Rohit Björkqvist

MATKANSEURANTASOVELLUS SYMBIAN ANNA -ALUSTALLE

Opinnäytetyö
KESKI-POHJANMAAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Helmikuu 2012

KESKI-POHJANMAAN AMMATTIKORKEAKOULU
MELLERSTA ÖSTERBOTTENS YRKESHÖGSKOLA

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Yksikkö Tietotekniikan yksikkö	**Aika** Tammikuu 2012	**Tekijä/tekijät** Rohit Björkqvist
Koulutusohjelma Tietotekniikan koulutusohjelma		
Työn nimi Matkanseurantasovellus Symbian Anna -alustalle		
Työn ohjaaja Sakari Männistö		**Sivumäärä** 31 + 2 liitettä
Työelämäohjaaja Petri Saviranta		

Tässä opinnäytetyössä on perehdytty Qt-sovelluskehitysympäristöön ja sillä tehtyyn matkanseurantasovellukseen. Tavoitteena oli toteuttaa karttasovellus matkanseurantajärjestelmällä, joka lähettää paikannustietoja palvelimelle. Opinnäytetyön toimeksiantajana toimi Petri Saviranta.

Työn alussa käydään läpi sovelluskehityksessä käytettyjä työkaluja, jotka ovat osana Qt:n sovelluskehitystyökalupakettia. Seuraavaksi käydään läpi opinnäytetyönä tehtyä sovellusta. Matkanseurantasovelluksen kehitys esitetään vaiheittain suunnittelusta valmiiseen sovellukseen. Työn lopussa tarkastellaan saavutettuja tavoitteita ja ongelmia, joita työn aikana on ilmennyt, sekä pohditaan sovelluksen jatkokehitystä.

Asiasanat
mobiilisovellus, ohjelmointi, Qt, sovelluskehitys, Symbian

CENTRAL OSTROBOTHNIA
UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

CENTRAL OSTROBOTHNIA UNIVERSITY OF APPLIED SCIENCES	**Date** January 2012	**Author** Rohit Björkqvist
Degree programme Information Technology		
Name of thesis Trip tracking application for Symbian Anna platform		
Instructor Sakari Männistö		**Pages** 31 + 2 Appendices
Supervisor Petri Saviranta		

This thesis is about Qt's application development environment and a trip tracking application created with it. The aim of this thesis was to develop a map application with the trip tracking system which sends GPS data from the mobile device to a server. The thesis was commissioned by Petri Saviranta.

The first part of the thesis consisted of information about the tools which are a part of the Qt source development kit. Next section described the map application. The development process of the application was presented in different stages from planning to the final product. The last part of the thesis focused on the objectives that were achieved, the problems during the development cycle and the future development of the application.

Key words
mobile application, programming, Qt, software development, Symbian

TIIVISTELMÄ
ABSTRACT
SISÄLLYS

1 JOHDANTO 1

2 Qt 2
2.1 Qt:n historia 2
2.2 Signal & Slot -järjestelmä 3
2.3 Qt SDK 4
2.3.1 Qt Creator 4
2.3.2 Qt Quick -käyttöliittymäkomponentit 5
2.3.3 Qt Simulator 5
2.3.4 Qt Mobility ja Location-kirjasto 6
2.4 Qt-dokumentaatio - Maps Demo 7

3 MATKALASKUKSI-SOVELLUKSEN MÄÄRITTELY 9

4 MATKALASKUKSI-SOVELLUKSEN SUUNNITTELU 10

5 MATKALASKUKSI-SOVELLUKSEN TOTEUTUS 11
5.1 Käyttöliittymä 11
5.1.1 Päävalikko 13
5.1.2 Karttapisteen hallintavalikko 14
5.1.3 Käyttöliittymän muut osat 15
5.2 Yleinen toiminta 17
5.2.1 Käynnistys 17
5.2.2 Päivitys 18
5.2.3 Asetukset 19
5.2.4 Piilotus 20
5.2.5 Lopetus 20
5.3 Kartan toiminta 20
5.3.1 Karttapisteiden hallinta 20
5.3.2 Osoitehaku 22
5.3.3 Navigointi 23
5.4 Matkanseurantajärjestelmän toiminta 24
5.4.1 Reittitunnuksen haku 24
5.4.2 Sijaintipisteiden kerääminen 24
5.4.3 Palvelimelle lähettäminen 25
5.4.4 Aikakatkaisu 25
5.4.5 Offline-tila 26

6 MATKALASKUKSI-SOVELLUKSEN TESTAUS 27
6.1 Sovelluksen testaus 27
6.2 Virheiden etsintä 28

7 MATKALASKUKSI-SOVELLUKSEN JATKOKEHITYS 29

8 JOHTOPÄÄTÖKSET 30

LÄHTEET **31**
LIITTEET
LIITE 1. Tietojen kerääminen
LIITE 2. Tietojen lähettäminen

KUVIOT
KUVIO 1. Signal & Slot -järjestelmä 3
KUVIO 2. Qt Creator -ohjelmointiympäristö 4
KUVIO 3. Qt Simulator 5
KUVIO 4. Maps Demo -ohjeet, osa 1 8
KUVIO 5. Suunniteltu käyttöliittymä 10
KUVIO 6. Toteutettu käyttöliittymä 10
KUVIO 7. Pystytila 12
KUVIO 8. Vaakatila 12
KUVIO 9. Kohdistimen tiedot 12
KUVIO 10. Kartan zoomauspainikkeet 12
KUVIO 11. Päävalikko 13
KUVIO 12. Karttapisteen hallintavalikko 14
KUVIO 13. Matkanseurantajärjestelmän tiedot 15
KUVIO 14. Virheilmoitus 16
KUVIO 15. Tilapalkki 16
KUVIO 16. Automaattinen päivitysten tarkistus 18
KUVIO 17. Sovelluksen asetukset 19
KUVIO 18. Karttapisteen tiedot 21
KUVIO 19. Osoitteiden haku 22
KUVIO 20. Osoitehausta lisätty karttapiste 22
KUVIO 21. Navigointi 23
KUVIO 22. Reitin lisääminen ja matkanseuranta 23
KUVIO 23. Palvelimelle lähettäminen 25
KUVIO 24. Valikko simulaattorissa 27
KUVIO 25. Virheiden etsintätila Qt Creator -työkalussa (Debug mode) 28

TAULUKOT
TAULUKKO 1. Työssä käytetyt Location-rajapinnan luokat 6
TAULUKKO 2. Kartan merkinnät 17

1 JOHDANTO

Nykyään yrityksissä ja muissa organisaatioissa tehdään paljon matkalaskuja ja niiden tekemiseen kuuluu huomattavan paljon työaikaa. Apuun on kehitetty erilaisia sovelluksia, yksi niistä on sovellus nimeltä Matkalaskuksi. Matkalaskuksi sovellus käyttää matkapuhelimen paikannuspalveluja ja lähettää reittipisteiden tietoja palvelimelle. Palvelin käsittelee nämä tiedot, jonka jälkeen käyttäjälle luodaan tiedot matkasta. Tämän avulla käyttäjä säästyy aikavievältä työltä, jossa hän joutuu itse kirjoittamaan matkalaskunsa.

Matkalaskuksi sovellus ei sisällä tukea Nokian uusimmille Symbian käyttöjärjestelmille, joten tämän opinnäytetyön tavoitteena on toteuttaa Matkalaskuksi sovellus Symbian Anna käyttöjärjestelmälle. Työssä ei hyödynnetä jo olemassa olevaa Matkalaskuksi sovellusta, vaan sovellus luodaan alusta alkaen eli rakennetaan toimiva matkanseurantajärjestelmä ja mahdollisuuksien mukaan monipuolinen karttapalvelu. Tavoitteisiin kuuluu myös käyttöliittymän tekeminen mahdollisimman helppokäyttöiseksi. Tämän työn toimeksiantajana toimii Petri Saviranta.

Työn alussa käydään läpi eri työkalut mitä Qt matkapuhelinsovelluksen kehittämiseen tarvitaan. Qt:n sovelluskehitystyökalupaketin tärkeimmät osat käydään läpi lyhyesti. Seuraavaksi keskitytään opinnäytetyönä tehtävään Matkalaskuksi sovellukseen. Sovelluksen kehitys käydään läpi vaiheittain eli sovelluksen määrittelystä valmiiseen tuotteeseen. Lopussa pohditaan sovelluksen kehitystyötä ja mitkä asetetuista tavoitteista saavutettiin.

2 Qt

Qt on kehitysympäristö alustariippumattomien sovellusten ja käyttöliittymien tuotantoon. Qt sisältää ohjelmointirajapinnan C++-ohjelmointiin sekä työkalut käyttöliittymien luomiseen.

