuori įvesties metodus taip, kad vartotojui reikėtų kuo mažiau pastangų norint įvesti tam tikrą informaciją.

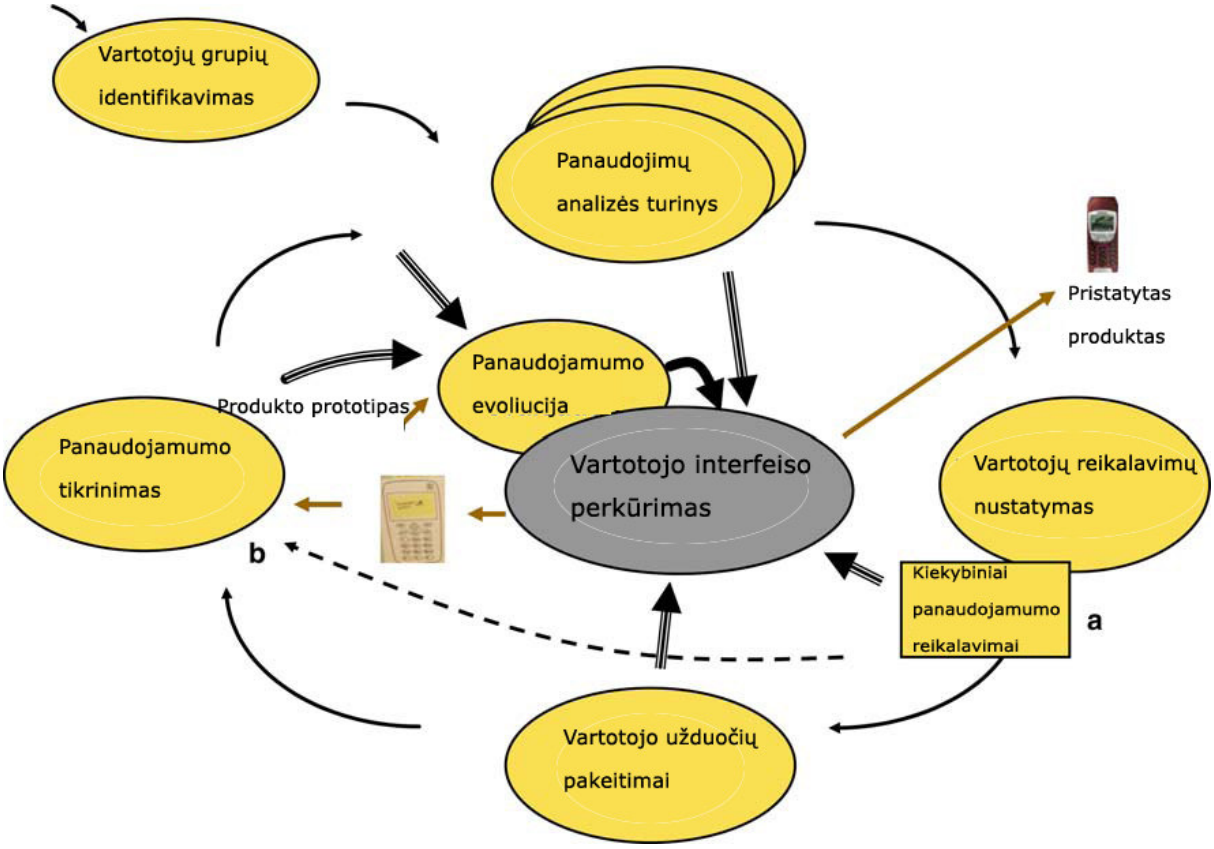
2.2.4 Panaudojamumui orientuotas kūrimas

Mobilaus ryšio telefonai tapo natūralia mūsų kasdienio gyvenimo dalimi. Jų draugiškumas vartotojui, vadinamas panaudojimo patogumu, turi vis didesnę paklausą. Panaudojimo patogumas atneša daug naudos: vartotojai panaudoja įvairias telefonų ir paslaugų tiekėjų galimybes, palaikomi vartotojų poreikiai. Tuo pačiu metu projektavimas vis daugiau ir daugiau stimuliuoja funkcijų skaičiaus didėjimą ir telefonų dydžio mažėjimą. Standartas ISO 13407 yra plačiai panaudota bendra rekomendacija panaudojimo patogumo projektavimui. Bendras panaudojimo patogumas turi būti užtikrinamas viso bevielio ryšio įrenginiams skirtų paslaugų kūrimo ciklo metu. [21]

Bevielių įrenginių projektavimo procesas skiriasi nuo įprasto kūrimo proceso, kadangi jam neįmanoma ir neprotinga pritaikyti egzistuojančius metodus. Čia tikslinga naudoti nestandartinius metodus. Šio ciklo žingsniai:

1. Vartotojų analizė. Pirma reikia atpažinti vartotojus. Naudojimo analizės kontekstas remiasi informacija apie vartotojus: kokie vartotojo tikslai, kokias užduotis jie daro ir kuriame kontekste. Vartotojų informacija yra pagrindas panaudojimo patogumo reikalavimams, kur yra nustatyti tikslo lygmenys panaudojimo patogumo produkto po sukūrimo.
2. Vartotojų užduočių analizė. Naujas produktas turi palaikyti efektyvesnes vartotojų užduotis; jos yra aiškiai suprojektuojamos kitame žingsnyje. Šioje stadijoje pradedamas projektuoti ir vartotojo interfeisas (UI).
3. Formuojama įvestis į sąveikos projektavimo fazę. Sąveikos projektas ir kokybiniai panaudojimo patogumo įvertinimai 2 tipiškai formuojasi, pasikartojantis procesas (kaip daroas visas panaudojimo patogumo projektavimo procesas). Įvairūs prototipų lygmenys yra dažnai naudojami šioje stadijoje.

Kai projektas pradeda subręsti, tikrinami panaudojimo patogumo reikalavimai. Panaudojimo patogumą galima pasiekti tik turint galimybę išmatuojami sąveikos panaudojimo patogumą.Negalint pamatuoti panaudojimo patogumo, nėra jokio būdo nustatyti, ar tikrai sistema įvykdo panaudojamumo reikalavimus. Jei jų negalima pamatuoti, nėra ir galimybių valdyti projektavimą. Matavimo metodai leidžia įvertinti atitikimą panaudojamumo reikalavimams praktiniame kontekste projektuojant UI mobilaus ryšio telefonui.



2 pav. Panaudojamumo užtikrinimas

2.3 Informacinių sistemų, pasiekiamų bevieliais įrenginiais, kūrimo metodai

Norint pasirinkti tinkamą metodą Is kūrimui, buvo nagrinėjami praktikoje labiausiai paplitę programinės įrangos (PĮ) kūrimo procesai:

- „Waterfall model" – krioklio modelis
- „Iterative development" – iteracijomis grįstas PĮ kūrimas.
- „Agile software development" – lankstusis programinės įrangos kūrimo procesas. [9]
- „Cowboy coding" – kaubojiškas programavimas.

Išsiaiškinus kiekvieno išvardinto PĮ kūrimo proceso privalumus bei trūkumus, buvo nuspręsta pasirinkti „Agile software development" – lankstųjį paprogramavimą. **Lankstusis programavimas** - programų kūrimo metodologijos, pasiūlytos nepelno organizacijos "Agile Alliance". Metodai buvo kuriami tiems atvejams, kai programų kūrimas sunkiai valdomas,

reikalavimai sparčiai kinta, tokiais atvejais lankstusis programavimas supaprastina tradicinę programų inžineriją.

Lanksčiojo programavimo metodais mėginama mažinti riziką, skaidant ilgalaikius projektus į smulkesnius tarpinius projektus, kuriuose dirbtų mažos komandos (2-3 žmonės). Kai komandos yra didesnės, kūrimo ciklai darosi per ilgi, kad suvaldyti komunikavimo sunkumus. Tipiškas programų kūrimo ciklas trunka nuo 1-4 savaitės, kiekvieno ciklo gale peržiūrimi projekto prioritetai, tai leidžia greitai prisitaikyti prie kintančių reikalavimų, suvaldyti atsirandančias rizikas.

Pagrindinis lanksčiojo programavimo skirtumas nuo tradicinių sunkesnių, į procesą orientuotų metodologijų (krioklio, spiralės modeliai) - orientacija į principus, ne į procesą.

Agile metodai yra visa procesų šeima, ne vienas žingsnis kaip kurti programinę įrangą. Lanksčiojo programavimo proceso metodo sukūrimo tikslas – kas PĮ kūrimo procesas būtų lengvesnis, greitesnis ir labiau orientuotas į žmogų (vartotoją). 2001 metais buvo sukurtas „Agile Manifesto" (Lankstusis manifestas).

Agile software development metodas „Ribinis programavimas" [8] (**XP** iš angl. *eXtreme Programming*) – vienas iš populiariausių lanksčiojo programavimo metodų, suformuluotas *Kent Beck*, *Ward Cunningham* ir *Ron Jeffries*.

Bazinės praktikos skirstomos į šias grupes:

- Pastovus grįžtamasis ryšys (feedback)
- Testavimu paremtas programavimas
- Planavimo žaidimas
- Vientisa komanda (įtraukiant užsakovą)
- Programavimas poromis
- Nenutrūkstamas procesas
- nuolatinis integravimas
- Projektavimo korekcijos
- Smulkūs išleidimai (releases)
- Bendras supratimas
- Paprastas projektas
- Sistemos metafora
- kolektyvinė atsakomybė už kodą
- Kodavimo standartai
- Programuotojo gerovė
- Pastovus tempas (jokių viršvalandžių)

Visos šios praktikos išvestos iš pripažįstamų geriausių praktikų, naudojant jas ribiniu atveju:

- Bendravimas tarp programuotojo ir kliento yra gerai. Todėl į ribinio programavimo komandą įtraukiamas užsakovas, detalizuojantis ir prioritetizuojantis darbus, taip pat galintis iškart atsakyti į iškilusius klausimus.
- Mokymasis gerai. Todėl programavimo laikas (ciklas) sutrumpinamas iki mažiausio galimo, o testuojama programavimo metu.
- Kuo paprastesnis kodas, tuo didesnė tikimybė, kad jis gerai veiks. Taigi, programuojama tik tai, ko reikia, prireikus, eliminuojamas kompleksiškumas ir kodo dubliavimas. Per sudėtingas kodas perrašomas.
- Kodo peržiūros yra naudingos. Todėl XP programuotojai dirba poromis, prie vieno monitoriaus ir klaviatūros, todėl visas kodas peržiūrimas rašymo metu.
- Kodo testavimas yra gerai. Todėl automatiniai testai rašomi prieš pradedant rašyti kodą. Užduotis laikoma baigta tik tada, kai visi testai baigiami sėkmingai. Periodiškai paleidžiami visi anksčiau rašyti automatiniai testai, užtikrinant kad nesugadintas ankstesnis kodas. Radus klaidą, jai taip pat sukuriamas automatinis testas.

Dažnai manoma, kad ribinis programavimas gali veikti tik mažose komandose iki 12 žmonių, tačiau kartais jis sėkminga veikia ir komandose su daugiau nei 100 programuotojų.

Ribinio programavimo metodologijas naudojantys programuotojai dažnai renkasi dinamiškas, greičiau įgalinančias pasiekti reikiamų rezultatų programavimo kalbas.

