TEKNIIKKA JA LIIKENNE

Tietotekniikka

Ohjelmistotekniikka

INSINÖÖRITYÖ

MOBILENOTE-PAIKKATIETOSOVELLUKSEN TOTEUTTAMINEN QT-KEHITYSYMPÄRISTÖSSÄ

Työn tekijä: Jaakko Mutikainen

Työn ohjaajat: Juha-Pekka Kämäri
Olli Alanko

Työ hyväksytty: 18.3.2011

Juha-Pekka Kämäri
Lehtori

Metropolia

ALKULAUSE

Tämä insinöörityö tehtiin yritykselle nimeltä Geometrix Oy. Kiitän toimitusjohtaja Olli Alankoa työn aiheesta ja mahdollisuudesta toteuttaa se. Kiitän myös työtoveriani Aleksi Virtasta, työn ohjaajaani Juha-Pekka Kämäriä ja kielenohjaajaani Jussi Alhorinnettä opastuksesta työn parissa.

Lisäksi kiitän erityisesti vanhempiani ja avopuolisoani Saaraa tuesta ja kannustuksesta opiskeluni eri vaiheissa.

Helsingissä 15.3.2011

Jaakko Mutikainen

Metropolia

TIIVISTELMÄ

Työn tekijä: Jaakko Mutikainen

Työn nimi: Mobilenote-paikkatietosovelluksen toteuttaminen Qt-kehitysympäristössä

Päivämäärä: 15.3.2011 **Sivumäärä:** 53 s. + 1 liite

Koulutusohjelma: Tietotekniikka

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Työn ohjaaja: Lehtori Juha-Pekka Kämäri

Työn ohjaaja: DI Olli Alanko

Tämä insinöörityö tehtiin osana pilottiprojektia, jonka tarkoituksena oli tutkia Qt-teknologian mahdollisuuksia paikkatietosovelluksen alustana. Tarkoituksena oli myös selvittää tarjoaako Qt käytännössä sellaista alustariippumattomuutta mitä sen dokumentointi antaa ymmärtää. Lisäksi tutkittiin uuden Qt Mobilityn tarjoamien karttapalveluiden soveltuvuutta Mobilenote-paikkatietojärjestelmän vaatimuksiin. Työn tuloksena syntyi paikkatietosovellus, jota pystytään suorittamaan Windows- ja Windows Mobile 6 -käyttöjärjestelmissä.

Työssä perehdytään ensin Qt:n lisenssointiin ja erityispiirteisiin. Erityispiirteistä käsitellään signaalit ja slotit, jotka helpottavat olioiden välistä kommunikointia, ja muistinhallinta. Erityispiirteiden lisäksi kerrotaan lyhyesti, mitä ovat Qt Mobility ja Qt Solutions Archive.

Mobilenote on Geometrix Oy:n tuottama paikkatietojärjestelmä, jonka toimintaperiaate esitellään lyhyesti. Projektin määrittely käsittää käyttötapaukset ja sovelluksen tekniset vaatimukset. Kehitysympäristön pystytys kuvataan tarkasti vain niiltä osin, joille ei ole riittävän tarkkaa kuvausta esimerkiksi Qt:n dokumentaatiossa.

Toteutetusta sovelluksesta esitellään arkkitehtuuri (moduulit), luokkarakenne, palvelurajapinnat, tietolähteiden käsittely ja käyttöliittymä. Havainnointiin käytettiin moduuli-, luokka- ja sekvenssikaavioita. Lopuksi esitellään sovelluksen toiminta Windows Mobile 6 -ympäristössä.

Avainsanat: Qt, paikkatieto, mobiilisovellus, Windows Mobile 6

Metropolia

ABSTRACT

Name: Jaakko Mutikainen

Title: Qt implementation of Mobilenote GIS client

Date: 15 March 2011 **Number of pages:** 53 pages + 1 appendice

Department: Information Technology **Study Programme:** Software Engineering

Instructor: Juha-Pekka Kämäri, Senior Lecturer

Supervisor: Olli Alanko, M.Sc. Tech

This thesis was made as a part of Qt pilot project. The goal of the project was to investigate Qt's possibilities as a platform for GIS client application. Also the practice of Qt's cross-platform capabilities was about to be examined. New version of Qt Mobility brings up the Maps API, which provides tools for GIS developers. Applicability of these tools for Mobilenote GIS client was tested. The result of the project is a GIS client, which runs on Windows and Windows Mobile 6 platforms.

The documentation starts with understanding the licensing and characteristics of Qt. The characteristics section clarifies the usage of signals and slots, which ease the communication between objects, and memory management. The idea of Qt Mobility and Qt Solutions Archive are explained.

Mobilenote is a geographical information system created by Geometrix Ltd. System functionality is introduced briefly in the documentation. The project specification includes the use cases and the application's technical requirements. Building of the development environment is fully explained only in parts where the Qt's documentation is inadequate.

Architecture (modules), classes, service interfaces, data handling and user interface of the resulted application are introduced in the documentation. Component, class and sequence diagrams are used to visualize the relations and intercourse between components. At the end, application usage in Windows Mobile 6 is presented.

Keywords: Qt, GIS, mobile application, Windows Mobile 6

SISÄLLYS

1 JOHDANTO

Paikkatietosovellukset ovat älykkäiden mobiililaitteiden myötä nousseet esiin ohjelmistojen tarjonnassa. Yritykset ovat havainneet paikkatiedon käyttömahdollisuudet tiedonkeruussa ja -hallinnassa, sekä erilaisissa kenttätehtävissä. Tavalliselle kuluttajalle paikkatieto on tuonut mm. lenkkeilyyn liittyviä tiedonkeruuohjelmia, ilmaisen navigaattorin (Nokia) ja aarteenetsinnän (geocaching).

Mobiilisovellusten kysynnän kasvaessa kehittimiin on lisätty työkaluja helpottamaan mobiilisovellusten kehittämistä. Useista mobiililaitteita tukevista kehitysympäristöistä (Qt, Java, Android) löytyykin jo esim. visuaaliseen käyttöliittymäsuunnitteluun tarvittavat välineet. Lisäksi ympäristöt osaavat usein kääntää sovelluksen suoraan kohdelaitteelle. Näin ollen kynnys ohjelmoida mobiililaitetta madaltuu jatkuvasti. Nykyään mobiililaitteille tarkoitetuista kehitysympäristöistä löytyvät usein tarvittavat työkalut myös paikkatietosovelluksen toteuttamiseen.

Työ tehtiin Geometrix Oy:lle, jossa olin ohjelmoijaharjoittelijana syksyllä 2010 ja keväällä 2011. Yrityksen toimialana ovat paikkatietojärjestelmät ja tiedonhallintaan liittyvät ohjelmistot. Yrityksellä on kaupallinen tuote, joka on nimeltään Mobilenote. Konsepti on paikkatietojärjestelmä, joka tarjoaa paikkatietokohteiden operoinnin työasemalla ja mobiililaitteessa. Ajatuksena on, että tiedon keruu (kohteiden luominen) tapahtuu mobiililaitteella ja työasemalla voidaan sitten tarkastella näitä tietoja. Kohteita tarkasteltaessa niistä voidaan luoda raportteja esimerkiksi erilaisten urakkatöiden seurantaa ja laskutusta varten. Tuote erottuu esimerkiksi ilmaisesta Ovi Kartat -ohjelmistosta mm. siten, että sillä voidaan tuottaa asiakkaan tarpeisiin räätälöityjä paikkatietokohteita ja taustakarttoina voidaan käyttää omia aineistoja.

Mobilenoten mobiilisovellus on alun perin toteutettu Java-tekniikalla Nokian S60-sovellusalustalle. Tämän työn tavoitteena oli toteuttaa konseptin uudella teknologialla, joka mahdollistaisi joustavamman kohdealustojen käytön. Uuden toteutuksen tulisi toimia Windows Mobile 6 -ympäristössä, jossa Java-sovellus ei toimi suoraan. Saman sovelluksen tulisi toimia myös Windowsin työpöydällä.

Uutta toteutusta ryhdyttiin tekemään Nokian Qt-teknologialla, joka tarjoaa tällaisen alustariippumattomuuden. Ratkaisuun päädyttiin, koska teoriassa tässä kehitysympäristössä tuotettu sovellus kääntyy suoraan niille kohdealustoille, joiden tukea uudelta toteutukselta vaaditaan. Vaatimuksena tässä työssä oli siis toimivuus seuraavilla alustoilla:

- Windows
- Windows Mobile 6.x

Rakenteellisesti sovelluksesta päätettiin tehdä mahdollisimman modulaarinen. Tällä tarkoitettiin sitä, että sovelluksen loogiset kokonaisuudet toteutettaisiin erillisiksi ohjelmakirjastoikseen, joita voidaan helposti päivittää tai jotka voidaan helposti vaihtaa kokonaan uuteen.

Projektin tavoitteena oli myös hyödyntää uuden Qt Mobility -version (1.1.0) Maps API -ohjelmointirajapintaa, joka helpottaa paikkatiedon käsittelyä Qt-sovelluksessa.

2 MIKÄ ON QT?

Qt on Nokian kehittämä kehitysympäristö, jonka avulla voidaan kirjoittaa ohjelmia käyttöliittymineen suoraan usealle alustalle [1]. Tämä tarkoittaa sitä, että täsmälleen sama ohjelmakoodi voidaan kääntää esimerkiksi Windowsille ja Windows Mobilelle. Yhteensopivuuden takaa ohjelmaan linkitettävät Qt-kirjastot, joiden rajapintoja ja palveluja vasten oma ohjelma kirjoitetaan.

2.1 Qt-lisenssit

Qt on saatavilla kolmen erilaisen lisenssin alla. Ensimmäinen niistä on kaupallinen lisenssi (Qt Commercial Developer License), joka mahdollistaa ns. suljettujen ohjelmistojen kehittämisen. Tällöin minkään ohjelman osan lähdekoodia ei tarvitse julkaista jakelun mukana. Lisenssin alla jopa Qt-kirjastoja voi muokata ilman, että muokattua lähdekoodia tarvitsee julkaista. Käytettäessä tätä vaihtoehtoa Qt-kirjastot voidaan linkittää sovellukseen staattisesti. [2.]

Kaksi muuta vaihtoehtoa ovat avoimen lähdekoodin lisenssien alla. Näistä ensimmäinen, Qt GNU LGPL, mahdollistaa ohjelman jakelun osittain suljettuna. Tämä tarkoittaa sitä, että oman sovelluksen lähdekoodia ei välttämättä

tarvitse julkaista [3], mutta jos Qt-kirjastoja on muutettu, on muutokset julkaistava [2]. Jos oma lähdekoodi halutaan pitää suljettuna, Qt-kirjastot on linkitettävä sovellukseen dynaamisesti. Staattinen linkitys tarkoittaisi sitä, että myös oman sovelluksen olisi noudatettava GNU LGPL-lisenssiä, jolloin lähdekoodi olisi julkaistava. [3.]

Viimeinen vaihtoehto on Qt GNU GPL, joka vaatii kaiken tuotetun ohjelmakoodin julkaisemisen. [4.]

2.2 Erityispiirteet

Qt-ohjelmoinnissa käytetty teknologia on yleensä C++, johon Qt-kirjastojen käyttö tuo joitakin lisäominaisuuksia. Tämä luku esittelee lyhyesti näistä tärkeimmät.

2.2.1 Periyttäminen Qt-luokista

Qt-kirjastojen tarjoamien luokkien kantaluokka on QObject. QObject on Qt:n luokkahierarkiassa ylimpänä [5], joten se on periytetty kaikkiin aliluokkiin joko suoraan tai epäsuorasti [6]. Kirjoitettaessa omia Qt-sovelluksen luokkia kannattaa myös ne periyttää tästä, koska se tarjoaa kaksi tärkeää ominaisuutta: automaattisen lapsiolioiden tuhoamisen, sekä signaalien ja slottien käytön [5].

Automaattinen lapsiolioiden tuhoaminen tarkoittaa sitä, että kun tuhotaan jokin QObject-olio, sen lapsioliot tuhotaan luokan destruktorissa automaattisesti. Tämä on toteutettu Qt:ssa siten, että kun luodaan vanhemman (parent) omaava lapsiolio (child), lapsiolio lisää itsensä vanhempansa children-listaan. Kun vanhempi tuhotaan, tuhotaan myös kaikki children-listan oliot. [5.]

2.2.2 Signaalit ja slotit

Signaalit ja slotit ovat käytettävissä kaikissa QObject-luokasta periytetyissä luokissa, joissa on määritetty Q_OBJECT-makro. Ne ovat Qt:n vaihtoehtoinen tapa callback-funktioiden toteutukselle. Callback-funktioita käytetään yleensä olioiden väliseen kommunikointiin. Qt:ssa tämä kommunikointi voidaan suorittaa määrittämällä oliolle signaaleja, joita se voi lähettää järjestelmän sisällä. Vastaanottaville olioille kirjoitetaan sitten funktioita (slotteja), joihin signaali voidaan ohjata. Tästä seuraa tilanne, että kun olio lähettää sig-

naalin, vastaanottava olio suorittaa signaaliin liitetyn funktion. Signaali on parametrisoitavissa, eikä sen parametrien määrä ole rajoitettu. Vastaanottava slotti voi itse päättää, mitä signaalin parametreja se haluaa käsitellä. Ylimääräiset voidaan jättää huomiotta. Signaalin parametrisoinnin on kuitenkin aina vastattava siihen kytketyn slotin parametrisointia. [6.]

Yksi signaali on mahdollista kytkeä useaan slottiin ja päinvastoin useampi signaali voidaan kytkeä yhteen slottiin. Signaalin voi myös kytkeä toiseen signaaliin, jolloin jälkimmäinen signaali lähetetään heti ensimmäisen jälkeen. [6.]

Qt käyttää ominaisuuksiensa luomiseen meta-objekteja, jotka sisältävät mm. signaali- ja slotti-funktioiden kuvaukset ja sijainnin muistissa. Meta-objektit luodaan luokkakuvauksen perusteella, kun niiden luomiseen tarkoitettu kääntäjä (meta-object compiler tai moc) ajetaan ennen ohjelman varsinaista kääntämistä. [6.]

Seuraavana on yksinkertainen esimerkki signaalin kytkemisestä slottiin. Esimerkissä on kaksi luokkaa (oliota), joista toinen lähettää signaalin ja toinen vastaanottaa sen. Kytkeminen tapahtuu pääohjelmassa. Huomataan, että olioiden ei tarvitse olla toisistaan tietoisia.

Signaalin määrittävä luokka

```
#include <QObject>

class Sender : public QObject
{
    Q_OBJECT

public:
    explicit Sender(QObject *parent = 0) {}

    void setNewValue(int newValue) {
        emit changeValue(newValue);
    }
signals:
    void changeValue(int newValue);
};
```

Luokka Sender lähettää signaalin changeValue(int), kun sen jäsefunktiota setNewValue(int newValue) kutsutaan. Signaalit määritellään aina avainsanan ’signals’ alle. Niiden paluuarvo on aina ’void’ eikä niille tule koskaan kir-

joittaa toteutusta [6]. Avainsana ’signals’ tukee näkyvyysmäärittelyjä normaalin luokkakuvauksen tapaan. Esim. ’public signals’ tai ’protected signals’.

Slotin määrittävä luokka

```
#include <QObject>

class Receiver : public QObject
{
    Q_OBJECT

public:
    explicit Receiver(QObject* parent = 0) {
        m_value = 0;
    }
    int value() const
    {
        return m_value;
    }
public slots:
    void setValue(int value) {
        m_value = value;
    }
private:
    int m_value;
};
```

Luokka Receiver vastaanottaa Senderin lähettämän signaalin changeValue(int). Signaali on ohjattu slottiin setValue(int). Slotit ovat luokan jäsenfunktioita, joita voidaan kutsua myös tavalliseen tapaan, mutta ne määritellään avainsanan ’slots’ alle [6]. Avainsana ’slots’ tukee näkyvyysmäärittelyjä normaalin luokkakuvauksen tapaan. Esimerkiksi ’public slots’ tai ’protected slots’.

Pääohjelma

```
#include "sender.h"
#include "receiver.h"
#include <QtCore/QCoreApplication>
#include <QDebug>

int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);

    Sender sender(&a);
    Receiver receiver(&a);

    QObject::connect(&sender, SIGNAL(changeValue(int)),
                     &receiver, SLOT(setValue(int)));

    sender.setNewValue(5);        // Lähettää signaalin
    qDebug() << receiver.value(); // Tulostaa luvun 5

    return a.exec();
}
```

Tässä esimerkissä pääohjelma huolehtii signaalien ja slottien kytkemisestä. Kytkeminen voidaan kuitenkin hoitaa missä tahansa, milloin tahansa kutsumalla funktiota QObject::connect(). Funktio ottaa parametrikseen sekä lähettäjän ja sen signaalin, että vastaanottajan ja sen slotin. Signaali ja slotti erotellaan makroilla SIGNAL ja SLOT. Ohjelma tulostaa luvun 5.

Signaalit ja slotit tarjoavat merkittäviä etuja callback-mekanismiin nähden. Signaalit ja slotit ovat mm. aina tyyppitarkistettuja (type-safe). Tämä tarkoittaa sitä, että kun slottia kutsutaan, voidaan olla varmoja siitä, että slotti käsittelee signaalin lähettämät parametrit oikein. Callback-mekanismia käytettäessä ei voi koskaan olla varma, kutsutaanko funktiota oikeilla parametreilla. Toinen merkittävä etu on se, että signaalin lähettäjän ja sen vastaanottajan ei tarvitse olla tietoisia toisistaan, kun taas callback-funktion kutsujan on tarkkaan tiedettävä mitä funktiota kutsutaan. [6.]

2.3 Qt Mobility

Qt-kirjastojen lisänä voi käyttää Qt Mobility -kirjastoja, jotka tarjoavat yksinkertaisen rajapinnan kohdelaitteen palvelujen ja sisään rakennettujen laitteiden käyttöön. Näitä palveluja ovat esimerkiksi kännykän GPS-paikannin, viestipalvelut, kamera ja erilaiset anturit. Qt Mobilitystä on saatavilla avoimen

lähdekoodin versio (LGPL) tai kaupallinen Qt Commercial License -lisenssin alainen versio. [14.]

Qt Mobility ei sisällä täyttä tukea Windows Mobilelle, joten Windows Mobile -laitetta käytettäessä täytyy osaan laitteistosta käyttää Windows Mobilen omia ohjelmointirajapintoja. [15.]

2.4 Qt Solutions Archive

Qt Solutions Archive on Qt Mobilityn tapaan kokoelma komponentteja, joita voi käyttää apunaan Qt-kehitystyössä. Qt Solutions Archive sisältää mm. komponentin, joka helpottaa MFC-sovelluksen integroimista Qt-kehitysympäristöön ja komponentin, joka tarjoaa tuen web-palvelun käyttöön SOAP-protokollan välityksellä. Ero verrattuna Qt Mobilityyn on, että Nokia ei osallistu aktiivisesti näiden komponenttien kehitystyöhön. Lisäksi tämän kokoelman komponentit ovat kaikki BSD-lisenssin alaisia. [16.]

