Sami Santala

KUVA- JA PAIKKATIEDON LÄHETTÄMINEN J2ME-MOBIILISOVELLUKSELLA

Opinnäytetyö
KESKI-POHJANMAAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Joulukuu 2009

KESKI-POHJANMAAN AMMATTIKORKEAKOULU
MELLERSTA ÖSTERBOTTENS YRKESHÖGSKOLA

TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Yksikkö Tekniikan ja liiketalouden yksikkö, Kokkola	**Aika** 13.12.2009	**Tekijä/tekijät** Sami Santala
Koulutusohjelma Tietotekniikan koulutusohjelma		
Työn nimi Kuva- ja paikkatiedon lähettäminen J2ME-mobiilisovelluksella		
Työn ohjaaja Kauko Kolehmainen		**Sivumäärä** 44
Työelämäohjaaja Ville Lehtinen		

Tämän opinnäytetyön aiheena oli toteuttaa kuva- ja paikkatietoa lähettävä mobiilisovellus J2ME-sovellusympäristöön Nokian S60-sarjan älypuhelimille, joista löytyy sisäänrakennettu GPS-vastaanotin ja kamera. Sovelluksen keskeisimpiin ominaisuuksiin kuuluu kuva- ja paikkatietojen lähettämisen lisäksi dynaaminen käyttöliittymä, jonka sovellus kykenee luomaan palvelimelta lataamansa XML-muotoisen asetustiedoston parseroidun sisällön perusteella.

Mobiilisovelluksen tekeminen aloitettiin kesällä 2008 kokkolalaiselle ohjelmointitalolle nimeltä Koodikolme Oy eräänlaisena tuotekehittelynä sekä tulikokeena tulevalle työntekijälle. Mobiilisovelluksen toteuttamiseen käytettiin NetBeans IDE -ohjelmointityökalua, sekä Wireless Toolkit -laajennusta.

Työssä toteutettiin mobiilisovelluksen lisäksi muutama yksinkertainen testisivu, joilla voitiin vastaanottaa lähetetty kuva- ja paikkatietodata esittämistä varten. Yksi näistä testisivuista käytti hyväkseen Coppermine Photo Gallery -kuvagalleriasovellusta, jonka tietokantaan kuva- ja paikkatieto onnistuttiin lähettämään. Tämän jälkeen vastaanotettuja kuvia pystyttiin selaamaan suoraan Copperminen avulla. Paikkatietojen esittämisessä käytettiin hyväksi Google Maps -karttasovellusta.

Asiasanat
Coppermine Photo Gallery, dynaaminen käyttöliittymä, Google Maps, GPS, J2ME, Java, kuva, paikkatieto, XML

CENTRAL OSTROBOTHNIA
UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

CENTRAL OSTROBOTHNIA UNIVERSITY OF APPLIED SCIENCES	Date 6.12.2009	Author Sami Santala
Degree programme Information Technology		
Name of thesis Sending Image and Location Data to a Server Using J2ME Mobile Software.		
Instructor Kauko Kolehmainen		**Pages** 44
Supervisor Ville Lehtinen		

The purpose of this Bachelor´s thesis was to create a mobile software into J2ME environment which would allow the user to send pictures and location information to a server. The target platform for this software was the S60-series cell phone by Nokia which had an integrated GPS-receiver and a camera. The main feature of this software, besides the ability to send pictures and location information, was the dynamic user interface, which was created based on the contents of a XML-file, downloaded from a server and then parsed in the software.

The programming of this software was started in the summer 2008 for a small IT-company in Kokkola called Koodikolme Oy. It was commissioned for product development and as a test for the new employee. The software was programmed by using Netbeans IDE and Wireless Toolkit.

After the development of the mobile software several web test pages were created for testing the software. These test pages were used to receive the data sent by this mobile software and then to display it to the users. One of these test pages used Coppermine Photo Gallery to display the received images and location data. The test pages used Google Maps to display the received location data from the mobile software.

Key words

Coppermine Photo Gallery, Dynamic User Interface, Google Maps, GPS, J2ME, Java, Picture, Location, XML

LYHENTEET JA TERMIT

API	Application Programmable Interface
Bugi	Virhe ohjelmassa
C#	C# (C Sharp) ohjelmointikieli
CLDC	Connected Limited Device Configuration
Debuggeri	Tietokoneohjelma, jonka avulla voidaan jäljittää ja korjata bugeja.
Emulaattori	Tietokoneohjelma, jonka avulla ohjelmaa voidaan ajaa erilaisessa ympäristössä, kuin mihin se on alun perin tarkoitettu.
GPS	Global Positioning System
GPS-laite	GPS-signaalia vastaanottava laite
IDE	Integrated Development Environment
J2EE	Java Enterprise Edition
J2ME	Java 2 Micro Edition
J2SE	Java 2 Second Edition
Java ME	Java Platform, Micro Edition
JSR	Java Specification Request
MIDP	Mobile Information Device Profile
Mobiilisovellus	mobiililaitteelle (esim. matkapuhelimelle) tehty sovellus
MySQL	yksi tunnetuimmista tietokannanhallintajärjestelmistä
PDA	Personal Digital Assistant (puhekielessä kämmentietokone)
PHP	PHP: Hypertext Preprocessor (palvelinpuolen ohjelmointikieli)
RMS	Record Management System
S60	Nokian älypuhelinsarja
Thumbnail	pienempi versio alkuperäisestä kuvasta
URL	Uniform Resource Locator
WGS84	World Geodetic System
XML	eXtensible Markup Language
XML-parserointi	XML-muotoisen tiedoston tai tekstin lukeminen siten, että ohjelmallisesti siitä saadaan kerättyä muuttujiin tarvittavat asiat.

TIIVISTELMÄ
ABSTRACT
LYHENTEET JA TERMIT

SISÄLLYS

1 JOHDANTO

Mobiililaitteiden kehittyessä, ja niiden hintojen laskiessa yleistyvät GPS-navigointiin kykenevät matkapuhelimet. Nämä tuovat mukanaan ominaisuuden liittää paikkatietoa monenlaisiin mobiilipalveluihin. Paikkatietoa voidaan käyttää esimerkiksi sääennusteiden kohdistamiseen käyttäjän sijainnin mukaan, tai tarjoamaan käyttäjille tietoa lähistöllä sijaitsevista nähtävyyksistä tai ruokailupaikoista. Soveltamismahdollisuuksia tällaiselle tekniikalle on lähes rajaton määrä. Mainonnan ja tiedottamisen lisäksi paikkatietojen avulla voidaan parantaa jo olemassa olevia sovelluksia.

Logistisia toimintojakin voidaan tehostaa, esimerkiksi metsuri voi ilmoittaa tukkikuskille tarkat paikkatiedot kuorman sijainnista suoraan maastosta tai vaikkapa taksin tilaajan sijainti voidaan välittää taksinkuljettajalle. Paikkatietojen liittäminen käyttäjän lähettämiin kuviin avaa uusia mahdollisuuksia moniin Internetissä toimiviin yhteisöpalveluihin ja luo uusia ulottuvuuksia lomakuvien tarkastelemiseen.

Tämän opinnäytetyön aiheena oli toteuttaa kuva- ja paikkatietoja lähettävä mobiilisovellus Nokian S60-sarjan älypuhelimille, joissa on kamera ja sisäänrakennettu GPS-vastaanotin. Opinnäytetyön aihe saatiin keväällä 2008 ja sen toteuttaminen alkoi kesällä 2008. Mobiilisovellusta tehtiin noin kolme kuukautta. Opinnäytetyö tehtiin kokkolalaiselle ohjelmointiyritykselle nimeltä Koodikolme Oy. Työnantajan motiivi opinnäytetyön teettämiseen oli tuotekehitys, ja samalla suorittaa eräänlainen tulikoe tulevalle työntekijälle.

2 AIHEESEEN LIITTYVIÄ KESKEISIÄ KÄSITTEITÄ

Opinnäytetyön sisällön ymmärtämisen kannalta on tarpeellista selvittää ensin muutama kuva- ja paikkatietoihin sekä satelliittipaikannusjärjestelmiin liittyvä keskeinen käsite. Samalla selitetään J2ME-sovellusympäristön peruskäsitteitä, sekä testauksessa käytettyjä Coppermine Photo Gallery -kuvagalleriasovellusta ja Google Maps -karttasovellusta.

2.1 GPS

Navstar GPS (Global Positioning System) on alun perin Yhdysvaltojen armeijan kehittämä ja rahoittama satelliittipaikannusjärjestelmä. Järjestelmän kehittäminen aloitettiin 1970-luvun puolessavälissä. Se oli tarkoitettu aluksi vain armeijan omaksi paikannusjärjestelmäksi, kunnes vuonna 1983 Yhdysvaltojen sen aikainen presidentti Ronald Reagan vapautti järjestelmän myös siviilien käyttöön. Navstar GPS järjestelmä koostuu nykyään 24 satelliitista ja neljästä varasatelliitista. Järjestelmän päähallinnointikeskus sijaitsee Yhdysvalloissa, Colorado Springssissä. Lisäksi järjestelmään kuuluu neljä tarkkailuasemaa, jotka on sijoitettu eri puolille päiväntasaajaa. (kowoma.de 2009; Aviation Explorer 2009.)

Sijainnin paikkatietojen laskemista varten on GPS-laitteen saatava olosuhteiden mukaan lukitus ainakin neljään satelliittiin. Tietyissä erikoistilanteissa satelliittilukituksen voi saada aikaan vähemmälläkin satelliittimäärällä. Esimerkiksi tilanteessa jossa ollaan merellä tai jossa ei tarvitse tietää paikan korkeuskoordinaattia, voidaan GPS-laitteen sijainti saada selville kolmellakin satelliitilla. Useamman satelliitin avulla paikkatieto saadaan selvitettyä tarkemmin. (kowoma.de 2009; Aviation Explorer 2009.)