2.1 Qt:n historia

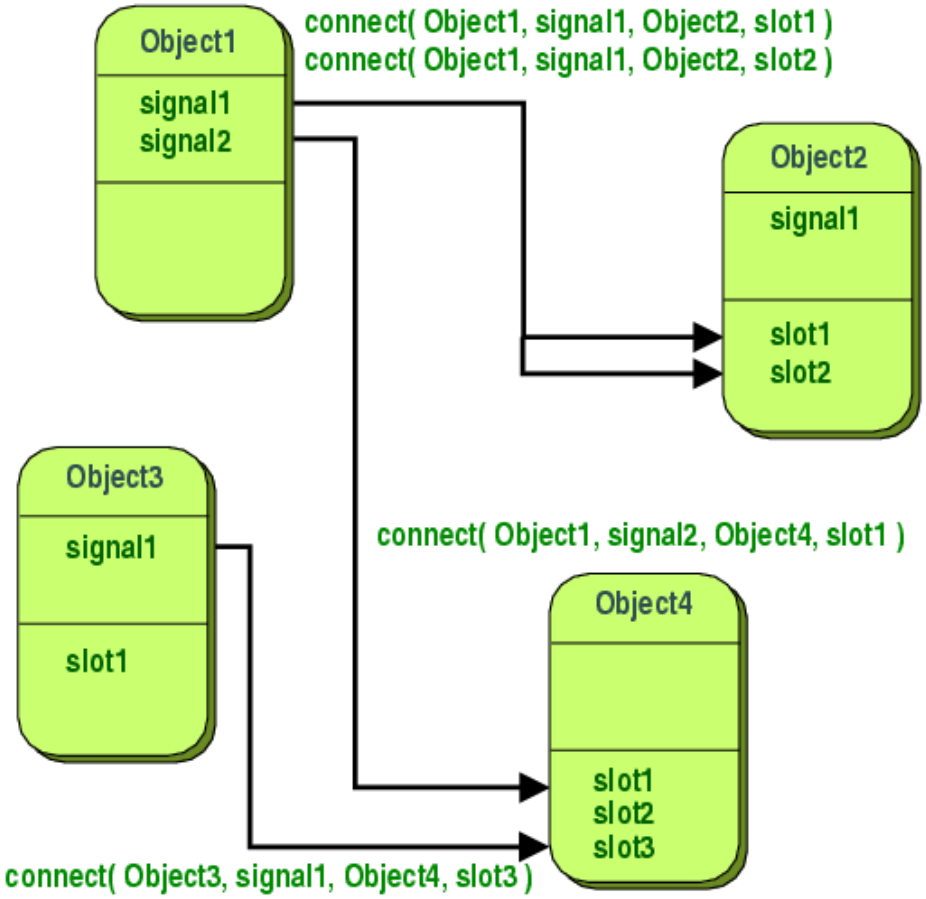
Qt-kehitysympäristön kehitys alkoi jo vuonna 1991 Haavard Nordin ja Eirik Chambe-Engin toimesta. Heidän visionaan oli kehittää maailman paras C++-pohjalta toimiva käyttöliittymärunko, joka toimisi eri alustoilla. He kehittivät Qt:n Signal & Slot -järjestelmän vuonna 1992. Trolltech, Nordin ja Chambe-Engin perustama yritys julkaisi ensimmäisen version Qt:stä toukokuussa 1995. Ensimmäistä versiota, versio 0.90:ää, voitiin käyttää sekä Windows-, että Unix- järjestelmällä, sillä Qt käytti samaa ohjelmointirajapintaa molemmilla alustoilla. Qt:tä oli saatavilla kahdella eri lisensillä: ilmainen versio, joka mahdollisti avoimen sovelluskehityksen, ja maksullinen versio, joka oli suunnattu yritysten sovelluskehitykseen. Qt 3.0 julkaistiin vuonna 2001 ja Qt 4.0 kesällä 2005. (Blanchette & Summerfield 2006, xv-xvi.)

Ensimmäinen Qt SDK (Source Development Kit), kehitystyökalupaketti, julkaistiin kesäkuussa 2010. Tämä työkalupaketti tunnettiin nimellä Nokia Qt SDK 1.0, sillä Trolltech oli siirtynyt aiemmin Nokian omistukseen. Työkalupaketti sisälsi monia eri sovelluksia ja työkaluja, joita tarvitaan Qt:n sovelluskehitykseen. Qt 4.7.0 SDK oli ensimmäinen versio, joka sisälsi Qt Quick -käyttöliittymätyökalun. Qt 4.7.0 SDK julkaistiin syyskuussa 2010. (Qt Nokia Qt SDK 1.0 released 2012.)

Seuraava suuri julkaisu, Qt 5.0, on suunniteltu vuoden 2012 ensimmäiselle puoliskolle. Qt 5.0 tulee hyödyntämään entistä enemmän Qt Quickiä ja tukee OpenGL/OpenGL ES -grafiikkakiihdytystä. (Qt 5.0 Roadmap 2012.)

2.2 Signal & Slot -järjestelmä

Qt:n keskeinen ominaispiirre on olioiden välinen Signal & Slot -järjestelmä. Alla olevasta kuviosta nähdään järjestelmän toiminta.



KUVIO 1. Signal & Slot -järjestelmä (Qt 4.7 : Signal & Slots 2012.)

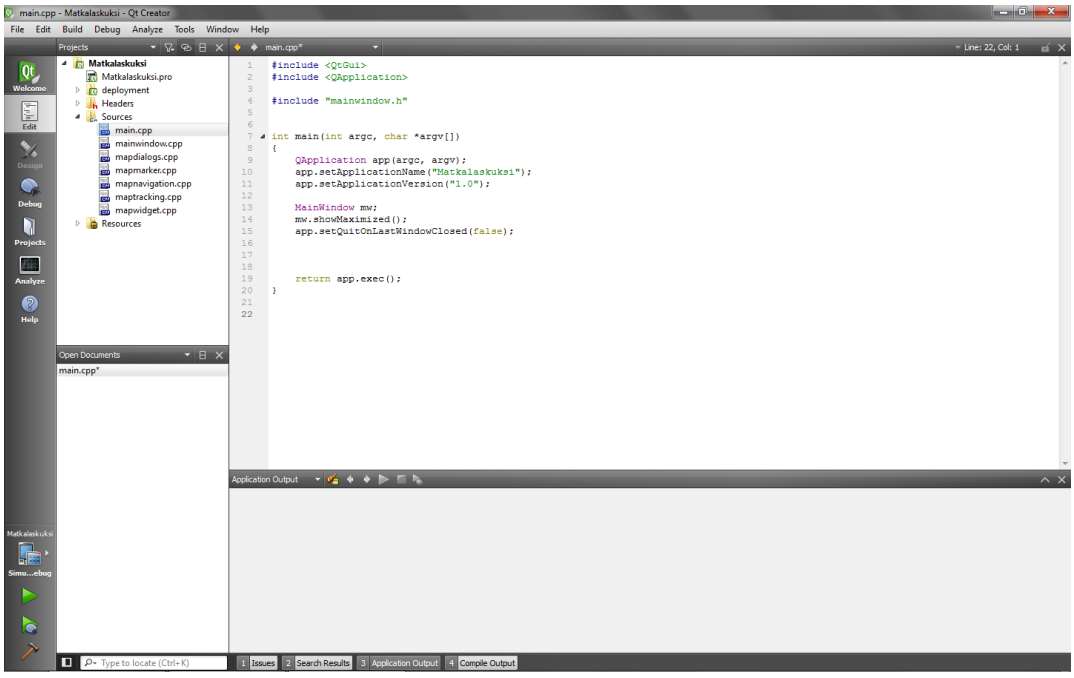
Järjestelmä toimii siten, että signaaleita lähettävä objekti yhdistetään toiseen objektin, jossa on vastaanottofunktio (Slot). Useampi signaali voidaan kytkeä samaan vastaanottofunktioon ja sama signaali voidaan kytkeä useaan vastaanottofunktioon. Signaaleja voidaan myös kytkeä yhteen; signaali lähettää toisen signaalin. Signaalit lähetetään käyttäen avainsanaa emit. Signaalien ja vastaanottofunktioiden täytyy käyttää samoja tietotyyppejä, jotta ne voidaan yhdistää. Esimerkiksi signaalia, joka lähettää numeromuuttujassa tietoja, ei voida yhdistää vastaanottofunktioon, joka käyttää tekstitietoa. Kaikki luokat, jotka perivät QObject-luokan, voivat käyttää Signal & Slot -mekaniikkaa. (Qt 4.7 : Signal & Slots 2012.)

2.3 Qt SDK

Qt SDK on kehitystyökalupaketti, joka sisältää eri työkaluja sovellusten tuottamiseen. Paketti sisältää Qt-kehitysympäristön alustariippumattomine luokkakirjastoineen, Qt Creator -ohjelmointiympäristön, Qt Quick -käyttöliittymätyökalun, Qt Simulator -sovellus testaukseen, Qt Mobility -ohjelmointirajapinnan mobiililaitteille sekä monia muita hyödyllisiä sovelluskehitykseen liittyviä työkaluja. Qt SDK sisältää myös tarvittavat tiedostot ja työkalut muun muassa Symbian-alustalle kehitettäviin sovelluksiin. Tämän työn alussa käytettiin Qt SDK 1.1.0 -versioita ja projektin edetessä työkalupaketti päivittyi versioon 1.1.4. (Qt SDK 2012.)

2.3.1 Qt Creator

Qt Creator, kuviossa 2, on Qt SDK:n ydintyökalu; se on alustariippumaton ohjelmointiympäristö, jolla voidaan tuottaa sovelluksia niin työpöytäkäyttöön kuin mobiililaitteille. Ohjelmointiympäristö sisältää tuen C++- sekä Java-koodille ja se sisältää myös versionhallinnan, kuten GIT, Subversion ja CVS. Qt Creator -työkalun mukana tulee valmiita sovelluspohjia, joita käyttäjä voi hyödyntää omissa projekteissa. (Qt Products 2012.)