BSD-lisenssi on hyvin löyhä avoimen lähdekoodin lisenssi. Sen nojalla komponenttia voi muuttaa ja jakaa eteenpäin ilman, että muutokset täytyy julkaista. Riittää, että komponentti sisältää lisenssissä mainitut tekijänoikeustiedot, lisenssin ehdot ja vastuuvapauslausekkeen. [17.]

3 MOBILENOTE LYHYESTI

Mobilenote on maastossa tapahtuvaan tiedonkeruuseen ja -selailuun tarkoitettu järjestelmä. Sovelluksella voidaan hakea ja päivittää haluttuja paikkatietokohteita halutun taustakartan ollessa taustalla. Sovellus osaa käsitellä useita taustakartta-aineistoja ja niitä voi lennossa vaihtaa. [10, s. 5]

Jokaisella paikkatietokohteella on ominaisuuksia (attribuuttitietoa) ja geometria. Geometria kertoo, minkälainen kuvio ja mihin kohtaan kartalle se piirretään. Kuvio voi olla joko piste, viiva tai alue. Lisäksi kohteilla voi olla liitteitä kuten kuvia, saneluja tai videoita, ja niitä voidaan poistaa tai lisätä sovelluksen avulla. Myös täysin uusia kohteita voidaan luoda. [10, s. 5]

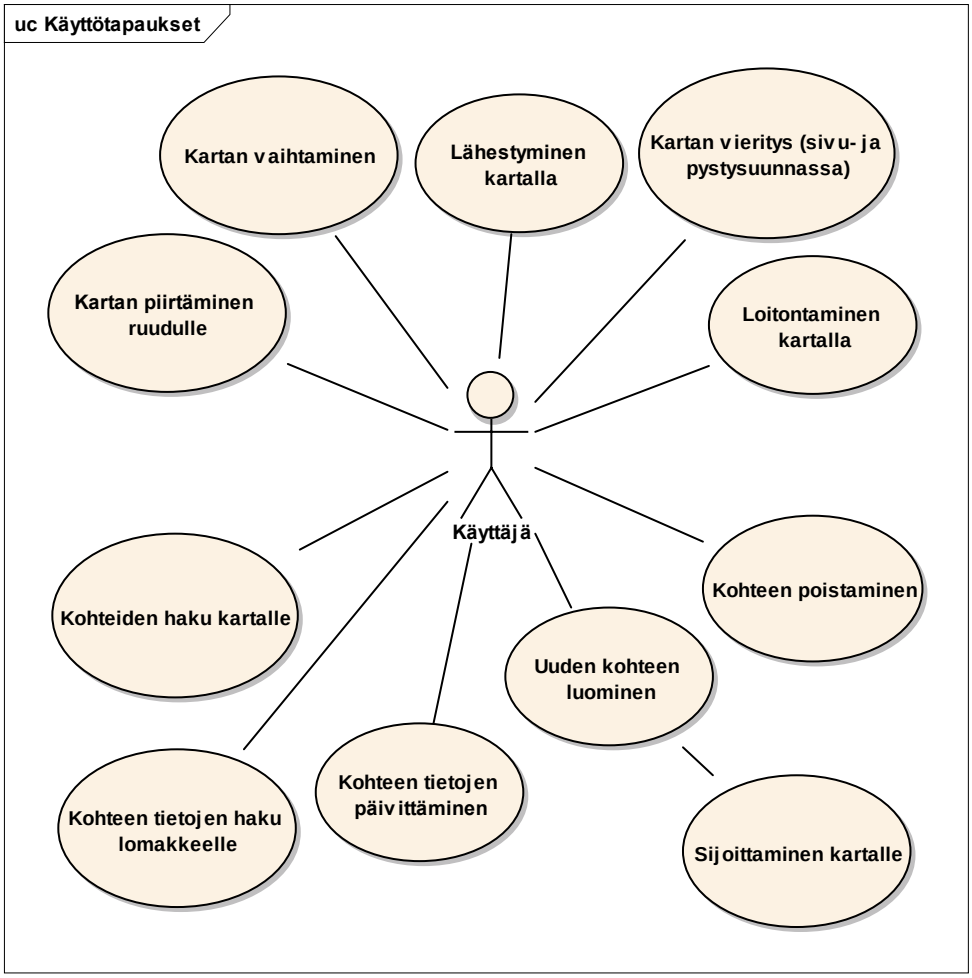
4 PROJEKTIN MÄÄRITTELY

Projektin määrittely oli toteutustavan kannalta väljä, koska tarkoitus oli lähinnä pilotoida Qt-teknologiaa ja kartoittaa sen mahdollisuuksia. Painopiste

määrittelyssä onkin käyttötapauksissa ja teknisissä vaatimuksissa. Tämä luku esittelee ensin käyttötapaukset ja sen jälkeen tekniset vaatimukset.

4.1 Käyttötapaukset

Sovellusta käytettäessä suoritetaan toimintoja, kuten kohteiden haku kartalle ja kohteen tietojen näyttäminen lomakkeella sitä klikattaessa. Käyttötapaukset on rajattu vain osaan Mobilenoten kaikista mahdollisista toiminnoista. Alle on kuvattu projektin vaatimuksiin luetut käyttötapaukset.



Kuva 1. Käyttötapauskaavio.

Käyttötapauksista kartan käsittelyyn liittyvät asiat rajattiin siten, että sovelluksen tulee pystyä käsittelemään yhtä rasterikarttaa. Päällekkäisten vektorikarttojen tuki jää jatkokehitysprojekteille. Rasterikartta tarkoittaa sitä, että yksi maantieteellinen piste esitetään ruudulla yhtenä pikselinä [12]. Käytännössä se tarkoittaa sitä, että kartta on kuvatiedosto, joka on formaatiltaan esimerkiksi JPEG tai PNG.

Kohteiden käsittelyn vaatimuksiin luettiin ainoastaan yhden kohdetyypin käsittely. Tämän kohdetyypin nimi on Merkintä, joka on pistemäinen kohde. Konseptina Mobilenote tukee muitakin kohdetyyppejä, jotka voivat olla esimerkiksi viivamaisia tai aluemaisia kohteita. Seuraava luku käsittelee määrittelyyn sisällytetyt tekniset vaatimukset.

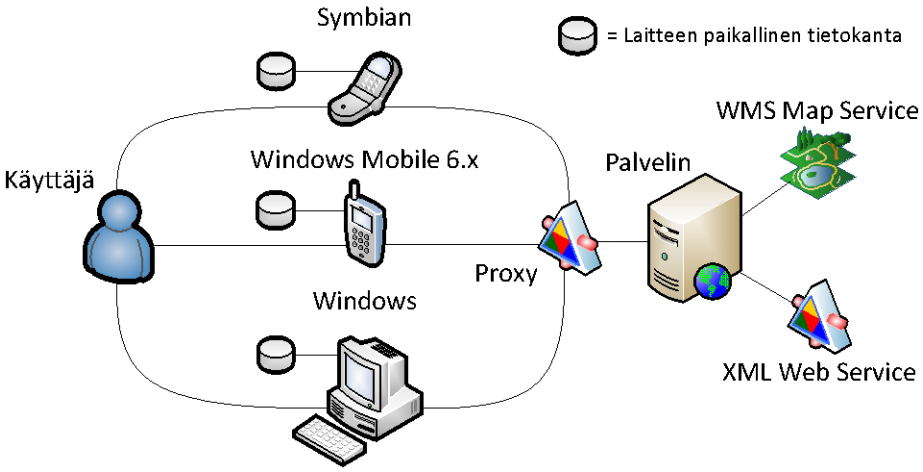
4.2 Tekniset vaatimukset

Määrittelyn tekniset vaatimukset pitävät sisällään vaatimuksia liittyen kohdealustoihin ja vaatimuksiin, joita Mobilenoten web-palvelu asettaa. Web-palvelun vaatimukset liittyvät HTTP-pyyntöihin, XML-muotoisen tiedon käsittelyyn ja kartta-aineiston käsittelyyn. Näiden lisäksi vaatimuksiin kuuluu web-palvelun tarjoaman tiedon käsittely paikallisessa tietokannassa.

Yleisesti ottaen sovelluksen on oltava helposti laajennettavissa, jolloin aikaisemmassa Java-toteutuksessa jo olevia, tai kokonaan uusia toimintoja voidaan helposti lisätä toteutuksen logiikkaan ja käyttöliittymään. Sovelluksen tulee toimia Windows-, Windows Mobile 6 -ympäristöissä.

4.2.1 Web-palvelut ja paikallinen tietokanta

Mobilenote tarjoaa kohdeaineistoaan Javalla toteutetun XML Web Servicen välityksellä. Kartat tarjotaan Web Map Servicen kautta. Nämä palvelut ovat saatavilla kolmannen web-palvelun, Proxyn, kautta. Proxy on se palvelu, johon sovelluksen tulee olla yhteydessä. Pyynnöt Proxylle lähetetään XML-muotoisina HTTP-pyyntöinä ja vastaus palauttaa XML-muotoista dataa. Sovelluksen on kyettävä muodostamaan oikean kaltaisia XML-pyyntöjä ja osattava purkaa tarvittava tieto pyynnön vastauksesta.



Kuva 2. Kaaviokuva järjestelmän rakenteesta.

Kartat Mobilenote tarjoaa palvelimen karttapalvelusta WMS-rajapinnan kautta. Myös WMS-palvelu on Proxyn takana, joten karttapyynnöt on osoitettava sille. Karttapyynnöt ovat tavallisia HTTP-pyyntöjä, joiden muodostamiseen ei tarvita XML-käsittelyä. Sovelluksen on kuitenkin kyettävä muodostamaan Proxyn määrityksiä noudattava pyyntö ja osattava käsitellä sen vastaus.

Kartat, jotka on haettu palvelimelta, on pystyttävä tallentamaan laitteen paikalliseen tietokantaan. Jos tieto on kertaalleen tallennettu tietokantaan, se osataan noutaa sieltä palvelimen sijaan.

4.2.2 Koordinaatisto

Web-palvelusta saatava kohde- ja kartta-aineisto on YKJ-koordinaatistossa. Sovelluksen on pystyttävä näyttämään kartta ja kohteet oikein näytöllä. Jos luodaan uusi kohde ja se sijoitetaan kartalle, tulee kohteen koordinaattien tallentua YKJ-koordinaatistoon.

KKJ on kaksiulotteinen kartastokoordinaattijärjestelmä, joka koostuu kuudesta (0-5) kolme astetta leveästä kaistasta, jotka kattavat koko Suomen alueen. YKJ-koordinaatisto on kartastokoordinaattijärjestelmän (KKJ) kolmas kaista, jota käytetään valtakunnan laajuisesti. Tästä kolmannesta kaistasta käytetään nimitystä KKJ3 tai Yhtenäiskoordinaatisto (YKJ). Koska KKJ on kaksiulotteinen koordinaatisto, eikä siihen liity mitään määritelmää korkeudesta, sovelluksen tarvitsee osata ainoastaan leveys- ja pituustietojen käsittely. [13.]

5 KEHITYSYMPÄRISTÖ

Tämä luku käsittelee kehitysympäristön osalta niitä osa-alueita, jotka eivät ole itsestään selviä aloittelevalle Qt-kehittäjälle. Käytetyn kehitysympäristön tuli kyetä tuottamaan ohjelma, joka toimii Windows- ja Windows Mobile 6 -ympäristössä. Kehitys tapahtui Windowsissa. Oleellista näiden vaatimusten puitteissa on ymmärtää, miten kehitysympäristö konfiguroidaan tukemaan kirjoitetun ohjelman Windows- ja Windows Mobile 6 -käännöksiä.

Seuraavaksi ohjeistetaan, miten Qt-kirjastot käännetään ja asennetaan vaadituille kohdealustoille. Ohjeistuksessa oletetaan, että liitteessä yksi mainitut ohjelmat on asennettuna. Toteutuksessa päätettiin käyttää Qt:n avointa, Qt GNU LGPL -lisenssin ehtojen alaista versiota.

5.1 Qt-kirjastot Windowsille ja Windows Mobilelle

Nokia Qt SDK ei tue kirjoitetun ohjelman Windows- tai Windows Mobile -käännöstä suoraan, vaan Qt-kirjastot näille alustoille on asennettava erikseen. Asennuksen voi suorittaa kahdella tavalla: Konfiguroimalla ja kääntämällä kirjastot lähdekoodista tai lataamalla valmiiksi käännetyt kirjastot Qt:n sivuilta. Kun kirjastot löytyvät järjestelmästä, niiden sijainti pitää kertoa koodieditorille (Qt Creator). Windows Mobile -ympäristöön tarkoitetut kirjastot kulkevat nimellä Qt for Windows CE.

5.1.1 Kirjastojen kääntäminen lähdekoodista

Kirjastojen avoimen lähdekoodin voi halutessaan kääntää itse haluttuun kohdeympäristöön. Alla ovat ohjeet, kuinka kirjastot käännetään Windowsille ja Windows Mobilelle.

Aluksi tarvitaan lähdekoodi, josta kirjastot halutaan kääntää. Qt tarjoaa paketin nimeltä qt-everywhere-opensource-src-4.7.0.zip, joka sisältää ainoastaan kirjastojen avoimen lähdekoodin. Everywhere-paketin voi ladata osoitteesta http://qt.nokia.com/downloads otsikon Qt:Framework Only alta. Paketti kannattaa purkaa hakemistoon C:\Qt.

Seuraavaksi käännetään lähdekoodista sekä Windows- että Windows Mobile 6 -käännös. Tämä tehdään käyttämällä ActivePerl:n tarjoamaa shadow-build -tekniikkaa. Shadow-buildin avulla samasta lähdekoodista voidaan kääntää eri hakemistoihin eri tavalla konfiguroituja käännöksiä. Siten lähdekoodia ei tarvitse monistaa useaan hakemistoon useita käännöksiä varten [7].

Käännöksissä komennot suoritetaan Visual Studion komentokehotteessa, joka löytyy käynnistä-valikon alta Microsoft Visual Studio 2005 / Visual Studio Tools / Visual Studio 2005 Command Prompt. Seuraavat tekstikehykset esittävät Visual Studion komentokehotetta, johon syötettävät komennot [7; 8] on merkitty lihavoituna.

Windows

```
Setting environment for using Microsoft Visual Studio 2005 x86
tools.

C:\Program Files\Microsoft Visual Studio 8.0\VC>mkdir
C:\Qt\vs2005-shadow

C:\Program Files\Microsoft Visual Studio 8.0\VC>cd C:\Qt\vs2005-
shadow

C:\Qt\vs2005-shadow>C:\Qt\qt-everywhere-opensource-src-
4.7.0\configure.exe -platform win32-msvc2005

C:\Qt\vs2005-shadow>nmake
```

Windows Mobile

```
Setting environment for using Microsoft Visual Studio 2005 x86
tools.

C:\Program Files\Microsoft Visual Studio 8.0\VC>mkdir
C:\Qt\mobile6-shadow

C:\Program Files\Microsoft Visual Studio 8.0\VC>cd
C:\Qt\mobile6-shadow

C:\Qt\mobile6-shadow>C:\Qt\qt-everywhere-opensource-src-
4.7.0\configure.exe -platform win32-msvc2005 -xplatform
wincewm60professional-msvc2005

C:\Qt\mobile6-shadow>cd bin

C:\Qt\mobile6-shadow\bin>setcepaths wincewm60professional-
msvc2005

C:\Qt\mobile6-shadow\bin>cd..

C:\Qt\mobile6-shadow>nmake
```

Komentojen suorittamisen jälkeen käännökset löytyvät shadow-päätteisistä kohdehakemistoista. Tätä työtä tehdessä törmättiin ongelmiin, joissa kaikkien koodiesimerkkien käännöstä ei voitu suorittaa. Tällöin komento ’nmake’ keskeytyy. Varsinaiset Qt-kirjastot on kuitenkin jo käännetty ennen esimerkkejä, joten tästä tilanteesta ei koidu haittaa ympäristön pystyttämisen kannalta.

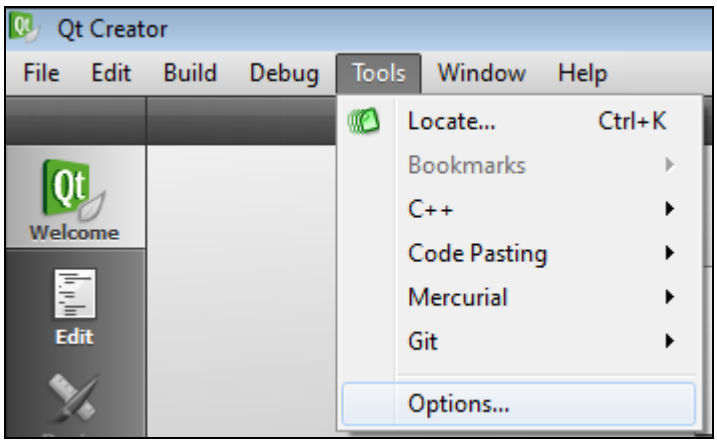
5.1.2 *Kirjastojen lataaminen netistä*

Valmiiksi käännetyt kirjastot eri kohdealustoille voi ladata myös netistä osoitteesta http://qt.nokia.com/downloads. Kun kirjastot on purettu, pitää vielä kertoa Qt Creatorille, missä ne sijaitsevat (ks. seuraava luku).

5.1.3 *Qt Creatorin konfigurointi*

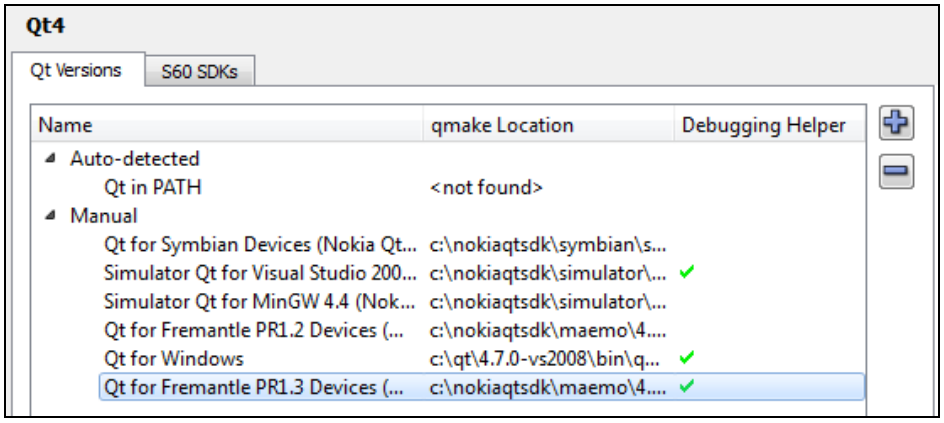
Kun halutaan käyttää Qt-kirjastoja, joita ei automaattisesti toimiteta Nokia Qt SDK:n mukana, pitää Qt Creatorille kertoa, missä hakemistossa kirjastot sijaitsevat. Alla olevassa esimerkissä Qt Creatoriin konfiguroidaan mahdollisuus käyttää Windows-kirjastoja. Nämä ohjeet pätevät käytettäessä Qt Creatorin versiota 2.0.1.