GPS:n toiminta perustuu välimatkojen mittaamiseen. Satelliitti lähettää navigaatiosignaalin lisäksi atomikellon ajan vastaanottavaan GPS-laitteeseen. GPS-laite laskee kauanko signaalilla menee matkaan selvittäen samalla etäisyytensä satellii-

tista. Koska vastaanottavan GPS-laitteen kello on huomattavasti epätarkempi kuin lähettävän satelliitin atomikello, on sen saatava useampi satelliittiyhteys tarkentaakseen tietonsa sijainnistaan. (kowoma.de 2009; Aviation Explorer 2009.)

GPS-vastaanottimen selvittämä paikkatieto ottaa häiriötä monista asioista. Häiriöitä aiheuttavat esimerkiksi sää, satelliittien rata ja kellovirheet, GPS-vastaanottimen virhe, käyttäjän virhe, satelliittien sijainti toisiinsa nähden sekä signaalin monitieeteneminen ja tahallinen häirintä. (kowoma.de 2009.) USA:n puolustusministeriö heikensi tahallisesti 1. toukokuuta 2000 asti siviilikäyttöön tarkoitettujen Navstar GPS -laitteiden tarkkuutta (U.S. Department of Defense 2007.)

Navstar GPS ei ole ainoa olemassa oleva satelliittipaikannusjärjestelmä, mutta se on niistä käytetyin ja yleisin. Muita satelliittipaikannusjärjestelmiä ovat Eurooppalaisen avaruusjärjestön (ESA) kehittämä Galileo, alun perin Neuvostoliiton luoma, nykyään Venäjän kehittämä GLONASS, Kiinan COMPASS, Intian IRNSS sekä Japanin QZSS.

2.2 GPS-koordinaatit

Opinnäytetyöni kannalta tärkeässä osassa ovat GPS-vastaanottimelta saatavat koordinaatit. Koordinaatit ovat tarkka tapa ilmoittaa sijainti. Koordinaatteja on tapana esittää monilla eri tavoin. Seuraavaksi selvitetään koordinaattien yleisimmät esitystavat ja niiden yhteydessä käytetyt symbolit.

Koordinaattisymboleja:

°	**Asteet (Degrees)**
'	**Minuutit (Minutes)**
"	**Sekunnit (Seconds)**

Kolme yleisintä koordinaattien esittämistapaa

DDD° MM' SS.S"	Asteet, minuutit ja sekunnit
DDD° MM.MMM'	Asteet ja desimaaliminuutit
DDD.DDDDD°	Desimaaliasteet

1. **Asteet, minuutit ja sekunnit**

DDD° MM' SS.S"
32° 18' 23.1" N 122° 36' 52.5" W
Tämä on yleisin tapa merkitä koordinaatteja karttoihin. Koordinaateissa 60 sekuntia on yksi minuutti (*60" = 1'*) ja jokaisessa asteessa on 60 minuuttia (*60' = 1°).* (Carnes 2008.)

2. Asteet ja desimaaliminuutit

DDD° MM.MMM'
32° 18.385' N 122° 36.875' W
Tätä formaattia käytetään yleensä elektronisten navigointilaitteiden yhteydessä. (Carnes 2008.)

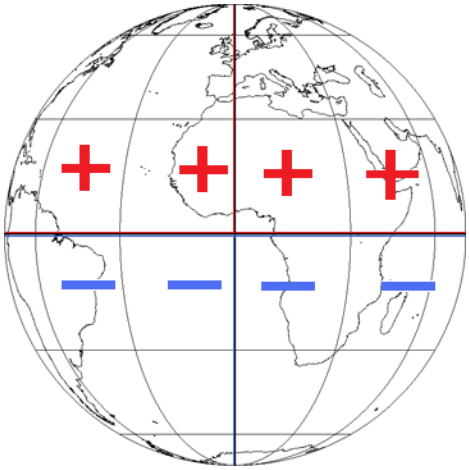
3. **Desimaaliasteet**

DDD.DDDDD°
32.30642° N 122.61458° W
tai +32.30642, -122.61458

Tätä formaattia käytetään suurimmassa osassa tietokonepohjaisista kartoitusjärjestelmistä. Koordinaatit esitetään desimaalilukuina ja ovat siten valmiina liukuluku-datatyypissä, ja eivät siksi tarvitse erillistä muutosta koodissa asteista minuu-

teiksi tai sekunneiksi. (Carnes 2008.) Tämä tekee koordinaattien käsittelystä koodissa huomattavasti yksinkertaisempaa. Tässä opinnäytetyössä käytetään koordinaattiformaattina tätä samaa desimaalimuotoa.

Usein desimaalien yhteydessä käytetään miinus- ja plusmerkkejä osoittamaan, sijaitseeko pituusaste (longitude) tai leveysaste (latitude) pohjoisella vai eteläisellä pallonpuoliskolla. Etumerkin määräytymistä sijainnin perusteella on selvennetty kuviossa 1.



KUVIO 1. Pituus- ja leveyspiirien desimaalimuodon etumerkin määräytyminen sijainnin mukaan

2.3 Coppermine Photo Gallery

Coppermine on ilmainen PHP:llä toteutettu kuvagalleriasofta. Se tukee useita käyttäjätilejä, joihin käyttäjät voivat lisätä omia kuviaan ja järjestellä niitä erilaisiin albumeihin. Kuvien lisäksi Coppermine tukee monia muitakin tiedostomuotoja, kuten videoita ja äänitiedostoja. (Coppermine dev team 2003.) Tässä opinnäytetyössä Coppermine Photo Gallery -kuvagalleriasovellusta käytetään testauksessa esittämään käyttäjän lähettämiä kuvia.

2.4 Google Maps

Google Maps on Googlen kehittämä karttapalvelu. Google Maps on lisenssiltään vapaa ei-kaupalliseen käyttöön, ja se on käytössä useissa karttapohjaisissa sovelluksissa. Google Maps on edelleen kehityksessä, ja siihen ilmestyy jatkuvasti uusia ominaisuuksia. Uusimmassa versiossa on jo katunäkymän mahdollistava ominaisuus, tosin alueita, joissa katunäkymää voidaan käyttää, ei ole vielä kovinkaan monta. (Google 2009.)

Ohjelmoija pääsee Google Mapsiin kiinni Googlen kehittämän Google Maps API -rajapinnan kautta. Google Maps API:n avulla onnistuu Javascript- ohjelmointikielellä esimerkiksi reittien, reittipisteiden ja valokuvien esittäminen maailman kartalla karttapalvelua käyttäen. Google Maps vaatii toimiakseen Google API Keyn joka on sivustokohtainen tunniste, jonka perusteella Google voi kerätä tietoja Google Mapsia käyttävistä sivustoista. Google API Key on saatavissa Google Mapsin sivuilta. (Google 2009.)

Tässä opinnäytetyössä Google Mapsia käytetään esittämään mobiilisovelluksen lähettämiä paikkatietoja kartalla.

2.5 J2ME

Java Micro Edition eli Java ME (tunnettiin aiemmin myös nimellä J2ME) on kevyempi ja rajoittuneempi sovellusympäristö Sun Microsystemsin kehittämästä Java-teknologiasta. Se on suunniteltu suorituskyvyiltään huomattavasti rajoittuneempien laitteiden sovellusten kehittämiseen. Se poikkeaa esimerkiksi J2SE:stä ja J2EE:stä siten, että siinä on käytettävissä pienemmät luokkakirjastot. Ohjelmoijan näkökulmasta ajatellen pienempien luokkakirjastojen määrä johtaa siihen, että ohjelmoijalla on käytettävissään vähemmän valmiita funktioita. J2ME:n voidaan ajatella koostuvan kolmesta suuremmasta käsitteestä. Nämä käsitteet ovat konfiguraatio, profiili ja valinnainen paketti. (Giguère 2002.)

Konfiguraatiot ovat kokonaisia Java-ajonaikaisia ympäristöjä, jotka koostuvat Java-virtuaalikoneesta (KVM tai CVM, konfiguraation mukaan), natiivista rajapinnasta järjestelmän koodiin sekä ryhmästä Javan perusluokkia. J2ME määrittelee kaksi erilaista konfiguraatiota: CDC:n ja CLDC:n. CDC (Connected Device Configuration) on konfiguraatio, jota käytetään monissa sulautetuissa järjestelmissä ja laitteissa, esimerkiksi tavallisista teollisuuskäyttöön tarkoitetuista PDA-laitteista digiboxeihin ja jääkaappeihin. Matkapuhelinten parissa CDC:tä tukevia laitteita on hyvin vähän. CLDC (Connected Limited Device Configuration) on konfiguraatio laitteille, joiden suorituskyky on vielä heikompi kuin CDC-laitteilla. CLDC käytetyin profiili matkapuhelinten parissa, jotka tukevat J2ME:tä. (Giguère 2002.)

Pelkällä konfiguraatiolla ei vielä päästä ohjelmoimaan kovinkaan ihmeellisiä sovelluksia, sillä konfiguraatioissa ei ole vielä edes tarvittavia luokkia käyttöliittymiin, ja ne ovat muutenkin ominaisuuksiltaan eräänlaisia peruskiviä, joiden päälle laajennetaan profiileilla ja valinnaisilla paketeilla sisältöä siten, että voidaan ohjelmoida monimutkaisempia sovelluksia. (Giguère 2002.)

Profiilit ovat konfiguraatioita laajentavia ryhmiä koodikirjastoja ja ohjelmointirajapintoja. Profiilit usein sisältävät tarvittavat rakennuspalaset käyttöliitymälle. Yksi näistä profiileista on MIDP eli Mobile Information Device Profile. MIDP:in kehittyneemmät versiot, esimerkiksi MIDP 2.0 vaatii toimiakseen uudemman puhelimen kuin MIDP 1.0. Uudemmissa versioissa on käytettävissä enemmän ominaisuuksia. Esimerkiksi MIDP 1.0 -versiossa ei ole tukea softakäyttöön tarkoitetuille ääniefekteille tai kokonäyttötilalle, kun taas MIDP 2.0 -versio tukee monenlaisten multimedia ja pelikäyttöön tarkoitettuja ominaisuuksia. Profiilit vaativat toimiakseen sen, että mobiililaite tukee kaikkia konfiguraation ominaisuuksia, sekä valitun profiilin ominaisuuksia. (Giguère 2002.)