Ribinis programavimas kritikuojamas dėl šių aspektų:

- Programos kūrimo metu nesukuriama detali dokumentacija.
- Programuotojai dirba poromis.
- Prieš pradedant darbą nėra atliekami rimti planavimai.
- Užsakovo atstovas privalo visą laiką dalyvauti projekte.
- Sunkiai prognozuojama projekto kaina

Agile metodai kartais laikomi opozicija metodams, kurie pagrįsti planavimu ir disciplina. Ši nuomonė yra klaidinanti, kai norima pasakyti, kad Agile metodai nėra suplanuoti ar juose trūksta disciplinos. Tiksliausiai apibūdinant skirtumą reikėtų sakyti, kad yra prisitaikantys („Adaptive”) ir prognozuojantys („Predictive”). Agile metodai priskiriami prie prisitaikančių („Adaptive”) metodų.

Prisitaikantys metodai orientuoti į greitą pasikeitimą, pasikeitus aplinkybėms, reikalavimams. Kai projekto reikalavimai keičiasi, keičiasi ir prisitaikanti komanda. Prisitaikanti komanda turės sunkumų detalizuojant kas bus daroma ateityje. Kuo tolimesnė ateitis, tuo sunkiau pasakyti kas bus atliekama. PĮ kūrėjai gali tiksliai pasakyti kas yra suplanuota padaryti ateinančiai savaitei, tačiau tik toms galimybėms, kurios suplanuotos artimiausiam mėnesiui. Tačiau naudojant Agile metodus negalima tiksliai pasakyti kurioje stadijoje bus projektas po, tarkim pusės metų. Galima įvardinti tik trumpą išdėstymą apie projekto pabaigą ir tikėtiną kainos ir kokybės santykį.

Jei palyginsime pranašaujančius metodus, jie koncentruojasi į detalų veiksmų planavimą ateičiai. Naudojant šį metodą, galima tiksliai pasakyti, kas bus daroma visą projektui skirtą laiką. Tačiau jei projekto eigoje reikia pakeisti projekto kryptį, „Predictive" metodai susiduria su tam tikrais sunkumais. Planas sudaromas pirminiams reikalavimams, ir keičiant juos gali atsitikti kad jau padarytas darbas nebebus reikalingas tolimesnėje projekto vykdymo eigoje ir jį reikės perdaryti. Kad to išvengti, programuotojų komanda dažnai turi svarstyti, kuriuos pasikeitimus verta atlikti.

2.4 Projektavimo metodų, priemonių pasirinkimas

Šiame punkte pateikiama projektavimų metodų lyginamoji analizė. Bus pagrįstas projektavimo metodo pasirinkimas. Analizuoti pasirinkti keturi gana populiarūs metodai: „Agile software development", „Waterfall model", „Iterative development", ir „Cowboy coding". Kiekvienas yra skirtingas, todėl bus įdomu analizuoti kuo jis skiriasi nuo pasirinkto „Agile software development" metodo.

Medodas	**Aprašymas**
„Iterative development" Iteracijomis grįstas PĮ kūrimas	Panašus į „Agile software development", nes kuriama programinė įranga išleidžiama kas trumpą laiko tarpą. Tačiau daugelis Agile metodų skiriasi ne tokiu griežtu darbo laiko paskirstymu, kaip „Iterative development" metodas.
„Waterfall model" (Krioklio modelis)	Agile metodai turi mažiau bendro su „Waterfall model" modeliu. Kai kurių žmonių akimis šis modelis yra praradęs pasitikėjimą, tačiau realybėje šis modelis labai dažnai naudojamas. Šis modelis yra labiausiai prognozuojantis, einantis per reikalavimus, analizavima, projektavimą, programavimą, testavimą labai griežta ir suplanuota seka. Projekto progresas iš esmės pamatuotas apimant reikalavimų

	specifikavimą, dokumentavimą, testavimo planus, kodo peržiūrą. Dirbant tokiu metodu projekto baigimo laikas gali svyruoti nuo kelių mėnesių iki kelių metų. Metodo platumas ir sudėtingumas šio prisitaikymo ir testavimo pastangų yra priežastys dėl kurių projektas gali žlugti. Jei palyginti Agile metodus, jie pristato sukurtas ir ištestuotas galimybes kas kelias savaites, tačiau tik nedidelę dalį jų.
„Cowboy coding”	Šis metodas reiškia jokio metodo nebuvimą. Komandos nariai dirbai taip, kaip jiems atrodo teisinga ir geriausia. Jis yra kartais painiojamas su Agile metodais dėl dažnų planų pasikeitimų, bei santykinai negausios dokumentacijos. Agile metodai numato darbų seką, beja, dažniausiai labai disciplinuotą ir griežtą.
„Agile software development” metodas „Ribinis programavimas”	Šiam tyrimui pasirinktas „Agile software development” metodas „Ribinis programavimas”. Tai vienas populiariausių lanksčiojo programavimo metodų. Jo pagrindiniai privalumai yra: pastovus grįžtamasis ryšys, testavimu paremtas programavimas, vientisa komanda (įtraukiant užsakovą), nenutrūkstamas procesas, nuolatinis integravimas, projektavimo korekcijos, smulkūs išleidimai (releases), bendras supratimas, kodavimo standartai, programuotojo gerovė, pastovus tempas (jokių viršvalandžių). Tai įdomus metodas, išsiskiriantis iš įprastinių.

Su visomis metodologijomis reikalinga programuotojų patirtis ir įgūdžiai. Tai dažniausiai nusako, ar projekto baigtis bus sėkminga ar ne. Griežta kontrolė, pritaikyta kiekvienam procesui ir didesnis atsakingumo lygis ypač reikalingas kuriant sudėtingas sistemas su bet kuriuo modeliu. Daugelio funkcijų nežinojimas ar nemokėjimas gali privesti prie veiksmų, vadinamų „Cowboy coding” programavimu.

Šiam tyrimui pasirinktas „Agile software development” [8] metodas „Ribinis programavimas”. Tai vienas populiariausių lanksčiojo programavimo metodų. Jo pagrindiniai privalumai yra: pastovus grįžtamasis ryšys, testavimu paremtas programavimas, vientisa komanda (įtraukiant užsakovą), nenutrūkstamas procesas, nuolatinis integravimas, projektavimo korekcijos,

smulkūs išleidimai (releases), bendras supratimas, kodavimo standartai, programuotojo gerovė, pastovus tempas (jokių viršvalandžių). Tai įdomus metodas, išsiskiriantis iš įprastinių.

Siekiant užtikrinti panaudojamumą, kartu su ribiniu programavimu bus taikomi panaudojamumo užtikrinimo principai, aprašyti 2.2.4 skyrelyje.

2.5 Organizacijos veiklos analizė

Organizacijos, šiuo atveju tai bus restoranų tinklas, kur bus galima rezervuoti sau vietą iš anksto bet kuriame filiale, veiklos analizės tikslas yra nustatyti veiksmus, kurių pagalba vartotojas galės užsisakyti paslaugas. Sistemoje bus numatomi trys tipai vartotojų. Pats „silpniausias" yra vartotojo tipas, kuris galės peržiūrėti restorano svetainę bei pateikti rezervavimo užsakymą. Šiam tipui nėra būtinybės suteikti prisijungimo vardų, nes tai gali atbaidyti užsakovus. Vartotojui atsidarius užsakymo puslapį reikės pasirinkti sekančius veiksmus:

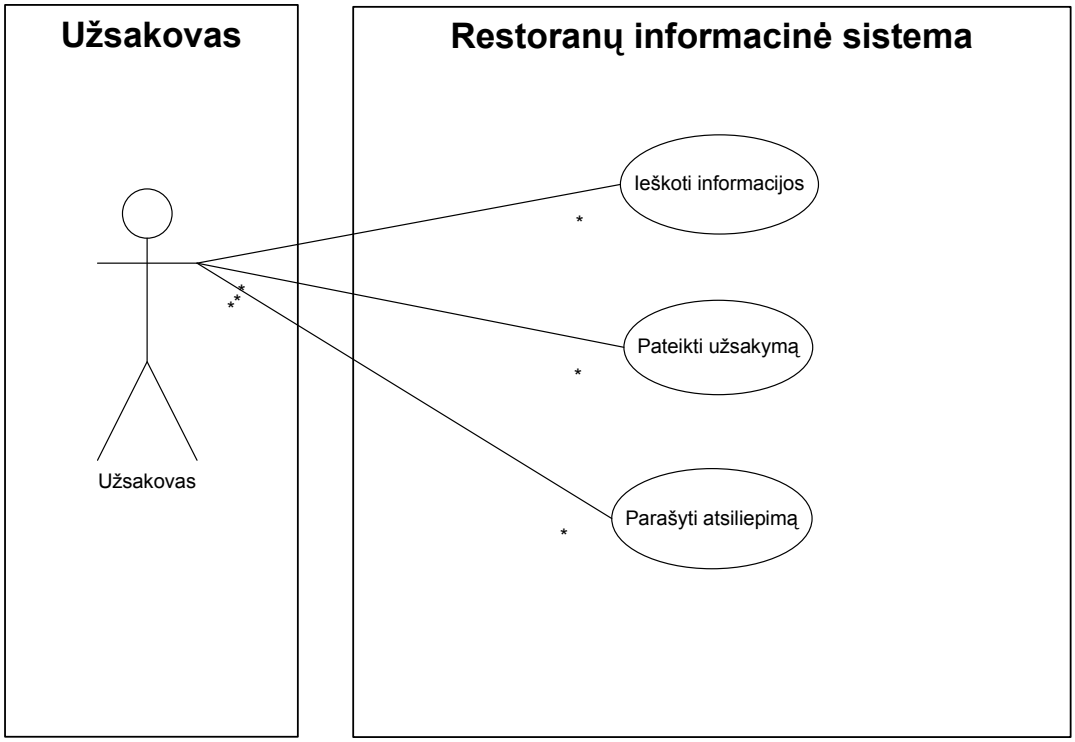
1. Miestas, kuriame pageidaujama atlikti rezervaciją.
2. Filialo pavadinimas, konkretus.
3. Laikas, nuo kada iki kada planuojama lankytis.
4. Pageidaujamų vietų skaičius
5. Kontaktiniai užsakovo duomenys

Nesant galimybės atlikti užsakymo norimu laiku, pageidaujamai asmenų grupei apie tai informuojama užsakymo metu. Tuomet galima pasirinkti kitą laiką arba vieną iš kitų restorano filialų. Kitu atveju sėkmingas rezervavimas registruojamas ir filialo skyriuje atsiranda informacija apie naują rezervavimą. Iškilus klausimams, susisiekiama su užsakovu nurodytu būdu.