1. Valitaan Qt Creatorin valikosta Tools / Options…



Kuva 3. Qt Creatorin asetuksiin siirtyminen.

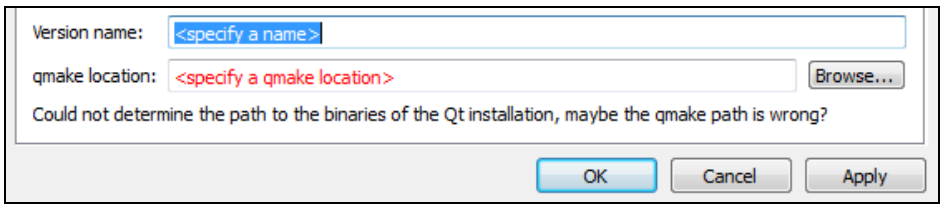
2. Valitaan vasemmalta listasta kohta Qt4. Eteen avautuu seuraava näkymä.



Kuva 4. Qt-käännösten hallinta.

Huom! Jos on käytössä useita kirjastoja, ei välttämättä kannata käyttää ympäristömuuttujaan PATH sijoitettua Qt-hakemistoa. Se tarkoittaisi sitä, että joka kerta, kun haluttaisiin muuttaa käännöksen kohdeympäristöä, pitäisi myös ympäristömuuttujan sisältöä muuttaa.

3. Asennettujen kirjastojen tiedot saatetaan Qt Creatorin tietoon lisäämällä ne kohtaan 'Manual'. Tämä tapahtuu klikkaamalla oikealla ylhäällä näkyvää plus-merkkiä.



Kuva 5. Uuden Qt-käännöksen lisääminen.

4. Syötetään version nimi (kuvaus) ja kerrotaan tiedoston qmake.exe sijainti.

5.2 Qt Mobility -kirjastojen käyttöönotto

Qt Mobility -kirjastot ovat osana Nokian Qt SDK:ta, mutta ainoastaan Symbianille ja Maemolle. Windowsille ja Windows Mobilelle ne täytyy kääntää erikseen.

Käyttöönottoa varten tarvitaan Qt Mobilityn lähdekoodi, jonka saa osoitteesta http://qt.nokia.com/products/qt-addons/mobility. Paketti kulkee nimellä qt-mobility-opensource-src-1.1.0.zip. Toisin kuten Qt-kirjastojen tapauksessa, Qt Mobility -kirjastojen käännös ei tue shadow-buildeja. Lähdekoodipaketti on siten syytä purkaa omiin kansioihinsa jokaista kohdealustaa varten. Alla olevissa esimerkeissä paketti on purettu seuraaviin hakemistoihin.

- Windows C:\QtMobility\vs2005
- Windows Mobile 6 C:\QtMobility\mobile6

Kuten Qt-kirjastojen käännöksissä, komennot suoritetaan Visual Studion komentokehotteessa. Seuraavat tekstikehykset esittävät tätä komentokehotetta. Syötettävät komennot [9] on merkitty lihavoituna.

Windows

```
Setting environment for using Microsoft Visual Studio 2005 x86
tools.

C:\Program Files\Microsoft Visual Studio 8.0\VC>cd
C:\QtMobility\vs2005

C:\QtMobility\vs2005>configure –prefix C:\QtMobility\vs2005-
install

C:\QtMobility\vs2005>nmake

C:\QtMobility\vs2005>nmake install
```

Windows Mobile

```
Setting environment for using Microsoft Visual Studio 2005 x86
tools.

C:\Program Files\Microsoft Visual Studio 8.0\VC>cd
C:\Qt\mobile6-shadow\bin

C:\Qt\mobile6-shadow\bin>setcepaths wincewm60professional-
msvc2005

C:\Qt\mobile6-shadow\bin>cd C:\QtMobility\mobile6

C:\QtMobility\mobile6>configure –prefix C:\QtMobility\mobile6-
install

C:\QtMobility\mobile6>nmake

C:\QtMobility\mobile6>nmake install
```

Esimerkeissä käytetty -prefix-optio määrittelee hakemiston, jonne komento ’nmake install’ asentaa käännetyt kirjastot ja käyttöön tarvittavat lähdekooditiedostot. Jos tätä optiota ei anneta, asennetaan käännös oletuksena lähdekoodihakemiston alle kansioon install. Windows-esimerkin tapauksessa polku olisi tällöin C:\QtMobility\vs2005\install. [9.]

Qt Mobility otetaan käyttöön kopioimalla kirjasto- ja lähdekooditiedostot oikeisiin hakemistoihin. Käytettäessä shadow-buildeja täytyy hakemisto install\inc kopioida hakemistoon, jossa shadow-buildin lähdekoodi sijaitsee

(esim. C:\Qt\qt-everywhere-opensource-src-4.7.0). Install-hakemiston muu sisältö kopioidaan sinne, missä varsinainen käännös sijaitsee (esim. C:\Qt\vs2005-shadow).

Jos Qt-kirjastojen käännöksessä ei ole käytetty shadow-buildia, voi -prefix-option asettaa suoraan Qt-hakemistoon (esim. C:\Qt\vs2005), jolloin kaikki tarvittavat tiedostot kopioituvat suoraan oikeisiin hakemistoihin.

Käännösten jälkeen Qt-projektin .pro-tiedostoon tulee lisätä alla olevat rivit. Esimerkissä projekti konfiguroidaan käyttämään mobilityn QtLocation-kirjastoa. Jos teksti ’location’ vaihdettaisiin tekstiin ’bearer’, käyttäisi projekti mobilityn QtBearer-kirjastoa. Useita kirjastoja voi käyttää erottamalla sanat toisistaan välilyönnillä.

```
CONFIG += mobility
MOBILITY += location
#MOBILITY += location bearer
```

5.3 Qt SOAP -komponentin käyttöönotto

Qt SOAP -komponentin käyttöönotto vaatii komponentin käännöksen halutulle kohdealustalle. Lähdekoodin voi ladata osoitteesta http://qt.gitorious.org/qt-solutions.

Komponentin lähdekoodipaketin mukana tulee projektitiedosto (.pro), jonka voi suoraan avata Qt Creatoriin ja projektin voi kääntää eri kohdeympäristöille. Lähdekoodihakemiston include-hakemisto pitää konfiguroida oman projektin .pro-tiedostoon, muuttujaan INCLUDEPATH. Kääntäjää varten myös LIBS-muuttujaan on sisällytettävä tiedoston QtSolutionsSOAP-2.7.lib sijainti. Seuraavana on esimerkki projektitiedostoon liitettävistä riveistä.

```
INCLUDEPATH += c:/Qt/qtsoap-2.7_1-opensource/src
LIBS += c:/Qt/qtsoap/lib/QtSolutions_SOAP-2.7.lib
```

5.4 SQLite-ajurin käyttöönotto

Qt pitää sisällään useita valmiita tietokanta-ajureita ja yksi niistä on tarkoitettu SQLite-tietokannan käsittelyyn (qsqlite4.dll). Tietokanta-ajuri voidaan kääntää osaksi QtSql4-kirjastoa tai sitä voidaan käyttää erillisenä komponenttina (plugin). Käyttötavan voi valita configure-käskyn parametrilla ennen

kuin Qt-kirjastot käännetään. Oletuksena (ilman parametrisointia) QtSql4 kääntyy siten, että ajuria voi käyttää pluginina.

Jos ajuria käyttää pluginina, se pitää sijoittaa ohjelman ajohakemiston hakemistoon plugins/sqldrivers. Pluginin tiedostonimen pitää olla qsqlite4.dll. Qt-projektin .pro-tiedostoon tulee lisätä seuraava rivi.

```
QT += sql
```

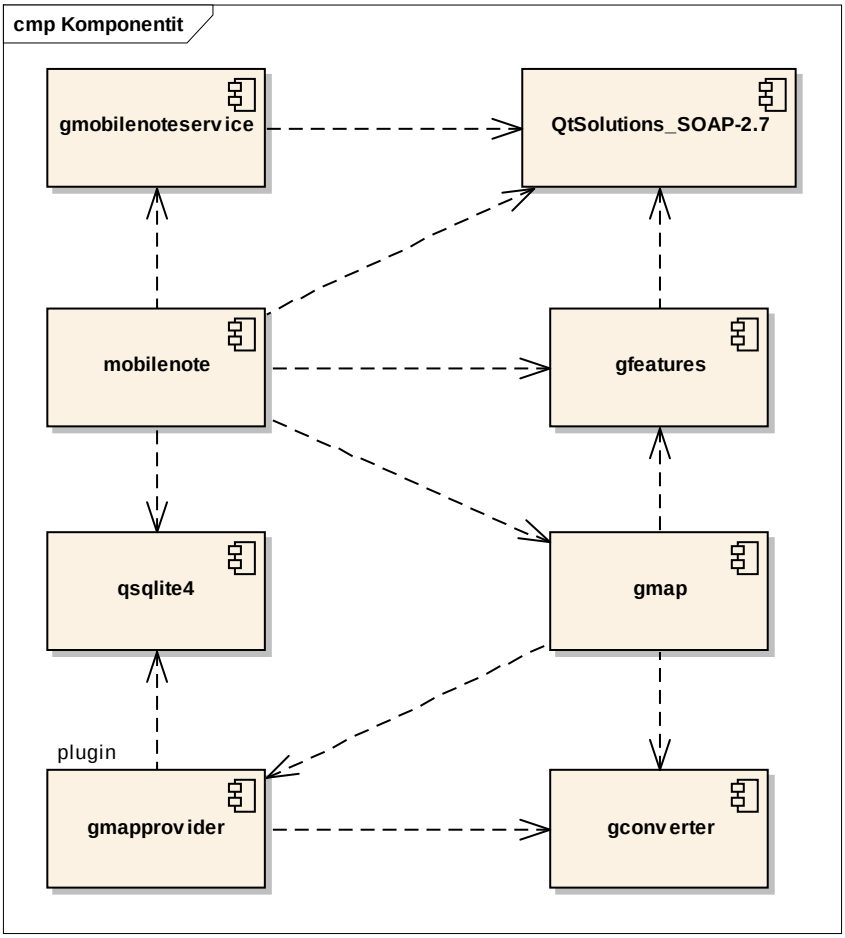
6 QT-TOTEUTUS

Qt-toteutus pohjautuu edellä esitettyihin määrityksiin. Tämä luku esittelee toteutuksen arkkitehtuurin, palvelurajapinnat, tietolähteiden käsittelyn ja sovelluksen käyttöliittymän.

6.1 Arkkitehtuuri (moduulit)

Arkkitehtuurisuunnittelussa pyrittiin modulaarisuuteen, jolloin jokainen loogi-nen kokonaisuus olisi toteutettu omaan moduuliinsa. Tämä luku esittelee työn tuloksena tuotetut moduulit ja käytetyt kolmannen osapuolen komponentit. Moduulien sisäiset luokkarakenteet käsitellään luvussa 6.3.

Moduulirakenne on toteutettu siten, että sovellus koostuu pääsovelluksesta (exe) joka itsessään sisältää palveluja, mutta se käyttää niitä myös ulkoisista kirjastoista (dll). Kuvassa 6 on moduulikaavio, josta käy ilmi moduulit ja niiden väliset suhteet.



Kuva 6. Moduulien väliset suhteet.

Taulukossa 1 on listattu sovellukseen toteutetut moduulit ja lyhyt kuvaus niiden toiminnan tarkoituksesta.

Taulukko 1. Toteutetut moduulit.

Moduuli	**Tiedosto**	**Kuvaus**
MobilenoteService	gmobilenoteservice.dll	Ohjelmakirjasto, joka määrittää web-palvelun rajapinnan ja tarjoaa sen käyttöön tarvittavat luokat.
Kartta	gmap.dll	Ohjelmakirjasto, joka tarjoaa karttapalvelut. Sisältää kartan logiikan ja käyttöliittymän toteutuksen.
Taustakartat	qtgeoservices_gmx.dll	Ohjelmakirjasto, johon on toteutettu taustakarttojen haku verkon yli ja niiden tallentaminen paikalliseen tietovarastoon.
Projektori	gconverter.dll	Ohjelmakirjasto, joka tarjoaa

		työkalut tarvittavien koordinaattimuunnosten tekemiseen.
Kohdekäsittely	gfeatures.dll	Ohjelmakirjasto, joka määrittää kohdekäsittelyn rajapinnan ja tarjoaa kohdekäsittelyssä tarvittavat luokat.
Mobilenote	mobilenote.exe	Sovelluksen pääohjelma. Moduuli käyttää muiden moduulien palveluja.

Taustakartat-moduuli on Qt-plugin, jonka Qt lataa ohjelman ajon aikana dynaamisesti. Qt olettaa sen sijaitsevan ajohakemiston alihakemistossa plugins\geoservices. Myös muut ohjelmakirjastot linkitetään pääohjelmaan dynaamisesti, mutta ne eivät ole Qt-plugineja. Jotta sovellus osaa ladata kirjastot, niiden täytyy oletuksena löytyä samasta hakemistosta pääohjelman kanssa. Taulukossa 2 ovat sovelluksen käyttämät kolmannen osapuolen komponentit.

Taulukko 2. Kolmannen osapuolen moduulit.

Moduuli	**Tiedosto**	**Kuvaus**
Qt SOAP	QtSolutions_SOAP-2.7.dll	Ohjelmakirjasto, helpottaa XML-tiedonsiirtoa verkon yli.
SQLite driver	qsqlite4.dll	Qt:n plugin, joka sisältää SQLite-tietokannan käsittelyn.

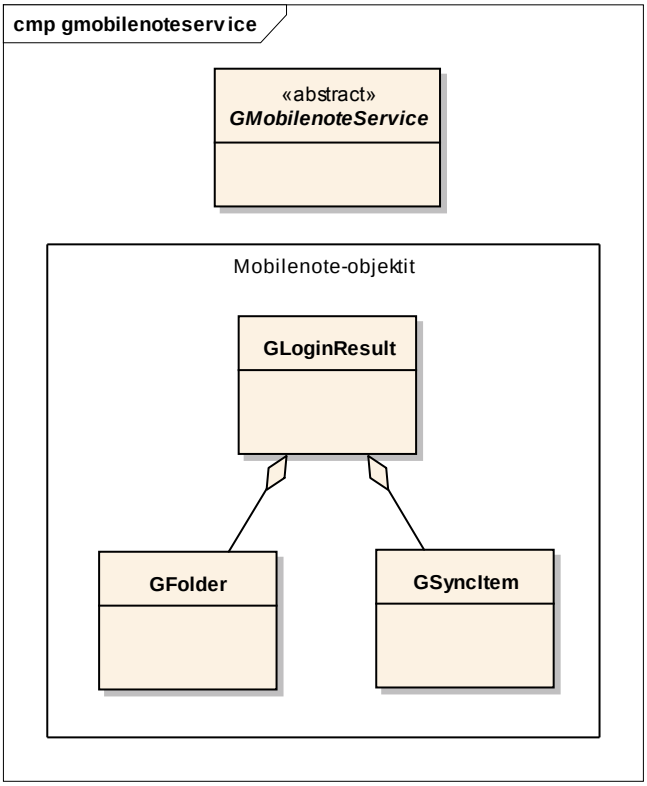
Kolmannen osapuolen kirjastoista SOAP linkitetään sovellukseen dynaamisena kirjastona, joka ei ole plugin. SQLiten ajuri on Qt-plugin, jonka Qt lataa hakemistosta plugins/sqldrivers. Seuraavassa luvussa käsitellään moduulien sisältämät luokat ja niiden väliset suhteet.

6.2 Luokkarakenne

Sovelluksen luokkarakenne esitetään seuraavaksi moduuleittain (komponenteittain). Moduulin luokkarakenteen kuvaus sisältää luokkakaaviot ja selitykset luokkien tehtävästä. Osa edellisessä luvussa esitetyistä moduulien välisistä riippuvuuksista ilmenee myös moduulien luokkakaavioista.

6.2.1 MobilenoteService

MobilenoteService-komponentti tarjoaa rajapinnan Mobilenoten web-palvelun käyttöä varten. Web-palvelun käyttö vaatii mm. sisäänkirjautumisen ja käytettävän kansion valinnan. Komponentti sisältää ainoastaan paluutiedon käsittelyyn tarvittavat luokat ja palvelun rajapinnan. Rajapinnan toteutus on kirjoitettu pääohjelmamoduuliin.



Kuva 7. MobilenoteService-komponentin luokkakaavio.

Taulukkoon 3 on selitetty luokkakaaviossa käytettyjen luokkien tehtävät.

Taulukko 3. MobilenoteService-komponentin luokkakuvaus.

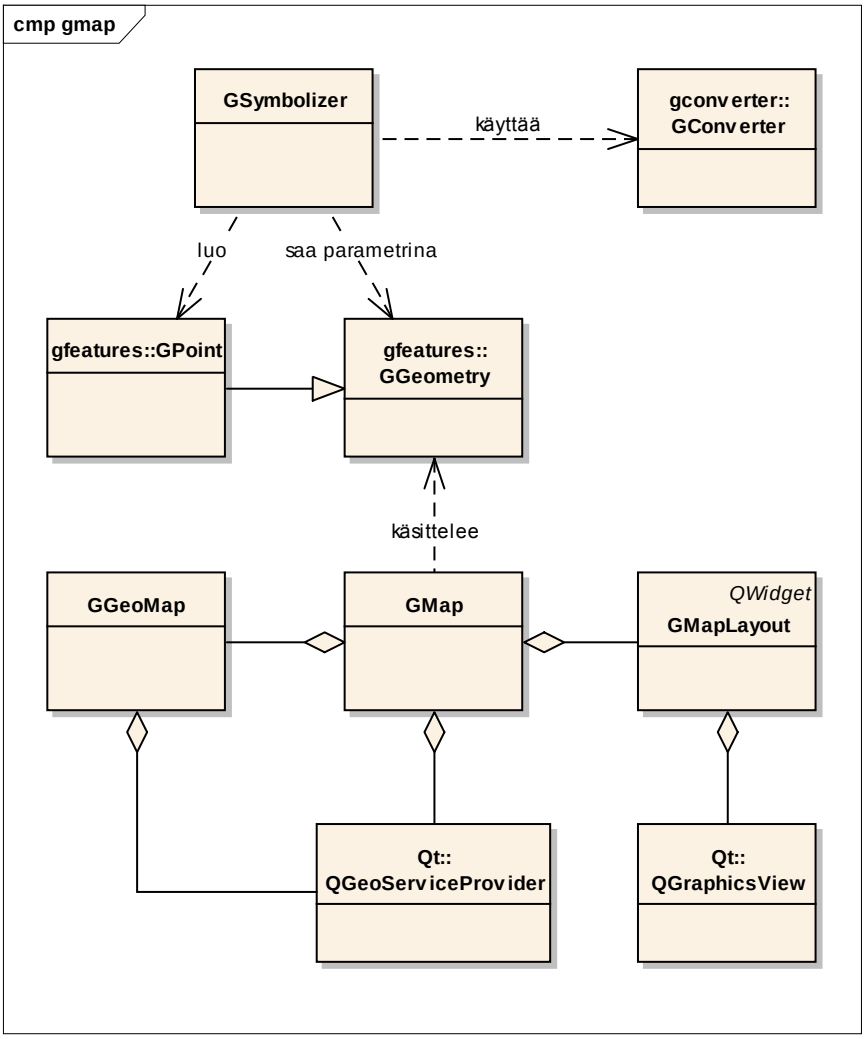
Luokka	**Kuvaus**
GMobilenoteService	Määrittää Mobilenoten web-palvelun rajapinnan.
GLoginResult	Objekti, johon tallennetaan Mobilenoten sisäänkirjautumisen yhteydessä saadut tiedot.
GFolder	Objekti, johon tallennetaan yhden Mobilenote-kansion tiedot.
GSyncItem	Objekti, johon tallennetaan Mobilenoten synkronointitietoa.