Valinnaiset paketit ovat eräänlaisia yksittäisiä laajennuksia, jotka eivät kuulu mihinkään profiiliin tai konfiguraatioon. Mikäli mobiililaite tukee tiettyä pakettia, voidaan sitä käyttää lisäämään tukea paketin antamille ominaisuuksille. Esimerkkinä tämäntyyppisistä paketeista ovat mm. Location API (JSR 179), jonka avulla voi-

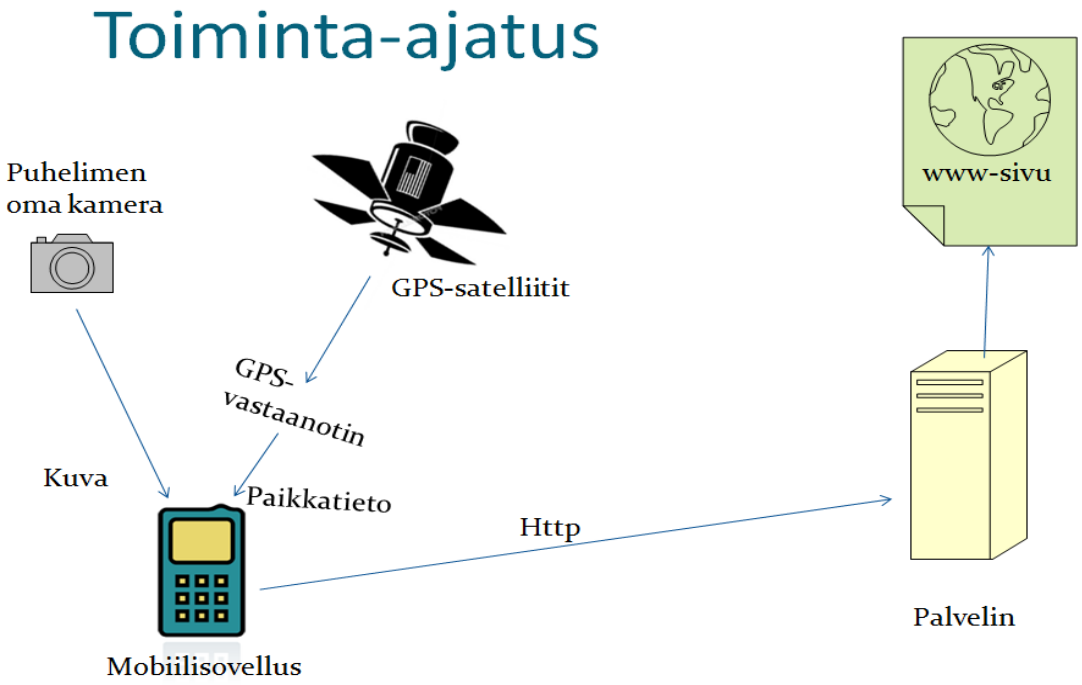
daan käyttää GPS-vastaanotinta ja MMAPI(JSR 135), jonka avulla päästään käsiksi kameraan, videoon yms. multimediaominaisuuksiin, sekä Bluetooth API (JSR 82), joka mahdollistaa Bluetooth-yhteydet mobiililaitteessa.

Koska J2ME-sovellus voi koostua näin monesta erilaisesta tekijästä on usein hämäävä puhua pelkästään "J2ME-sovelluksesta", koska se ei yksinään vielä ilmoita kovin paljoa kuulijalle. Käytetyn profiilin, konfiguraation ja valinnaisten pakettien perusteella voidaan kertoa hieman tarkempi kuva siitä, millaiselle alustalle sovellus on tehty ja millaisia ominaisuuksia se vaatii mobiililaitteelta toimiakseen. (Giguère 2002).

3 VAATIMUSANALYYSI

Ennen opinnäytetyön vastaanottamista sain työnantajalta varsin perusteelliset ohjeet mobiilisovelluksen toteuttamiseksi. Ohjeissa selvitettiin varsin yksityiskohtaisesti mitä, sovelluksen tulisi pystyä tekemään ja millaisella tavalla vaaditut ominaisuudet tultaisiin toteuttamaan.

Tehtävänä oli toteuttaa mobiilisovellus, jonka avulla käyttäjä voisi lähettää kuva- ja paikkatietoa mobiilisovelluksella palvelimelle, jossa lähetetyt tiedot vastaanotettaisiin, ja niiden avulla ne esitettäisiin käyttäjille www-sivuilla. Mobiilisovelluksen toiminta-ajatusta on esitelty kuviossa 2.

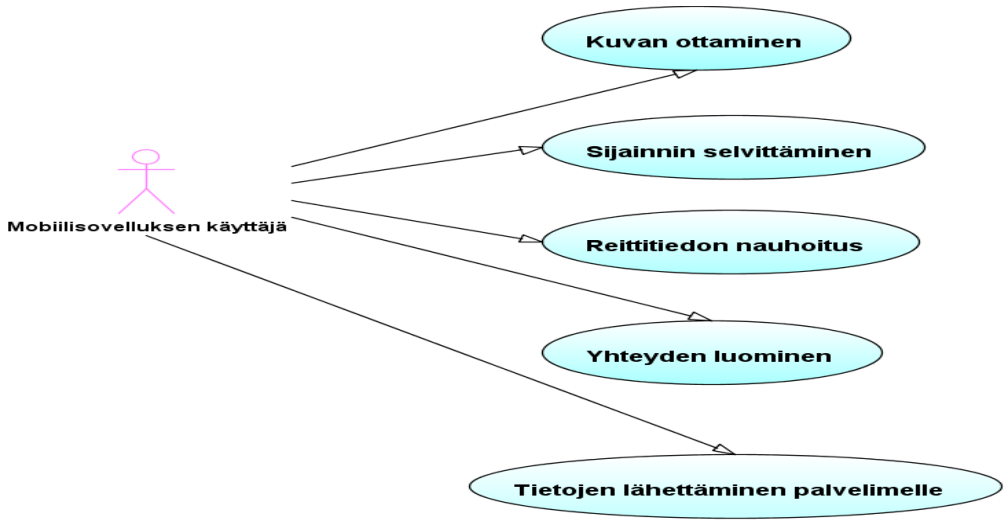


KUVIO 2. DataSender-mobiilisovelluksen toiminta-ajatus

Yksi tärkeimmistä mobiilisovelluksen vaatimuksista oli se, että samalla mobiilisovelluksella voitaisiin lähettää kuva- ja paikkatietoa useampaan eri osoitteeseen. Kohdeosoitteen piti olla muutettavissa mobiilisovelluksen sisältä ilman että sovellusta jouduttaisin muokkaamaan koodinpuolelta jokaista uutta yhteyttä varten. Li-

sävaatimuksina toteutettavaan sovellukseen oli käyttäjänimen ja salasanan vaativa yhteyskohtainen kirjautuminen. Lähetettäessä kuvatietoa tuli myös olla mahdollista lähettää lyhyt teksti, jossa kuvailtiin kuvaa muutamalla sanalla. Palvelimen ja mobiilisovelluksen välinen keskustelu tuli toteuttaa XML-muotoisten tekstien avulla.

Ohjelman toiminnan tuli lisäksi olla muutettavissa palvelimenpuolelta XML–muotoisen asetustiedoston sisällön perusteella. Tämän tiedon avulla voitaisiin määrätä yhteyskohtaisesti, tarvitseeko jokin tietty yhteys kirjautumista, lisätietokenttiä, tai esimerkiksi paikkatiedon lähetystä. Mobiilisovelluksen käyttötapauksia on kuvattu kuviossa 3.



KUVIO 3. Mobiilisovelluksen käyttötapaukset

Työnantajan yritys oli toteuttanut aikaisemminkin kuvan ja paikkatiedon lähettämisen mahdollistavia mobiilisovelluksia, esimerkiksi Pixender ja Mobiilipostikortti (Koodikolme Oy. 2009), mutta tällainen dynaaminen käyttöliittymä oli käsittääkseni se asia mikä yritystä eniten kiinnosti tämän opinnäytetyön teettämisessä.

Alun perin oli tarkoituksena, että toteuttaisin ainoastaan tietoa lähettävän mobiilisovelluksen ja vastaanottavat palvelintiedostot tekisi työnantaja. Tämä muuttui myöhemmin kuitenkin siten, että pääsin itse toteuttamaan myös vastaanottavat testisivut.