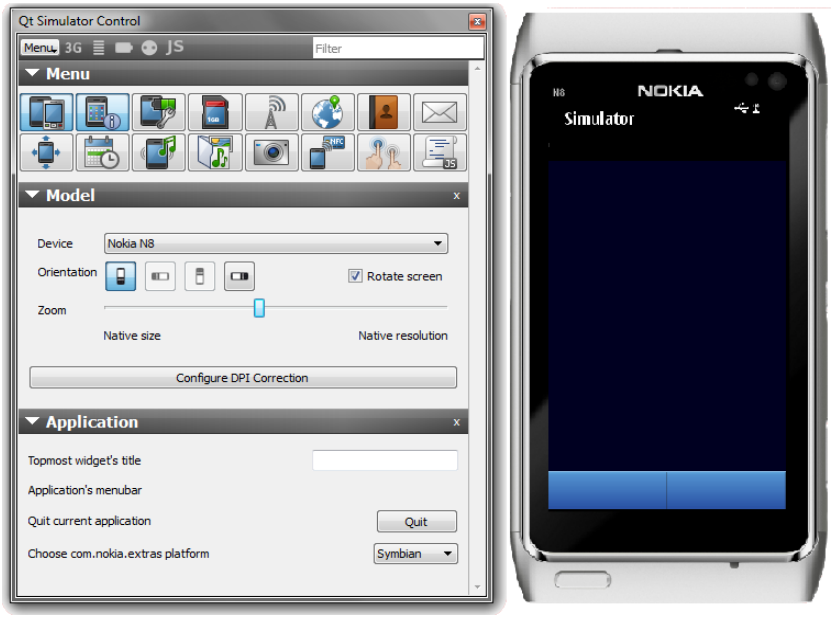
KUVIO 2. Qt Creator -ohjelmointiympäristö

2.3.2 Qt Quick -käyttöliittymäkomponentit

Qt SDK sisältää Qt Quick -käyttöliittymän rakennustyökalun ja siihen tarvittavat käyttöliittymäkomponentit. Qt Quickin tarkoituksena on helpottaa ja nopeuttaa käyttöliittymän rakennusta eri sovelluksille. Työkalu sisältää valmiita käyttöliittymäkomponentteja eri alustoille, jotka tuovat käyttöjärjestelmän mukaisen ulkonäön käyttöliittymälle. Qt Quick käyttää QML-kieltä koodissa ja tämä kieli muistuttaa hyvin paljon CSS- ja Javascript-kieltä. (Qt Products 2012.)

2.3.3 Qt Simulator

Qt Simulator, kuviossa 3, on Qt SDK:n mukana tuleva sovellus, jolla voidaan simuloida mobiililaitteen toimintaa. Simulaattorin tarkoitus on auttaa sovelluskehittäjää havaitsemaan mahdollisia virheitä koodissa ja estää oikean laitteen vahingoittumista. Simulaattoria voidaan käyttää silloin, kun mobiililaitetta ei ole saatavana tai kun halutaan testata sovellusta ilman laitteen hajoamisen riskiä. Simulaattori ei kuitenkaan korvaa oikeaa mobiililaitetta, sillä kaikkia toimintoja ei voida simuloida. Simulaattori ei myöskään vastaa oikean mobiililaitteen toimintaa kaikissa tapauksissa. Projektin edetessä yleiseksi säännöksi muotoutui se, että mikä toimii simulaattorissa, ei välttämättä toimi oikeassa laitteessa.



KUVIO 3. Qt Simulator

2.3.4 Qt Mobility ja Location-kirjasto

Qt Mobility tarjoaa ohjelmointirajapinnan mobiililaitteille. Se sisältää tärkeimmät ominaisuudet, mitä tarvitaan mobiililaitteiden sovelluskehityksessä, kuten esimerkiksi paikannus- ja multimediatoiminnot.

Tässä työssä käytettiin Location- ja System Information -kirjastoja. System Information sisältää tärkeimmät menetelmät laitteen tietojen käsittelyyn. Se sisältää myös moduulin SystemDeviceInfo, jonka avulla matkapuhelimen IMEI, jota tarvitaan matkanseurantajärjestelmässä, voidaan hakea. Location-rajapinta tarjoaa monipuolisen valikoiman toimintoja, jotka liittyvät paikannukseen. Alla olevasta taulukosta nähdään käytetyt Location-rajapinnan luokat.

TAULUKKO 1. Työssä käytetyt Location-rajapinnan luokat (Qt Mobility 1.1 : Location 2012.)

Nimi	Selitys
QGeoAddress	Osoite.
QGeoCoordinate	Koordinaatti.
QGeoPlace	Paikka, sisältää osoitteen ja koordinaatit.
QGeoPositionInfo	Paikannustieto, sisältää mm. koordinaatit ja aikaleiman.
QGeoPositionInfoSource	Paikannustietojen lähde.
QGeoServiceProvider	Palvelutarjoaja, välittää paikannustiedot.
QGeoMappingManager	Kartan hallinta.
QGraphicsGeoMap	Näyttää kartan ja toimii rajapintana käyttäjän ja kartan välillä.
QGeoMapObject	Käytetään karttapisteiden piirtämiseen kartalle.
QGeoRoute	Reitti kahden pisteen välillä.
QGeoMapRouteObject	Käytetään reitin piirtämiseen kartalle.
QGeoRouteRequest	Reitinhakupyyntö.
QGeoRouteReply	Reittihaun vastaus.
QGeoRoutingManager	Reittien hallinta.
QGeoSearchReply	Osoite- ja koordinaattihaun vastaus.
QGeoSearchManager	Osoite- ja koordinaattihaun hallinta.

2.4 Qt-dokumentaatio - Maps Demo

Qt Mobility -dokumentaatiosta löytyvä Maps Demo -opetussovellus on ainoa täysin dokumentoitu esimerkkisovellus. Tätä esimerkkisovellusta käytettiin pohjana opinnäytetyölle. Esimerkkisovelluksen dokumentaatiossa on käyty vaiheittain läpi se, miten karttasovellus rakennetaan. Esimerkissä käydään läpi kartan käyttö, se miten karttaa liikutellaan ja miten se piirretään käyttöliittymään. Karttapisteiden hallintaan sekä osoitteiden ja reittien hakuun löytyy ohjeet. Maps Demo -esimerkki on rakennettu alussa työpöytäsovelluksena, mutta ohjeiden lopussa perehdytään siihen, miten sovellusta tulee muokata mobiililaitteelle yhteensopivaksi. Loppuvaiheessa muun muassa katsotaan, miten verkkoyhteys hoidetaan ja miten karttaa liikutetaan matkapuhelimen näppäinten avulla. Täydellinen lähdekoodi löytyy Qt:n mukana tulevasta esimerkkisovelluskansiosta. (Qt Mobility 1.1 Maps Demo Tutorial 2012.)

Lähdekoodia soveltamalla saatiin haluttu toiminnallisuus karttapalveluun. Kaikkia toimintoja ei kuitenkaan esimerkistä löytynyt. Nämä toiminnot oli toteutettava siten, että ensin etsittiin tarvittavat komponentit ja sen jälkeen luettiin dokumentaatiosta, miten niitä käytetään. Tämän jälkeen sovellukseen lisättiin kyseiset komponentit ja koodia muokattiin halutun toiminnon saavuttamiseksi. Osa Maps Demo -esimerkistä löytyy seuraavalta sivulta, kuviosta 4.

[Previous: Maps Demo Tutorial] [Next: Part 2 - Searching for locations]

Part 1 - The Map Widget

To begin with, we will start defining the map widget, which is the central part of the application's user interface. Enough of the map widget will be defined here to work satisfactorily on most desktop platforms -- full consideration for mobile use will be made later along with other parts of the application.

Contents

- The very basics
- Pan & zoom
- Map icons

The very basics

The Location module provides the QGraphicsGeoMap which is a simple, easy way to insert maps into a QGraphicsScene. Since we're going to be extending the map later, we'll create a subclass of QGraphicsGeoMap called `GeoMap`, as below:

```
class GeoMap : public QGraphicsGeoMap
{
    Q_OBJECT

public:
    GeoMap(QGeoMappingManager *manager, MapsWidget *mapsWidget);
    ~GeoMap();

private:
    MapsWidget *mapsWidget;
};

GeoMap::GeoMap(QGeoMappingManager *manager, MapsWidget *mapsWidget) :
    QGraphicsGeoMap(manager), mapsWidget(mapsWidget)
{
}
```

And next we define a QWidget subclass, MapsWidget, which handles the creation of QGraphicsView and QGraphicsScene to put the GeoMap into. We make use of the Pimpl idiom on this class, since (as we will see) it will grow later to have a large complement of private data members, and some of these have naming conflicts with public methods.

KUVIO 4. Maps Demo -ohjeet, osa 1

3 MATKALASKUKSI-SOVELLUKSEN MÄÄRITTELY

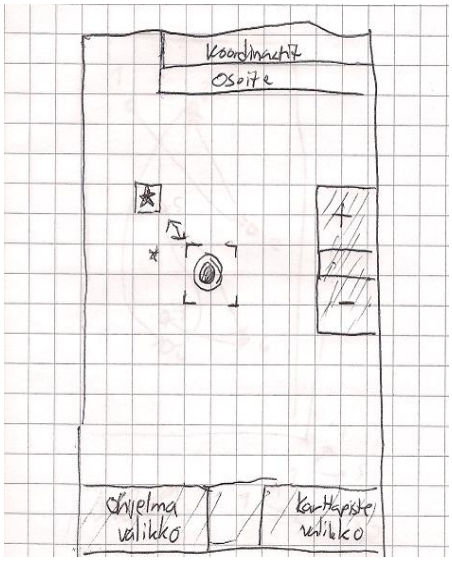
Sovelluskehityksen alussa käydään läpi, minkälaista sovellusta kehitetään ja mitä siltä vaaditaan. Mitä tarkemmat vaatimukset saadaan määriteltyä, sitä helpompaa on laatia suunnitelma sovelluksen toteutukseen. Kaikkia vaatimuksia ei välttämättä voida toteuttaa, sovellusalustan ja käyttöjärjestelmän asettamien rajoitusten vuoksi.

Tarkoituksena oli kehittää matkanseurantasovellus älypuhelimeen, joka sisältäisi matkanseurantajärjestelmän. Sovelluksen tärkeimpänä osana olisi matkanseurantajärjestelmä, jonka täytyisi toimia mahdollisimman virheettömästi. Tästä syystä vaatimuksissa määriteltiin hyvin tarkasti, miten järjestelmä tulisi toimia. Järjestelmän toiminta, kuten käynnistys, lopetus ja se, mitä tietoja sijaintipisteistä sovellus kerää ja mihin ne lähetetään, määriteltiin alkuvaiheessa. Toimintapoikkeustiloissa, kuten verkkoyhteyden katkeamisessa, seurantajärjestelmän on kerättävä tietoja vähintään 15 min, jos verkkoyhteyttä ei ole. Tämä tarkoitti sitä, että suunnitteluvaiheessa oli otettava huomioon tietojen käsittely verkkoyhteydettömässä toimintatilassa. Kartan toimintaan ei määritelty yhtä tarkkoja vaatimuksia, vaan navigointi ja karttapisteiden hallinta olivat lisätoimintoja. Sovelluksen asetuksiin vaadittiin mahdollisuus muuttaa palvelimen IP-osoitetta sekä kerättyjen sijaintipisteiden määrää, ennen kuin paketti lähetetään palvelimelle.