Luokkakaaviosta käy ilmi, että luokka GLoginResult riippuu luokista GFolder ja GSyncItem. Tämä johtuu siitä, että GLoginResult sisältää kansio- ja synkronointitietoja, jotka on tallennettu luokkien GFolder- ja GSyncItem-instansseihin.

Kansio on Mobilenoten käsite, jonka avulla voidaan ryhmitellä kohteita: Kohde voi kuulua tiettyyn kansioon, jonka perusteella se voidaan hakea kartalle. Synkronointitiedot sisältävät aikaleimoja kohdetyyppien ja karttojen kuvauksien tiedoista. Aikaleimoja käytetään, kun tutkitaan, onko sovelluksen paikallinen tietokanta ajan tasalla.

6.2.2 Kartta

Karttakomponentti tarjoaa pääohjelmalle yhden luokan (GMap), jossa ovat pääohjelman näkökulmasta katsottuna tarvittavat toiminnot. Moduuli ei siis tarjoa abstraktia rajapintaa, vaan valmiin toteutuksen ja siten eroaa esimerkiksi MobilenoteService- tai Kohdekäsittely-moduulista. Kuvan 8 luokkakaavioon on merkitty moduuliin toteutetut luokat, komponentin kannalta olennaiset Qt-luokat ja luokkien väliset kytkökset.



Kuva 8. Kartta-komponentin luokkakaavio.

Taulukkoon 4 on selitetty luokkakaaviossa käytettyjen luokkien tehtävät.

Taulukko 4. Kartta-komponentin luokkakuvaus.

Luokka	**Kuvaus**
GMap	Kokoaa kartan komponentit yhden luokan alle, jota pääohjelma kutsuu. Instantioi tarvittavat komponentit ja käyttöliittymän. Hoitaa kommunikoinnin Taustakartat-komponentin kanssa. Päämoduuli instantioi ainoastaan tämän luokan tästä moduulista.
GMapLayout	Toteuttaa karttanäkymän (luo graafisen QGraphicsView-komponentin). Pääohjelma pyytää ja näyttää ruudulla tämän luokan instanssin.
GGeoMap	Qt Mobility Maps API:n karttakomponentin toteuttava widget-luokka. Sisältää käyttäjän ja kartan välisen interaktion toteutuksen. GGeoMap lisätään GMapLayoutiin, joka huolehtii sen piirtämisestä ruudulle.

GSymbolizer	Singleton, joka tekee GGeometry-objekteista kartalle piirrettäviä objekteja.

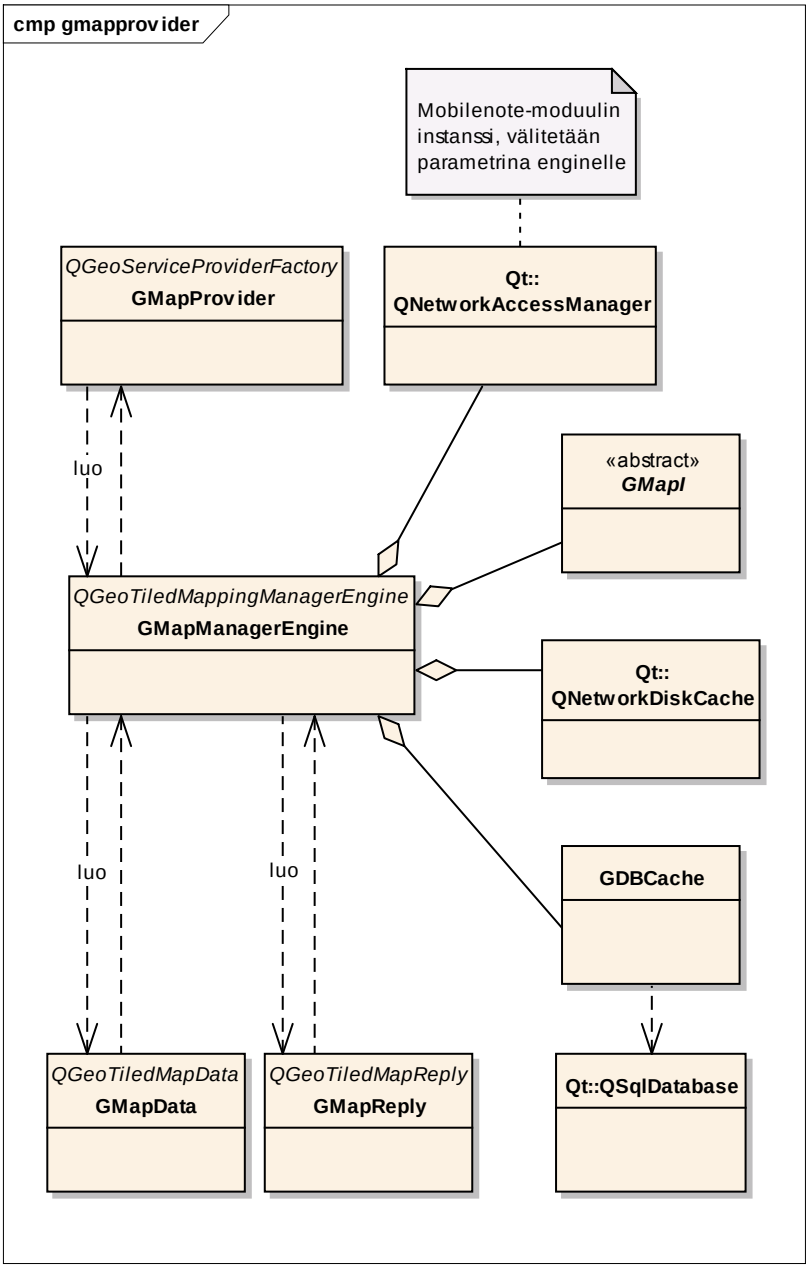
Luokkakaavion keskellä on ydinluokka GMap, joka luo käyttöönsä karttanäkymän (GMapLayout) ja Maps API:n karttakomponentin (GGeoMap). Karttakomponentti piirtyy ruudulle, kun GGeoMap-luokan instanssi liitetään QGraphicsView-komponentin näkymään (scene).

Ydinluokka luo QGeoServiceProvider-luokan instanssin, jota GGeoMap käyttää taustakarttojensa käsittelyyn. Luodun instanssin mappingManager() (GMapManagerEngine) välitetään GGeoMap-oliolle sen konstruktorin parametrina. GMapManagerEngine on osa Taustakartat-moduulia.

GMap käyttää GSymbolizer-luokkaa silloin, kun kohteen geometria halutaan näyttää kartalla. Se osaa muuntaa geometrian graafiseksi elementiksi. Luokka on singleton ja sitä käytetään vain silloin, kun sitä tarvitaan.

6.2.3 Taustakartat (Qt-plugin)

Taustakartat-komponentti on Qt-plugin, jonka Kartta-komponentti lataa ajon aikana. Komponentti tarjoaa taustakarttojen käsittelyyn tarvittavat työkalut. Seuraava luokkakaavio esittelee moduulin keskeisimmät luokat ja niiden väliset riippuvuudet.



Kuva 9. Taustakartat-komponentin luokkakaavio.

Taulukkoon 5 on selitetty luokkakaaviossa käytettyjen luokkien tehtävät.

Taulukko 5. Taustakartat-komponentin luokkakuvaus.

Luokka	**Kuvaus**
GMapProvider	Tehdasluokka, joka luo GMapManagerEnginen.
GMapManagerEngine	Moottori, jota Kartta-komponentin GGeoMap-luokka kutsuu halutessaan tietyn karttaruudun. Sisältää logiikan, joka määrää haetaanko karttaruutu paikallisesta tietolähteestä vai verkon yli.

GMapI	Rajapinta, jonka avulla kommunikoidaan GMap-luokan instanssin kanssa.
GMapData	Sisältää funktioita, joita Kartta-komponentin GGeoMap kutsuu, kun se haluaa tehdä koordinaattimuunnoksia. Funktioita voi kutsua myös itse.
GMapReply	Huolehtii karttapyynnon vastauksen käsittelystä. Osaa käsitellä sekä paikalliset, että verkon yli lähetettävät pyynnöt.
GDBCache	Tietokantakäsittelijä, joka osaa tallettaa karttaruutuja paikalliseen tietokantaan ja hakea niitä sieltä.

Kun edellisessä luvussa esitelty Kartta-komponentti luo itselleen taustakarttojen käsittelijän, se kutsuu GMapProvider-luokan tehdasmetodia, joka palauttaa GMapManagerEngine-luokan instanssin. Tehdasmetodi ottaa parametrikseen listan ohjelmoijan määräämiä parametreja, jonka se välittää GMapManagerEnginen konstruktorille.

Parametrilista sisältää osoittimen päämoduulin QNetworkAccessManager-tyyppiseen jäsenmuuttujaan. GMapManagerEngine määrätään käyttämään tätä samaa instanssia. Parametrilista sisältää myös osoittimen kutsujaansa, eli Kartta-komponentin GMap-instanssiin. GMapI:n toteuttajaksi sijoitetaan tämä GMap-instanssi, jotta GMapManagerEngine voi reagoida GMapin lähettämiin signaaleihin.

Sovellukseen on konfiguroitavissa joko tiedosto- tai tietokantapohjainen karttaruutujen säilömistapa. QNetworkDiskCache-luokkaa moduuli käyttää tiedostopohjaisen välimuistin luomiseen eikä sen käyttö vaadi ohjelmointia. Pelkästään QNetworkDiskCache-olion luominen QNetworkAccessManagerin käyttöön riittää [11]. GDBCache on ohjelmoitu toteutus karttaruutujen tietokantaan tallentamista varten.

GMapData on objekti, joka liitetään jokaiseen ruudulla näkyvään karttaelementtiin. Se tarjoaa koordinaattimuunnoksiin käytettävät funktiot, joita tarvitaan, koska karttaelementtien koordinaatit näyttöruudulla poikkeavat niiden maantieteellisistä koordinaateista. Esim. näyttöruudun koordinaatti voi olla suhteellinen näytön vasemman yläkulman nurkkapikselistä sata pikseliä alas ja sata pikseliä oikealle (100, 100). Tätä vastaava maantieteellinen koordinaatti voi olla (lat 60.16324, lon 24.93254). Muunnosfunktiot osaavat koordinaattien muunnokset molempiin suuntiin.

6.2.4 Projektori

Projektori-moduuli sisältää ainoastaan yhden luokan, joka tarjoaa koordinaattimuunnoksiin liittyviä funktioita.

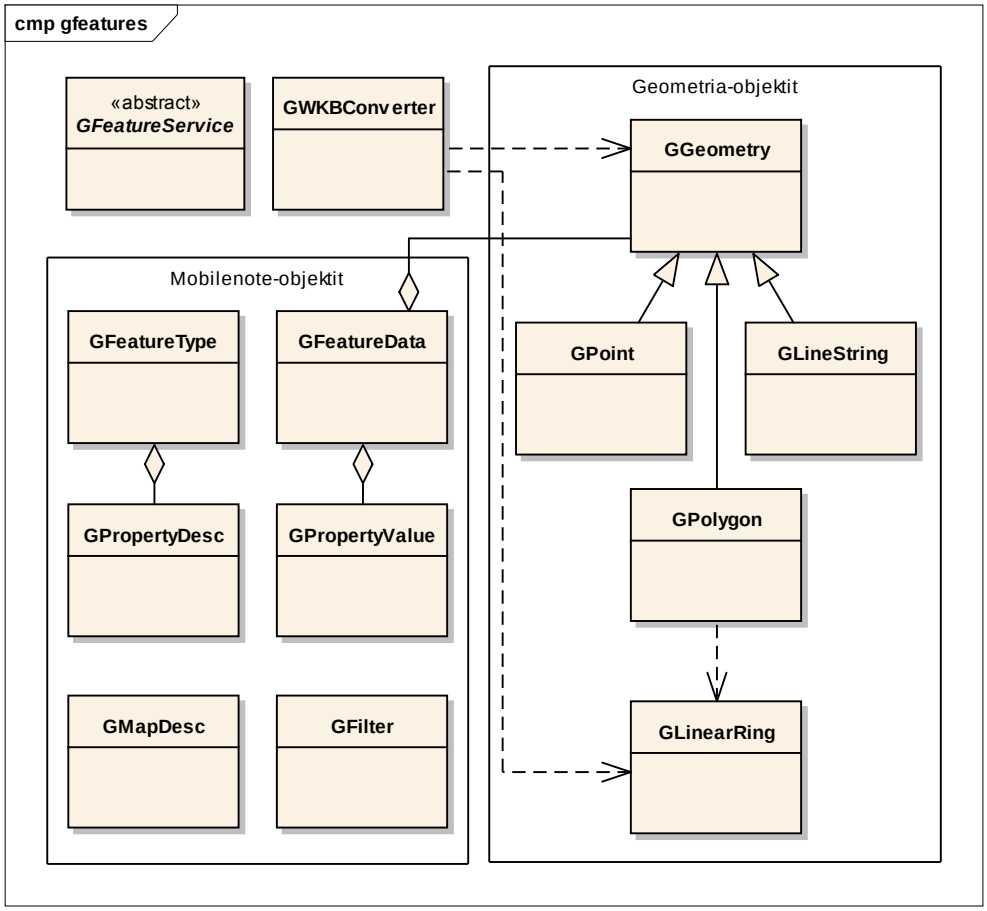
Taulukko 6. Projektori-moduulin luokkien kuvaus.

Luokka	**Kuvaus**
GConverter	Singleton-luokka sisältää koordinaattimuunnoksiin käytettäviä funktioita.

6.2.5 Kohdekäsittely

Kohdekäsittely-komponentti tarjoaa rajapinnan kohteiden käsittelyä varten ja sisältää tähän tarvittavat luokat. Kohteiden käsittely käsittää kohdetyyppien tietojen (metadata) ja kohteiden varsinaisen datan käsittelyn. Metadata sisältää tietyn kohteen tietokenttien tiedot (esim. nimi, tietotyyppi) ja varsinainen data taas tietokenttien sisällön ko. kohteelle. Moduuli osaa käsitellä molempia. Kohteen varsinainen data sisältää lisäksi kohteen geometrian, jonka käsittelyyn tarvittavat luokat löytyvät tästä moduulista.

Moduuli osaa käsitellä karttojen metatietoja (esim. nimi, julkaisija), mutta karttojen varsinainen data (karttaruudut) käsitellään Taustakartat-moduulissa. Kuvaan 10 on kuvattu moduulin luokkakaavio.



Kuva 10. Kohdekäsittely-komponentin luokkakaavio.

Taulukkoon 7 on selitetty luokkakaaviossa käytettyjen luokkien tehtävät.

Taulukko 7. Kohdekäsittely-moduulin luokkien kuvaus.

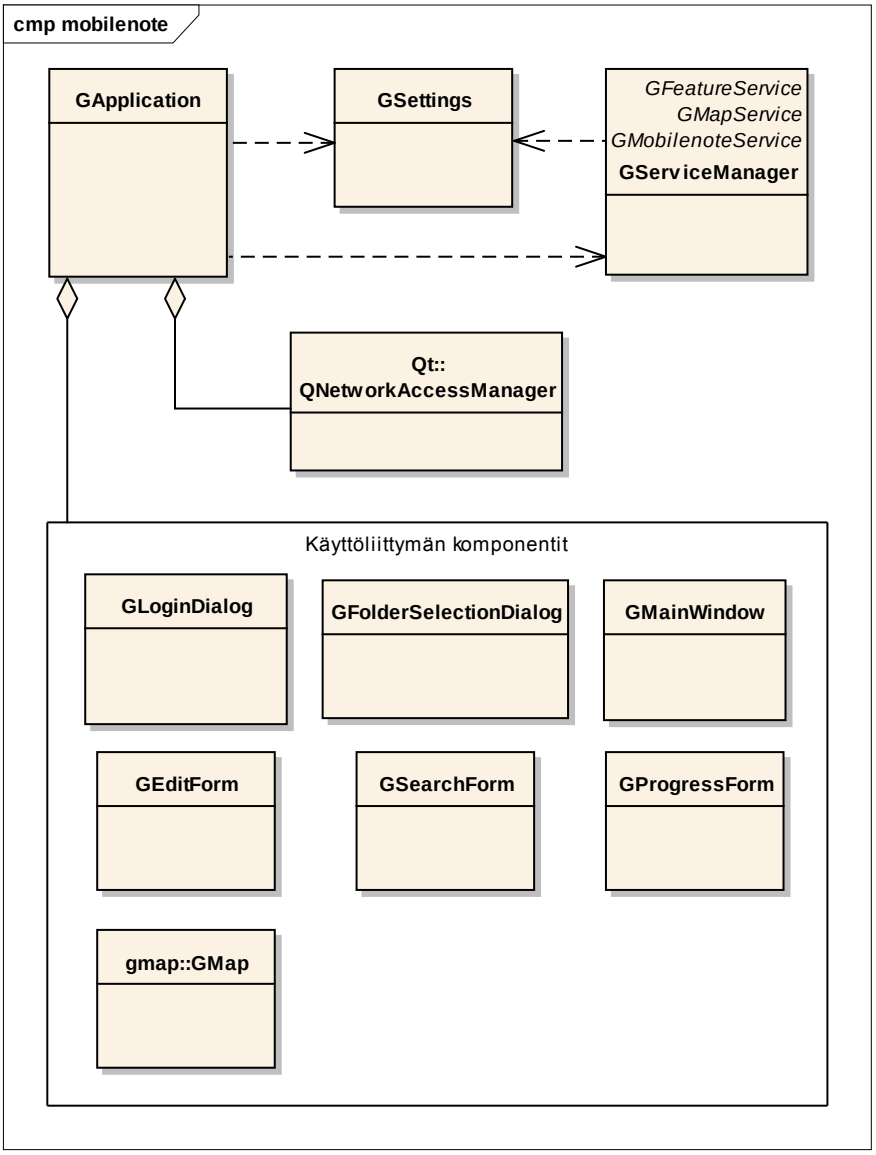
Luokka	**Kuvaus**
GFeatureService	Rajapinta, joka määrää kohdekäsittelyssä käytettävät funktiot.
GMapDesc	Sisältää yhden kartan metatiedot.
GFilter	Objekti, joka sisältää kohteiden haun hakukriteerit.
GFeatureType	Sisältää yhden kohdetyypin metadatan. Metadata sisältää mm. 0..n kappaletta ominaisuuksia (property), jotka ovat niitä tietoja, jotka näytetään kohdetietolomakkeella.
GPropertyDesc	Sisältää yhden kohteen ominaisuuden (property) metadatan. Esim. kuvauksen (”Numero”) ja tietotyypin (int).
GFeatureData	Sisältää yhden kohteen varsinaisen datan. Sisältää kohdetyypin metadatan määräämän kappalemäärän ominaisuuksien arvoja (property value) ja kohteen geometrian.
GPropertyValue	Sisältää yhden kohteen yhden ominaisuuden arvon.