4 SUUNNITTELU

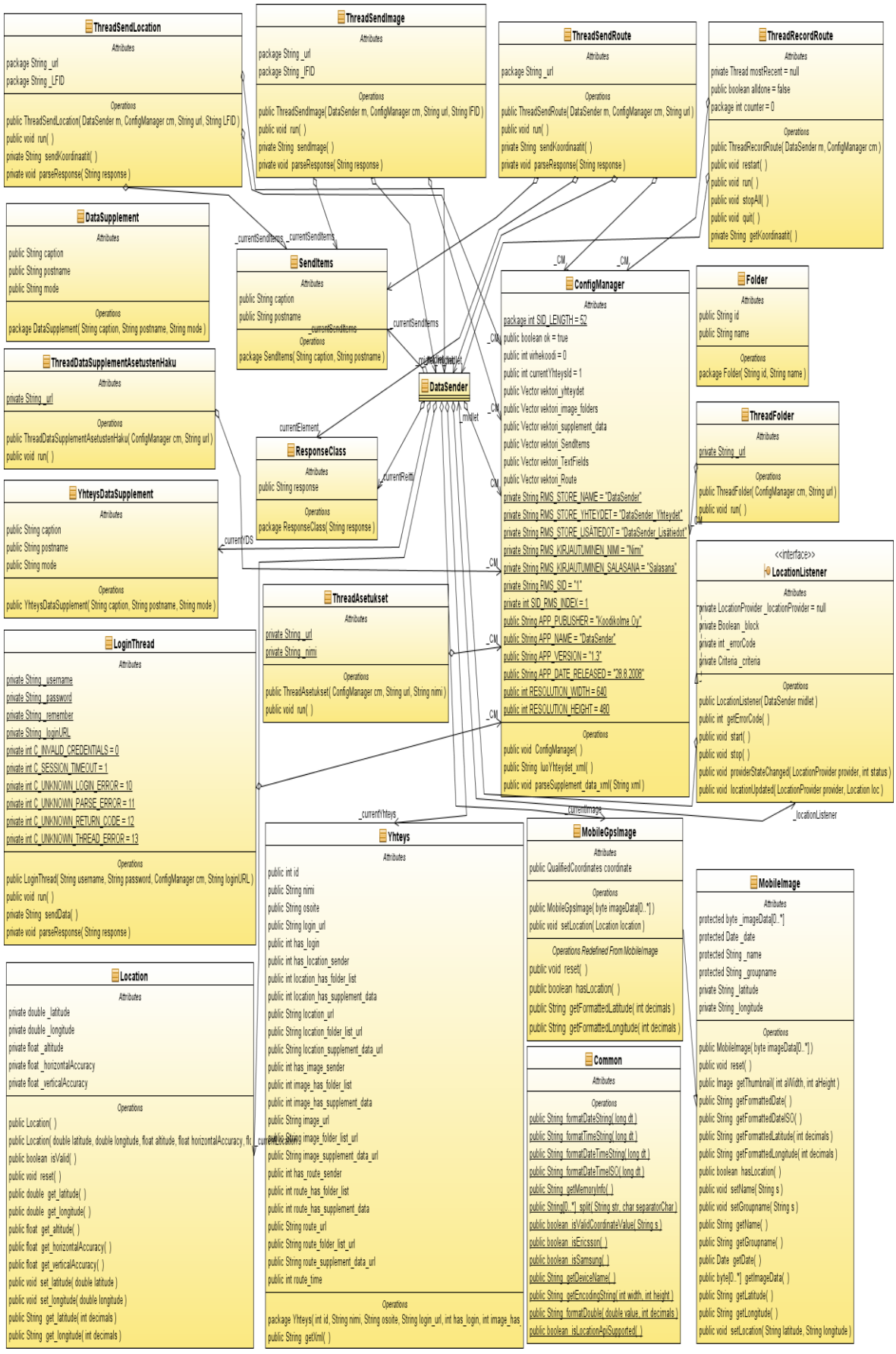
Mobiilisovelluksen suunnittelussa yritettiin pitää mielessä helppokäyttöisyyttä ja tekniikoita, joiden avulla käyttäjä kykenisi välttämään turhauttavien asioiden tekemisen yhä uudelleen ja uudelleen. Esimerkkinä tästä oli yhteyksienhallinta, jonka avulla käyttäjä pystyi luomaan useita yhteyksiä. Luodut yhteydet tallennettiin RMS-tietovarastoon siten, että ne säilyvät siellä vaikka sovellus suljettaisiin ja puhelimesta katkaistaisiin virta. Näin ollen käyttäjä välttyy yhteyksien luomiselta uudelleen jokaisella ohjelman käynnistyskerralla.

Samaa asiaa mielessä pitäen suunniteltiin asetustiedostot, joiden perusteella sovellus rakensi dynaamisesti käyttöliittymäänsä. Kun kaikki käyttäjän mahdolliset valinnat pystyttiin pitämään palvelinpuolella yhdessä tiedostossa, vältyttiin sellaisilta tilanteilta, missä käyttäjä esimerkiksi vahingossa yrittäisi lähettää tietoja väärin kirjoittamaansa osoitteeseen. Samalla ehkäistiin sellaisen tilanteen syntyminen, jossa käyttäjä lähettäisi kuvadataa osoitteeseen, jossa ei esimerkiksi olisi toteutettu kuvadatan vastaanottoa. Toki tällaisella keskitetyllä hallinnalla on myös omat haittapuolensa. Esimerkiksi, jos yhteyden asetustiedosto palvelimella ei ole saatavilla tai on väärin muodostettu, on sovelluksen käyttäminen mahdotonta, ja niinpä edes kokenut käyttäjäkään ei voi tuollaisessa tilanteessa korjata ongelmaa, ottamatta yhteyttä palveluntarjoajaan.

Dynaaminen käyttöliittymä suunniteltiin siten, että sovelluksessa oli kaikki mahdollisuudet tietojen lähettämiseen, mutta yhteyskohtaisen asennustiedoston perusteella määrättiin, mitkä näistä ominaisuuksista olivat käyttäjän nähtävissä ja käytettävissä.

Palvelimen ja mobiilisovelluksen välinen keskustelu suunniteltiin suurimmilta osin toimimaan XML-muotoisten tekstien lähetyksen, vastaanottamisen ja jäsentämisen varaan. Mobiilisovellus päätettiin toteuttaa Java-ohjelmointikielellä S60-sarjan Nokian älypuhelimille käyttäen NetBeans 6.5 IDE -työkalua ohjelman koodaamiseen.

Mobiilisovelluksessa käytettiin hyväksi muutamaa Koodikolme Oy:n kehittämää luokkakirjastoa. Nämä luokkakirjastot sisälsivät lähinnä paikkatietokoordinaattien, ja päivämäärien muotoiluun liittyviä funktioita sekä kuvan ottamiseen ja tietojen lähettämiseen liittyviä funktioita. Kuviossa 4 on nähtävissä mobiilisovelluksen luokkakaavio. Kuviossa on visual MIDletin attribuutit ja operaatiot supistettu pois näkyvistä tilan säästämiseksi. Luokkakaavio on generoitu automaattisesti koodista käyttäen NetBeansin omaa UML-laajennusta.



KUVIO 4. Mobiilisovelluksen luokkakaavio

5 TOTEUTUS

Seuraavissa luvuissa 5.1–5.12 selvitetään tarkemmin kuva- ja paikkatietoa lähettävän DataSender-mobiilisovelluksen toimintaa ja sen toteuttamistapaa. Luvuissa perehdytään tarkemmin sovelluksella lähetettäviin kuva-, paikkapiste- ja reittitietoihin, sekä palvelimen ja mobiilisovelluksen välillä tapahtuvaan kirjautumiseen. Lisäksi selvitetään dynaamisen käyttöliittymän toimintaa ja toteuttamistapaa.

5.1 Mobiilisovellus

Sovelluksen koodaaminen lähti suhteellisen hyvin käyntiin, sillä olin käyttänyt samaa NetBeans IDE -työkalua ja tutustunut osaan sen toiminnoista koulussa pidetyllä J2ME-kurssilla. Monimutkaisemmat toiminnot, kuten emulaattoriin ja debuggerin käyttö, tulivat tutuiksi sovelluksen koodaamisen edetessä. Sovellusta tehtiin pala kerrallaan eteenpäin. Jokaisen lisätyn ominaisuuden jälkeen testattiin emulaattorin ja debuggerin avulla ominaisuuden toiminnallisuutta. Bugien korjaamisen jälkeen siirryttiin toteuttamaan seuraavaa toiminnallisuutta mobiilisovellukseen.

NetBeansin kanssa käytetyn Wireless Toolkit 2.5.1 emulaattorin avulla ohjelman testaamisessa huomasin, että ihan kaikkia ominaisuuksia ei voi testata emulaattorin avulla, ja niinpä sovellusta alettiin testata myös Nokian 6110 Navigator -puhelimella, jonka sain käyttööni työpaikalta.

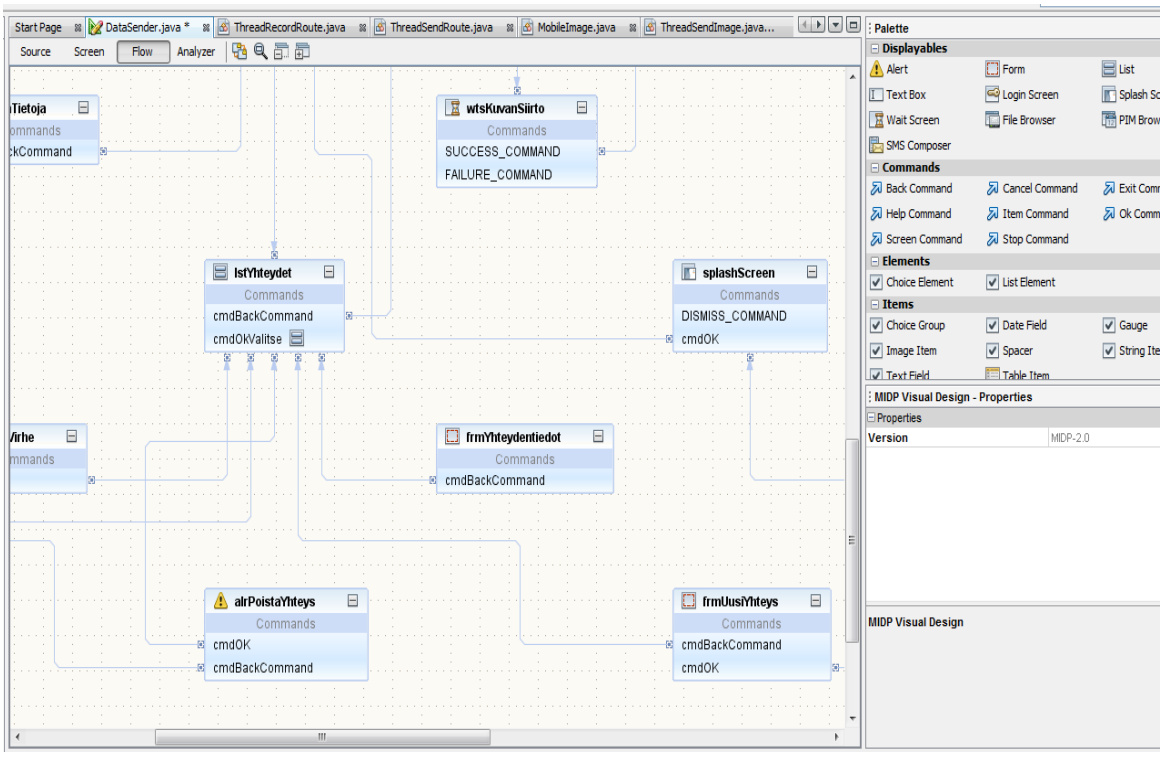
Sovelluksen toteuttamista kuitenkin helpotti huomattavasti se, että suunnittelu oli toteutettu varsin selkeästi, ja itsellä oli sen ansiosta jatkuvasti selvillä idea siitä, mitä ollaan tekemässä, ja mikä pitää tehdä seuraavaksi, kun tekeillä oleva asia saadaan valmiiksi.