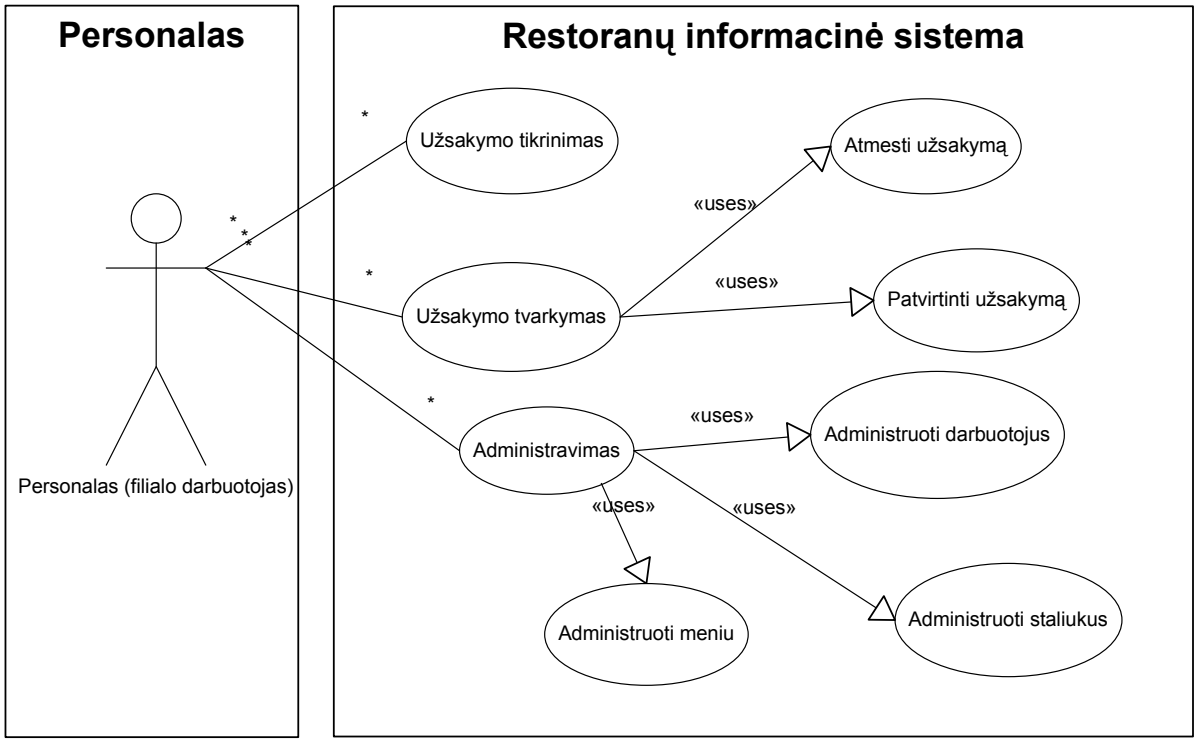
Kiekvienas restoranų tinklo filialas turi priėjimą prie sistemos kaip registruotas vartotojas. Filialo teisėmis galima valdyti tik tuos rezervavimus, kurie skirti tik tam filialui. Filialo vartotojas gali atlikti šiuos veiksmus:

- Panaikinti užsakymą, jei užsakymas akivaizdžiai neteisingas.
- Filialo vartotojas gali užregistruoti sistemoje naujus staliukus bei vietas, juos redaguoti, šalinti.
- Gali įterpti, redaguoti, šalinti darbuotojus

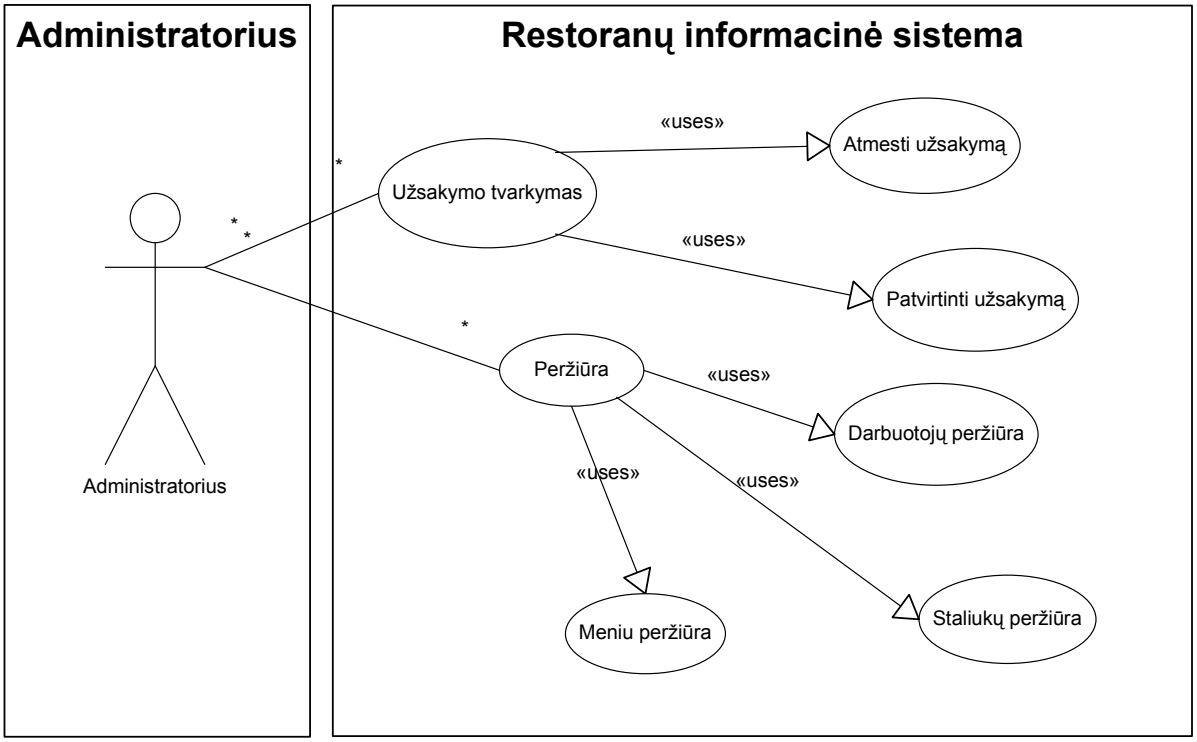
Sistemos „stipriausias" vartotojas yra administratorius. Jis gali matyti viską, kas vyksta sistemoje, valdyti filialų vartotojus, tačiau neturi teisės kištis į konkretaus filialo darbą. Tai būtų užsakymų, darbuotojų valdymo veiksmai.



(*2.1 pav.*) Restorano užsakymų sistemos užsakovo vartotojo panaudojimo atvejų modelis

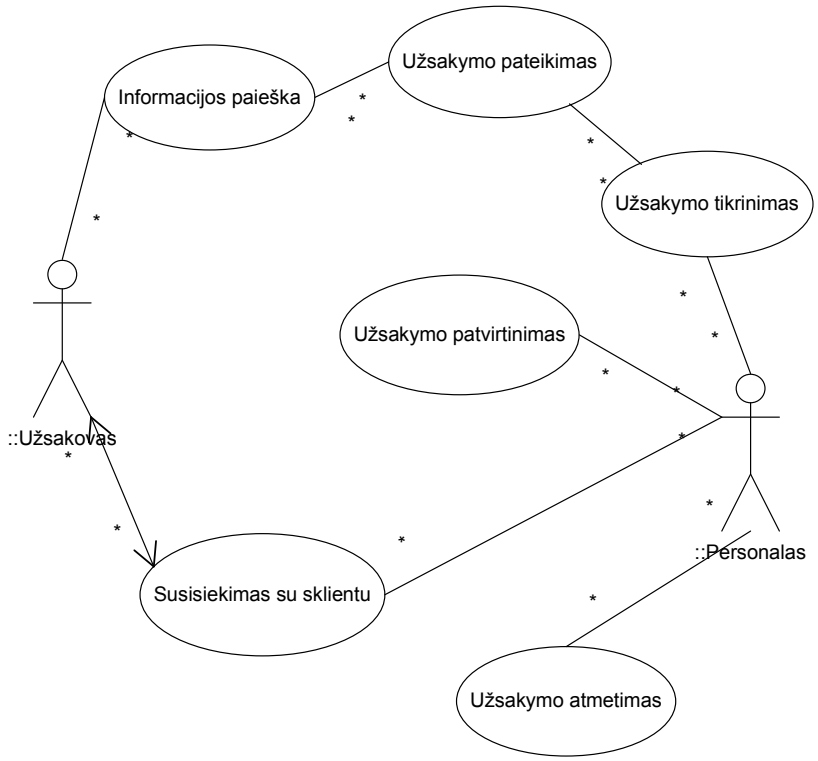


(*2.2 pav.*) Restorano užsakymų sistemos personalo vartotojo panaudojimo atvejų modelis



(*2.3 pav.*) Restorano užsakymų sistemos administratoriaus vartotojo panaudojimo atvejų modelis

Restorano užsakymų sistemos vartotojų ir panaudojimo atvejų modelyje (*2.1-2.3 pav.*) matomi veiksmai, kurie numatomi klientams, restoranų filialų personalui ir administratoriams.



(*3 pav.*) Užsakymų pateikimo proceso modelis

Sistemų būsenos diagramoje (*3 pav.*) matomas pateikimo proceso modelis. Užsakymas pradedamas formuoti nuo kliento, tada jis pateikiamas sistemai. Sistema užsakymą registruoja. Filialas, kuriam buvo skirtas užsakymas peržiūri užsakymą ir turi galimybę jį atmesti. Jei užsakymas atmetamas, jis „miršta" ir nėra patvirtinamas, tačiau bandoma pranešti klientui apie tai (tačiau tik tuo atveju, jei kontaktiniai duomenys yra teisingi). Jei užsakymas priimtinas, vadinasi sistema patvirtina užsakymą ir grįžtamasis ryšys grįžta atgal klientui apie sėkmingą užsakymo patvirtinimą.

2.6 Interneto ir WAP technologijų integravimo sprendimų analizė

Lentelėje pateikiami galimi sprendimai interneto sistemų ir bevielių įrenginių integravimui, jų teigiamos ir neigiamos savybės.

Pavadinimas	Šaltinis	Aprašymas
H2W servlet 1.84	www.servletsuite.com/servlets/h2w.htm	Ši programa veikia serverio pusėje ir jei svetainė pasiekiama mobiliuoju įrenginiu, ji konvertuoja html kodą į WML. Tai yra daroma kiekvieną kartą, kai aplankomas puslapis ir kiekvienam puslapiui tai daroma iš naujo ir iš naujo. O kas bus jei vienu metu daug vartotojų naršys? Tai gali turėti labai neigiamos įtakos serverio darbui, kuris skirtas anaiptol ne html kodo konvertavimui. Be to, šis sprendimas negarantuoja WML kodo korektiškumo, bei korektiško veikimo.
Mobile converter	http://www.cgiexpert.com/	Tai yra įskiepis į web serverį, kuris leidžia atpažinti ar kreipiamasi iš mobiliojo įrenginio. Tačiau kiekvieną kartą pasikreipus puslapio turinys generuojamas iš naujo. Teigiama, jog šis įrankis gali būti naudojamas kaip WML validavimo įrankis klaidoms aptikti, tačiau to nerekomenduojama daryti.
HTML2WML	http://html2wml.sourceforge.net/	Tai yra atviro kodo paketas, kuris dinamiškai konvertuoja HTML kodą į WML, kai kreipiamasi iš mobilaus įrenginio. Tačiau ši programa turi kelis nemažus trūkumis: nekonvertuoja .jpg ir .gif paveiksliukų į WBMP, ir susiduria su sunkumais su nuorodomis WML kode.
Java	www.java.com	Daugiau skirta mobilioms programoms ir mobiliems žaidimams, nei WAP puslapiams kurti.