GGeometry	Kuvaa kohteen geometrian yleisellä tasolla ja sisältää kaikille geometrioille yhteiset ominaisuudet. Toimii yläluokkana erilaisille geometriatypeille (GPoint, GPolygon, GLineString).
GPoint	Sisältää kohteen pistemäisen geometrian tiedot.
GPolygon	Sisältää kohteen aluemaisen geometrian tiedot.
GLineString	Sisältää kohteen viivamaisen geometrian tiedot.
GWKBConverter	Singleton-luokka, joka osaa muuntaa kohteen geometrian GGeometry-objektiksi ja päinvastoin.

Moduulin tarkoituksena on tarjota Mobilenote-moduulille työkalut, joilla se pystyy helposti tuottamaan kohteisiin liittyvät tietokentät kohdetietolomakkeelle ja jotka tarjoavat vaivattoman kohteiden tietojen (ominaisuustietojen ja geometrian) käsittelyn.

6.2.6 Mobilenote

Mobilenote-moduuli on sovelluksen päämoduuli, joka sisältää pääohjelman, käyttöliitymän ja palvelurajapintojen toteutukset. Kartalle päämoduuli ainoastaan varaa tilaa käyttöliittymässään, koska kartan käyttöliittymä (GMapLayout) on toteutettu karttamoduuliin itseensä. Lisäksi Mobilenote-moduuli sisältää toteutuksen konfigurointitiedoston (settings.ini) käsittelyn ja pitää yllä instanssia QNetworkAccessManagerista, jota käytetään HTTP-pyyntöjen muodostamiseen. Kuvassa 11 on luokkakaavio päämoduulin pääluokista ja niiden välisistä suhteista. Palvelurajapinnat toteuttava GServiceManager esitetään omana luokkakaavionaan myöhemmin.



Kuva 11. Mobilenote-komponentin pääluokkakaavio.

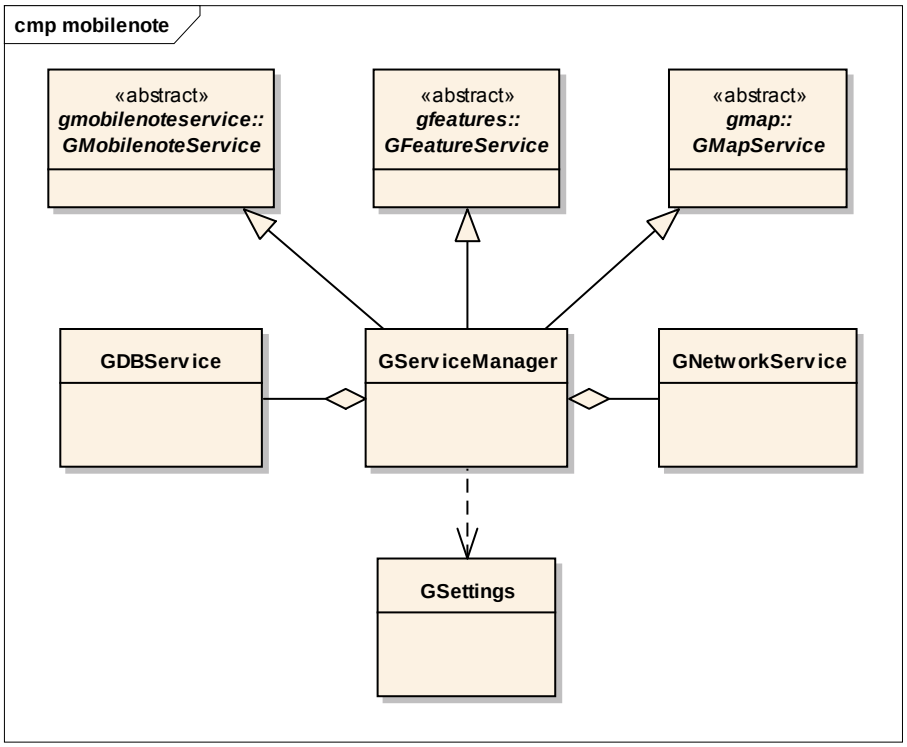
Taulukkoon 8 on selitetty pääluokkakaaviossa käytettyjen luokkien tehtävät.

Taulukko 8. Mobilenote-moduulin pääluokkien kuvaus.

Luokka	**Kuvaus**
GApplication	Sovelluksen ydinluokka. Luo käyttöliittymät ja palvelut. Ohjailee sovelluksen toimintaa.
GSettings	Luokka konfigurointitiedoston käsittelyyn. Osaa lukea ja kirjoittaa arvoja.
GServiceManager	Palvelurajapintojen toteuttaja.
GLoginDialog	Luokka, joka näyttää sisäänkirjautumis-dialogin.
GFolderSelectionDialog	Luokka, joka näyttää ’Kansion valinta’ -dialogin.
GMainWindow	Sovelluksen pääikkuna. Sisältää raamit kartalle ja

	lomakkeille, ja valikon.
GEditForm	Lomake kohteen tietojen editointiin. Lomake näytetään GMainWindow-ikkunassa sille tarkoitetussa raamissa.
GSearchForm	Lomake kohteiden hakua varten. Lomake näytetään GMainWindow-ikkunassa sille tarkoitetussa raamissa.
GProgressForm	Luokka, joka näyttää dialogin, jossa voidaan ilmaista kaksivaiheisen prosessin edistymistä.
GMap	Kartta-komponentin ydinluokka.

Päämoduuli on suoritettava ohjelma (exe), josta koko sovellus käynnistyy. Moduuli instantioi kaiken tarvittavan käyttöönsä ja huolehtii niiden varaaman muistin vapauttamisesta. Se pitää muistissaan koko ohjelman suorituksen ajan mm. kohdetyyppien tiedot. Kuvan 12 luokkakaaviossa kuvataan palvelurajapinnat toteuttava GServiceManager.



Kuva 12. Palvelurajapintojen toteutuksen luokkakaavio.

Taulukkoon 9 on selitetty luokkakaaviossa käytettyjen luokkien tehtävät.

Taulukko 9. Palvelurajapintojen toteuttavien luokkien kuvaus.

Luokka	**Kuvaus**
GServiceManager	Tämä luokka toteuttaa kaikki kolme rajapintaa, jotka määräävät web-palveluun kohdistuvat pyynnöt. Sisältää logiikan, joka päättää haetaanko pyyntö paikallisesti tietokannasta, vai tehdäänkö se verkon yli.
GDBService	Luokka, joka suorittaa palvelurajapintojen määrittämät pyynnöt tietokantaan.
GNetworkService	Luokka, joka suorittaa palvelurajapintojen määrittämät pyynnöt verkon yli.

GApplication luo itselleen GServiceManagerin suorittaakseen palvelurajapintojen pyyntöjä. GServiceManager luo alustamisen yhteydessä itselleen sekä tietokanta-, että verkkokäsittelijän (GDBManager ja GNetworkManager). Kun GApplication suorittaa pyyntöjä, GServiceManager päättää, mihin tietolähteeseen pyyntö lopulta päätyy, eli kumpaa luokkaa se käskee.

6.3 Palvelurajapinnat

Tässä luvussa esitellään sovelluksessa käytetyt palvelurajapinnat, eli moduulien MobilenoteService, Kartta ja Kohdekäsittely määrittämät rajapinnat.

6.3.1 MobilenoteService

GMobilenoteService-rajapinta määrittää metodit Mobilenoten web-palveluun kirjautumista varten. Funktiot sisältävät sisään- ja uloskirjautumismetodit ja kansion avaus- ja luontimetodit. Seuraavaksi on taulukoitu kaikki rajapinnan metodit selityksineen. Metodit ovat kaikki abstrakteja, vailla toteutusta (pure virtual).

Taulukko 10. Login-metodin kuvaus.

void login (const QString& username, const QString& password)	
• Suorittaa kirjautumisen Mobilenoteen annetulla käyttäjätunnuksella ja salasanalla.	
const QString& username	Mobilenoten käyttäjätunnus
const QString& password	Käyttäjätunnuksen salasana

Taulukko 11. CreateFolder-metodin kuvaus.

void createFolder (const QString& name)	
• Luo kansion nimeltä *name*.	
const QString& name	Kansion nimi.

Taulukko 12. OpenFolder-metodin kuvaus.

void openFolder (int id)	
• Avaa kansion, jonka id on *id*.	
int id	Kansion yksilöivä tunniste.

Taulukko 13. Logout-metodin kuvaus.

void logout (GFolder* folder)	
• Kirjaa käyttäjän ulos Mobilenotesta.	
GFolder folder*	Kansion tiedot sisältävä objekti.

Toteutetussa sovelluksessa tämän rajapinnan toteuttajana toimii Mobilenote-moduulin luokka GServiceManager, joka suorittaa kaikki nämä pyynnöt ainoastaan verkon yli. Toteutus ei tue istuntojen tai kansioiden käsittelyä tietokannassa.

6.3.2 Kartta

GMapService-rajapinta määrittää metodin, jonka avulla Mobilenoten web-palvelusta voidaan kysellä karttoihin liittyviä metatietoja. Lisäksi se sisältää metodin, jolla verkosta haettu kartan kuvaus voidaan tallentaa paikalliseen tietokantaan. Seuraavaksi on taulukoitu kaikki rajapinnan metodit selityksineen. Metodit ovat kaikki abstrakteja, vailla toteutusta (pure virtual).

Taulukko 14. DescribeMap-metodin kuvaus.

void describeMap (const QString& layer, qlonglong updated)	
• Suorittaa pyynnön, joka hakee kartan *layer* metatiedot.	
const QString& layer	Karttatason nimi (WMS-taso).
qlonglong updated	Sisäänkirjautumisen yhteydessä saatu aikaleima, joka kertoo milloin ko. kartan tietoja on päivitetty web-palveluun. Tätä aikaleimaa verrataan tietokannan aikaleimaan ja mikäli ne poikkeavat, haetaan tuore tieto verkosta.

Taulukko 15. InsertMapDescToDB-metodin kuvaus.

void insertMapDescToDB (GMapDesc* mapDesc)	
• Tallentaa kartan metatiedot paikalliseen tietokantaan.	
GMapDesc mapDesc*	Objekti, joka sisältää kartan metatiedot.

Sovelluksessa tätä rajapintaa toteuttaa Mobilenote-moduulin GServiceManager. Tietokannan käsittely on keskeinen osa tämän rajapinnan toteutusta, koska verkon yli haetaan tietoa vain silloin, kun sitä ei ole saatavilla tietokannasta.

6.3.3 Kohdekäsittely

GFeatureService-rajapinta määrittää metodit, joiden avulla sovellus voi käsitellä kohteiden kuvauksia ja dataa. Seuraavaksi on taulukoitu kaikki rajapinnan metodit selityksineen. Metodit ovat kaikki abstrakteja, vailla toteutusta (pure virtual).

Taulukko 16. DescribeFeature-metodin kuvaus.

void describeFeature (const QString& feature, qlonglong updated)	
• Suorittaa pyynnön, joka hakee kohteen *feature* kuvauksen.	
const QString& feature	Kohteen feature-ominaisuus, joka ei ole kuitenkaan sama kuin käyttäjälle näkyvä kohteen nimi.
qlonglong updated	Sisäänkirjautumisen yhteydessä saatu aikaleima, joka kertoo milloin ko. kohteen kuvauksen tietoja on päivitetty web-palveluun. Tätä aikaleimaa verrataan tietokannan aikaleimaan ja mikäli ne poikkeavat, haetaan tuore tieto verkosta.

Taulukko 17. QueryFeatures-metodin kuvaus.

void queryFeatures (QString feature, QList<GFilter> filters, int max)	
• Hakee kohteiden tiedot, joiden feature-ominaisuus on *feature* ja joiden ominaisuudet vastaavat suodattimien *filters* hakuehtoja. Vastauksen kohteiden lukumäärä on rajoitettu parametrilla *max*.	
QString feature	Kohteiden feature-ominaisuus, joka ei ole kuitenkaan sama kuin käyttäjälle näkyvä kohteen nimi.

QList<GFilter> filters	Lista, joka sisältää hakukriteerit. Yksi listan GFilter-olio sisältää yhden hakukriteerin.
int max	Määrittää vastauksessa palautettavien kohteiden maksimilukumäärän.

Taulukko 18. GetFeature-metodin kuvaus.

void getFeature (const QString& id)	
• Hakee yhden kohteen varsinaisen datan.	
const QString& id	Kohteen yksilöivä tunniste.

Taulukko 19. InsertFeature-metodin kuvaus.

void insertFeature (int folderId, GFeatureData* featureData)	
• Tallentaa uuden kohteen *featureData* kansioon, jonka tunniste on *folderId*.	
int folderId	Kansion yksilöivä tunniste.
GFeatureData featureData*	Objekti, joka sisältää kohteen varsinaisen datan.

Taulukko 20. UpdateFeature-metodin kuvaus.

void updateFeature (int folderId, GFeatureData* featureData)	
• Päivittää kohteen *featureData* tiedot kansioon, jonka tunniste on *folderId*.	
int folderId	Kansion yksilöivä tunniste.
GFeatureData featureData*	Objekti, joka sisältää kohteen päivitetyn datan.

Taulukko 21. DeleteFeature-metodin kuvaus.

void deleteFeature (int folderId, GFeatureData* featureData)	
• Poistaa kohteen *featureData* kansiosta, jonka tunniste on *folderId*.	
int folderId	Kansion yksilöivä tunniste.
GFeatureData featureData*	Objekti, joka sisältää kohteen varsinaisen datan.

Taulukko 22. InsertFeatureTypeToDB-metodin kuvaus.

void insertFeatureTypeToDB (GFeatureType* featureType)	
• Tallentaa kohdetyypin kuvauksen paikalliseen tietokantaan.	
GFeatureType featureType*	Objekti, joka sisältää kohdetyypin kuvauksen.

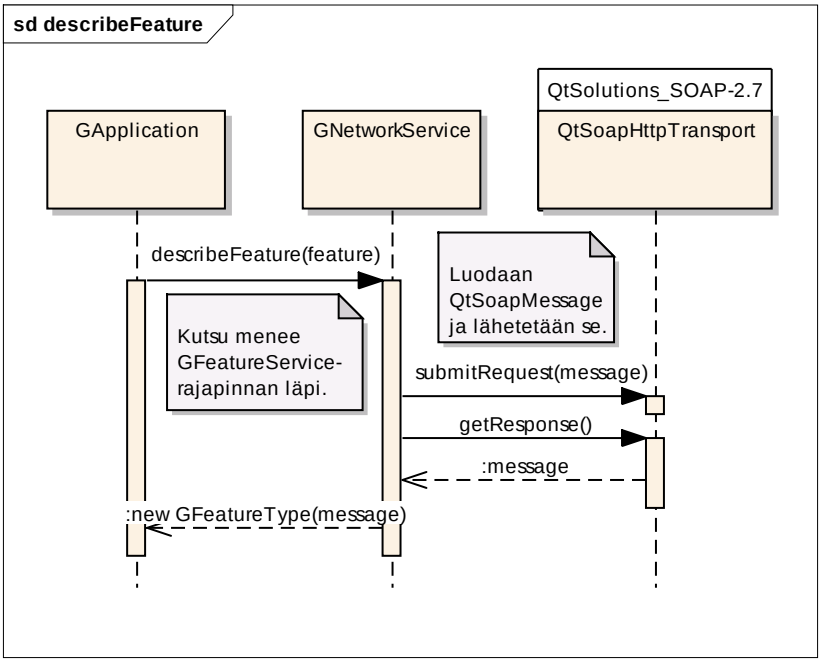
Sovelluksessa tätä rajapintaa toteuttaa Mobilenote-moduulin GServiceManager. Tietokannan käsittely on keskeinen osa tämän rajapinnan toteutusta, koska kohdetyyppien kuvaukset haetaan ensisijaisesti paikallisesta tietokannasta. Kohteiden varsinaisen datan käsittely sen sijaan tapahtuu aina verkon yli.

6.4 Tietolähteiden käsittely

Tämä luku esittelee, kuinka sovellus käsittelee palvelimella ja paikallisessa tietokannassa sijaitsevia tietolähteitä. Palvelimen tarjoamat palvelut ovat jo olemassa olevia toteutuksia yrityksen verkossa eikä niiden rakennetta tai toimintaa käsitellä.

6.4.1 XML Web Service

Sovellus käyttää kohdetietojen lähteenään Mobilenoten web-palvelua, jolta tietoja kysellään SOAP-protokollan välityksellä. Sekä lähetetty pyyntö että palvelun vastaus on tällöin XML-muotoista dataa. Kyselyt muodostetaan käyttäen valmista Qt Solutions Archive:n komponenttia nimeltään Qt SOAP. Komponentti tarjoaa valmiin toteutuksen SOAP-protokollan transaktioiden käsittelyyn, joka tekee XML-datan lukemisen ja kirjoittamisen helpoksi. Kuvan 13 sekvenssikaavio esittää, kuinka sovellus pyytää kohdetyypin kuvausta web-palvelulta.



Kuva 13. Kohdetyypin kuvauksen pyytäminen verkon yli.