Kuva- ja paikkatietojen ottamisessa ja lähettämisessä käytettiin apuna Koodikolme Oy:n luomia luokkia. Nämä luokat sisälsivät funktioita paikkatietojen koordinaattien muotoiluun, ja kameran käyttämiseen. Lisäksi apuna käytettiin XML-muotoisen

tekstin jäsentämisen J2ME-mobiilisovelluksessa mahdollistavaa Xparse-J-luokkaa.

5.2 Käyttöliittymä

Mobiilisovelluksen toteutus lähti käyntiin käyttöliittymän tekemisestä. NetBeansillä on hyvin vaivatonta tehdä käyttöliittymiä Java ME -ohjelmille. Käyttöliittymän sai tehtyä vetämällä tarvittavat Displayable-komponentit Flow-näkymään ja yhdistämällä niiden väliset siirtymiset viivoilla toisiinsa. Monimutkaisemmat siirtymiset ja niiden ehdot siirtymisissä ruuduista toiseen jouduttiin kuitenkin tekemään koodin puolella. Kuviossa 5 on näkyvissä NetBeansin käyttöliittymän luomiseen tarkoitettu Flow-näkymä.



KUVIO 5. Käyttöliittymän luominen NetBeansin Flow-näkymän avulla

Displayable-komponentit ovat J2ME:n valmiita käyttöliittymäkomponentteja, joiden avulla voidaan luoda nopeasti yksinkertaisia käyttöliittymiä VisualMidletejä hyödyntäen. Valmiita displayable-käyttöliittymäkomponentteja on erityyppisiä, ja ne eroavat asetuksiltaan sekä toiminnoiltaan toisistaan. Tässä opinnäytetyössä toteutetussa mobiilisovelluksessa käytettiin käyttöliittymän luomiseen seuraavia Disp-

layable-komponentteja:

Käyttöliittymässä käytetyt Displayable-käyttöliittymäkomponentit:

- SplashScreen
- List
- Form
- WaitScreen
- Alert

SplashScreen-komponentti on eräänlainen sovelluksen oma mainos- tai tervehdysruutu. Sitä käytetään yleensä sovelluksen ensimmäisenä sivuna ilmoittamaan käyttäjälle käynnistetyn ohjelman nimi- ja versiotietoja. Se voi samalla toimia eräänlaisena latausruutuna isommissa sovelluksissa. Tosin käyttöliittymäkomponenteissa on latausruutua varten oma komponenttinsa nimeltä WaitScreen.

List eli listakomponentti on kaikessa yksinkertaisuudessaan lista, johon voidaan lisätä rivejä. Jokaiselle listassa näkyvälle riville voidaan lisätä oma kuvansa ja toimintonsa. Tämän takia listaa on näppärä käyttää käyttöliittymässä eräänlaisena pääsivuna, mistä käyttäjä voi valita siirtymisen ohjelman muihin osioihin. Listaa voidaan myös käyttää esimerkiksi tilanteissa joissa käyttäjän täytyy valita jokin asia ennen seuraavalle sivulle siirtymistä.

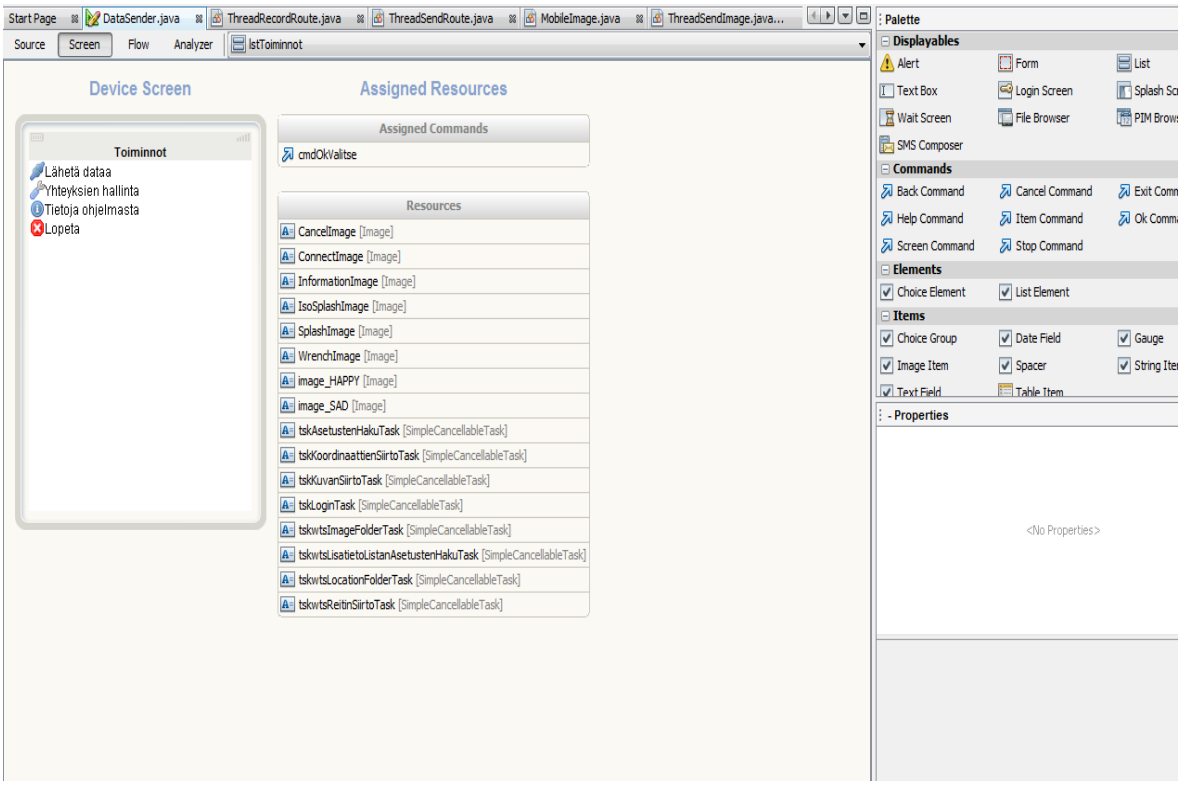
Form eli lomake on monipuolinen komponentti, johon voi asettaa tekstikenttiä, otsikoita ja kuvia sekä monenlaisia valintakontrolleja. Form on kaikista displayableista monipuolisin muokattavuutensa ansiosta. Tässä opinnäyteyössä formeja käytettiin mm. tietojen kyselyyn käyttäjältä, kuvan ottamiseen ja ohjelman tietojen esittämiseen.

WaitScreen on nimensä mukaisesti displayable-käyttöliittymäkomponentti odottamista varten. Odotusruudulle voidaan asettaa tehtävä, jonka ohjelman on suoritettava ennen kuin odotusruudusta voidaan siirtyä eteenpäin. Siirtymisen voi asettaa tapahtumaan eri tavalla sen mukaan onnistuiko odotusruudulle määrätty tehtävä vai ei. Tekemässäni mobiilisovelluksessa käytetään WaitScreeniä esimerkiksi

http-liikennetilanteissa, joissa sovellus lataa tai lähettää palvelimelle tietoja.

Alert displayable -käyttöliittymäkomponentti on tarkoitettu hälytys- ja varoitustilanteita varten. Sitä voidaan käyttää esimerkiksi tilanteissa, joissa käyttäjä on vahingossa syöttänyt tekstikenttään epäkelpoa tekstiä, esimerkiksi tilanne jossa käyttäjä syöttää "minuutit" - tekstikenttään numeroiden sijaan kirjaimia. Alert-komponenttia voidaan käyttää myös tilanteissa joihin ei liity mitään virheitä tai ongelmia. Esimerkiksi tässä opinnäytetyössä Alert-komponenttia käytetään ilmoittamaan käyttäjälle tieto kuvanlähetyksen onnistumisesta.

Netbeansin käyttöliittymässä on erillinen Screen-näkymä sitä varten, että käyttäjä voi muokata valitun Displayablen asetuksia. Kuviossa 6 on esillä ruutukaappaus tästä Screen-näkymästä.



KUVIO 6. Yksittäisen käyttöliittymäkomponentin muokkaaminen NetBeansin Screen-näkymän avulla

5.3 Asetukset

Tämän mobiilisovelluksen tarkoitus oli lähettää kuva- ja paikkatietoa palvelimelle. Palvelimen tai palvelimien ja mobiilisovelluksen välinen liitos tapahtuu niiden välille luodun yhteyden avulla. Yhteys luodaan XML-muotoisesta asetustiedostosta, joka ladataan mobiililaitteeseen uuden yhteyden luomisen aikana. XML-muotoinen asennustiedosto on nähtävissä kuviossa 7.

```
<options>
    <has_login>0</has_login>
    <login_url>0</login_url>
    <has_location_sender>0</has_location_sender>
    <location_has_folder_list>0</location_has_folder_list>
    <location_url>0</location_url>
    <location_folder_list_url>0</location_folder_list_url>
    <location_has_supplement_data>0</location_has_supplement_data>
    <location_supplement_data_url>0</location_supplement_data_url>
    <has_image_sender>1</has_image_sender>
    <image_has_folder_list>0</image_has_folder_list>
    <image_url>http://lt.kapsi.fi/DataSender/Kuvalista/Testi/DataSender.php</image_url>
    <image_folder_list_url>0</image_folder_list_url>
    <image_has_supplement_data>1</image_has_supplement_data>
    <image_supplement_data_url>http://lt.kapsi.fi/DataSender/Kuvalista/ds_lisatiedot.xml</image_supplement_data_url>
    <has_route_sender>0</has_route_sender>
    <route_url>0</route_url>
    <route_has_folder_list>0</route_has_folder_list>
    <route_folder_list_url>0</route_folder_list_url>
    <route_has_supplement_data>0</route_has_supplement_data>
    <route_supplement_data_url>0</route_supplement_data_url>
    <route_time>1000</route_time>
</options>
```

KUVIO 7. XML-muotoinen asetustiedosto

5.3.1 Asetuksien merkitys

Asetustiedosto on erittäin tärkeässä osassa tämän ohjelman toimintaa ajatellen. Sen sisällön perusteella määräytyy, mitkä lähetystoiminnot ovat käyttäjälle avoinna kyseisellä yhteydellä. Lisäksi asetustiedostosta selviää, onko yhteyden lähetystoiminnoilla, esimerkiksi kuvanlähetyksellä, käytössään lisätietojen syöttämisen mahdollistava lisätietoruutu, jota voidaan käyttää esimerkiksi kuvan lähettämisessä lyhyen teksti kommentin lisäämiseen lähetykseen. Toinen lisäominaisuus, joka lähetykselle lisättiin, oli niin sanottu kansiolistaus, jota hyödyntäen voitiin vastaanottaa palvelimelta listaus kansioista, joihin lähetyksen tiedosto pystyttiin ohjaa-

maan. Enemmän tietoa lisätiedoista ja kansioista on luvuissa 5.8 ja 5.9.