2.7 Projekto tikslas ir jo pagrindimas, kokybės kriterijų apibrėžimas

Projektas yra orientuotas tiek į vartotoją, tiek į paslaugos teikėją. Jo tikslas yra suteikti galimybę užsisakyti paslaugas ne tik įprastu būdu per interneto naršyklę, tačiau ir per WAP sąsają. Tai duoda naudos paslaugos tiekėjui, nes yra papildomas būdas privilioti klientą naudojantis naujomis technologijomis. Vartotojo požiūriu taip pat yra labai patogu užsisakyti tam tikras paslaugas, šiuo atveju rezervuoti restorano staliuką net ir neturint kompiuterio šalia, o per WAP važiuojant automobiliu ar turint laisvą minutėlę.

Kokybiškam staliuko rezervavimui reikia užtikrinti tokius kriterijus:

- Paprastumas naudotis:
    - Minimalus veiksmų skaičius
    - Minimali hierarchija
    - Grįžimo į ankstesnį puslapį galimybė
- Korektiškas veikimas.
- Grįžtamasis ryšys (įspėjimai apie negalimus užsakymus).
- Galimybė pakeisti užsakymą, jei toks negalimas.

Kadangi WAP dėl mažų bevielių įrenginių resursų skirtas daugiausiai tekstinei informacijai vaizduoti, bus stengiamasi kreipti dėmesį į sistemos funkcionalumą, o ne į kosmetinį grožį. Tai suteiks WML kortelėms paprastumo ir aiškumo. Per WAP, taip pat bus galima naršyti restoranų tinklo žiniatinklyje ir skaityti informaciją, kuri bus publikuojama.

Pateikiama iliustracija (4 *pav.*), kuri atspindi priėjimo būdus prie sistemos. Pasiekiant sistemą per WAP, sistemos pasiekiamumas tiesiogiai priklauso nuo WAP ryšio patikimumo.

(*4 pav.*) Prieigos būdai prie sistemos.

2.8 Kompiuterizuojamos sistemos varianto parinkimas

Sistemoje bus kompiuterizuojama daugelis veiksmų. Vartotojo pusė, filialo dalis ir administravimo terpė skirsis viena nuo kitos savo funkcionalumu ir realizacijos priemonėmis. Vartotojas užsakymą galės pateikti ne tik įprastu būdu bet ir mobiliuoju įrenginiu per WAP. Pati sistema bus kuriama Microsoft .NET programų paketu. Vartotojui numatoma kompiuterizuoti šiuos uždavinius:

- Informacijos paieška.
- Užsakymo pateikimas (per WAP ir įprastiniu būdu).
- Atsiliepimo siuntimas.

Filialo tipo vartotojui kompiuterizuoti numatoma sekančius veiksmus, tačiau priėjimas nebus daugiamodalinis :

- Užsakymo priėmimas arba atmetimas.
- Staliukų administravimas.
- Darbuotojų administravimas.
- Filialo užsakymų istorija (administravimas nenumatomas).

Aukščiausias teises turinčiam administratoriaus vartotojui bus galima:

- Administruoti filialo vartotojus.
- Kiekvieno filialo peržiūrėjimas (administravimas nenumatomas).
- Filialo užsakymų istorija (administravimas nenumatomas).

2.9 Analizės išvados

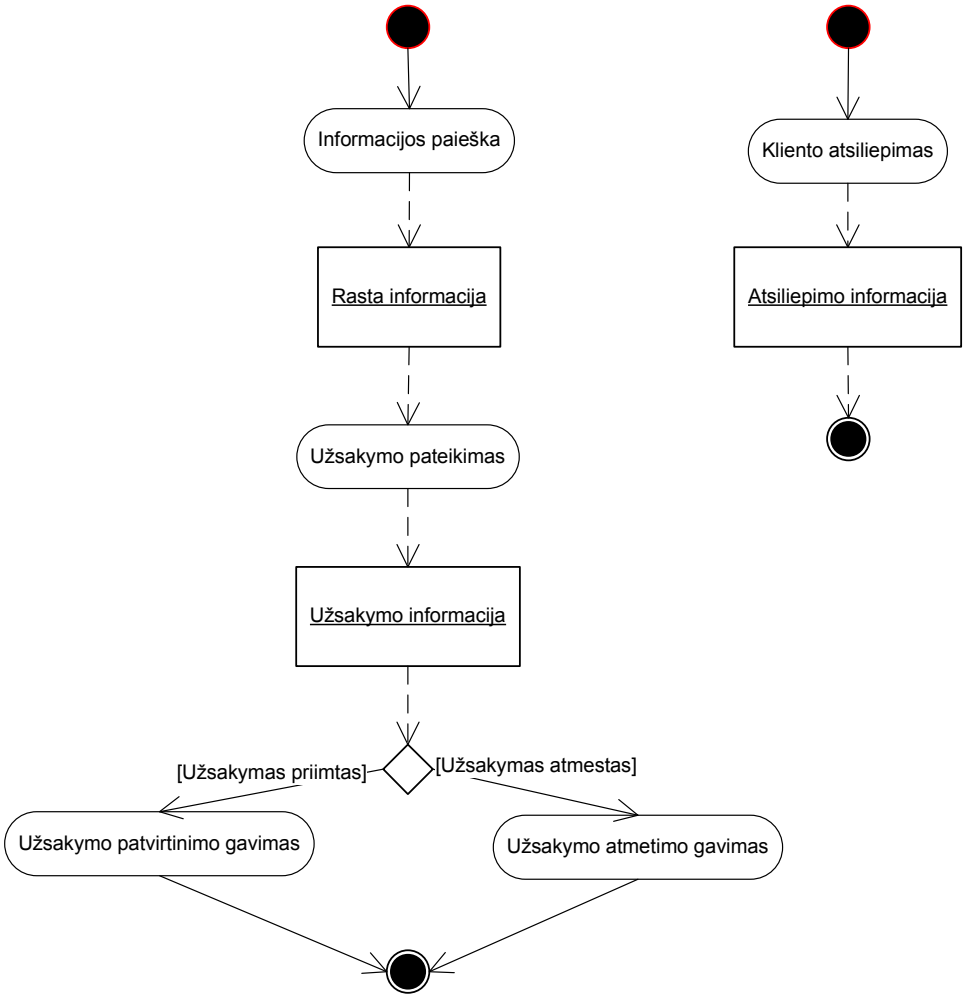
1. Išanalizuotos WAP galimybės ir pagal tai nuspręsta kokius veiksmus galės atlikti vartotojas žiniatinklyje su mobiliuoju įrenginiu.
2. Palyginus populiariausius projektavimo metodus, buvo nuspręsta pasirinkti „Agile software development" ribinio programavimo (Extreme Programming) metodą bei taikyti mobiliems įrenginiams skirtą projektavimo metodiką panaudojamumui užtikrinti.
3. Dėl savo didelių galimybių buvo pasirinkta Microsoft .Net paketas, kuriuo bus realizuota informacinė sistema.
4. Išanalizuotas kuriamos sistemos modelis, kuris leistų:
    a) Užsakymų registravimą, atsiliepimų siuntimą, teikiamų paslaugų peržiūrą ir kitos informacijos pasiekimą tiek per internetą, tiek mobiliuoju įrenginiu naudojant WAP technologijas.
    b) Surinkti duomenis apie klientų užsakymus, siekiant padidinti restorano pelningumą;
    c) Filialams patogiai pateikti informaciją apie rezervavimus, peržiūrėti statistinius duomenis apie rezervavimus;
    d) Administratoriams administruoti filialus bei stebėti kas vyksta kiekviename filiale atskirai;

3 Reikalavimų specifikacija ir analizė

3.1 Reikalavimų specifikacija

3.1.1 Klientas

Klientas yra mažiausiai teisių turintis, tačiau bene svarbiausias sistemos dalyvis. Jis, naudodamasis IS ieško informacijos tarp pateiktų variantų (5 pav.), kuriuos tvarko kiti sistemos vartotojai. Kliento vartotojas turi galimybę atlikti paieška ir išsirinkti sau maitą iš siūlomo meniu bei nusiųsti užsakymo duomenis sistemai, kad kiti sistemos vartotojai gautų užsakymo duomenis ir atliktų jiems skirtus veiksmus. Kliento veiklos diagrama:



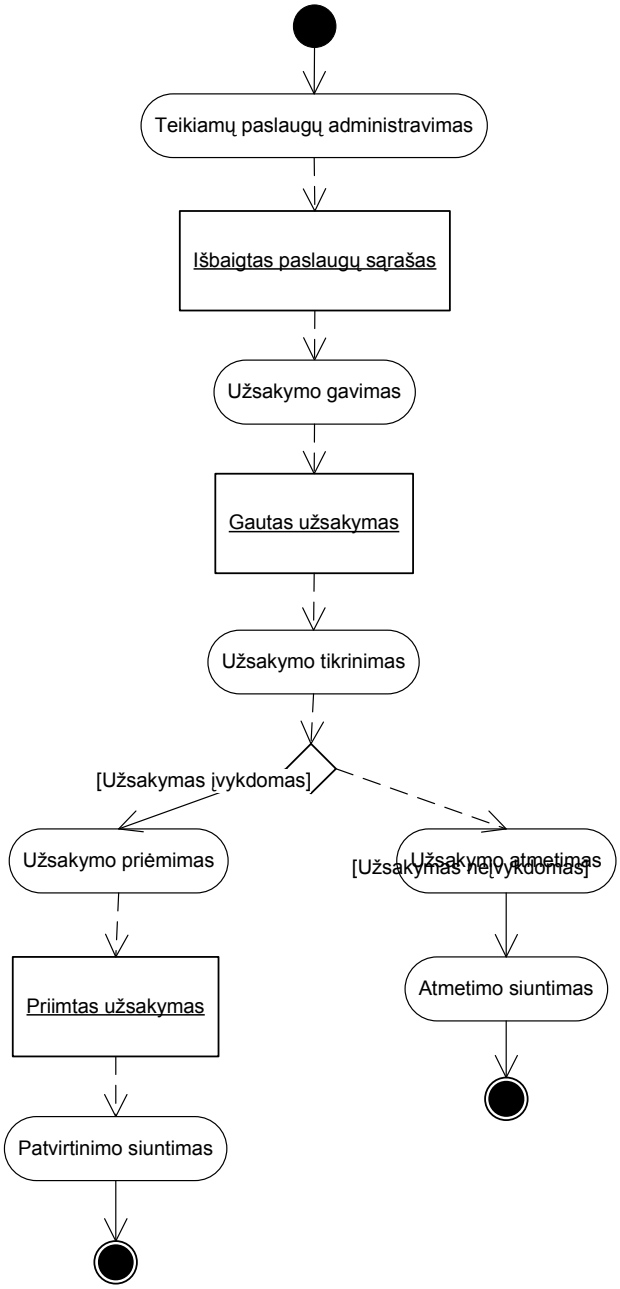
3.1. Kliento veiklos diagrama

Klientas šiuos veiksmus galės atlikti per įprastinę sąsają internetu, ir naudodamasis savo telefonu per WAP. Būtent dėl šio vartotojo yra reikalinga prieiga per WAP.