Kun QtSoapMessage luodaan, se rakentaa automaattisesti oikeanlaisen XML-rakenteen pyyntöä varten. Palvelun osoite, johon pyyntö lähetetään, on asetettu QtSoapHttpTransport-olioon jo instantiointivaiheessa. Web-palvelussa suoritettava metodi määrätään funktiolla message.setMethod() ja pyynnön parametrit asetetaan kutsumalla message.setMethodArgument(). Pyyntö lähetetään kutsumalla transport-olion submitRequest()-metodia.

```
QtSoapMessage request;
request.setMethod("describeFeature", m_tns);
request.addMethodArgument("feature", "", feature);
```

Palvelulta pyydetään vastausta kutsumalla transport-olion getResponse()-metodia. Palvelu palauttaa pyynnön vastauksen SOAP-komponentille, joka palauttaa sen QtSoapMessage-oliona sovellukselle. Message-oliosta GFeatureTypen konstruktori osaa purkaa tiedot jäsenmuuttujiinsa.

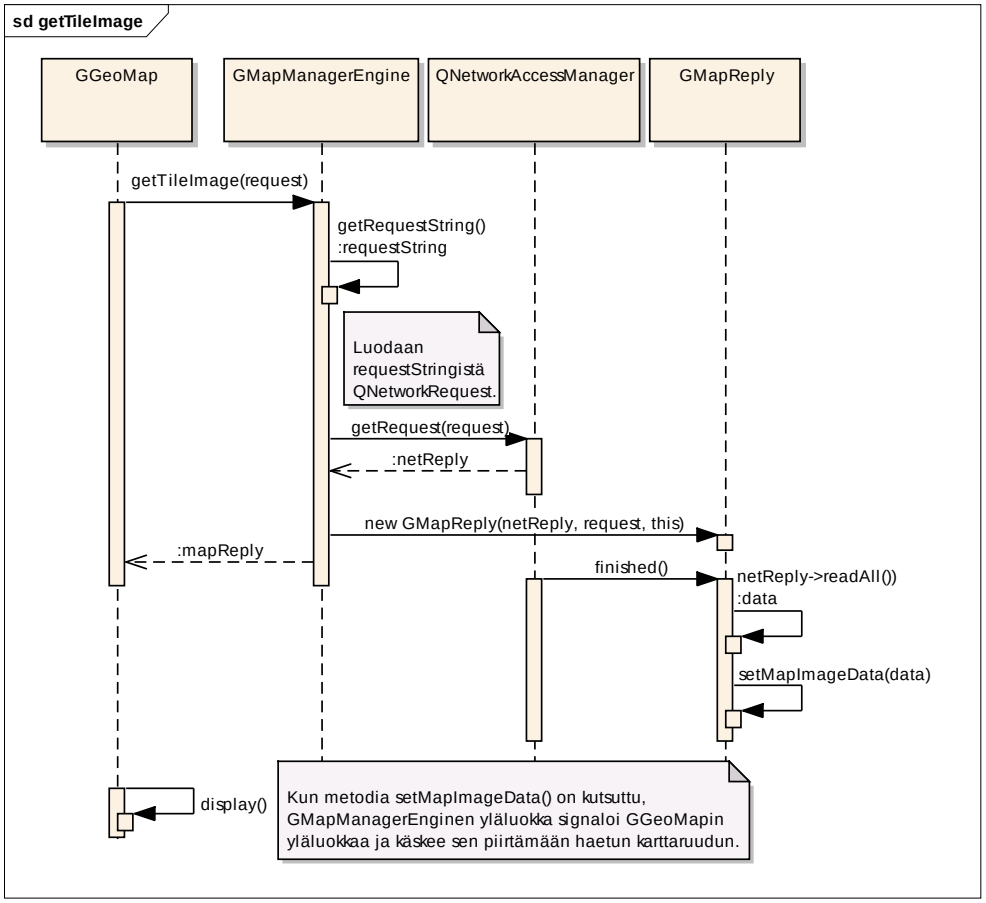
Samalla periaatteella toimii kaikki kommunikointi web-palvelun kanssa, oli kyse sitten kohteen tietojen päivittämisestä, kohteen poistamisesta tai uuden kohteen luomisesta.

6.4.2 WMS Map Service

Kartta-aineisto on palvelimella WMS-rajapinnan takana. Karttaruutuja voisi pyytää suoraan WMS-palvelusta, mutta sovellus pyytää niitä palvelimelle ra-

kennetulta Proxyltä, koska sen käyttö on verraten nopeampaa. Proxyn rajapinta poikkeaa standardista WMS-rajapinnasta, joten pyynnöt muodostetaan Proxyn määritysten mukaan.

Karttapyyntöjen muodostaminen tapahtuu Taustakartat-moduulissa. Sovellus kutsuu GMapManagerEnginen metodia getTileImage() aina, kun se haluaa ladata karttaruudun verkon yli. GMapManagerEngine osaa muodostaa ja lähettää oikeanlaisen pyynnön Proxylle, ja käsitellä sen vastauksen. Sovellus saa paluuarvona kuvan valmiina näytettäväksi. Kuvan 14 sekvenssikaavio kuvaa ohjelman kulun, kun karttaruutua pyydetään palvelimelta.



Kuva 14. Karttaruudun lataaminen palvelimelta.

GGeoMap huolehtii karttaruutujen pyytämisestä. Se pyytää ruutuja automaattisesti silloin, kun karttanäkymä siirtyy niiden rajojen yli, missä edellinen ruutu loppuu. Se pyytää uudet ruudut myös silloin, kun käyttäjä lähentää tai loitontaa karttaa.

GGeoMap välittää pyyntönsä parametrina sen ruudun tiedot, jonka se haluaa. GMapManagerEngine rakentaa näiden tietojen perusteella verkkopyyn-

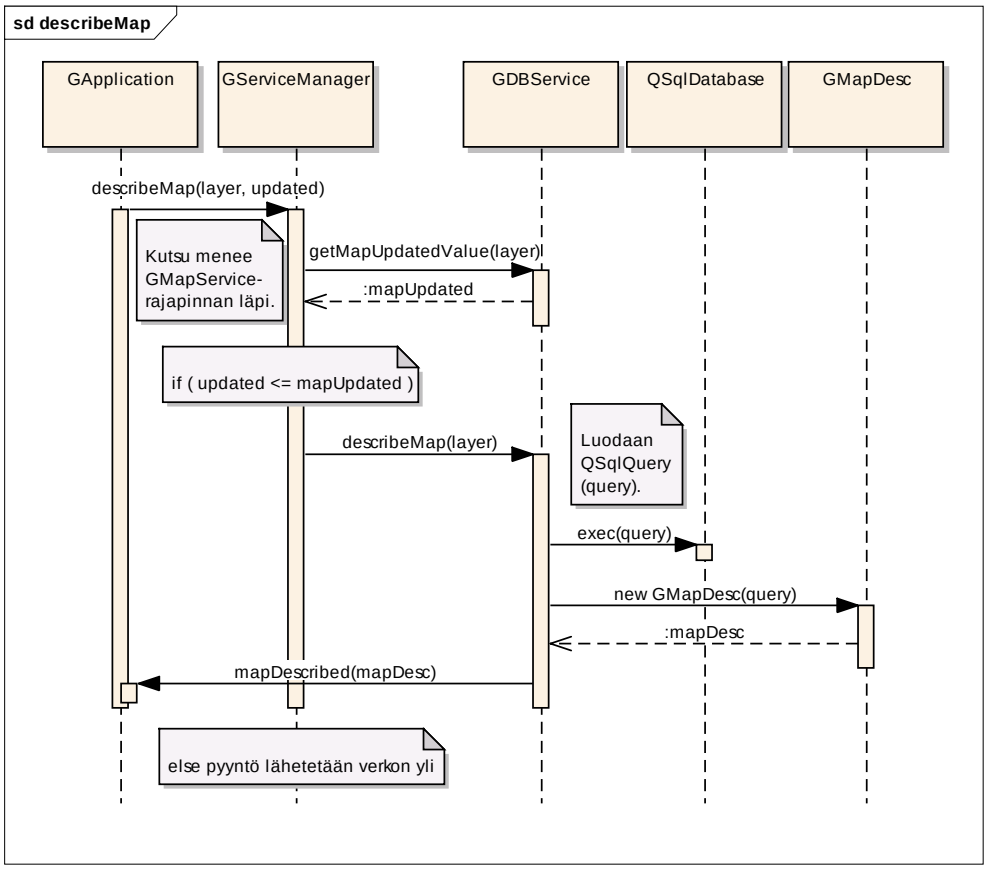
nön, joka lähetetään palvelimelle kutsumalla QNetworkAccessManagerin metodia getRequest(). Metodi palauttaa enginelle QNetworkReply-olion (netReply). GMapReply-olio luodaan netReplyn perusteella ja se palautetaan GGeoMapille. GGeoMap odottaa, kunnes pyyntö on suoritettu. Pyyntö on suoritettu silloin, kun QNetworkAccessManager lähettää GMapReplylle finished()-signaalin. Tällöin GMapReply asettaa GGeoMapille palautettuun olioon karttaruudun datan metodilla setMapImageData(). Kun data on asetettu, GGeoMap voi näyttää ruudun näytöllä.

6.4.3 SQLite-tietokanta

Sovellus tarjoaa alustavan tietojen käsittelyn SQLite-tietokannassa. Toteutettu sovellus käyttää tietokantaa sekä kohdetyyppien ja taustakarttojen kuvausten, että karttaruutujen tallentamiseen laitteen massamuistiin. Kohteiden varsinainen data haetaan aina verkon yli. Tämä luku esittelee tietokantojen käytön logiikan, mutta ei kantojen rakennetta.

Sovellus luo käynnistyksen yhteydessä itselleen kaksi tietokantaa. Toinen on tarkoitettu kohdetyyppien ja karttojen kuvauksia varten ja toinen karttaruutuja varten. Jos jokin tietokanta on jo olemassa, sitä ei korvata.

Tietokantoja käytetään silloin, kun tehdään tiettyjä pyyntöjä web-palveluun. Tällainen pyyntö on esimerkiksi describeMap(), joka suoritetaan silloin, kun halutaan hakea tietyn kartan kuvaus. Sovellus reagoi pyyntöön siten, että se hakee kuvauksen ensisijaisesti tietokannasta. Kuvan 15 sekvenssikaavio kuvaa describeMap()-pyynnön suorituksen kulun silloin, kun se kohdistuu tietokantaan.



Kuva 15. Kartan kuvauksen lataaminen tietokannasta.

Pyynnön describeMap() yhteydessä välitetään aikaleima, joka kertoo, milloin viimeksi kartan kuvaus on päivitetty web-palveluun. Aikaleima on saatu login()-pyynnön vastauksessa. Sovellus vertaa tätä aikaleimaa tietokantaan tallennettuun aikaleimaan ja suorittaa pyynnön tietokantaan, jos aikaleimat täsmäävät. Jos aikaleimat eivät täsmää, pyyntö suoritetaan verkon yli ja tietokanta päivitetään ajan tasalle.

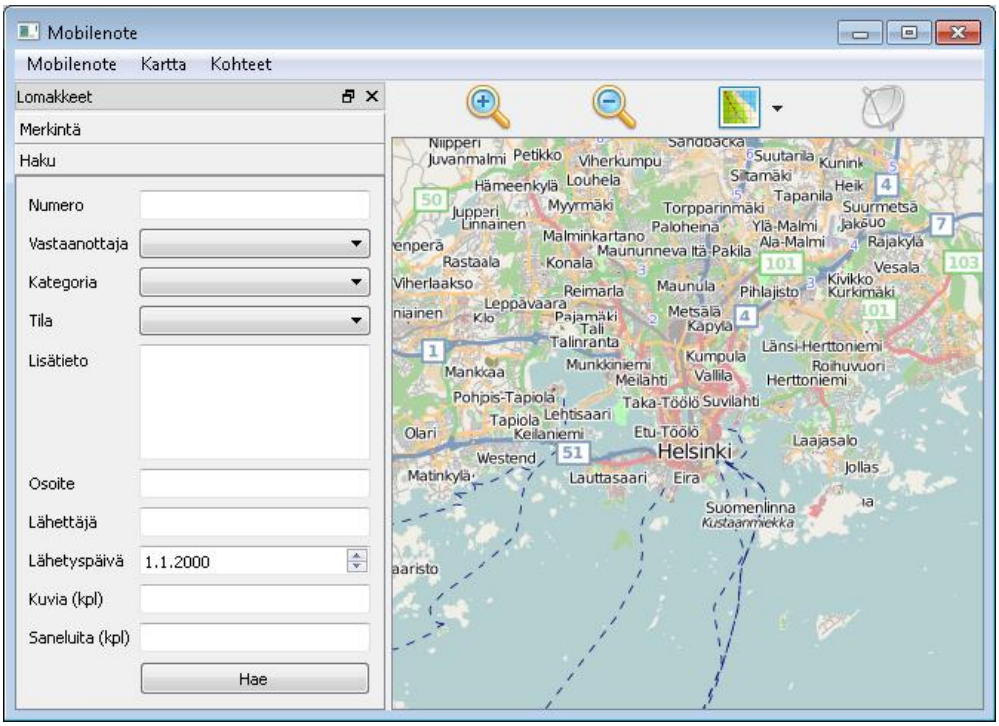
Tulevaisuudessa tietokantaa on tarkoitus hyödyntää myös silloin, kun sovellusta käytetään ns. offline-tilassa. Tällöin sovellus ei ole lainkaan yhteydessä web-palveluihin, vaan kirjautuminen suoritetaan paikallisesti ja kohteiden varsinaiset tiedot haetaan paikallisesta (laitteen sisäisestä) tietokannasta.

6.5 Käyttöliittymä

Sovelluksen käyttöliittymän perusperiaate on jokaisella kohdealustalla sama. Kartta, lomakkeet ja hakulista löytyvät omasta raamistaan. Näyttöjen asettelusta (pysty / vaaka) ja resoluutiosta johtuen eri alustoille on kuitenkin käytetty erilaisia käyttöliittymäkomponentteja.

Esimerkiksi sovelluksen Windows-käännöksessä käytetään käyttöliittymäkomponentteja, jotka asettelevat kartan, lomakkeet ja hakulistan esiin samalla kertaa (ks. 5.5.1). Mobiililaitteissa näytön koko on rajatumpi, joten sama ajattelutapa ei tuota selkeää lopputulosta. Windows Mobilessa kartta, lomakkeet ja hakulista ovat kaikki omina välilehtinään, jolloin yksi niistä voi olla kerrallaan näkyvissä. Sovellus tutkiikin, missä käyttöjärjestelmässä toimitaan ja näyttää käyttöliittymän sen perusteella. Tässä luvussa esitellään vain Windows-käyttöliittymä. Windows Mobilen käyttöliittymä tulee tutuksi luvussa 6.6.

Windows käyttöliittymässä näytetään kaikki komponentit samaan aikaan. Ainoastaan lomakkeet on ladottu päällekkäin. Sovelluksen toiminnallisuus ja toiminnot ovat Windows Mobilen kanssa samat.



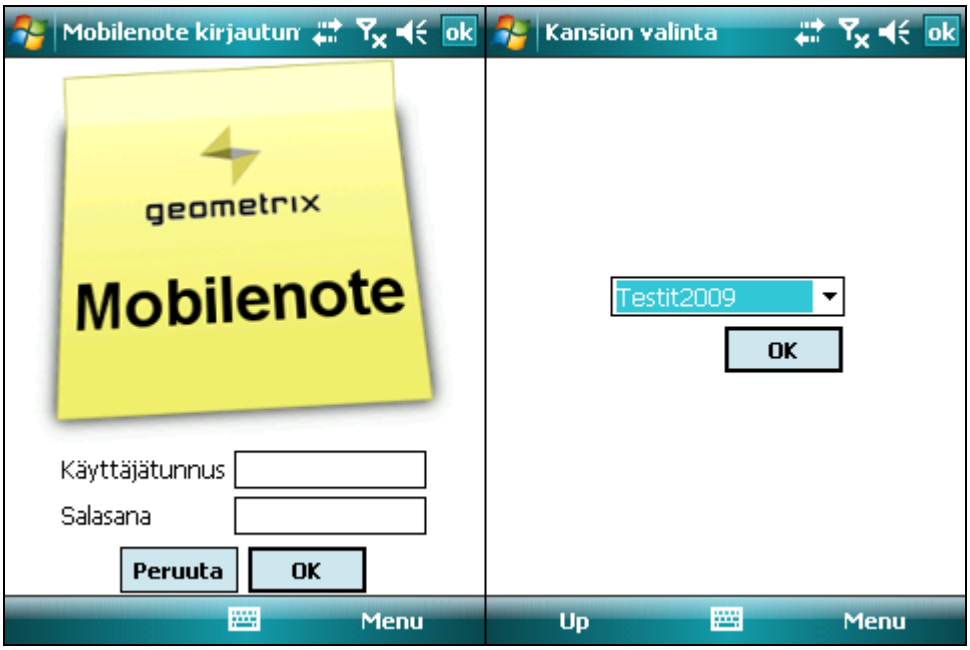
Kuva 16. Windowsin käyttöliittymä.

6.6 Sovelluksen toiminta

Tässä luvussa esitellään sovelluksen toimintaa. Suurennuslasin alla on ohjelman toiminnan logiikka tietyissä tilanteissa. Tilanteet käsittelevät istunnon ylläpitoa ja karttojen ja kohteiden käsittelyä. Esittelyn apuna on käytetty käyttöliittymäkuvia ja sekvenssikaavioita. Käyttöliittymän kuvina on käytetty Windows Mobile 6 -alustaa. Luku etenee loogisesti sisäänkirjautumisesta ohjelman käyttöön ja lopulta uloskirjautumiseen.

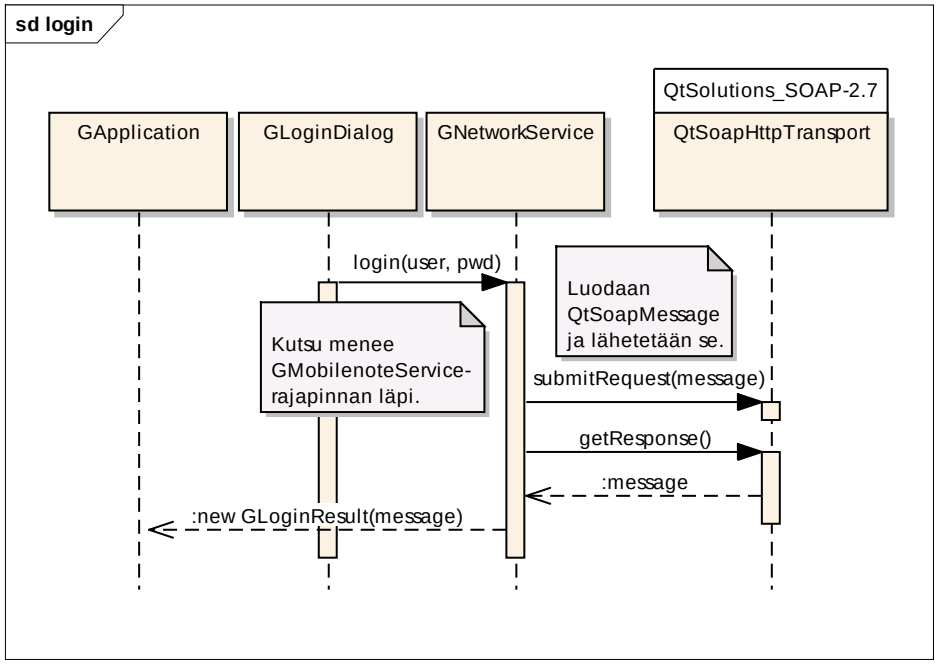
6.6.1 Sisäänkirjautuminen

Jotta Mobilenoten web-palveluita voi käyttää, on oltava voimassa oleva istunto. Tämä luodaan heti sovelluksen käynnistyttyä kysymällä käyttäjältä käyttäjätunnus ja salasana.