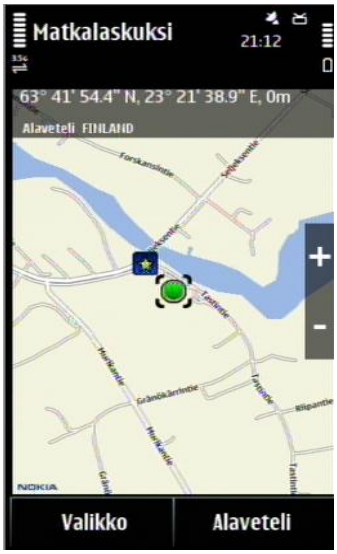
4 MATKALASKUKSI-SOVELLUKSEN SUUNNITTELU

Suunnittelu on tärkeä osa ohjelmistoprojektia, se toimii sovelluskehityksen ohjenuorana ja helpottaa ohjelmoijan työtä. Suunnittelun pohjana käytetään vaatimuksia ja määrityksiä, joita projektin alussa on käyty läpi; tarkoituksena on saada kokonaiskuva sovelluksesta. Tärkeää sovelluksen suunnittelussa on erilaiset kehitysalustan asettamat rajoitukset. Nämä rajoitukset on otettava huomioon varsinkin matkapuhelimeen tehtävissä sovelluksissa.

Työn alussa tutkittiin, minkälaista matkanseurantasovellusta on mahdollista toteuttaa Qt:llä Nokia N8 -älypuhelimeen. Suunnitteluvaiheessa Qt:n opetussovelluksista löytyi Maps Demo -sovellus ja tätä sovellusta käytettiin apuna Matkalaskuksi-sovelluksen suunnittelussa. Suunnittelussa tutkittiin myös Nokia Ovi Maps -palvelun käyttöliittymän ulkonäköä ja kartan toimintoja. Kuviosta 5 ja 6 nähdään suunniteltu ja toteutettu.



KUVIO 5. Suunniteltu käyttöliittymä



KUVIO 6. Toteutettu käyttöliittymä

Suunnittelussa huomioitiin yleisiä sääntöjä, jotka liittyvät matkapuhelinsovellusten kehittämiseen. Matkapuhelinsovelluksille on olemassa tiettyjä sääntöjä siitä, mitä sovelluksen täytyy noudattaa, esimerkiksi jos sovellus halutaan ladata Nokian sovelluskauppaan. Tärkeimpiä sääntöjä ovat, että sovellus ei estä hätänumeroon soittamista ja että sovellus ei käytä mitään toimintoja, jotka voivat aiheuttaa käyttäjälle kustannuksia ilman tämän tietoa.

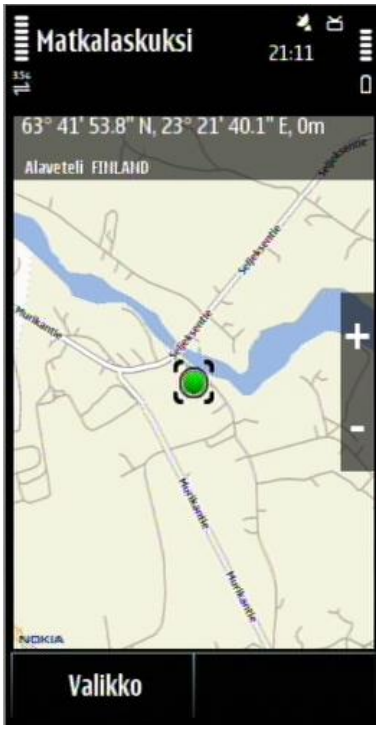
5 MATKALASKUKSI-SOVELLUKSEN TOTEUTUS

Toteutusvaiheessa sovelluksen koodaaminen aloitetaan suunnitelman pohjalta. Mitä parempi suunnitelma on, sitä helpompi sovellus on toteuttaa. Toteutuksen alussa on tärkeää saada kokonaiskuva sovelluksesta, jotta voidaan määritellä sen laajuus ja se, mistä osasta toteutus tulisi aloittaa. Usein joudutaan kuitenkin palaamaan takaisin suunnitteluun, sillä toteutusvaiheessa tehtäviä sovellusosia ei välttämättä voidakaan toteuttaa. Siksi vaihtelua tapahtuu toteutusvaiheesta takaisin suunnitteluvaiheeseen ja toisinpäin. Matkalaskuksi-toteutuksessa käydään läpi, miten sovelluksen eri osat on toteutettu ja miten ne toimivat.

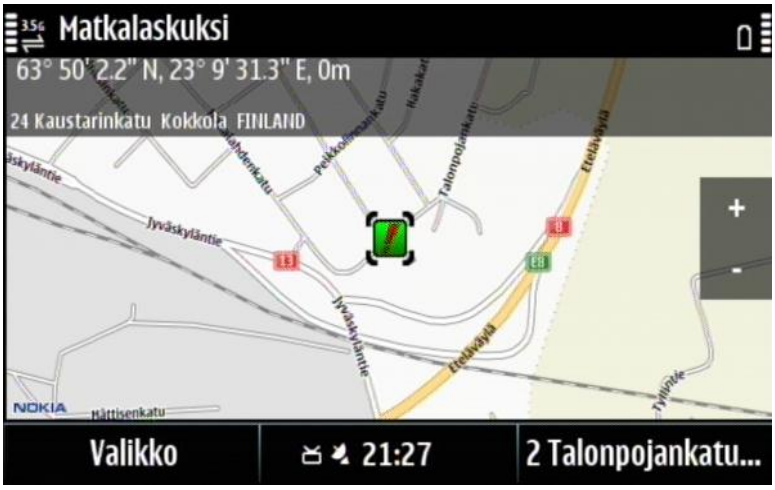
5.1 Käyttöliittymä

Yksi tärkeimmistä asioista, joka tulee huomioida sovelluskehityksessä, on sovelluksen käytettävyys. Käyttäjä kokee turhautumista, jos sovelluksen käyttö on liian monimutkaista. On siis tärkeää, että käyttöliittymä on hyvin selkeä ja helppokäyttöinen. Erityisesti matkapuhelinsovellusten ohjelmoinnissa käyttöliittymä on suunniteltava hyvin yksityiskohtaisesti, sillä näyttöjen koot vaihtelevat eri malleissa. Joissakin puhelinmalleissa on mahdollisuus kääntää näyttöä, mutta tämä edellyttää sitä, että käyttöliittymän on pystyttävä sopeutumaan eri kuvatiloihin. On myös tärkeää selvittää, tuleeko sovellus laitteeseen, jossa on näppäimet vai kosketusnäyttö tai molemmat.

Matkalaskuksi-sovelluksen käyttöliittymä on suunniteltu kosketusnäyttölaitteelle, mutta tuki löytyy myös muun tyyppisille matkapuhelimille. Käyttöliittymä on suunniteltu helppokäyttöiseksi ja selkeäksi. Kaikki valikot ovat dynaamisia, eli valikkojen sisältö riippuu sovelluksen tilasta. Käyttöliittymän kaikki valikot eivät ole näkyvissä, ennen kuin tietyt komponentit on ladattu valmiiksi, esimerkiksi matkanseurantaa ei voi käynnistää eikä karttapisteitä voi lisätä, ennen kuin oma sijainti kartalla on löydetty. Sovellus havaitsee, missä tilassa älypuhelin on ja sopeuttaa käyttöliittymän joko pysty- tai vaakatilaan. Kuviossa 7 on esitetty pystytila. Kuviossa 8 käyttöliittymä on vaakatilassa.

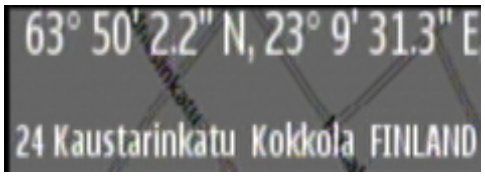


KUVIO 7. Pystytila



KUVIO 8. Vaakatila

Sovelluksen näyttö koostuu pääikkunasta ja kartasta sekä tämän käyttöliittymäkomponenteista. Kartan keskipisteestä löytyy kohdistin, jonka tiedot näkyvät kartan yläosassa. Kuviosta 9 nähdään esimerkki kohdistimen tiedoista. Pääikkuna sisältää kaksi valikkoa. Toinen näistä valikoista on päävalikko, ja toinen on karttapisteen hallintavalikko, joka on ainoastaan näkyvissä, kun karttapiste on valittu. Päävalikko sisältää sovelluksen tärkeimmät ominaisuudet; täältä voidaan käynnistää matkanseuranta, etsiä osoitteita ja muuttaa asetuksia. Karttapisteen hallintavalikosta löytyvät muun muassa navigointi ja karttapisteen tiedot. Sovelluksen oikeassa laidassa ovat painikkeet kartan lähentämiseen ja loitontamiseen, kuvio 10.



KUVIO 9. Kohdistimen tiedot

KUVIO 10. Kartan zoomauspainikkeet

5.1.1 Päävalikko

Päävalikosta, joka on esitetty kuviossa 11, löytyy sovelluksen tärkeimmät toiminnot, kuten matkanseurannan aloitus ja osoitehaku.

KUVIO 11. Päävalikko

Avoimet sovellukset: Tämä toiminto on käyttöjärjestelmän sisäänrakennettu toiminto, joka näyttää kaikki avoimet sovellukset. Koska tämä toiminto on käyttöjärjestelmän toiminto, sitä ei piiloteta ja se löytyy molemmista valikoista, sekä päävalikosta että karttapiste valikosta.