Käytössä olevien toimintojen ja lisäominaisuuksien lisäksi asetustiedostossa on määrätty osoitteet, joissa lähetyksen vastaanottavat palvelintiedostot sijaitsevat, sekä osoite, josta mobiilisovellus voi ladata tarvittavat lisätiedot.

5.3.2 Yhteyksien hallinta

Yhteyksien hallinnointia varten tehtiin sovellukseen oma osio. Yhteyksien hallinnan avulla onnistuu uusien yhteyksien luominen, muokkaaminen ja poistaminen. Yhteyden yksittäisiä asetuksia ei pystytä muuttamaan mobiilisovelluksen puolelta, sillä asetuksia voi muokata ainoastaan palvelinpuolelta XML-tiedostoa muokkaamalla. Tämä siksi, että esimerkiksi kuvadatan lähettäminen palvelimelle, jota ei ole asetettu ottamaan sitä vastaan on täysin turhaa, ja niinpä päätös siitä "Mitä tietoja voidaan lähettää ja vastaanottaa" tehdään palvelimen puolella. Yhteyden luomisessa yhteydelle asetetaan nimi, jonka perusteella käyttäjä voi tunnistaa sen ja samalla erottaa sen muista luomistaan yhteyksistä. Toinen vaadittu tieto yhteyden luomisessa on osoite, josta ohjelma löytää tarvittavan XML tiedoston, jonka parseroinnin jälkeen ohjelma osaa muodostaa ja luoda uuden yhteyden.

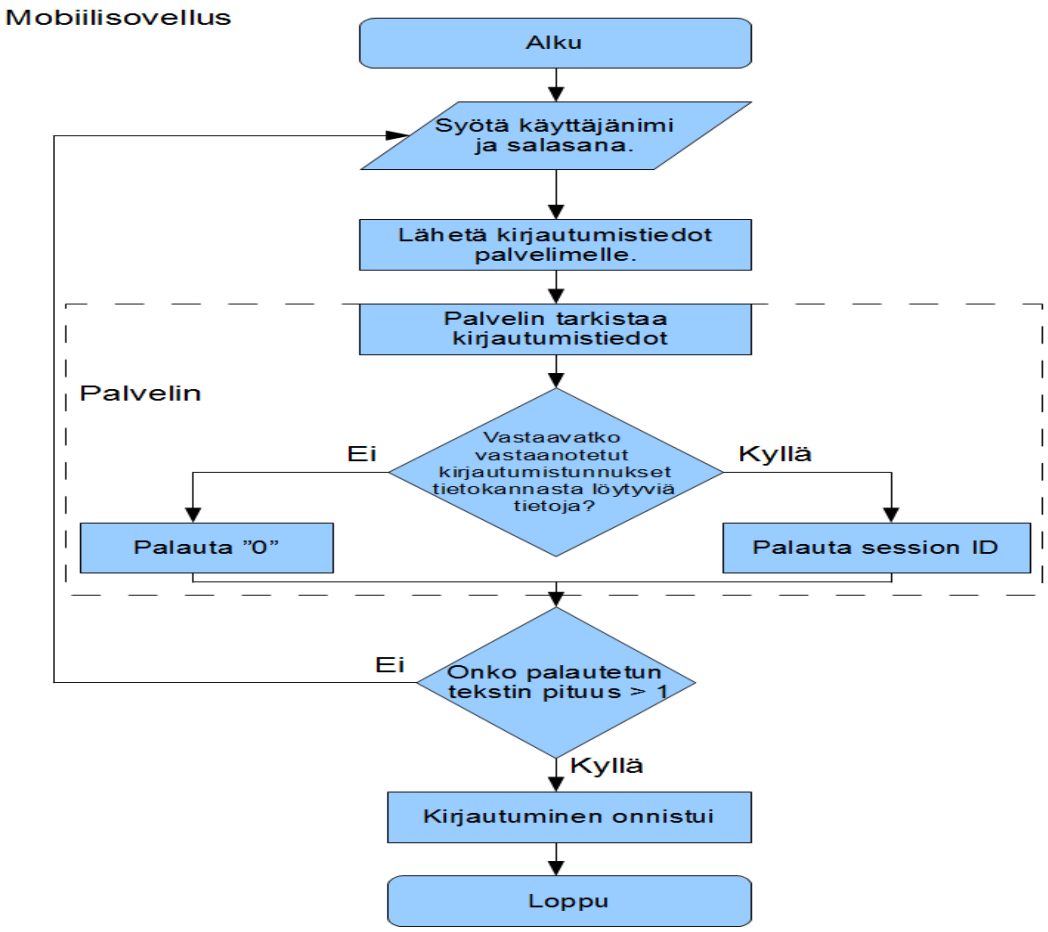
5.3.3 RMS

RMS (Record Management System) –rajapintaa hyödyntäen mobiisovellukseen luodut yhteydet pystyttiin tallentamaan mobiililaitteen omaan tiedostomuistiin ja näin päästiin eroon yhteyksien luomisesta uudelleen jokaista käyttökertaa varten. RMS:n avulla tallennetut yhteydet säilyvät mobiililaitteessa, kunnes ne sieltä poistetaan, ja säilyttävät siis tilansa vaikka mobiililaitteesta katkaistaisiin virta tai sovellus suljettaisiin.

5.4 Kirjautuminen

Kirjautumista käytetään tunnistamaan käyttäjä, ja samalla estämään asiattomien käyttäjien pääsy järjestelmään. Tässä mobiilisovelluksessa kirjautuminen on yhteyskohtaista, ja yhteyden mukaan siis tarpeellista tai tarpeetonta. Mikäli asetustiedostossa on määrätty kirjautuminen tarpeelliseksi, pyydetään käyttäjää kirjautumaan, aina kun tämä tahtoo käyttää valitsemaansa yhteyttä. Mikäli kirjautuminen on asetettu tarpeettomaksi, ohitetaan koko kirjautumisruutu käyttöliittymässä ja siirrytään suoraan tietojen ottamiseen ja niiden lähettämiseen. Kirjautuessaan käyttäjä lähettää käyttäjänimensä ja salasanansa palvelimelle, jossa nämä tiedot tarkistetaan esimerkiksi tietokannassa olevaan tauluun tai tauluihin vertailemalla. Mikäli kirjautuminen onnistuu, palauttaa palvelin mobiilisovellukseen tiedon kirjautumisen onnistumisesta palauttamalla Session id -merkkijonon. Epäonnistunut kirjautuminen palauttaa vain virhekoodin, jonka perusteella mobiilisovellus osaa ilmoittaa käyttäjälle virhesanoman.

Session id on palvelimen generoima merkkijono, joka toimii eräänlaisena avaimena myöhemmin tapahtuville toiminnoille, joissa palvelin ja mobiililaite vaihtavat tietoja. Session id välitetään myöhemmin tapahtuvissa lähetyksissä palvelimelle muiden lähetettävien tietojen kanssa siksi, että palvelin pystyy myös vastaanottaessaan tunnistamaan "keneltä nämä tiedot tulevat". Kuviossa 8 on esitetty kirjautumisen toiminta vuokaaviossa.



KUVIO 8. Kirjautumista kuvaava vuokaavio

5.5 Kuva

Mobiililaitteella kuvan ottaminen hyödyntää mobiililaitteen omaa kameraa. Kameraan päästään käsiksi The Mobile Media API (MMAPI) –rajapinnan kautta. Kuvan ottaminen tapahtuu koodin puolella getSnapshot()-funktion avulla ottamalla kuva ruudulla näkyvästä videosta, joka näyttää kameralta tulevaa kuvaa.

5.5.1 Kuvan käsittely

Kuvan ottamisen jälkeen otettu kuva tallennetaan mobiilisovelluksen muistiin. Kuvan resoluutio on 640 x 480, vaikka mobiililaitteen kamera pystyisi puhelinmallin

mukaan kuvaamaan korkeammankin resoluution kuvia. Pienemmällä kuvakoolla vältetään vanhempien puhelinmallien virhetilanteet, joissa muisti loppuu kesken kuvauksen aikana. Kuvaamisen jälkeen puhelimen muistiin ladatusta kuvasta tehdään thumbnail (pienempi esikatselukuva), joka asetetaan valmiiksi seuraavaksi avattavaan esikatseluruutuun.