Dirbant kliento vartotojui, sistema atliekamas užsakymo užregistravimas. Užsakymui užregistruoti reikalinga matyti informaciją šiuo atveju meniu, tada apsisprendus suvesti norimas reikšmes į formą ir galiausiai registruoti užsakymą sistemoje. Pateikus užsakymą pranešama apie sėkmingą/nesėkmingą užregistravimą ir grįžtama atgal į meniu.

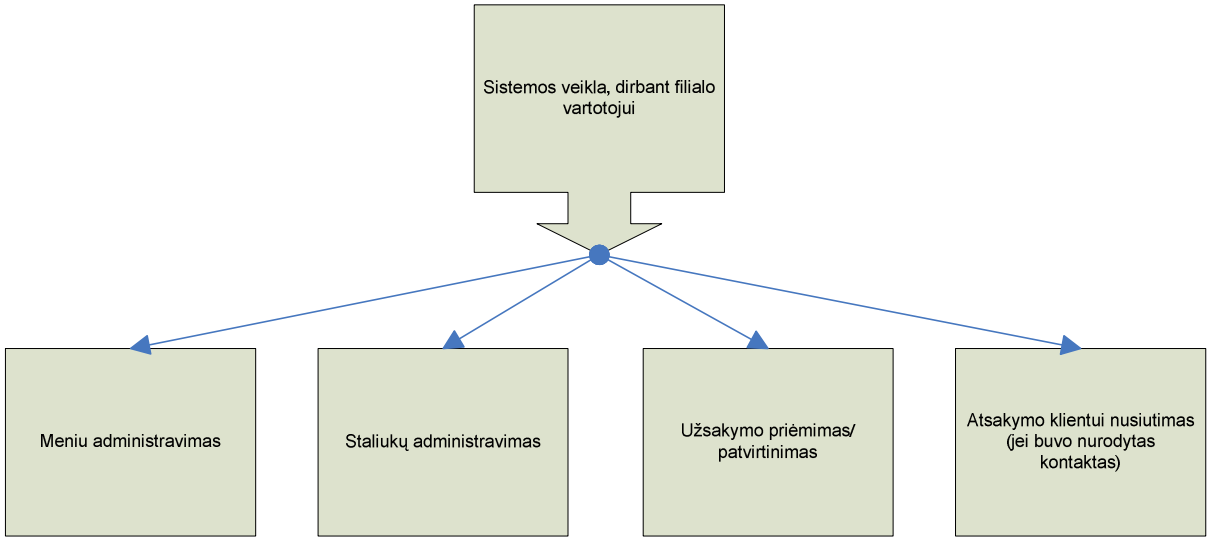
3.1.2 Restorano filialo vartotojas

Restorano filialo vartotojo funkcijos yra daug platesnės nei kliento (8 pav). Visų pirma tai yra jau registruotas vartotojas, į kurio numatomas funkcijas įeina šie veiksmai: staliukų administravimas, meniu administravimas, užsakymo priėmimas, užsakymo patvirtinimas. Visa tai šis vartotojas galės atlikti per sistemą įprastine sąsaja. Prieiga per WAP šiam vartotojų tipui nenumatoma, nes nėra tokio poreikio.



3.2 Filialo vartotojo veiklos diagrama

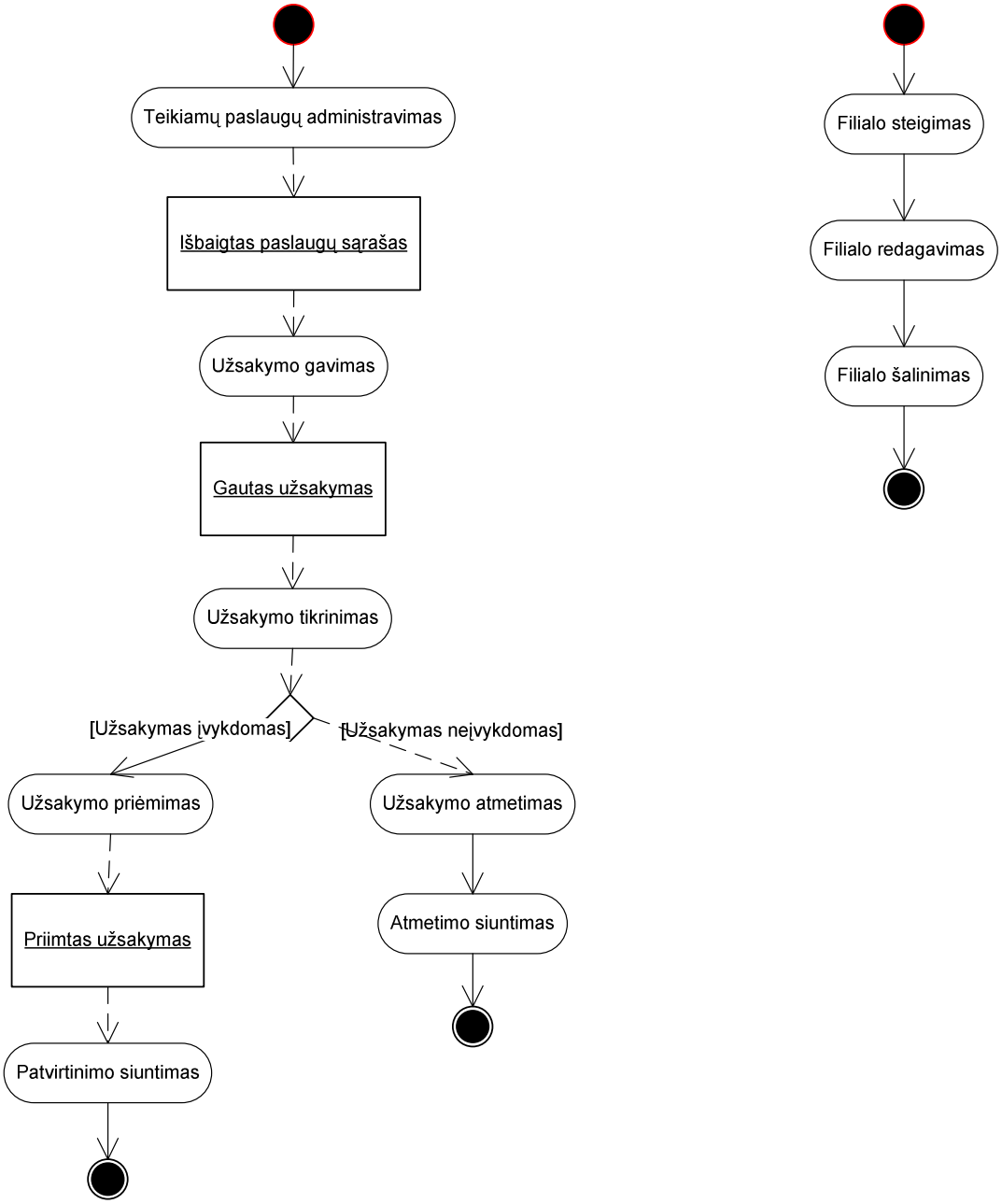
Filialo vartotojui dirbant sistema gali atlikti tokias veiklas: meniu administravimas, staliukų administravimas, užsakymo priėmimas, atsakymo klientui siuntimas (7 pav).



Filialo vartotojo interfeiso navigavimo planas (7 pav.)

3.1.3 Sistemos administratorius

Tai daugiausiai teisių turintis registruotas sistemos vartotojas, į kurio galimų atlikti veiksmų sąrašą pateka visi filialo vartotojo veiksmai (8 pav.). Beto, šis vartotojas dar gali administruoti filialus, t.y. juos steigti, redaguoti, trinti ir t.t.

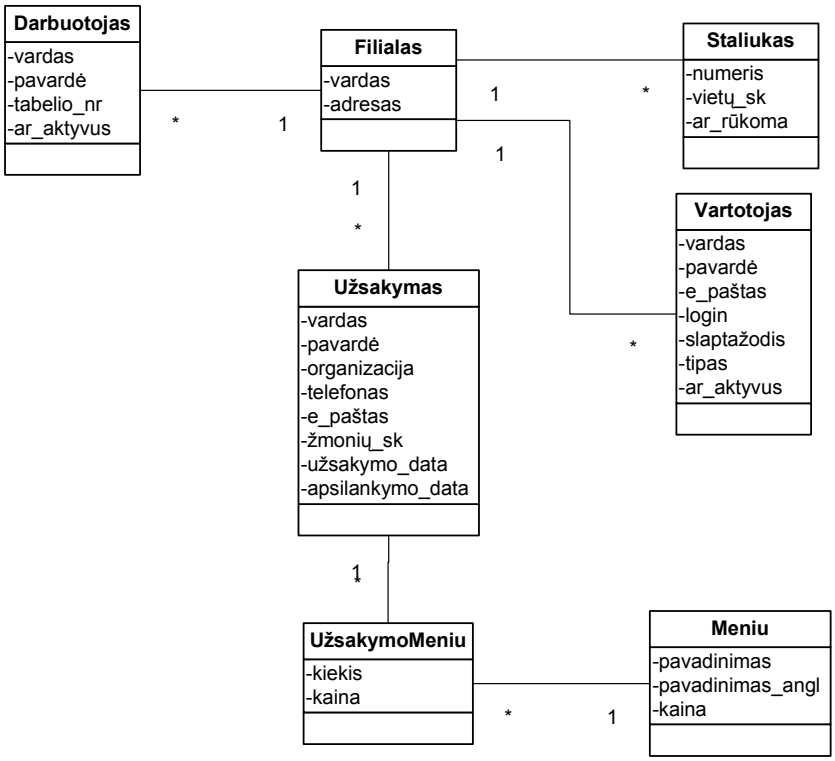


3.3 Administratoriaus veiklos diagramos

Sistemos administratoriui dirbant sistemos veikla yra labai panaši, tik prisideda filialų bei darbuotojų administravimo veiklos.