Oma sijainti: Toiminto siirtää kartan näkymän käyttäjän tämänhetkiseen sijaintiin ja alkaa seurata käyttäjän sijaintia. Tämä toiminto ei ole sama kuin matkanseuranta, sillä mitään tietoja sijainnista ei kerätä. Toiminnon tarkoitus on auttaa käyttäjää seuraamaan omaa sijaintiaan liikkuessa. Oman sijainnin seuranta kytkeytyy automaattisesti pois päältä, kun käyttäjä liikuttaa karttaa.

Osoitehaku: Tämän toiminnon avulla voidaan etsiä osoitteita eri puolelta maailmaa.

Lisää karttapiste: Toiminto lisää kartan keskelle, kohdistimen kohdalle, karttapisteen.

Aloita matkanseuranta: Tämä on sovelluksen ydintoiminto. Käyttäjän sijainnin seuraus ja paikkatietojen keräys aloitetaan matkanseurannasta.

Tyhjennä kartta: Tämä on helppokäyttöisyystoiminto, jolla käyttäjä voi tyhjentää kartan kaikista karttapisteistä tarvitsematta poistaa pisteitä yksitellen.

Asetukset: Asetukset on valikko, josta voidaan säätää sovelluksen eri asetuksia.

Piilota: Tämä toiminto piilottaa sovelluksen taustalle.

Käyttöohjeet (ei näy kuvassa): Toiminto näyttää sovelluksen käyttöohjeet.

Lopeta sovellus (ei näy kuvassa): Sovellus suljetaan tästä toiminnosta.

5.1.2 Karttapisteen hallintavalikko

Hallintavalikko on näkyvissä ainoastaan silloin, kun karttapiste on valittu. Valitun karttapisteen nimi näkyy valikossa. Kuviosta 12 nähdään avoinna oleva karttapisteen hallintavalikko.

KUVIO 12. Karttapisteen hallintavalikko

Näytä tiedot: Toiminto näyttää valitun karttapisteen tarkemmat tiedot.

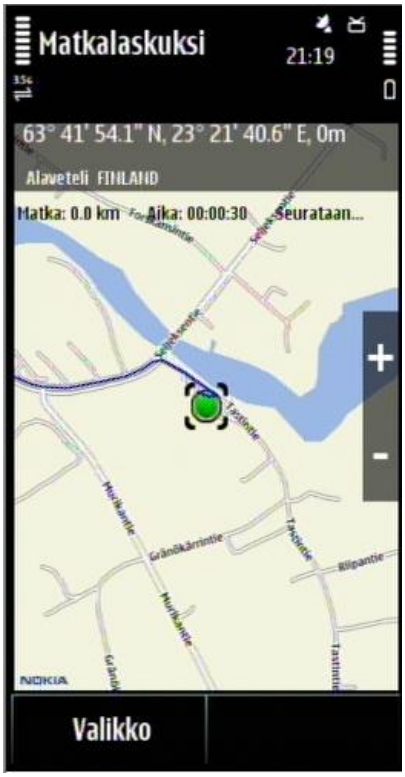
Näytä pisteen sijainti: Toiminto siirtää kohdistimen valitun pisteen kohdalle.

Navigointi: Käyttäjä voi aloittaa tästä toiminnosta navigoinnin.

Poista karttapiste: Tämä toiminto poistaa valitun karttapisteen kartalta.

5.1.3 Käyttöliittymän muut osat

Sovelluksesta löytyy käyttöliittymäkomponentteja, jotka ovat näkyvissä käyttäjälle ainoastaan sovelluksen ollessa tietyssä tilassa. Näihin komponentteihin kuuluu matkanseurantatiedot, joka on esitetty kuviossa 13, ja palvelimelle lähetyksen tila sekä tilapalkki- ja virheilmoitukset.



KUVIO 13. Matkanseurantajärjestelmän tiedot

Matkanseurantatiedot: Nämä tiedot ilmestyvät näkyviin, kun matkanseuranta on päällä tai pysäytettynä.

Seurataan…: Teksti vilkkuu, kun seuranta on aktiivinen. Tiedot ilmestyvät kohdistintietojen alapuolelle.

Lähetetään palvelimelle…: Teksti ilmestyy, kun sovellus aloittaa tietojen lähetyksen palvelimelle. Teksti vilkkuu ja on esillä niin kauan, kunnes sovellus on saanut vastauksen palvelimelta.

Virheilmoitus: Virheilmoitus ilmestyy, kun sovelluksessa tapahtuu virhe. Esimerkki virheilmoituksesta nähdään kuviosta 14, jossa osoitehaussa on ilmennyt virhe. Käyttäjä voi sulkea ikkunan painamalla ok-painiketta.

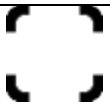
Tilapalkki: Sovelluksen suorittaessa toimintoja tai tilan muuttuessa kartan alaosasta nousee näkyviin tilapalkki. Tilapalkki on ajastettu, ja se laskeutuu alas piiloon tietyn ajan kuluttua. Tilapalkki on esillä kuviossa 15.

KUVIO 14. Virheilmoitus

KUVIO 15. Tilapalkki

Taulukosta 2 nähdään kartan eri kuvakkeet. Valittu piste käyttää samaa kuvaketta mutta erivärisenä.

TAULUKKO 2. Kartan merkinnät

Oma sijainti	
Oma karttapiste ja valittu karttapiste erivärisenä	
Osoitteenhakupiste ja valittu hakupiste erivärisenä	
Päämäärä	
Kohdistin	

5.2 Yleinen toiminta

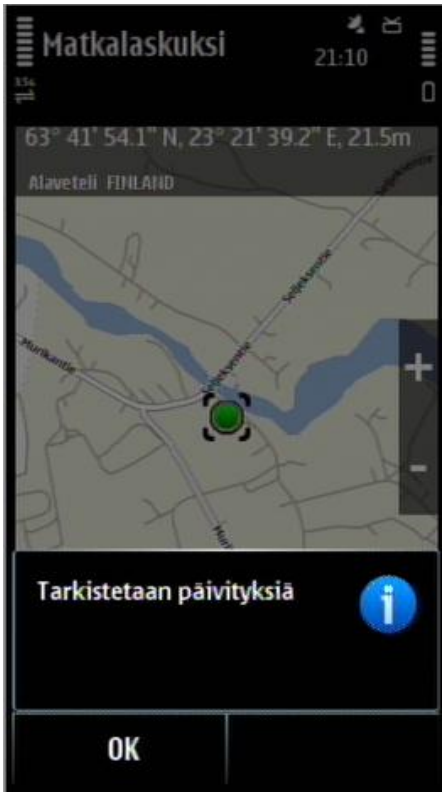
On erittäin tärkeää, että sovelluksen toiminta on sulavaa ja ettei sovellus kaadu tai sammu itsestään, jolloin tärkeitä tietoja saatetaan menettää.

5.2.1 Käynnistys

Sovelluksen käynnistyessä tarkistetaan verkkoyhteyden saatavuus, ja jos verkkoyhteyttä ei ole saatavilla, sovellus ei käynnistä karttapalvelua. Saatuaan verkkoyhteyden sovellus käynnistää karttapalveluun ja lataa sovelluksen asetukset tiedostosta. Tämän jälkeen aloitetaan oman sijainnin jäljitys. Samalla tarkistetaan, onko uusia versioita saatavilla. Oman sijainnin löytyessä kartta kohdistetaan tähän sijaintiin.

5.2.2 Päivitys

Automaattinen päivitysten tarkistus helpottaa ylläpitoa ja auttaa käyttäjää, sillä hänen ei itse tarvitse tarkistaa, onko uutta versioita saatavilla ja etsiä, mistä tämä päivitys ladataan. Käynnistyessä sovellus tarkistaa onko uusia päivityksiä saatavilla ja tästä ilmoitetaan käyttäjälle, joka nähdään kuviosta 16. Uuden version löytyessä sovellus ilmoittaa käyttäjälle uuden version numeron, mutta käyttäjän ei ole kuitenkaan pakko ladata uutta päivitystä. Käyttäjän halutessa päivityksen sovellus aukaisee verkkoselaimen ja sulkeutuu. Verkkoselain avautuu osoitteeseen, josta päivitys ladataan. Käyttäjä joutuu kuitenkin itse asentamaan päivityksen.



KUVIO 16. Automaattinen päivitysten tarkistus

5.2.3 Asetukset

Asetukset-valikosta voidaan säätää asetuksia, jotka vaikuttavat sovelluksen toimintaan. Asetukset-valikko on nähtävissä kuviosta 17. Sovelluksen yleiset tiedot näkyvät valikon ala-osasta. Matkanseurantaan kuuluvat asetukset ovat päivitysväli ja palvelimen IP. Päivitysväli vaikuttaa siihen, kuinka usein sijaintipisteitä pyydetään paikannustietolähteeltä. Päivitysväli on rakennettu siten, että se ei vaikuta oman sijainnin päivitykseen kartalla. Palvelimen IP on osoite, minne sijaintitiedot lähetetään, kun sijaintitietoja on kerätty tietty määrä.

Käyttäjä voi valita Asetukset-valikosta myös kartan tyypin. Valittavissa on katu-, satelliitti- ja maastonäkymä. Näkymän muuttaminen saattaa vaihtaa käyttöliittymänkomponenttien tekstin väriä näkyvyyden parantamiseksi. Käyttäjä voi halutessaan palauttaa asetukset alkuperäisiin asetuksiin painamalla Palauta asetukset -painiketta. Tiedot tallentuvat käyttäjän painaessa Tallenna-painiketta. Asetukset-valikosta voidaan nähdä sovelluksen nimi sekä tämänhetkinen versio.