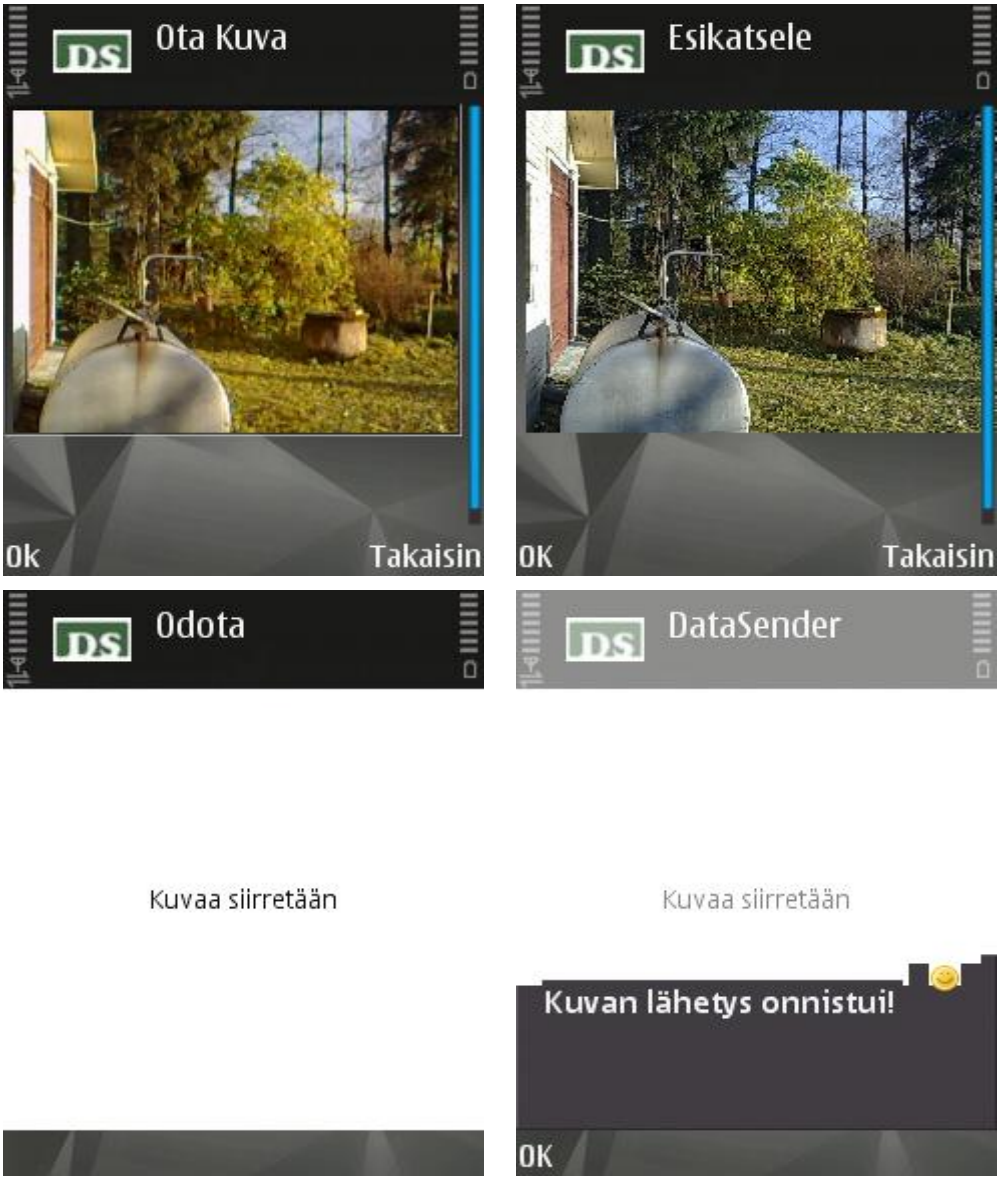
5.5.2 Kuvan esikatselu

Kuvan ottamisen jälkeen käyttäjä ohjataan esikatseluruudulle. Esikatseluruudulla käyttäjä voi vielä katsella ottamaansa kuvaa ennen sen lähettämistä. Tämä on varsin tarpeellista mobiilisovelluksessa, koska jos kuvaaminen epäonnistuu, esimerkiksi tilanteessa, missä käyttäjän käsi heilahtaa viime hetkellä ja kuvasta tulee epätarkka tai kuvattavaa asiaa ei saada rajattua oikein, voi käyttäjä hylätä ottamansa kuvan ennen sen lähettämistä palvelimelle. Lisäksi on ymmärrettävää, että varsinkin hitaammilla datapaketeilla epäonnistuneiden kuvien lähettäminen on käyttäjälle varsin turhauttavaa.

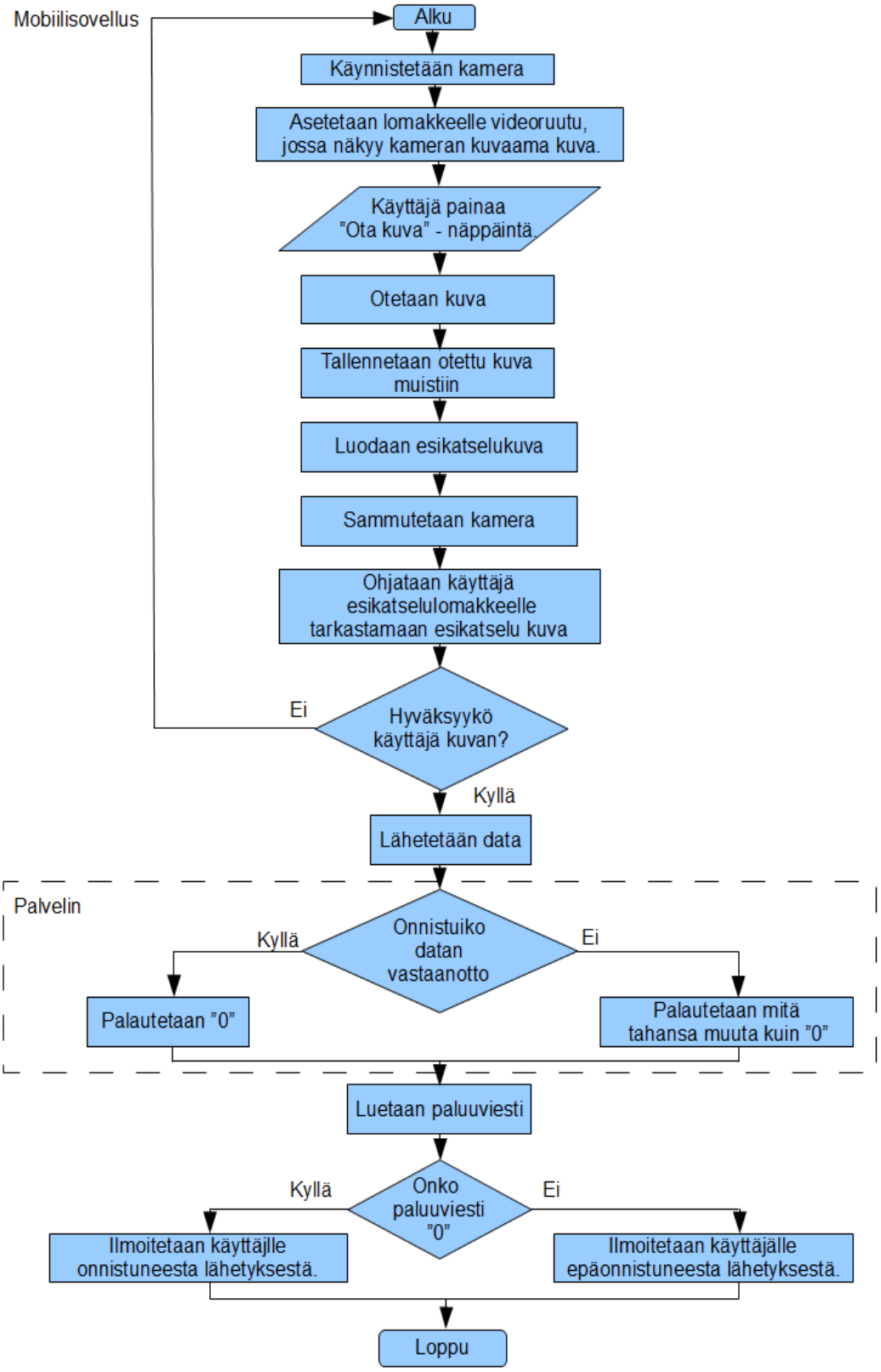
5.5.3 Kuvan lähettäminen

Kuva tallennetaan väliaikaisesti puhelimen muistiin ja lähetetään sieltä HTTP:n yli asetustiedoston määräämään osoitteeseen. Kuvan lähettäminen tapahtuu sitä varten tehdyssä säikeessä. Lähettämisen ajaksi käyttäjä ohjataan WaitScreen-odotusruudulle, jossa käyttäjän on odotettava tiedon siirtämiseen kuluvan ajan verran. Palvelin palauttaa tiedoston vastaanottamisen jälkeen virhekoodin, jonka perusteella sovellus voi selvittää onnistuiko kuvan lähetys vai ei. Kuviossa 9 on nähtävissä mobiilisovelluksesta napatut ruutukaappaukset, joista selviää miltä kuvan ottaminen ja lähettäminen näyttää käytännössä. Kuviossa 10 on nähtävissä kuvan ottamisen ja lähettämisen toiminta esitettynä vuokaaviossa.

Lähetettävä data on tiedostokooltaan kuvasta riippuen noin 50 kilotavua. Kuvan siirtämiseen menee puhelimen tietoliikenneyhteyden nopeudesta ja verkon tilasta riippuen eripituinen aika. Omalla puhelimellani sovellusta testatessani huomasin, että kuvan siirtämiseen menee noin minuutti halvimmalla palveluntarjoajani GPRS-datapaketilla. Lähettämisen aikana välitetään myös GPS-koordinaatit, mikäli ne on saatu selvitettyä siihen mennessä. Paikkatiedon välityksestä kerrotaan lisää luvussa 5.6.