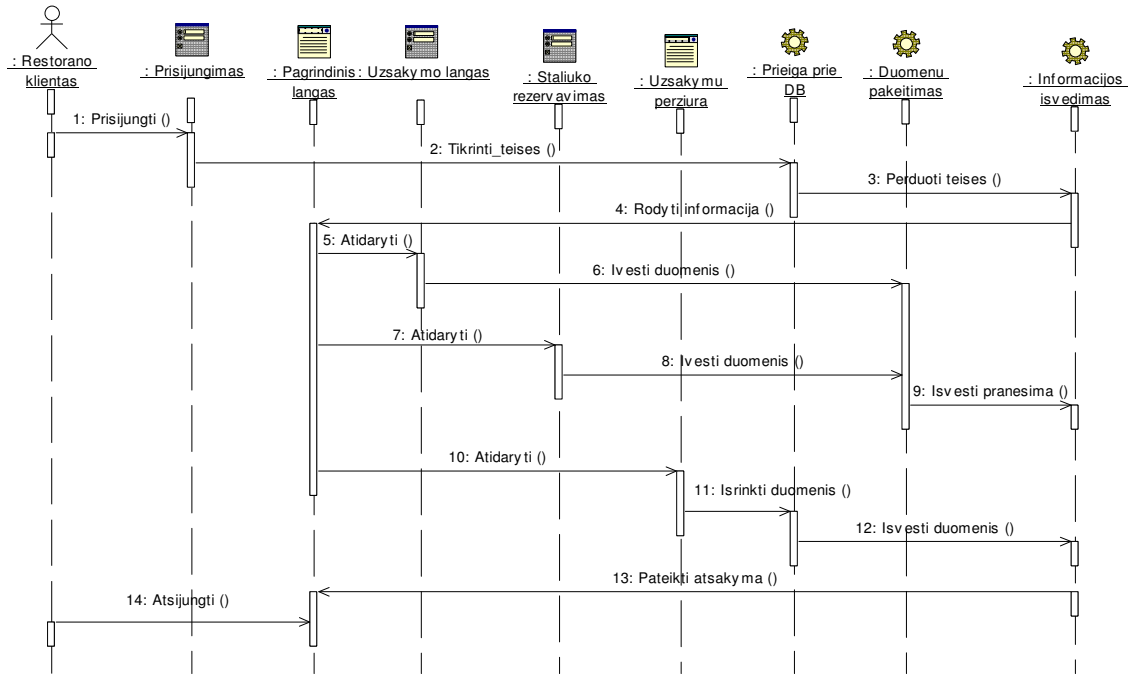
3.2 Dalykinės srities modelis

Esybių klasių diagramoje (9 pav), pavaizduotos numatomos sistemos klasės.



3.4 Dalykinės srities klasių diagrama

3.3 Kliento sekų diagrama



Kliento sekų diagrama (10 pav.)

3.4 Reikalavimų analizė

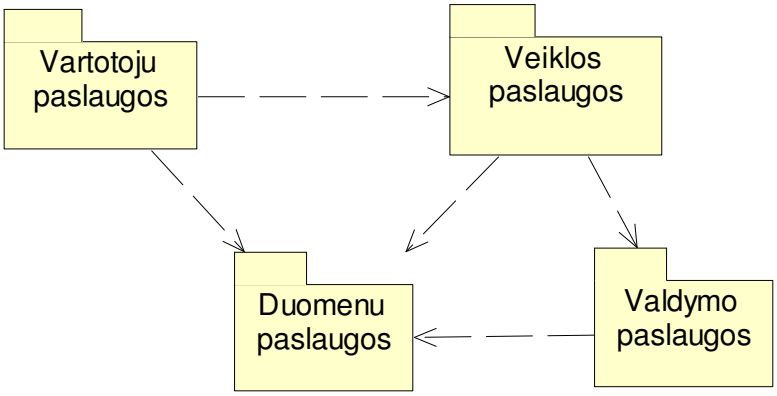
Pradiniai reikalavimai, išdėstyti ankstesniuose punktuose yra pakankamai aiškūs ir suprantami. Natūralu, jog projekto vykdymo eigoje reikalavimuose bus pasikeitimų, bet naudojant vikriuosius kūrimo metodus šie pakeitimai neatneš labai didelių nuostolių laiko ir projektavimo klausimais.

Aišku, jog daugiausiai dėmesio reikalauja kliento vartotojas, nes visa sistema yra orientuota į jį. Taip pat žiniatinklio pritaikymas prie WAP bus atliekamas tik kliento vartotojui. Iš reikalavimų suprantama, jog pirmiausia turi būti sukurta pilnai veikianti sistema, o tada atliekamas žiniatinklio pritaikymas kliento vartotojui.

Atsižvelgiant į sistemos nefunkcinius reikalavimus, svarbiausias dalykas yra funkcionalumas bei korektiškas veikimas. Klientui pateikus neteisingą užsakymą, sistema turi pranešti ir leisti užsakyti dar kartą. Filialo darbuotojui priėmus užsakymą sistema atsiunčia pranešimą, jei klientas pateikė kontaktinę informaciją.

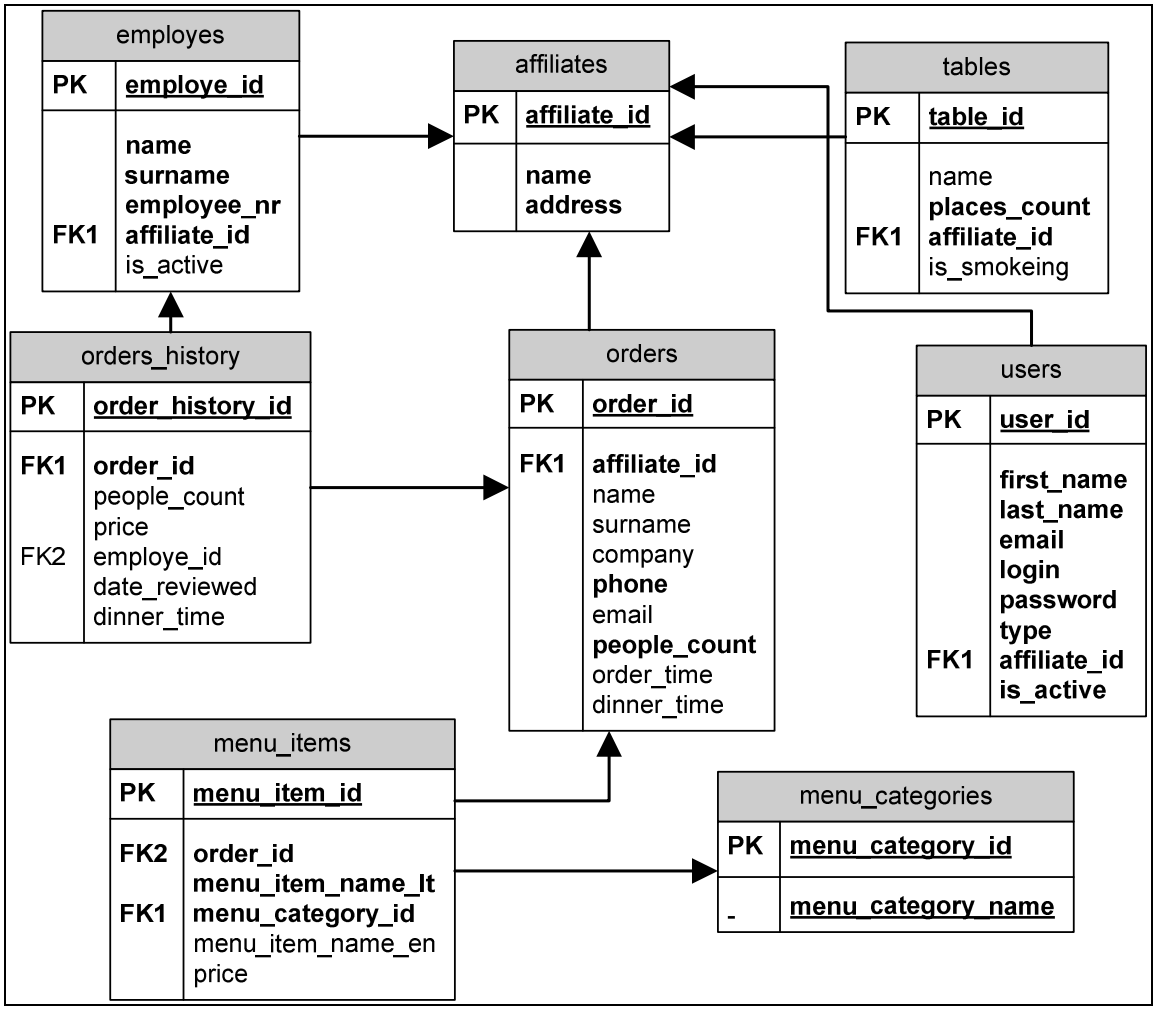
4 Projektas

4.1 Loginė sistemos architektūra



Loginė sistemos architektūra (11 pav.)

4.1.1 Duomenų bazės schema



Duomenų bazės schema (12 pav.)

Lentelė „employes“

Lauko pavadinimas	Duomenų tipas	Aprašymas
Employe_id	integer	Darbuotojo identifikatorius, unikalus, pirminis raktas
name	Nvarchar	Darbuotojo vardas
Surname	Nvarchar	Darbuotojo pavardė
Employe_nr	Nvarchar	Darbuotojo numeris, unikalus kiekvienam darbuotojui
Affiliate_id	Integer	Filialo identifikatorius

Is_active	bit	Informacinis laukas, nurodantis ar darbuotojas aktyvus

Lentelė „affiliates“

Lauko pavadinimas	Duomenų tipas	Aprašymas
affiliate_id	integer	Filialo identifikatorius, unikalus, pirminis raktas
name	Nvarchar	Filialo pavadinimas
address	Nvarchar	Filialo adresas

Lentelė „tables“

Lauko pavadinimas	Duomenų tipas	Aprašymas
table_id	integer	Staliuko identifikatorius, unikalus, pirminis raktas
name	Nvarchar	Staliuko pavadinimas
Places_count	integer	Vietų skaičius prie staliuko
Affiliate_id	Nvarchar	Filialo identifikatorius
Is_smoking	bit	Ar staliukas rokymo zonoje

Lentelė „orders_history“

Lauko pavadinimas	Duomenų tipas	Aprašymas
order_history_id	integer	Užsakymo istorijos identifikatorius, unikalus, pirminis raktas
People_count	nvarchar	Tikrasis žmonių kiekis, kuriam priimtas užsakymas
Price	Money	Sąskaita už čekį
Employe_id	int	Darbuotojo identifikatorius
Date_reviewed	datetime	Data, kai užsakymas buvo patvirtintas
Dinner_time	datetime	Pietų laikas

Lentelė „orders“

Lauko pavadinimas	Duomenų tipas	Aprašymas
order_id	integer	Užsakymo identifikatorius, unikalus, pirminis raktas
Affiliate_id	integer	Filialo identifikatorius
name	nvarchar	Užsakovo vardas
surname	nvarchar	Užsakovo pavardė
company	nvarchar	Įmonės pavadinimas (jei užsako jurininis asmuo)
Phone	nvarchar	Kontaktinis telefono nr.
email	nvarchar	El. Pašto adresas
People_count	nvarchar	Pageidaujamas žmonių kiekis
Order_time	datetime	Užsakymo laikas
Dinner_time	datetime	Pageidaujamas pietų laikas

Lentelė „users“

Lauko pavadinimas	Duomenų tipas	Aprašymas
user_id	integer	vartotojo identifikatorius, unikalus, pirminis raktas
firstname	nvarchar	Vartotojo vardas
lastname	nvarchar	Vartotojo pavardė
email	nvarchar	Vartotojo el. paštas
login	nvarchar	Prisijungimo vardas
password	nvarchar	Slaptažodis
type	integer	Vartotojo tipas (0 – administratorius, 1 – personalo darbuotojas)
Affiliate_id	integer	Filialo identifikatorius (jei vartotojo tipas = 1)
Is_active	bit	Ar vartotojas aktyvus