KUVIO 17. Sovelluksen asetukset

5.2.4 Piilotus

Piilotus-toiminto ei sulje sovellusta, vaan se laitetaan piiloon. Tämä tarkoittaa sitä, että sovellus ajetaan taustalla. Sovellus on siis käynnissä, vaikka se ei ole näkyvissä. Sovellus tuodaan takaisin näkyviin käyttämällä **Avoimet sovellukset** -toimintoa tai painamalla sovelluksen kuvaketta.

5.2.5 Lopetus

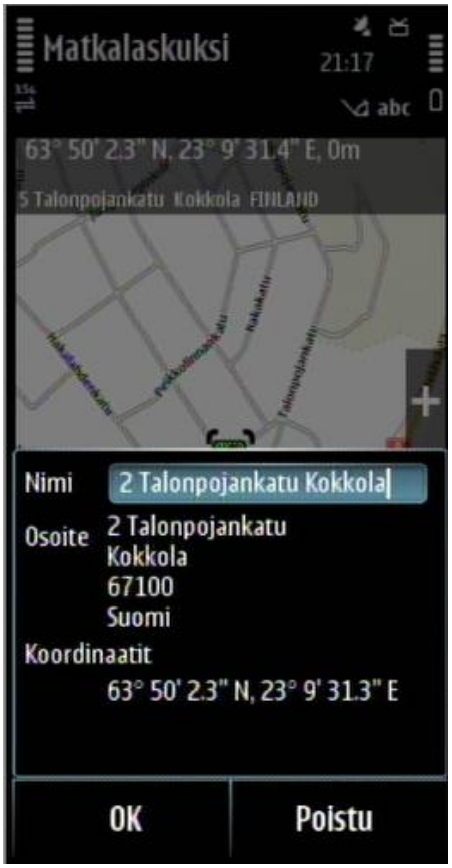
Käyttäjän valitessa **Lopeta sovellus** -toiminnon sovellus tarkistaa, onko verkkoyhteyttä saatavilla, ja jos verkkoyhteyttä ei ole, sovellus sulkeutuu. Seuraavaksi sovellus tarkistaa, onko matkanseurantajärjestelmä päällä, ja jos järjestelmä on päällä, suoritetaan palvelimelle lähettäminen, vaikka pakettimäärä ei olisi täynnä. Tämän jälkeen sovellus sulkeutuu.

5.3 Kartan toiminta

Käyttäjä hallitsee karttaa kosketuksella, ja kartan liikuttaminen onnistuu siten, että käyttäjä siirtää sormeaan näytöllä. Käyttäjä voi myös siirtää näkymää näpäyttämällä karttaa kahdesti, jolloin kohdistin siirtyy näpäytettyyn kohtaan. Karttaa voidaan loitontaa ja lähentää kosketuspainikkeiden avulla. Kartta lähenee tai loittonee yksi taso kerrallaan, ja uusien karttakuvien haun kesto riippuu käytettävän verkkoyhteyden nopeudesta.

5.3.1 Karttapisteiden hallinta

Karttapisteitä voidaan lisätä kahdella eri tavalla kartalle. Käyttäjä voi itse lisätä karttapisteen käyttämällä **Lisää karttapiste** -toimintoa tai käyttää osoitehakua; onnistunut haku ja kartalle lisäys valitsee karttapisteen ja tuo näkyviin karttapisteen hallintavalikon.
Käyttäjän lisätessä oman karttapisteen, **Lisää karttapiste** -toiminnon avulla, sovellus hakee automaattisesti koordinaatteja vastaavan osoitteen ja asettaa tämän karttapisteen tietoihin. Nämä tiedot voidaan lukea **Näytä tiedot** -toiminolla hallintavalikossa. Karttapisteen **Näytä tiedot** -toiminto on esitetty kuviossa 18.



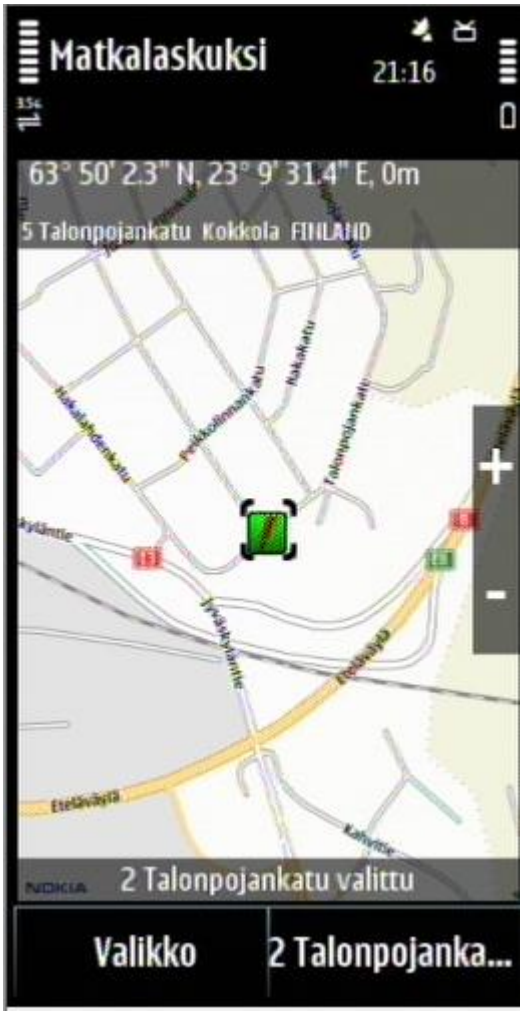
KUVIO 18. Karttapisteen tiedot

Karttapisteen tiedoissa näkyy nimi, osoite sekä pisteen koordinaatit, ja näistä tiedoista käyttäjä voi muokata ainoastaan nimeä. Karttapisteen voi poistaa hallintavalikosta tai käyttämällä **Tyhjennä kartta** -toimintoa. Käyttämällä **Tyhjennä kartta** -toimintoa kaikki pisteet poistetaan kartalta paitsi oma sijainti. Navigoinnin ollessa käytössä kartalle jätetään myös päämäärä. Karttapisteiden valinta onnistuu laitamalla kohdistimen halutun karttapisteen kohdalle ja painamalla kohdistinta. Valitun karttapisteen ollessa poissa näkyvyyskentästä voidaan valittuun karttapisteeseen siirtyä hallintavalikosta toiminolla **Näytä pisteen sijainti**, jolloin kohdistin siirtyy valitun kartta pisteenkohdalle.

5.3.2 Osoitehaku

Osoitehaun toiminta on kaksivaiheinen: ensin haetaan kaikki osoitteet, joita hakukriteerit vastaavat, ja sen jälkeen käyttäjä valitsee halutun osoitteen. Osoitehaku nähdään kuviosta 19. Valitun osoitteen kohdalle lisätään karttapiste, tämä ominaisuus nähdään kuviosta 20. Hakutoiminnossa osoitemuodon täytyy olla tietyssä muodossa, jotta haku onnistuu. Käyttäjä voi hakea katuosoitteen perusteella osoitetta, mutta tällöin on määriteltävä myös kaupunki. Jos käyttäjä jättää katuosoitteen sekä kaupungin pois, sovellus hakee valitun maan pääkaupungin. Käyttäjä voi valita hakualueen, josta valittavana on alle 10 km, alle 25 km, alle 100 km sekä maailmanlaajuinen. Onnistuneen karttapisteen lisäyksen jälkeen karttanäkymä siirtyy valittuun osoitteeseen, karttapiste valitaan ja hallintavalikko tulee näkyviin

KUVIO 19. Osoitteiden haku

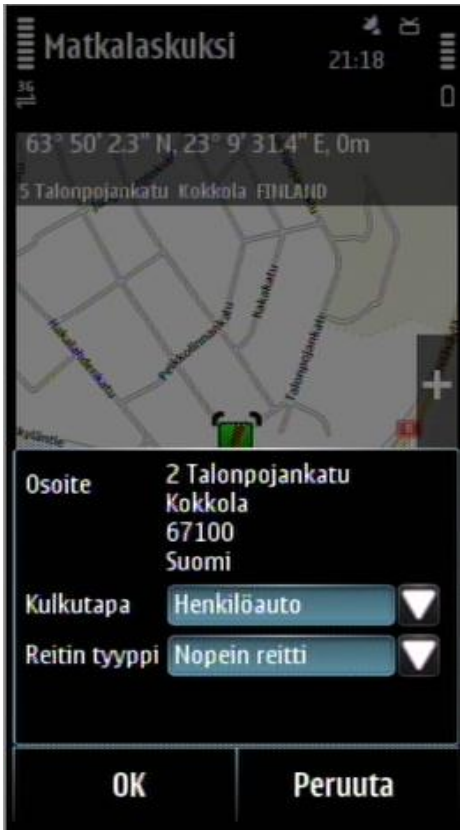


KUVIO 20. Osoitehausta lisätty karttapiste

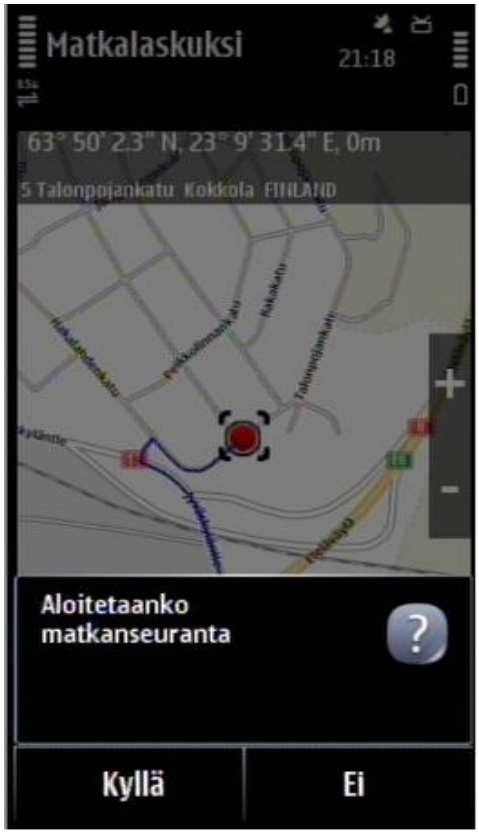
5.3.3 Navigointi

Karttapisteen hallintavalikosta valittu **Aloita navigointi** avaa ikkunan, kuvio 21, jossa käyttäjä voi määritellä kulkutavan sekä reitin tyypin, ja myös karttapisteen osoite näkyy ikkunassa. Eri kulkutapoja on kaksi, henkilöauto sekä kuorma-auto. Valittavia reittityyppejä on kolme, lyhin reitti, nopein reitti ja taloudellisin reitti. Ok -painiketta painettaessa sovellus alkaa etsiä reittiä, joka vastaa käyttäjän valintoja.