Kuva 17. Sisäänkirjautumisen ja kansion valinnan näkymä.

Kun tunnukset on syötetty ja klikataan OK, sovellus lähettää pyynnön palvelimelle. Jos kirjautuminen onnistui, näytetään ’Kansion valinta’ -ikkuna. Seuraava sekvenssikaavio selventää, mitä ohjelman sisällä tapahtuu sisäänkirjautumisen yhteydessä.



Kuva 18. Sisäänkirjautumisen sekvenssikaavio.

Erityistä tapahtumien kulussa on se, että GNetworkService ei vastaa kutsujalleen, eli luokalle GLoginDialog. Sen sijaan se vastaa sovelluksen ytimelle (GApplication). Jos sisäänkirjautuminen epäonnistuu, GMobilenoteService palauttaa ytimelle tyhjän osoittimen. Ydin reagoi viestiin joko näyttämällä virheilmoituksen tai ’Kansion valinta’ -ikkunan (GFolderSelectionDialog).

Ennen sovelluksen pääikkunan näyttämistä sovellus listaa sisäänkirjautumisen vastauksena tulleet Mobilenote-kansiot ’Kansion valinta’ -ikkunaan. Kansion valinnasta seuraava pyyntö toimii täysin samalla logiikalla kuin sisäänkirjautuminen, ja pyyntö tehdään aina verkon yli. Erona on, että pyynnön lähettäjä on GFolderSelectionDialog, pyyntömetodi on nimeltään openFolder() ja pyynnön vastaus on tyyppiä GFolder. Kansion valinnan jälkeen ydin alkaa valmistella sovelluksen pääikkunan käynnistämistä.

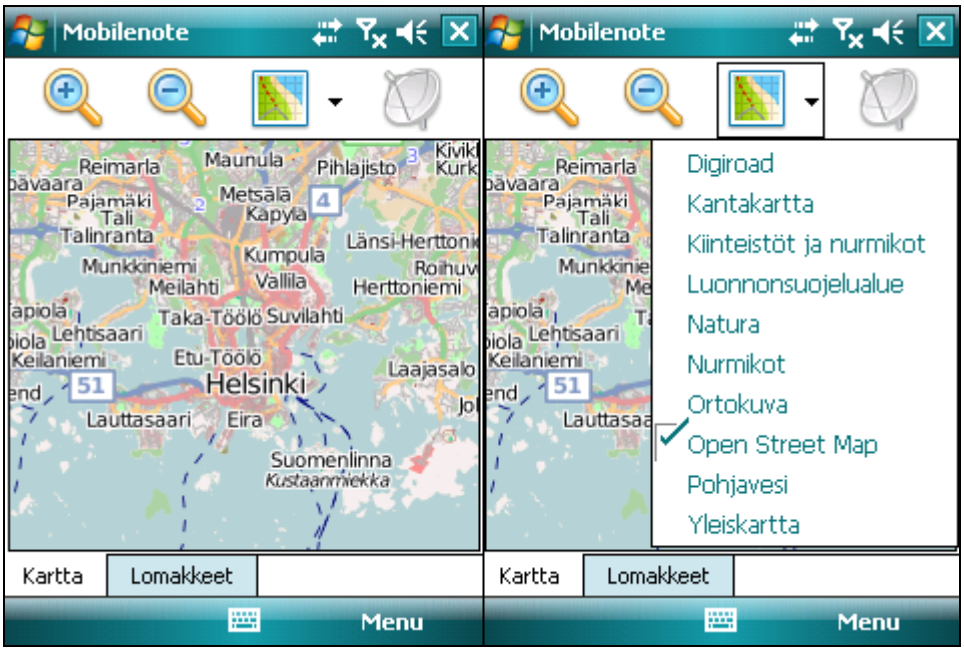
6.6.2 Karttojen ja kohdetyyppien kuvaukset

Ennen pääikkunan näyttämistä sovellus hakee muistiinsa karttojen ja kohdetyyppien kuvaukset. Karttojen kuvauksista sovellus luo käyttöliittymään karttalistan, josta käyttäjä voi valita käytettävän kartta-aineiston. Kohdetyyppien kuvauksista muodostetaan kullekin kohdetyypille ominainen muokkaus- ja hakulomake. Toteutettu sovellus tukee vain Merkintä-kohdetyyppiä, mutta mekanismi usean kohdetyypin tukea varten on jatkokehitysprojekteja ajatellen tehty jo valmiiksi.

Sovellus käyttää kuvausten hakemiseen pyyntöjä describeMap() ja describeFeature(). Tapahtumien kulku kummassakin tapauksessa on esitelty luvussa 6.4. Luvussa 6.4.1 esitellään kohdetyypin kuvauksen haku verkon yli ja luvussa 6.4.3 kartan kuvauksen haku tietokannasta. Pyynnöt ovat logiikaltaan identtisiä, joten tietokanta- ja verkkopyyntöjen sekvenssikaavioita voi soveltaa ristiin. Ainoastaan pyyntömetodien nimet, niiden parametrit ja pyynnön paluuarvot eroavat.

6.6.3 Sovelluksen pääikkuna

Kun sovellus on ladannut kuvaustiedot muistiinsa, se käynnistää sovelluksen pääikkunan. Seuraavaksi on kuva pääikkunan aloitusnäkymästä ja karttalistasta.



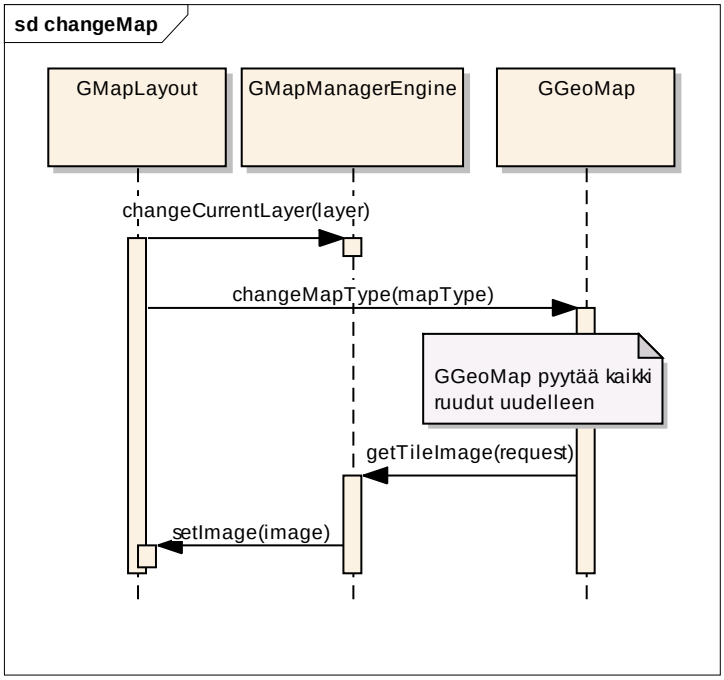
Kuva 19. Sovelluksen aloitusnäkymä ja karttalista.

Vasemmalla kuvassa on sovelluksen käyttöliittymä. Se sisältää kartan käsittelyyn liittyvät toimintopainikkeet (ylälaita), jotka käsittävät lähentämisen, loitontamisen ja karttalistan. Karttalista on luotu ajon aikana sovelluksessa olevien karttojen kuvausten perusteella. Alalaidassa ovat välilehdet, joiden avulla näkymä voidaan vaihtaa karttanäkymästä lomakenäkymään. Karttaa vieritetään joko laitteen nuolinäppäimillä tai raahaamalla karttaa kosketusnäytöllä.

Kuvassa oikealla on klikattu karttalistaikonia. Se avaa valikon, josta voi valita yhden kartan kerrallaan näytettäväksi. Menu-painike sisältää sovelluksen päävalikon.

6.6.4 Kartan vaihtaminen

Kun käyttäjä vaihtaa kartan, Kartta-moduuli lähettää uuden karttatason nimen Taustakartat-moduulille ja ilmoittaa karttakomponentille (GGeoMap), että kartta on vaihdettu. GGeoMap reagoi ja pyytää kaikki ruudut uudelleen uusilla asetuksilla. Kuvassa 20 oleva sekvenssikaavio selittää tapahtumien kulkua.



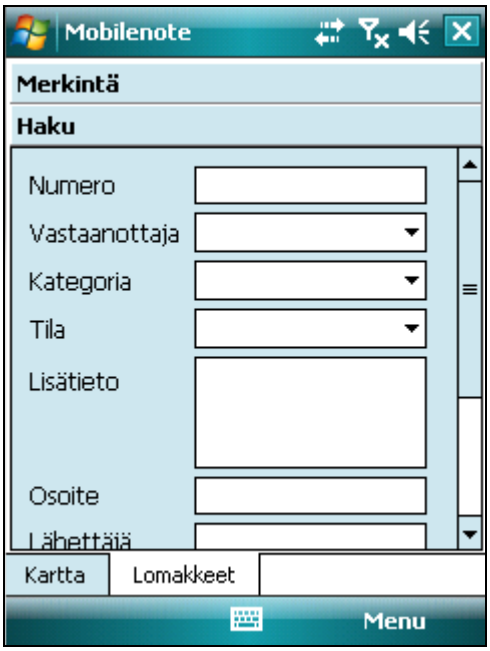
Kuva 20. Kartan vaihtamisen sekvenssikaavio.

6.6.5 Kartan muut toiminnot

Kartan muut toiminnot, kuten lähentäminen, loitontaminen ja vieritys ovat osana Maps API:n karttakomponenttia. Niiden käyttäminen tapahtuu yksinkertaisilla funktiokutsuilla, kuten setZoomLevel() ja pan(). Näihin toimintoihin liittyvää toiminnan logiikkaa ei käsitellä. Käyttöliittymän ylälaidassa näkyy myös satelliitin kuva. Se on tarkoitettu GPS:n käyttöä varten, jota ei tämän projektin yhteydessä toteutettu.

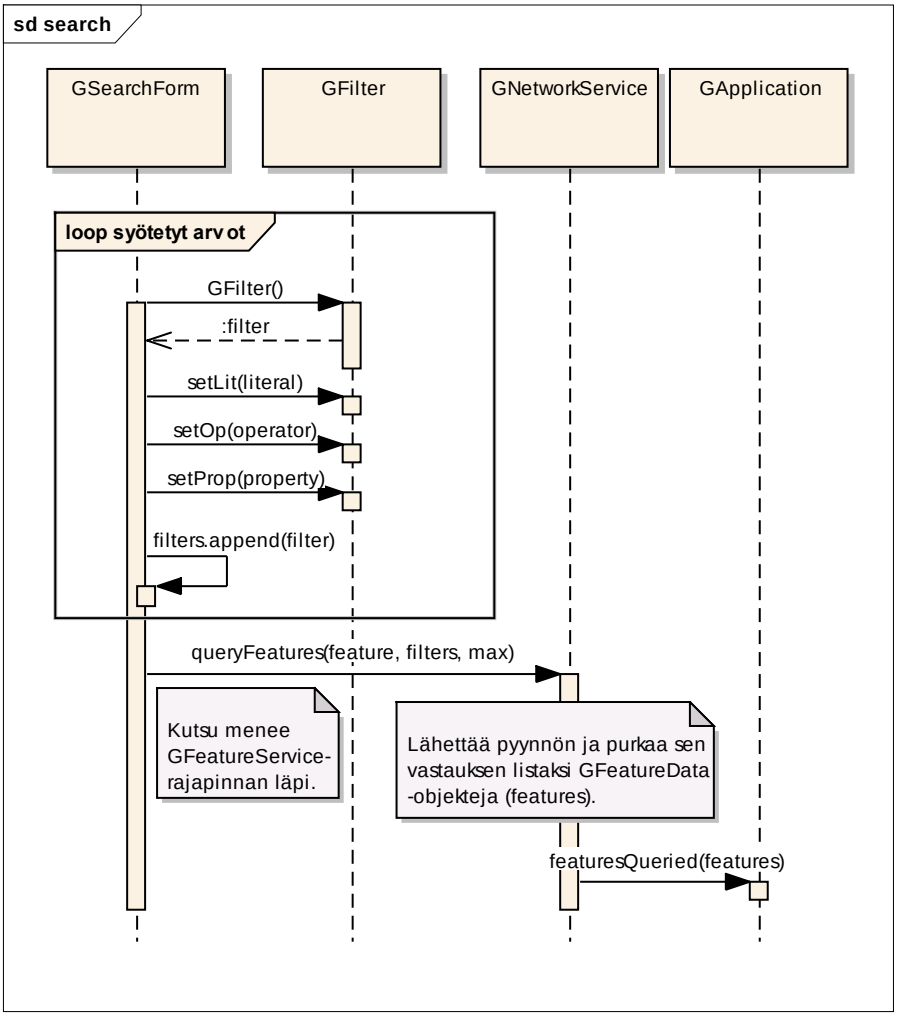
6.6.6 *Merkintöjen haku kartalle*

Merkintöjen haku kartalle tapahtuu Lomakkeet-välilehdeltä. Välilehti sisältää kaksi sivua, joista toinen on tarkoitettu merkintöjen editointiin ja toinen hakukriteerien syöttämiseen. Seuraava kuva on Lomakkeet välilehden näkymästä.



Kuva 21. Lomakkeet-välilehden näkymä.

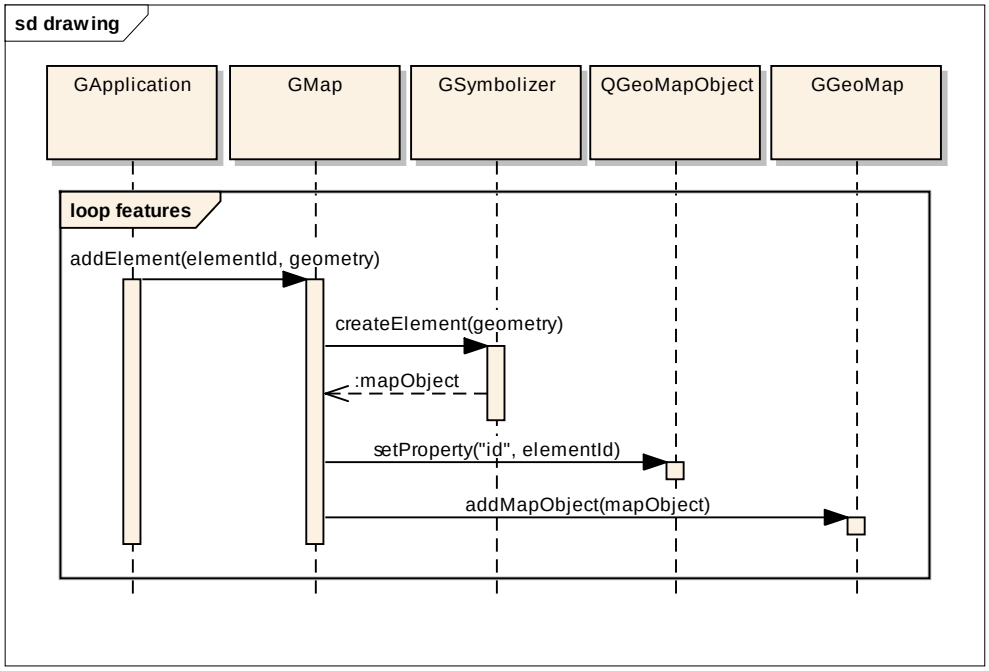
Hakusivu sisältää merkinnän ominaisuustietoja, joita voi käyttää hakukriteereinä kohteita haettaessa. Haku käynnistetään painikkeesta Hae, joka jää kuvassa piiloon. Se löytyy lomakkeen alalaidasta, kun sivua rullaa alas vierityspalkista. Kuvan 22 sekvenssikaavio selittää ohjelman sisäistä toimintaa, kun käyttäjä tekee kohteen haun.



Kuva 22. Kohteiden haun sekvenssikaavio.

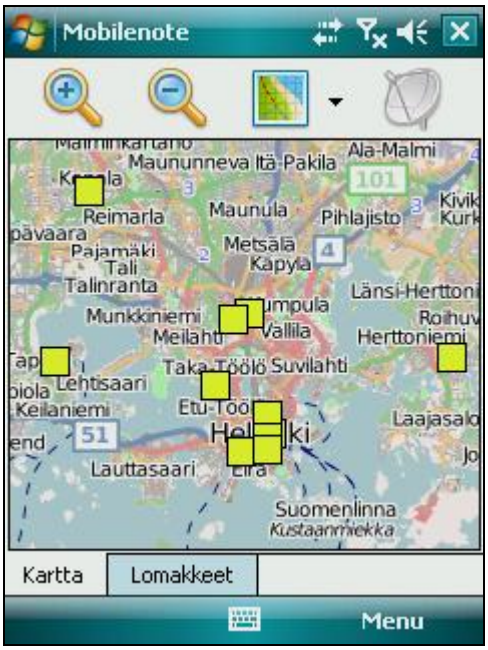
Kun käyttäjä klikkaa Hae-painiketta, sovellus tekee jokaisesta syötetystä hakukriteeristä GFilter-olion, jotka se kerää listaan. Kohdetyypin tunnus (feature), hakukriteerilista (filters) ja vastauksen kohteiden maksimilukumäärä (max) lähetetään pyyntöjä käsittelevälle GFeatureManagerille. Pyynnön muodostaminen ja vastauksen käsittely on esitelty luvun 6.4.1 sekvenssikaaviossa, joten tähän kaavioon sitä ei ole merkitty. Vastaus puretaan listaksi kohdeobjekteja, jotka palautetaan sovelluksen ytimelle.

Sovelluksen ydin käy kohdelistan läpi ja käskee Kartta-komponenttia piirtämään ne kartalle. Kuvan 23 sekvenssikaaviosta käy ilmi, miten kohteiden piirto kartalle tapahtuu.



Kuva 23. Sekvenssikaavio kohteiden piirtämisestä kartalle.

Sovelluksen ydin antaa jokaisen listan kohteen id-tunnisteen ja geometrian Kartta-moduulille. GMap luo GSymbolizerin avulla geometriasta karttaelementtejä. Jokaiseen karttaelementtiin liitetään kohteen id ja ne lisätään kartalle. Kun kaikki kohteet on piirretty, sovellus aktivoi karttanäkymän. Kuvassa 24 on näkymä, kun haku on suoritettu.