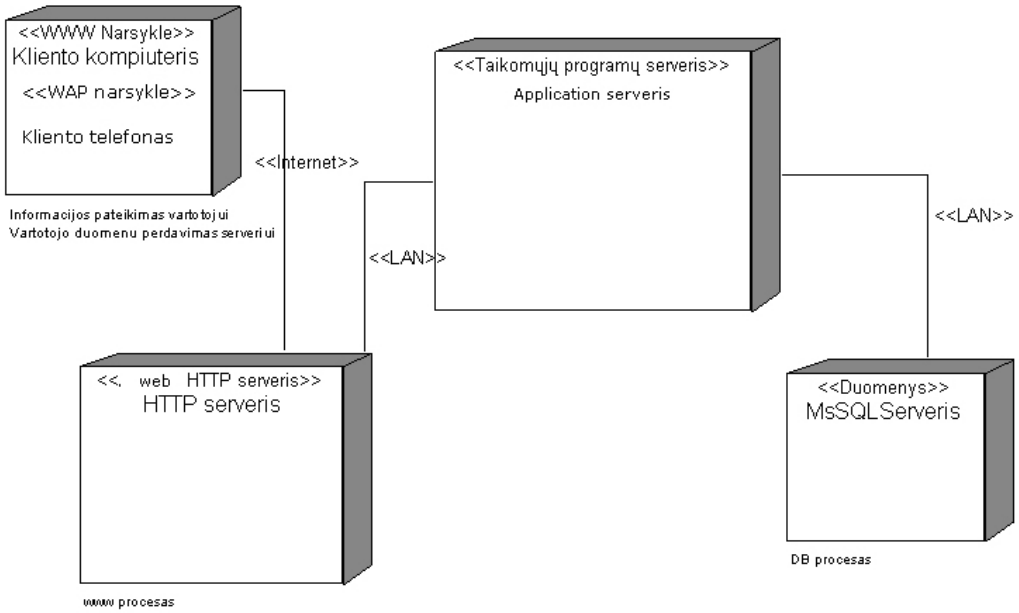
Lentelė „menu_items“

Lauko pavadinimas	Duomenų tipas	Aprašymas
Menu_item_id	integer	meniu identifikatorius, unikalus, pirminis raktas
Order_id	integer	Užsakymo identifikatorius
Menu_item_name_lt	nvarchar	Meniu įrašo pavadinimas, lietuviškai
Menu_item_name_lt	nvarchar	Meniu įrašo pavadinimas, angliškai
Menu_category_id	integer	Meniu kategorijos identifikatorius

Lentelė „menu_categories“

Lauko pavadinimas	Duomenų tipas	Aprašymas
Menu_category_id	integer	Meniu kategorijos identifikatorius, unikalus, pirminis raktas
Menu_category_name	integer	Meniu kategorijos pavadinimas

4.2 Sistemos įdiegimo schema



Sistemos įdiegimo schema (13 pav.)

4.3 Sistemos detalus projektas

Mano kuriama sistema – tai yra pavyzdys, kaip pritaikyti jau sukurtą .Net paketu informainę sistemą pritaikyti prie WAP bevielio ryšio įrenginiams. Bus kuriama restoranų IS, kuri bus pritaikyta prie WAP. Tam tikslui Microsoft turi „Microsoft Mobile Internet Toolkit", dar vadinamą „.Net Mobile", kuris kaip tik ir skirtas mobiliems įrenginiams. „.Net Mobile" yra ne kas ita kaip ASP.Net ir .Net Framework praplėtimas. Procesas, kai sistema naudojamasi moobiliu įreginiu atrodo taip:

- Mobilūs įrenginiai;
- Iternetas;
- Interneto informacijos servisas – IIS;
- .Net Framwork karkasas;
- ASP.NET;
- .NET Mobile.

Kuriant mobilias apikacijas, veismas yra toks:

- Sukuriamas ASP .Net puslapis;
- Įterpiamos mobilios bibliotekos „Include System.Mobile.UI";
- Sukuriamos mobilūs komponentai puslapiui (Mobile controls).

Mobilūs komponentai yra pagridinės mobilaus puslapio sudedamosios dalys. Šie komponentai yra labai panašūs į paprastus komponentus. Šis payzdys parodo kaip naudojami mobilūs komponentai:

```
<%@ Page
Inherits="System.Web.UI.MobileControls.MobilePage" %>
<%@ Register
TagPrefix="mob"
Namespace="System.Web.UI.MobileControls"
Assembly="System.Web.Mobile" %>
<mob:Form runat="server">
   <mob:Label runat="server">Labas</mob:Label>
</mob:Form>
```

Pavyzdys parodo, kad reikia naudoti mobilaus puslapio generavimą vietoj įprastinio. Čia yra panaudotas mobilūs Form ir Label komponentai. Jie aprašomi atitinkamai <mob:Form> ir <mob:Label>.

Štai WML kodo pavyzdys, kuris buvo sugeneruotas paleidus programą:

```
<?xml version='1.0'?>
<!DOCTYPE wml PUBLIC
'-//WAPFORUM//DTD WML 1.1//EN'
'http://www.wapforum.org/DTD/wml_1.1.xml'>
<wml>
<card>
<p>Labas</p>
</card>
</wml>
```

Mobile .Net patogumas yra tame, jog jis atpažįsta mobilų įrenginį ir sugeneruoja kodą būtent tokį, kurį tas įrenginys palaiko. Štai kaip atrodo tas pats kodas jį pasiekus kišeniiu kompiuteriu (Pocket PC):

```
<html>
<body><form id="ctrl1" name="ctrl1" method="post"
action="example.aspx">
<div>Hello W3Schools</div>
</form></body>
</html>
```

Kiekvienas mobilus puslapis privalo turėti bent vieną formą, kuri savyje gali turėti komponentus. Puslapis gali turėti ir kelias formas, bet kadangi mobilių įrenginių ekranai yra nedideli, visai normalu jog navigacija tarp formų bus realizuota elementariausiomis nuorodomis. Vienos formos pavyzdys:

```
<%@ Page
Inherits=
"System.Web.UI.MobileControls.MobilePage"%>
<%@ Register
TagPrefix="Mobile"
Namespace="System.Web.UI.MobileControls"
Assembly="System.Web.Mobile" %>
<Mobile:Form runat="server">
   <Mobile:Label runat="server">Hello W3Schools
   </Mobile:Label>
</Mobile:Form>
```

Dviejų formų pavyzdys:

```
<%@ Page
Inherits=
"System.Web.UI.MobileControls.MobilePage"%>
<%@ Register
TagPrefix="Mobile"
Namespace="System.Web.UI.MobileControls"
Assembly="System.Web.Mobile" %>
<Mobile:Form id="f1" runat="server">
 <Mobile:Label runat="server">Labas</Mobile:Label>
 <Mobile:Link runat="server" NavigateURL="#f2">2
 </Mobile:Link>
</Mobile:Form>
<Mobile:Form id="f2" runat="server">
 <Mobile:Label runat="server">Labas dar kartą
 </Mobile:Label>
 <Mobile:Link runat="server" NavigateURL="#f1">1
 </Mobile:Link>
</Mobile:Form>
```

Mobilių komponentų sąrašas ir paaiškinimai:

Komponentas	**Funkcija**
Command	Atlieka veiksmą
Form	Mobilių komponentų konteineris
Image	Aprašo paveiksliuką
Label	Aprašo tekstą
Link	Aprašo nuorodą
List	Aprašo sąrašą
MobilePage	Mobilių komponentų konteineris
ObjectList	Aprašo duomenų sąrašo objektą
Panel	Kitų mobilių komponentų konteineris
SelectionList	Pasirenkamas sąrašas
StyleSheet	Aprašo stilius, kurie pritaikomi komponentams
TextBox	Vienos eilutės įvedimo laukas

TextView	Daugelio eilučių įvedimo laukas

4.4 Sistemos realizacija

4.4.1 Sistemos veikimas, langai, vartotojo instrukcija

Restoranų užsakymo informacinė sistema tyri du interfeisus, tai – prieiga per WAP ir paprasta internetinė prieiga. Pirmiausiai apžvelgsime prieiga per WAP. Užsikrovus sistemą savo telefone, matomas pagrindinis užsakymo langas, kuriame reikia suvesti reikalingus duomenis į užsakymo formą. Privalomi laukai pažymėti žvaigždute (*). Suvedus duomenis, saudžiame mygtuką „Užsakyti“. Jums išvedamas pranešimas apie sėkmingą arba nesėkmingą užsakymą bei galimybė grįžti atgal į užsakymo formą.

Užsakovo vartotojo sąsaja. Užsakymo forma (14 pav.)

Užsakymas buvo sėkmingai
užregistruotas.
Greitu metu susisieksime su Jumis.

Naujas užsakymas

Personalas, prisijungęs prie sistemos, patenka į naujų užsakymų langą. Jame jis mato tik naujus užsakymus bei gali juos administruoti. Jei užsakymas klaidingas, arba duomenys nekorektiški, galima jį pašalinti. Norint patvirtinti užsakymą, reikia jį redaguoti. Pagal sistemos idėją, šiame žingsnyje personalo darbuotojas turėtų kontaktuoti su užsakovu pagal jo pateiktus duomenis ir tik tada tvirtinti užsakymą.

Nauji užsakymai Užsakymų istorija

Naujų užsakymų sąrašas

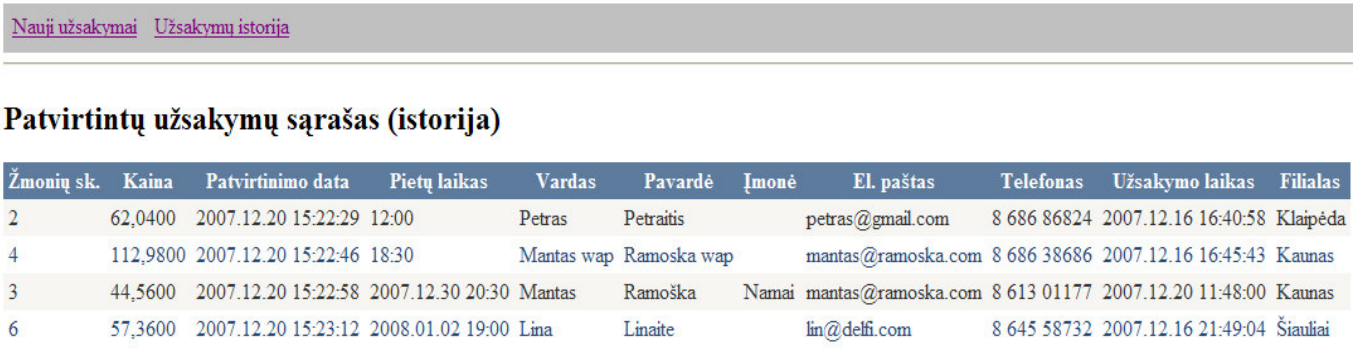
Vardas	Pavardė	Įmonė	El. paštas	Telefonas	Žmonių skaičius	Užsakymo data	Pietų data (pageidaujama)	Filialo pavadinimas		
Petras	Petraitis		petras@gmail.com	8 686 86824	2	2007.12.16 16:40:58	12:00	Klaipėda	Redaguoti	Trinti
Mantas wap	Ramoska wap		mantas@ramoska.com	8 686 38686	4	2007.12.16 16:45:43	18:30	Kaunas	Redaguoti	Trinti
Lina	Linaite		lin@delfi.com	8 645 58732	6	2007.12.16 21:49:04	2008.01.02 19:00	Šiauliai	Redaguoti	Trinti
Mantas	Ramoška	Namai	mantas@ramoska.com	8 613 01177	3	2007.12.20 11:48:00	2007.12.30 20:30	Kaunas	Redaguoti	Trinti

Personalo vartotojo sąsaja. Užsakymų sąrašas (14 pav.)