Reitin laskennan aikana valitun karttapisteen kuvake muuttuu, ja tämän nimeksi tulee päämäärä. Kun reitin laskenta on valmis, se piirretään kartalle. Piirtäminen saattaa kestää, jos kyseessä on pitempi reitti. Piirretty reitti näkyy sinisenä kartalla. Sovellus, kysyy halutaanko matkanseuranta aloittaa, jos se ei ole käynnissä. Tämä nähdään kuviosta 22. Navigoinnin lopetus onnistuu **Lopeta navigointi** -toiminolla, joka sijaitsee päävalikossa. Lopetuksen yhteydessä karttapisteen tyyppi muutetaan takaisin alkuperäiseen tyyppiin ja pisteen valinta poistetaan. Matkanseurannan ollessa päällä sovellus kysyy, halutaanko seuranta lopettaa.



KUVIO 21. Navigointi



KUVIO 22. Reitin lisäys ja matkanseuranta

5.4 Matkanseurantajärjestelmän toiminta

Matkanseuranta on sovelluksen tärkein osa, se lähettää kerätyt sijaintitiedot palvelimelle. Käyttäjä käynnistää matkanseurannan päävalikosta. Tämän jälkeen käynnistetään reitti-ID:n eli reittitunnuksen haku. Käyttäjä voi pysäyttää matkanseurantapalvelun, mikäli se on aktiivinen. Tällöin käyttöliittymässä olevan matkanseurannan tiedot näyttävä komponentti ei vilkuta **Seurantaan…**-tekstiä ja tietoja ei piiloteta. Käyttäjä voi lopettaa matkanseurannan tai jatkaa matkanseurantaa tilan ollessa pysäytetty.

5.4.1 Reittitunnuksen haku

Reittitunnus haetaan siten, että matkapuhelimen IMEI, joka on 15 -merkkinen yksilöllinen laitetunnus, lähetetään palvelimelle. IMEI toimii yksilöllisenä tunnuksena Matkalaskuksi-palvelussa. Palvelin palauttaa numeron, jota käytetään reittien tunnistuksessa. Virheen sattuessa palvelin palauttaa numeron 0. Reittitunnus yritetään hakea uudestaan vain kerran, jos haku epäonnistuu. Matkanseuranta ei käynnisty ilman reittitunnusta. Kun reittitunnus on saatu, se tallennetaan käyttöä varten. Uusi reittitunnus haetaan ainoastaan matkanseurannan käynnistysvaiheessa, ei pysäytetyn seurannan jatkamisvaiheessa.

5.4.2 Sijaintipisteiden kerääminen

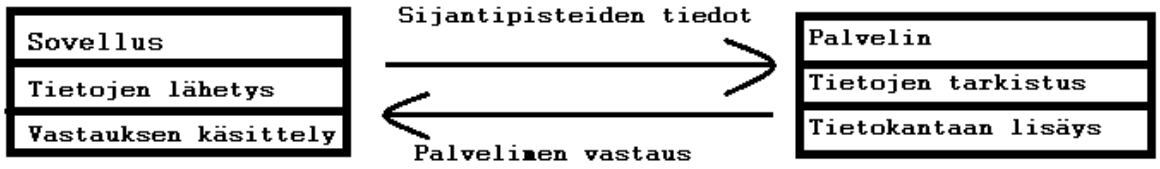
Matkapuhelin kerää sijaintipisteitä, kunnes tietty määrä on saavutettu. Ennen kuin sijaintipiste hyväksytään, tarkistetaan, että etäisyys edelliseen pisteeseen on suurempi tai yhtä kuin 3,0 metriä. Tämän avulla turhia pisteitä ei kerätä ja GPS-paikannusjärjestelmän antamat sijaintivirheet eivät vaikuta lopputulokseen.

Sijaintipisteiden tiedot kerätään NMEA-lauseisiin. NMEA-lause sisältää sijaintipisteen IMEI:n, latitudin, longitudin, aikaleiman, reitti-ID:n ja etäisyyden edelliseen pisteeseen. Aikaleima sisältää päivämäärän ja tarkan ajan, kun sijaintipiste on kerätty. Näitä NMEA-lauseista kerätään tietty määrä, kunnes ne lähetään palvelimelle.

Sovelluksen Matkanseuranta-palveluun on toteutettu dynaaminen NMEA-lauseiden määrä. Tämä tarkoittaa sitä, että tarvittavien NMEA-lauseiden määrä vaihtelee, ennen kuin lähetys tapahtuu. Lähetyksen ollessa käynnissä tarvittavien NMEA-lauseiden määrää kasvatetaan, ja jos lähetys onnistuu heti, NMEA-lauseiden määrää vähennetään. Tämän toiminnon lähdekoodi löytyy liitteestä 1.

5.4.3 Palvelimelle lähettäminen

Palvelimelle lähettäminen tapahtuu aina, kun tietty määrä NMEA-lauseita on kerätty tai käyttäjä sulkee sovelluksen **Lopeta sovellus** -toiminnon kautta. Ensin sovellus tarkistaa, sisältääkö lähetettävä tieto NMEA-lauseita. Tämän jälkeen lisätään parametrit ja poistetaan alusta ensimmäinen pilkku ja muunnetaan tieto oikeaan muotoon, jotta se voidaan lähettää. Lähetyksen jälkeen palvelin palauttaa reittitunnuksen, jota lähetettävissä NMEA-lauseissa käytetään. Virhetilanteissa palvelin palauttaa 0:n. Sovellus yrittää lähettää uudestaan, jos ensimmäinen lähetys epäonnistuu. Toiminnon lähdekoodi löytyy liitteestä 2. Palvelimelle lähettäminen on kuvattu kuviossa 23.



KUVIO 23. Palvelimelle lähettäminen

5.4.4 Aikakatkaisu

Kaikkiin sovelluksen toimintoihin, jotka lähettävät tietoja verkkoon, on lisätty aikakatkaisu-ominaisuus, sillä Qt:n verkkohallintamoduuli ei sisällä lainkaan omaa aikakatkaisua. Aikakatkaisun aika on 30 sekuntia. Tämän jälkeen verkkotoimintoa yritetään uudelleen tai käyttäjälle ilmoitetaan aikakatkaisuvirheestä.

5.4.5 Offline-tila

Sovellus saattaa menettää verkkoyhteyden, ja tällöin käyttäjälle ilmoitetaan asiasta ja matkanseuranta asetetaan offline-tilaan. Offline-tilassa matkanseuranta kerää suuremman määrän NMEA-lauseita. Määrän täyttyessä sovellus pysäyttää matkanseurannan. Tämän jälkeen käyttäjän on tarkistettava verkkoyhteys ja käynnistettävä matkanseuranta pysäytetystä tilasta.

6 MATKLASKUKSI-SOVELLUKSEN TESTAUS

Testaus on sovelluskehityksessä eniten aikaa vievä vaihe. Mitä enemmän testaukseen sijoitetaan aikaa, sitä paremmin sovelluksen toimintaa pystytään kartoittamaan ja virheet pystytään havaitsemaan.

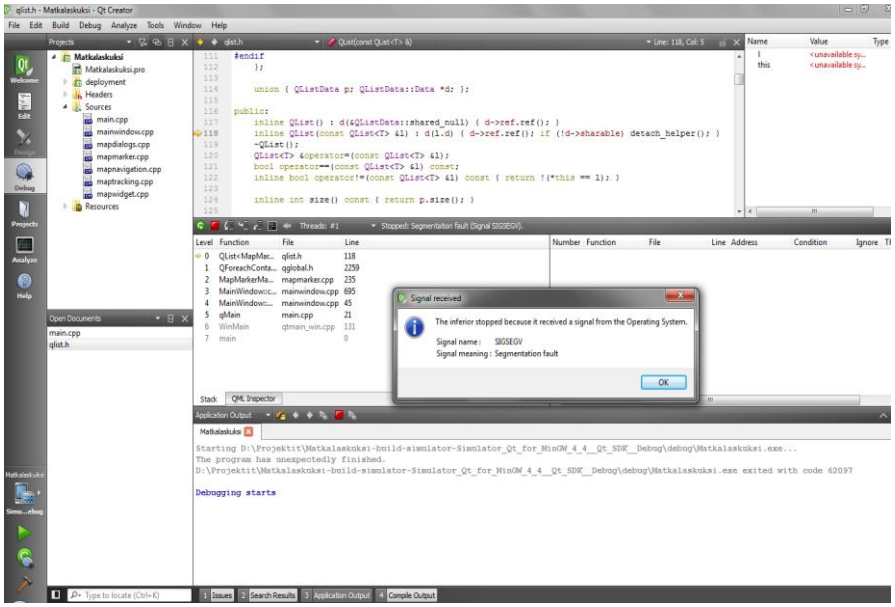
6.1 Sovelluksen testaus

Qt-kehitystyökalupaketissa oleva simulaattori toimi alussa hyvin testausalustana, mutta verkkoyhteyksien testaaminen suoritettiin oikealla laitteella. Simulaattori ei kuitenkaan ole täysin hyvä testaustyökalu, sillä joitakin ongelmia testauksessa ilmeni. Projektin aikana voitiin todeta, että se mikä toimii simulaattorissa, ei välttämättä toimi matkapuhelimessa ja se mikä toimii puhelimessa, ei välttämättä toimi simulaattorissa. Tästä hyvä esimerkki on sovelluksen käyttöliittymä. Käyttöliittymä näkyi erilaisena simulaattorissa kuin matkapuhelimessa; valikot saattoivat olla näkyvissä simulaattorissa, kun taas matkapuhelimessa ne olivat piilotettuina. Kuviosta 24 nähdään sovelluksen käyttöliittymä simulaattorissa.