Kuva 24. Näkymä, kun kohteiden haku kartalle on suoritettu.

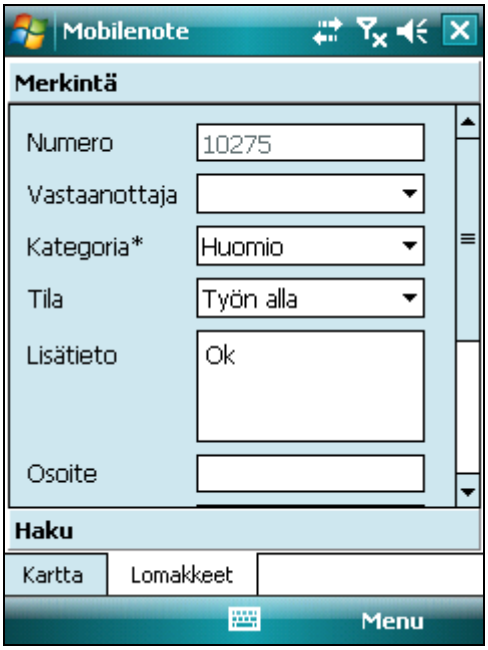
Näkymässä näkyvät kohteet ovat osoitettavia kohteita. Kohdetta voi osoittaa koskettamalla sitä kosketusnäytöllä. Kun kohdetta on osoitettu, muokkauslomake aktivoituu ja kohteen tiedot näytetään siinä.

6.6.7 Kohteen tietojen muokkaaminen

Kohteen tietojen muokkaaminen käynnistyy, kun kohde osoitetaan kartalta. Ensin sovellus hakee kyseisen kohteen tiedot web-palvelusta. Kun käyttäjä osoittaa kohteen, GGeoMap signaloi kohteen id-tunnisteen GApplicationille. GApplication suorittaa kohteen haun id-tiedon perusteella.

Yhden kohteen tietojen haun sovelluslogiikka on sama kuin usean kohteen haussa, mutta pyyntömetodi on tällöin getFeature() ja paluuarvona vain yksi GFeatureData-olio. Toisin kuin usean kohteen haussa, yhden kohteen haussa pyynnön vastauksesta puretaan kohteen ominaisuuksien tiedot. Kohteen id-tunnistetta tai geometriaa ei käsitellä.

Kun sovellus on hakenut kohteen tiedot, niitä voi muokata siihen tarkoitetulla lomakkeella. Kohdetyyppien kuvauksen mukana on tullut myös tietoa kohteen ominaisuustietojen oikeuksista. Jos kohdetyyppikuvauksessa määrätään, että Numero-kentän arvoa ei voi muuttaa, niin sovellus näyttää tekstikentän harmaannutettuna. Tällöin käyttäjä ei voi muuttaa kentän arvoa. Seuraava kuva on näkymä muokkauslomakkeesta.

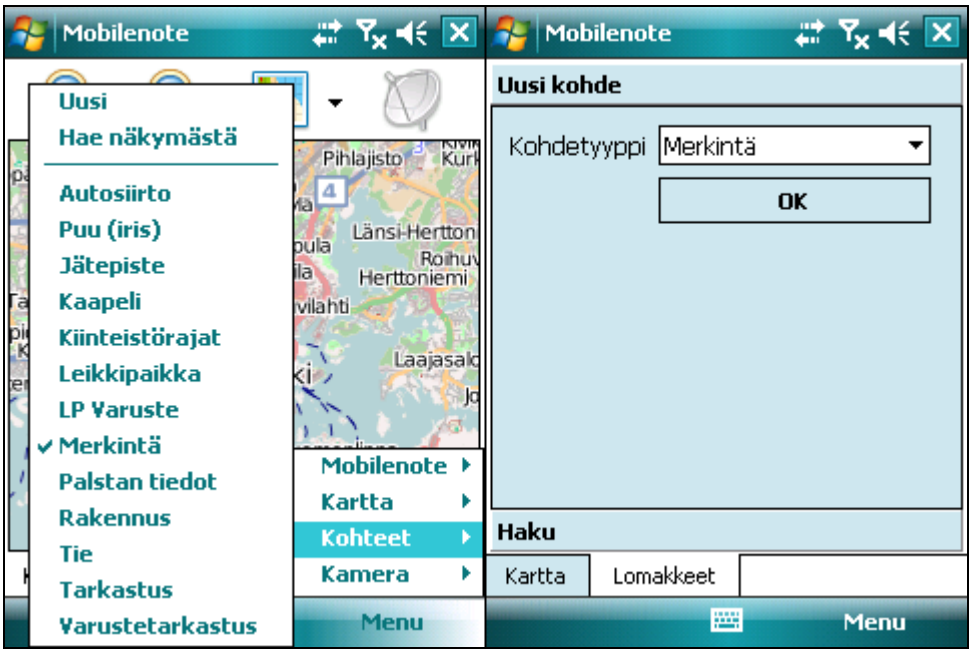


Kuva 25. Kohteen tietojen näyttäminen / muokkaaminen.

Muutetut tiedot tallennetaan painikkeesta Tallenna, joka jää kuvassa piiloon. Se löytyy lomakkeen alalaidasta, kun sivua rullaa alas vierityspalkista. Tallennus kutsuu GFeatureServicen updateFeature()-metodia, joka suorittaa pyynnön web-palveluun.

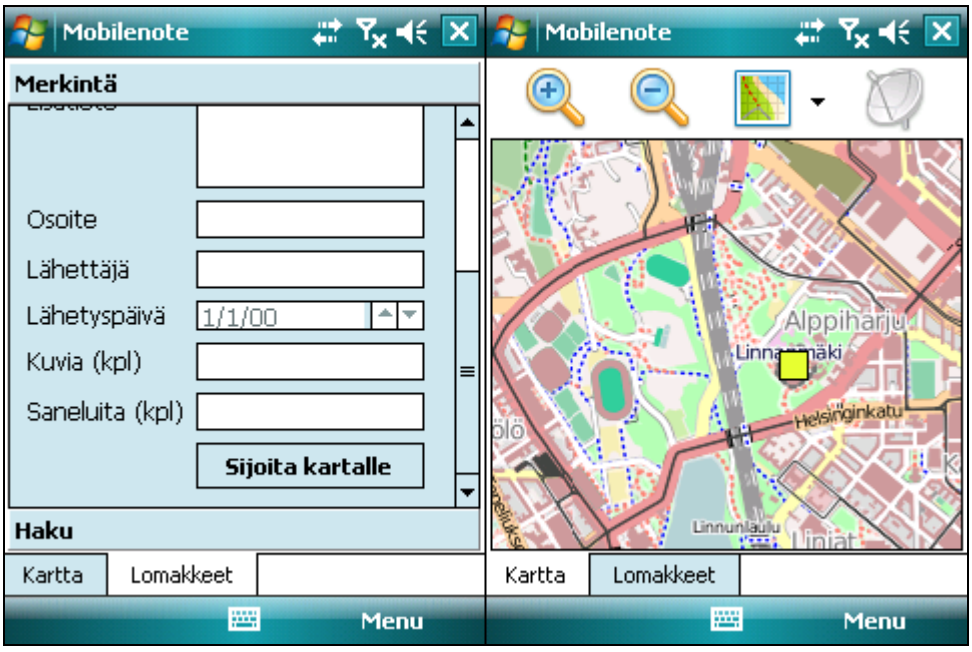
6.6.8 Uuden karttakohteen luominen

Uuden karttakohteen luominen aloitetaan valitsemalla päävalikosta (Menu) Kohteet / Uusi. Sovellus aktivoi Lomakkeet-välilehden, josta ensin valitaan uuden kohteen tyyppi. Toteutettu sovellus tukee ainoastaan Merkintä-tyyppistä kohdetta. Kuvassa 26 on kuva käyttöliittymästä.



Kuva 26. Uuden karttakohteen luominen.

Kun käyttäjä klikkaa OK, lomake (GEditForm) luo lennossa itselleen valitun kohdetyypin määräämät tietokentät. Kun uuden kohteen tiedot on syötetty, se pitää sijoittaa kartalle. Tämä tapahtuu klikkaamalla lomakkeen alalaidasta löytyvää painiketta. Painikkeen painaminen aktivoi Kartta-välilehden ja kohteen voi sijoittaa kartalle koskettamalla kosketusnäyttöä.



Kuva 27. Kohteen sijoittaminen kartalle.

Kun kohde on sijoitettu kartalle, se voidaan tallentaa siirtymällä Lomakkeet-välilehdelle ja painamalla Tallenna-painiketta. Sovellus suorittaa GFeatureServicen metodin insertFeature(), joka tallentaa uuden kohteen web-palveluun. Metodin toiminnan logiikka on sama, kuin missä tahansa vain verkon yli suoritettavassa pyynnössä.

6.6.9 Kartan tyhjentäminen

Kartan voi tyhjentää sovelluksen päävalikosta löytyvästä Kartta-valikosta. Sovellus tällöin poistaa kaikki kohteet kartalta, mutta jättää kartan näkyviin. Kohteiden poistaminen kartalta on yksinkertainen toimenpide, jonka Maps API:n karttakomponentti suorittaa clearMapObjects()-metodin kutsulla.

6.6.10 Sovelluksen sulkeminen

Sovellus suljetaan päävalikon Mobilenote-valikosta. Ennen prosessin tuhoamista sovellus suorittaa uloskirjautumisen. Uloskirjautuminen suoritetaan kutsumalla GMobilenoteServicen logout()-metodia, joka suoritetaan aina verkon yli.

7 YHTEENVETO

Työn tarkoituksena oli pilotoida Qt-teknologiaa mobiili- ja paikkatietosovelluksessa. Qt:n etuja Javaan verrattuna lähdettiin tutkimaan toteuttamalla Mobilenote-paikkatietojärjestelmän asiakassovellus Qt-kehitysympäristössä.

Työssä perehdyttiin ensin Qt-teknologian erityispiirteisiin ja siihen, kuinka kehitysympäristön konfiguraatio saadaan halutulle tasolle. Nokian Qt SDK on sellaisenaan valmis paketti Maemo- ja Symbian-kehitykseen, mutta Windows- ja Windows Mobile 6 -alustojen konfigurointi vaatii noviisilta yllättävän paljon ponnisteluja. Tästä johtuen työssä esiteltiin yksityiskohtaisesti Qt-kirjastojen kääntäminen Windows- ja Windows Mobile 6 -kohdealustoille.

Mobilenote-paikkatietojärjestelmän periaate käytiin läpi yleisellä tasolla. Projektin määrittelyssä esitettiin käyttötapaukset, joiden perusteella projektissa tuotettua sovellusta ryhdyttiin tekemään. Työssä tuotettu sovellus onnistui täyttämään määrittelyn asettamat vaatimukset. Vaatimuksina olivat mm. arkkitehtuurin modulaarisuus, onnistunut kommunikointi web-palveluiden kanssa ja paikallisen tietokannan hyödyntäminen. Modulaarisuus saavutettiin luomalla dynaamisia ohjelmakirjastoja sovelluksen käyttöön. Kommunikointi web-palvelun kanssa toteutettiin hyödyntämällä Qt Solutions Archiven komponenttia Qt SOAP, joka on työkalu SOAP-protokollan mukaisen XML-datan käsittelyyn. Paikallinen tietokanta toteutettiin Qt:n työkaluilla. Tietokannan tyypiksi valittiin SQLite, jota Qt osaa käsitellä SQLite-liitännäisen avulla.

Qt osoittautui käteväksi työkaluksi, kun halutaan operoida usealla kohdealustalla. Kun konfiguraatio on kunnossa, sama ohjelma kääntyy eri alustoille vaivatta. Konfigurointi voisi tosin olla suoraviivaisempaa. Lisäksi Qt:lle ominaiset erityispiirteet, kuten signaalit ja slotit sekä muistinhallinta, tuovat mielekkyyttä perinteiseen C++-ohjelmointiin.

Paikkatietosovelluksen ohjelmointiin jo pelkästään Qt:n komponentit tarjoavat riittävästi työkaluja. Uusi Qt Mobility tuo kuitenkin mukanaan Maps API:n, joka helpottaa paikkatiedon käsittelyä. Ohjelmointirajapinta sisältää luokkia, jotka käsittelevät koordinaatteja, karttakohteita ja taustakarttoja. Rajapinnan toimintaa oli osittain hankala omaksua ja se tuntui olevan liikaa sidottu yleisesti käytettyyn WGS84-koordinaatistoon. Rajapinnan komponentit saatiin kuitenkin toimimaan halutulla tavalla ja sitä käytettiin toteutuksessa.

VIITELUETTELO

[1] Qt Development Frameworks. *Qt* [verkkosivu], päivitetty 10.10.2010 [viitattu 13.11.2010]. Saatavissa: http://qt.nokia.com/.

[2] Qt Development Frameworks. *Qt Licensing — Qt* [verkkosivu], päivitetty 8.10.2010 [viitattu 13.11.2010]. Saatavissa: http://qt.nokia.com/products/licensing.

[3] Free Software Foundation (FSF). *GNU Lesser General Public License v2.1 - GNU Project* [verkkosivu], päivitetty 8.9.2010 [viitattu 13.11.2010]. Saatavissa: http://www.gnu.org/licenses/lgpl-2.1.html.

[4] Free Software Foundation (FSF). *GNU General Public License v3.0 - GNU Project* [verkkosivu], päivitetty 9.10.2010 [viitattu 13.11.2010]. Saatavissa: http://www.gnu.org/licenses/gpl.html.

[5] Qt Reference Documentation. *Qt 4.7.0: QObject Class Reference* [verkkosivu], päivitetty 24.11.2010 [viitattu 24.11.2010]. Saatavissa: http://doc.trolltech.com/4.7/qobject.html.

[6] Qt Reference Documentation. *Qt 4.7.0: Signals & Slots* [verkkosivu], päivitetty 22.9.2010 [viitattu 24.11.2010]. Saatavissa: http://docs.huihoo.com/qt/4.7/signalsandslots.html.

[7] Qt Reference Documentation. *Qt 4.7.0: Windows CE - Using shadow builds* [verkkosivu], päivitetty 27.11.2010 [viitattu 27.11.2010]. Saatavissa: http://doc.qt.nokia.com/4.7/shadow-builds-wince.html.

[8] Qt Reference Documentation. *Qt 4.7: Installing Qt for Windows* [verkkosivu], päivitetty 27.11.2010 [viitattu 27.11.2010]. Saatavissa: http://doc.qt.nokia.com/4.7-snapshot/install-win.html.

[9] Qt Mobility Project Reference Documentation. *Qt Mobility 1.1: Installation Guide* [verkkosivu], päivitetty 8.12.2010 [viitattu 8.12.2010]. Saatavissa: http://doc.qt.nokia.com/qtmobility-1.1.0/installation.html.

[10] Geometrix Oy. *Mobilenote - Toiminnallinen kuvaus* [sähköinen dokumentti], päivitetty 3.11.2008 [viitattu 14.2.2011].

[11] Qt Reference Documentation. *Qt 4.7.1: QNetworkDiskCache Class Reference* [verkkosivu], päivitetty 15.2.2011 [viitattu 15.2.2011]. Saatavissa: http://doc.qt.nokia.com/4.7/qnetworkdiskcache.html.

[12] Advanced Wireless Map. *Raster Map* [verkkosivu], päivitetty 24.4.2010 [viitattu 16.2.2011]. Saatavissa: http://www.gsm-modem.de/raster-map.html.

[13] Uikkanen, Eino. *Suomalaiset koordinaatistot* [verkkosivu], päivitetty 17.12.2010 [viitattu 17.2.2011]. Saatavissa: http://www.kolumbus.fi/eino.uikkanen/geodocs/kkjgps.htm.

[14] Qt Development Frameworks. *New Qt APIs - Qt Mobility* [verkkosivu], päivitetty 8.12.2010 [viitattu 19.2.2011]. Saatavissa: http://qt.nokia.com/products/qt-addons/mobility.

[15] Qt Reference Documentation. *Qt Mobility 1.1* [verkkosivu], päivitetty 19.2.2011 [viitattu 19.2.2011]. Saatavissa: http://doc.qt.nokia.com/qtmobility-1.1.0/index.html#platform-compatibility.

[16] Qt Development Frameworks. *Qt Solutions Archive — Qt* [verkkosivu], päivitetty 9.11.2010 [viitattu 19.2.2011]. Saatavissa: http://qt.nokia.com/products/qt-addons/solutions-archive.

[17] Aaron Toponce. *BSD License Explained In Layman Terms* [verkkosivu], päivitetty 18.2.2011 [viitattu 19.2.2011]. Saatavissa: http://pthree.org/2007/08/08/bsd-license-explained-in-layman-terms/.

Kehitysympäristön tarkka kuvaus

Alla olevan taulukkoon on listattu kehityksen kannalta kriittisimmät kehitysympäristön osat.

Huom!

Microsoft Visual Studio 2010 ei enää tue Windows Mobile 6 -ympäristöä, vaan tuettu versio on ainoastaan Windows Mobile 7. Visual Studio ei ole välttämätön osa kehitysympäristöä kun halutaan tuottaa Windows Mobile-sovelluksia, mutta silloin tietyt Microsoftin komponentit pitää asentaa erikseen. Tässä työssä ei oteta kantaa näihin komponentteihin tai niiden asennukseen vaan oletetaan, että Visual Studio 2005 tai 2008 on saatavilla.

Sovellus	**Kuvaus**
Microsoft Visual Studio 2005 (8.0) tai 2008 (9.0)	Kehitin C-kielisten ja .NET-pohjaisten sovellusten luomiseen. Sisältää osia, joita Windows Mobile 6 -kehittäminen vaatii.
Nokia Qt SDK 1.0.1	Sisältää koodieditorin (Qt Creator 2.0.1) ja valmiiksi käännetyt Qt- ja Qt Mobility-kirjastot Symbian- ja Maemo-alustoille (Qt 4.6.2 ja Qt Mobility 1.0.2). Tarjoaa lisäksi simulaattorin, joka simuloi ohjelman ajamista mobiililaitteella.
.NET Compact Framework	Tarvitaan Windows Mobilen emulointiin.
Active Sync 4.5 (WinXP) Windows Device Center (Win7)	Tarvitaan Windows Mobile -emulaattorin, tai Windows Mobile -pohjaisen laitteen ja Windowsin väliseen tiedonsiirtoon.
Windows Mobile SDK 6 Refresh	Kehitysympäristö Windows Mobile 6 -sovellusten tuottamiseen.
ActivePerl	Tarvitaan, kun käännetään Qt-kirjastot Windows- ja Windows Mobile 6 -ympäristöön.
Debugging tools for Windows	Tarvitaan Windows-käännöksen debuggaamiseen.

Nokia Qt SDK:n mukana tuleva vanhempi Qt-versio 4.6.2 päivitettiin työn tekemisen aikana uuteen versioon 4.7.0 ja Qt Mobility versioon 1.1.0.