Redaguojant užsakymą matomi užsakymo duomenys. Personalo darbuotojas turi galimybę redaguoti duomenis, nes susisiekus su užsakovu duomenys gali keistis.

Personalo vartotojo sąsaja. Užsakymų patvirtinimas (15 pav.)

Patvirtinus užsakymą jis atsiduria užsakymų istorijoje.



Žmonių sk.	Kaina	Patvirtinimo data	Pietų laikas	Vardas	Pavardė	Įmonė	El. paštas	Telefonas	Užsakymo laikas	Filialas
2	62,0400	2007.12.20 15:22:29	12:00	Petras	Petraitis		petras@gmail.com	8 686 86824	2007.12.16 16:40:58	Klaipėda
4	112,9800	2007.12.20 15:22:46	18:30	Mantas wap	Ramoska wap		mantas@ramoska.com	8 686 38686	2007.12.16 16:45:43	Kaunas
3	44,5600	2007.12.20 15:22:58	2007.12.30 20:30	Mantas	Ramoška	Namai	mantas@ramoska.com	8 613 01177	2007.12.20 11:48:00	Kaunas
6	57,3600	2007.12.20 15:23:12	2008.01.02 19:00	Lina	Linaite		lin@delfi.com	8 645 58732	2007.12.16 21:49:04	Šiauliai

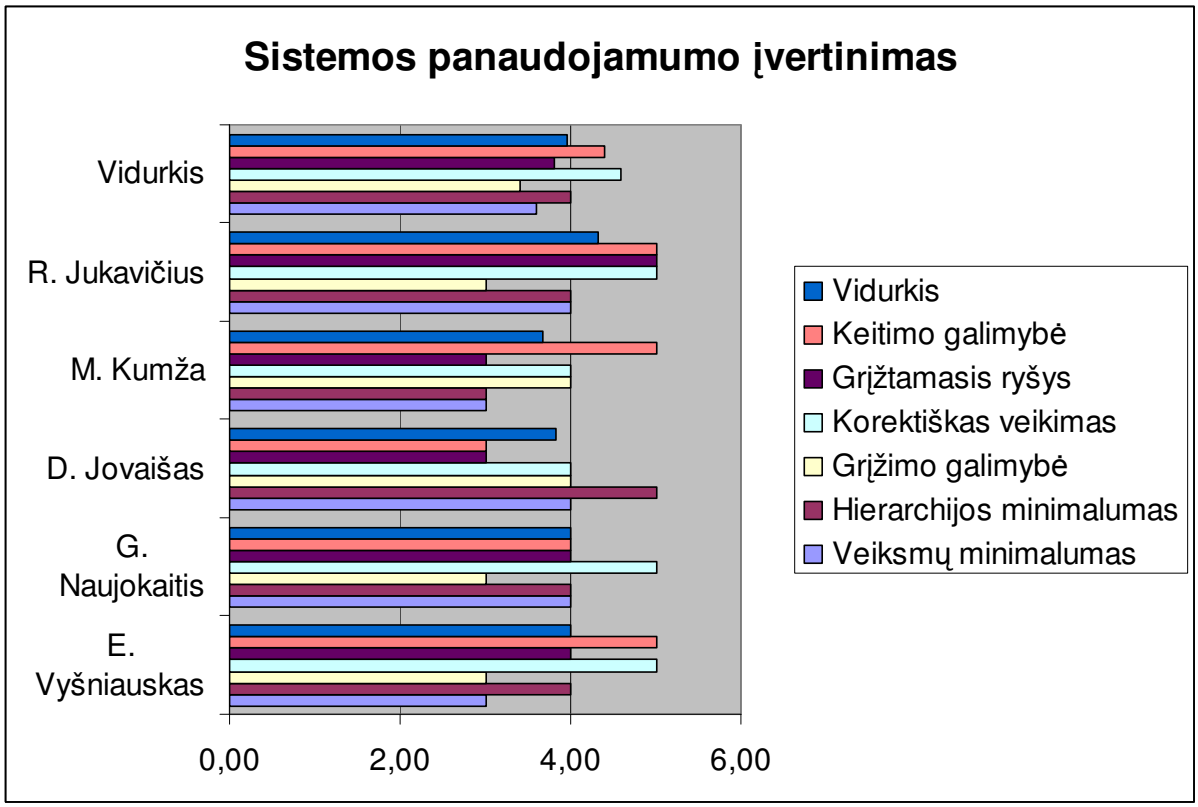
Personalo vartotojo sąsaja. Užsakymų istorija (16 pav.)

4.5 Sistemos panaudojamumo įvertinimas

Kuriant sistemą, didelis dėmesys buvo skiriamas sistemos panaudojamumui užtikrinti [21]. Buvo atsižvelgta į tris pagrindinius panaudojamumo įvertinimo kriterijus: ekrano dydis, navigacija ir įvesties budai. Užsakovui atsidarius užsakymo langą, visa užsakymo forma netelpa ekrane, dėl didelio reikalingų duomenų kiekio. Tačiau tai didelio sunkumo nesukelia, nes: įvesties laukai yra tekstinio tipo ir nesudėtingi vartotojui. Vartotojui tiesiog reikia įvesti tekstą, o užsakymo forma nesunkiai galima paslinkti aukštyn arba žemyn.

Sistemos panaudojamumo įvertinimą atliko 5 vartotojai (penkiabalė sistema)

Kokybės kriterijus	E. Vyšniauskas	G. Naujokaitis	D. Jovaišas	M. Kumža	R. Jukavičius	Vidurkis
Veiksmų minimalumas	3	4	4	3	4	3.6
Hierarchijos minimalumas	4	4	5	3	4	4
Grįžimo galimybė	3	3	4	4	3	3,4
Korektiškas veikimas	5	5	4	4	5	4,6
Grįžtamasis ryšys	4	4	3	3	5	3,8
Keitimo galimybė	5	4	3	5	5	4,4
Vidurkis	4	4	3,83	3,67	4,33	3,97



Sistemos panaudojamumo įvertinimas (17 pav.)

4.6 Išvados

1. ASP.Net technologijos suderinamumo su bevieliais mobiliais įrenginiais analizė parodė, kad suderinamumą galima pasiekti naudojant ASP.Net mobiliąsias bibliotekas.
2. Mobilių paslaugų kūrimo problemų ir galimų kūrimo metodų analizės metu nustatyta, kad mobilias paslaugas geriausia kurti naudojant ribinį programavimą ir panaudojamumui orientuotą kūrimą.
3. Kadangi panaudojamumui orientuotas kūrimas reikalauja nuolat vertinti sistemos panaudojamumą, jam matuoti buvo pasirinkta 5 balų skalė.
4. Sistemos realizacijos išbandymas parodė, kad kelti reikalavimai tenkinami, darbo tikslas pasiektas.
5. Galutinės realizacijos panaudojamumo įvertinimas parodė, kad sukurtos mobilios restoranų užsakymo paslaugos tenkina vartotojų reikalavimus panaudojamumui.

4.7 Literatūra

Analizuotant naudotasi šia literatūra:

1. Internetinė enciklopedija Vikipedija, informacija apie Agile metodus http://en.wikipedia.org/wiki/Agile_software_development [žiūrėta 2006 11 14]
2. Informacija apie Agile software development metodus http://www.agilealliance.org/ [žiūrėta 2006 11 18].
3. www.W3.org WEB konsorciumo WAP pristatymas http://www.w3schools.com/wap/wap_intro.asp [žiūrėta 2006 10 05]
4. Straipsnis „Methods for quantitative usability requirements: a case study on the development of the user interface of a mobile phone". Autoriai: Timo Jokela, Jussi Koivumaa, Jani Pirkola, Petri Salminen, Niina Kantola. Data: 2005 10 08.
5. Informacija apie „MS .Net" programų paketą http://www.microsoft.com/net/ [žiūrėta 2006 12 05]
6. www.W3.org WEB konsorciumo informacija apie WMLScript kalbą http://www.w3schools.com/wmlscript/default.asp [žiūrėta 2006 10 17]
7. Internetinė enciklopedija Vikipedija, informacija apie Agile metodą Ribinis programavimas http://en.wikipedia.org/wiki/Extreme_Programming [žiūrėta 2006 11 29]
8. Straipsnis „What is Agile software development" http://www.stsc.hill.af.mil/crosstalk/2002/10/highsmith.html [žiūrėta 2006 12 20]
9. Internetinė enciklopedija Vikipedija, informacija apie PĮ kūrimo procesus http://en.wikipedia.org/wiki/Software_development_process [žiūrėta 2006 12 12]
10. Internetinė enciklopedija Vikipedija, Microsof .Net paketo specifikacijos ir aprašymas http://en.wikipedia.org/wiki/Microsoft_Net [žiūrėta 2006 12 28]
11. W3Schools .Net pritaikymas mobiliems įrenginiams http://www.w3schools.com/dotnetmobile/default.asp [žiūrėta 2007 09 27]
12. MSDN biblioteka. „Inside the ASP.NET Mobile Controls" http://msdn2.microsoft.com/en-us/library/08e3b0ck(VS.71).aspx [žiūrėta 2007 11 18].
13. Roadmap for Mobile Web Development with ASP.NET http://www.asp.net/mobile/mobileroadmap.aspx?tabindex=3&tabID=44 [žiūrėta 2007 04 30].
14. The .NET Compact Framework vaizdo pristatymas http://msdn.microsoft.com/archive/default.asp?url=/archive/en-us/theshow/Episode029/default.asp [žiūrėta 2007 05 11].

15. ASP.NET Tutorial - with ASP.NET 2.0 http://www.w3schools.com/aspnet/default.asp [žiūrėta 2007 05 19].
16. Mobile Controls Reference http://msdn2.microsoft.com/en-us/library/4e0ewas2(VS.71).aspx [žiūrėta 2007 05 22].
17. On the Road with .NET Mobile Controls http://www.sitepoint.com/article/road-net-mobile-controls [žiūrėta 2006 06 05]
18. Fundamentals of Microsoft .NET Compact Framework Development for the Microsoft .NET Framework Developer http://msdn2.microsoft.com/en-us/library/aa446549.aspx [žiūrėta 2007 06 12]
19. Mobile Web User Controls http://www.dotnetforce.com/Content.aspx?t=a&n=177 [žiūrėta 2007 10 04]
20. Personalizing Mobile Web Applications http://www.dotnetforce.com/Content.aspx?t=a&n=165 [žiūrėta 2007 11 07]
21. Straipsnis. Methods for quantitative usability requirements: a case study on the development of the user interface of a mobile phone. Autoriai: Timo Jokela, Jussi Koivumaa, Jani Pirkola, Petri Salminen, Niina Kantola [žiūrėta 2007 10 14]
22. Straipsnis. Getting the Right Design and the Design Right: Testing Many Is Better Than One. Autoriai: Maryam Tohidi, William Buxton, Ronald Baecker, Abigail Sellen. [žiūrėta 2007 12 05]
23. Straipsnis. User Profiles and User Requirements in Mobile Services. Autoriai: Xiaosong Zheng and Zheying Zhang. [žiūrėta 2007 11 15]