KUVIO 24. Valikko simulaattorissa

6.2 Virheiden etsintä

Virheiden etsintään kuului paljon aikaa, sillä jotkin virheet olivat vaikeasti havaittavissa. Aikaa olisi kulunut paljon enemmän, ellei Qt Creatorin virheiden etsintätilaa olisi käytetty. Tässä tilassa ohjelman kulkua voidaan seurata vaiheittain ja paikantaa, missä osassa koodia virhe ilmestyy. Kuviosta 25 nähdään Qt Creator -työkalun virheiden etsintätila.



KUVIO 25. Virheiden etsintätila Qt Creator -työkalussa (Debug mode)

Mitä vähemmän virheitä lopputuotteesta löytyy, sitä parempi tuote käyttäjälle. Virheiden käsittely on myös sovelluksen sisällä tärkeä, sillä jos virheitä ei käsitellä, saattaa sovellus kaatua ja käyttäjä menettää tärkeitä tietoja. Kaikkia virheitä on mahdotonta saada pois koodista, joten siksi sovellukseen on lisätty päivitysmahdollisuus. Ylläpito voi korjata sovelluksen lähdekoodia ja julkaista käyttäjille päivityksiä, jotka korjaavat virheitä.

Matkalaskuksi-sovelluksessa on pyritty mahdollisimman sulavaan toimintaan. Toiminnot on rakennettu siten, että virheiden mahdollisuus on hyvin pieni. Käyttäjän aiheuttamat mahdolliset virheet on huomioitu ja sovelluskoodin on lisätty eri varotoimenpiteitä.
Virheiden hallinta-sovelluksessa on toteutettu virhe ilmoituksilla ja virheen tyypillä. Virhe voi olla kriittinen, jolloin käyttäjälle ilmoitetaan vakavasta virheestä. Tämän jälkeen sovellus sulkeutuu. Muut virheet eivät aiheuta sovelluksen sulkeutumista.

7 MATKALASKUKSI-SOVELLUKSEN JATKOKEHITYS

Jatkokehityksessä on tarkoitus parantaa jo olemassa olevia toimintoja tai lisätä uusia. Jatkokehityksessä voidaan myös korjata virheitä, joita aikaisemmin ei ole voitu korjata.
Matkalaskuksi-sovelluksen jatkokehityksessä kartan käyttöliittymään voitaisiin lisätä pinch-zoom -toiminto. Tällä toiminolla käyttäjä voi lähentää ja loitontaa näkymää kahden sormen avulla. Jatkokehityksessä voitaisiin myös tehdä koko käyttöliittymä uusiksi Qt Quickin avulla, tällöin C++-luokat hoitaisivat sovelluksen logiikan, kun taas Qt Quick hoitaisi käyttöliittymän. Qt Quickllä saataisiin myös käyttöliittymään käyttöjärjestelmän oma ulkoasu.

Jatkokehityksessä voitaisiin myös korjata navigointiosasta löytyviä toimintoja, joita ei voitu ottaa käyttöön valmiissa sovelluksessa. Yksi näistä toiminnoista on navigoinnin lopetus, kun oma sijainti on tietyn etäisyyden päässä päämäärästä. Toimintoja ei voitu ottaa käyttöön, sillä koodissa käytetty Location-moduulin luokka sisälsi virheitä. Tämä toiminto voidaan ottaa käyttöön vasta sitten, kun Qt-luokkakirjastoon tulee korjauksia.

8 JOHTOPÄÄTÖKSET

Opinnäytetyönä tehty sovellus, Matkalaskuksi, osoittautui hyvin haasteelliseksi projektiksi, sillä sovellus koostui monista eri osista. Syvällisestä sovelluskehityksestä Qt:llä ei ollut juurikaan kokemusta mutta Qt:n opetussovelluksen avulla projekti saatiin hyvin käyntiin. Pelkästään kartan ja karttapisteiden piirtäminen näytölle oli monimutkaista; piirtämiseen tarvittiin erilaisia graafisia komponentteja ja omia luokkia jouduttiin luomaan, jotta haluttu toiminnallisuus saatiin esiin.

Tavoitteena oli saada toimiva matkanseurantasovellus. Navigoinnin lisäämistä sovellukseen, mikäli mahdollista, oli pohdittu sovelluskehityksen alussa. Lopullisesta sovelluksesta nähdään, että kaikki tavoitteet saavutettiin kirkkaasti. Lisäksi sovellukseen tuli monia lisätoimintoja, jotka helpottavat käytettävyyttä ja tuovat lisäsisältöä sovellukseen. Näistä toiminnoista voidaan mainita omien karttapisteiden lisääminen ja sovelluksen automaattinen päivitys. Testaukseen ja käyttöliittymän helppokäyttöisyyteen käytettiin hyvin paljon aikaa. Helppokäyttöisyys ei välttämättä näy käyttäjälle, mutta esimerkiksi se, että käyttäjän painaessa Return-painiketta tekstinsyötön yhteydessä, ei sulje avattua ikkunaa. Myös virheiden käsittelyyn ja niiden havaitsemiseen kulutettiin runsaasti aikaa, muun muassa tietojen lähetykseen rakennettiin aikakatkaisuvirhe, eli mikäli palvelin ei ole vastannut tietyn ajan päästä, käyttäjälle ilmoitetaan virheestä.

Ainoat ongelmat, joita sovelluskehityksen aikana ilmentyi, olivat esimerkkien puute. Tilanteissa, joissa esimerkkejä ei ollut, ainoa ratkaisu oli kokeilla muuttamalla koodia vähitellen, jotta haluttu toiminnallisuus saatiin sovellukseen. Myös Qt-simulaattorin kanssa oli ongelmia, sillä käyttöliittymä saattoi näkyä aivan erilaisena simulaattorissa kuin oikeassa laitteessa.

Lopuksi voidaan todeta, että ilman Qt:n dokumentaatiossa olevaa opetussovellusta ja Qt:n laajaa luokkadokumentaatiota tätä opinnäytetyötä ei olisi voitu toteuttaa niin hyvin kuin se lopulta toteutui.

LÄHTEET

Blanchette, J. & Summerfield, M. 2006. C++ GUI Programming with Qt 4. New Jersey, Yhdysvallat: Prentice Hall

Qt Mobility 1.1 : Location. Www-dokumentti. Saatavissa: http://doc.qt.nokia.com/qtmobility-1.1/location-overview.html. Luettu: 9.1.2012.

Qt Mobility 1.1 Maps Demo Tutorial. Www-dokumentti. Saatavissa: http://doc.qt.nokia.com/qtmobility-1.1/tutorials-mapsdemo.html. Luettu: 9.1.2012.

Qt Nokia Qt SDK 1.0 released. Www-dokumentti. Saatavissa: http://labs.qt.nokia.com/2010/06/23/nokia-qt-sdk-10-released/. Luettu: 9.1.2012.

Qt Products. Www-dokumentti. Saatavissa: http://qt.nokia.com/products. Luettu: 9.1.2012.

Qt 5.0 Roadmap. Www-dokumentti. Saatavissa: http://developer.qt.nokia.com/wiki/Qt_5.0. Luettu: 9.1.2012.

Qt SDK. Www-dokumentti. Saatavissa: http://qt.nokia.com/products/qt-sdk. Luettu: 9.1.2012.

Qt 4.7 : Signal & Slots. Www-dokumentti. Saatavissa: http://doc.qt.nokia.com/4.7/signalsandslots.html. Luettu: 9.1.2012.

Tietojen kerääminen

```
void MapTracking::packetData()
{
if(distance < 3.0 ) {
   return;
}
if(!gotRouteID) {
   return;
}

if(currentUpdateNumber != updateInterval) {
    currentUpdateNumber++;
    return;
}

currentUpdateNumber = 1;
gpsData =
QString(",%1;%2;%3;%4;%5;%6").arg(imei).arg(currentPos.latitude())
.arg(currentPos.longitude()).arg(time).arg(routeID).arg(distance);

    dataString.append(gpsData);
    itemCount++;
    if(itemCount == packetLimit) {
        if(sending) {
            packetLimit = packetLimit + 10;
            return;
        }
        else {
            if((packetLimit - 10) >= 30) {
                packetLimit = packetLimit - 10;
            }
        }
        if(offlineMode) {
            setTrackingStatus(Pause);
        }
        else {
            emit startDataTransfer(dataString);
            dataString = "";
            itemCount = 0;
        }
    }
}
```

Tietojen lähettäminen

```
void MapTracking::dataTransfer(QString data)
{
    if(data.length() <= 0) {
        emit transferStatus(false);
        emit notActive();
        return;
    }
    emit transferStatus(true);
    retryData = data;
    QString param = "gps=";
    data = data.remove(0,1);
    QByteArray sendData;
    sendData.append(param.toAscii());
    sendData.append(data);

    dataReply = networkManager->post(QNetworkRequest(QUrl(dataAddress)),
sendData);

    sending = true;
    timeoutTimerServer->start(30000);
    if(dataReply->isFinished()) {
        if(dataReply->error() == QNetworkReply::NoError) {
            serverReplyFinished(dataReply);
        }
        else {
            serverReplyError(dataReply,dataReply->error(),
dataReply->errorString());
        }
        return;
    }
    connect(networkManager,SIGNAL(finished(QNetworkReply*)),
this,SLOT(serverReplyFinished(QNetworkReply*)));
    serverConnected = true;
}
```