KUVIO 9. Kuvasarja kuvan ottamisesta ja lähettämisestä mobiilisovelluksessa



KUVIO 10. Vuokaavio kuvanottamisesta ja lähettämisestä

5.6 Paikkatieto

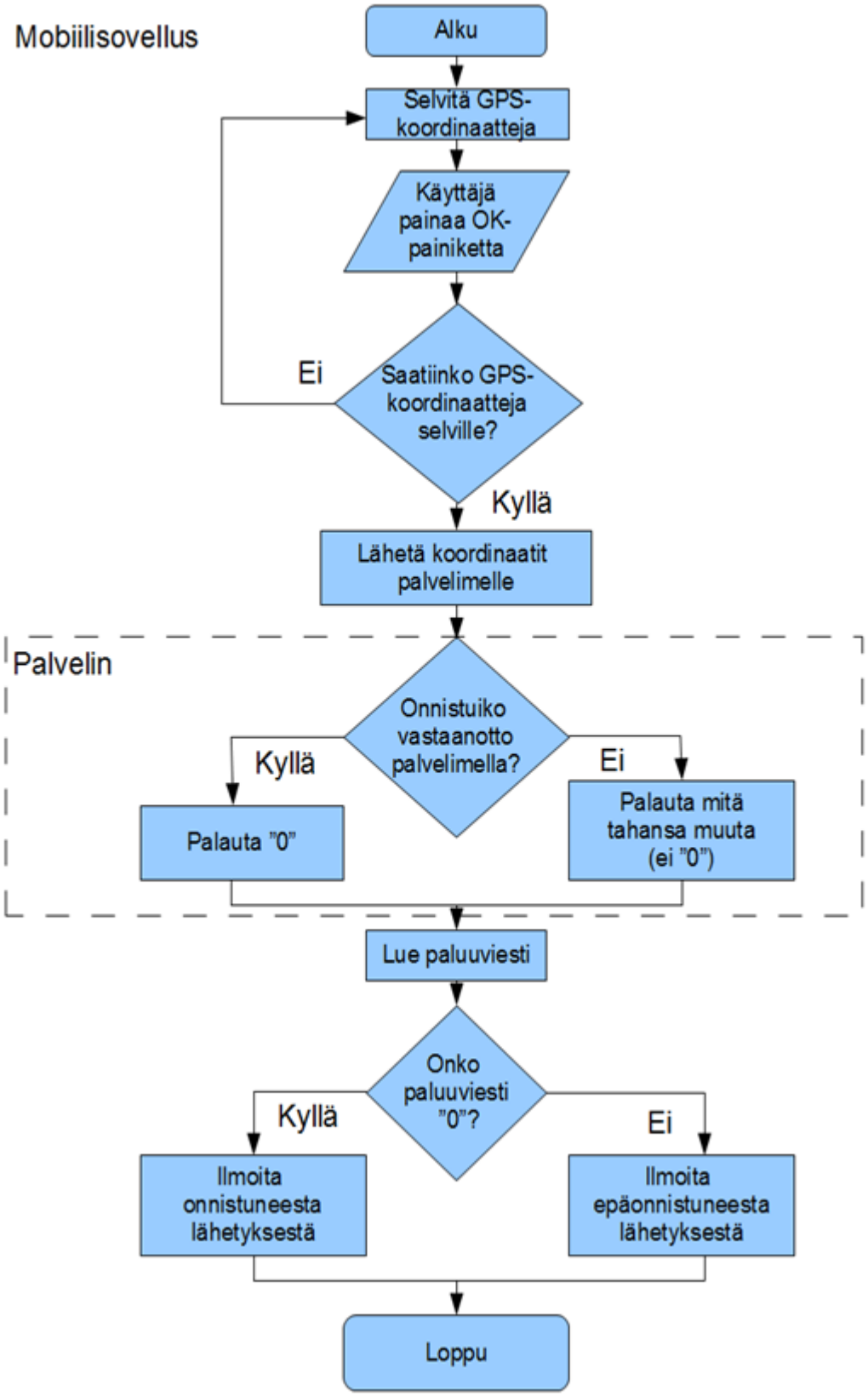
Paikkatiedon selvittäminen tapahtuu mobiililaitteen sisään rakennetun GPS-vastaanottimen avulla. GPS-vastaanottimeen päästään Javassa käsiksi Location API –rajapinnan avulla.

Paikkatietojen vastaanottaminen ja lähetys on toteutettu siten, että käyttäjä ei voi lähettää paikkatietoja, ennen kuin ne on saatu selvitettyä. Paikkatietojen selvittämiseen menee GPS-laitteen mallin ja ominaisuuksien sekä paikan mukaan muutamasta sekunnista jopa 10 minuuttiin. Rakennuksien sisätiloissa, tai paikoissa joissa GPS-laite ei saa suoraa näköyhteyttä satelliitteihin, ei GPS-laite saa välttämättä koordinaatteja selvitettyä ollenkaan.

Kun yhteys lopulta saadaan muodostettua, ilmoittaa ohjelma käyttäjälle siitä. Samalla sovellus alkaa esittää selvitettyjä pituus- ja leveyspiirikoordinaatteja sekä korkeutta kuvaavaa tietoa käyttöliittymän lomakkeella. Nämä koordinaatit esitetään desimaalimuodossa. Mikäli käyttäjä haluaa lähettää tiedot palvelimelle, on hänen valittava "Lähetä"-valinta. Mobiilisovelluksen tapa esittää koordinaatteja käyttäjälle on nähtävissä kuviossa 11.

Paikkatietojen lähettäminen mobiilisovelluksesta vastaanottavalle palvelimelle tapahtuu säikeen sisällä. Säikeen suorituksen aikana suoritetaan http-yhteydenotto palvelimelle ja välitetään koordinaatit POST-parametreinä. Palvelin palauttaa lähetyksen onnistumisen perusteella virhekoodin, jonka perusteella mobiilisovellus voi selvittää, onnistuiko lähetys vai ei. Paikkatietojen lähettämistä on kuvattu vuokaaviolla kuviossa 12.

KUVIO 11. Paikkatietojen selvittäminen



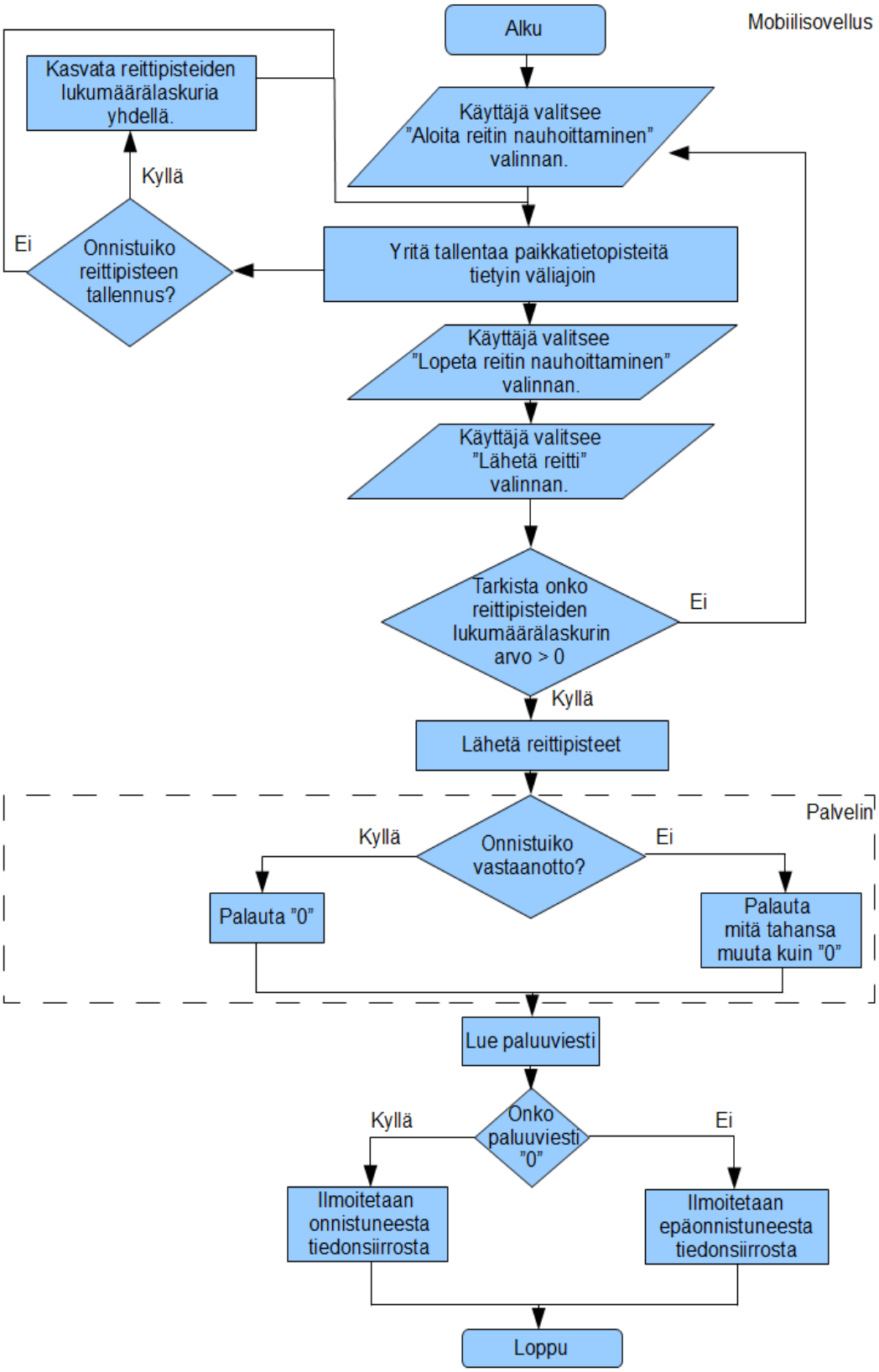
KUVIO 12. Paikkatietojen selvittämistä ja lähettämistä kuvaava vuokaavio

5.7 Reitti

Reitillä tarkoitetaan tässä tapauksessa useasta sijaintipisteestä koostuvaa joukkoa sijaintipisteitä, joista voidaan vastaanottavassa palvelinpäässä muodostaa yksi yhtenäinen reitti, jolla on alku ja loppupiste. Reittipisteiden selvittäminen tapahtuu samalla tavalla kuin paikkatietojen selvittämisessä, eli Location API –rajapintaa käyttäen.

Reitin nauhoittaminen toteutettiin siten, että käyttäjän käynnistäessä reitin nauhoittamisen ohjelma aloittaa reittipisteiden tallentamisen ja jatkaa sitä, kunnes käyttäjä pysäyttää nauhoittamisen. Nauhoittaminen tapahtuu säikeen avulla siten, että ThreadRecordRoute-luokka tallentaa tietyin väliajoin saamiaan GPS-koordinaatteja vektoriin. Tämä tietty väliaika on toteutettu pysäyttämällä säikeen suoritus Thread.sleep()-funktion avulla asetustiedostosta haettavan millisekuntimäärän ajaksi.

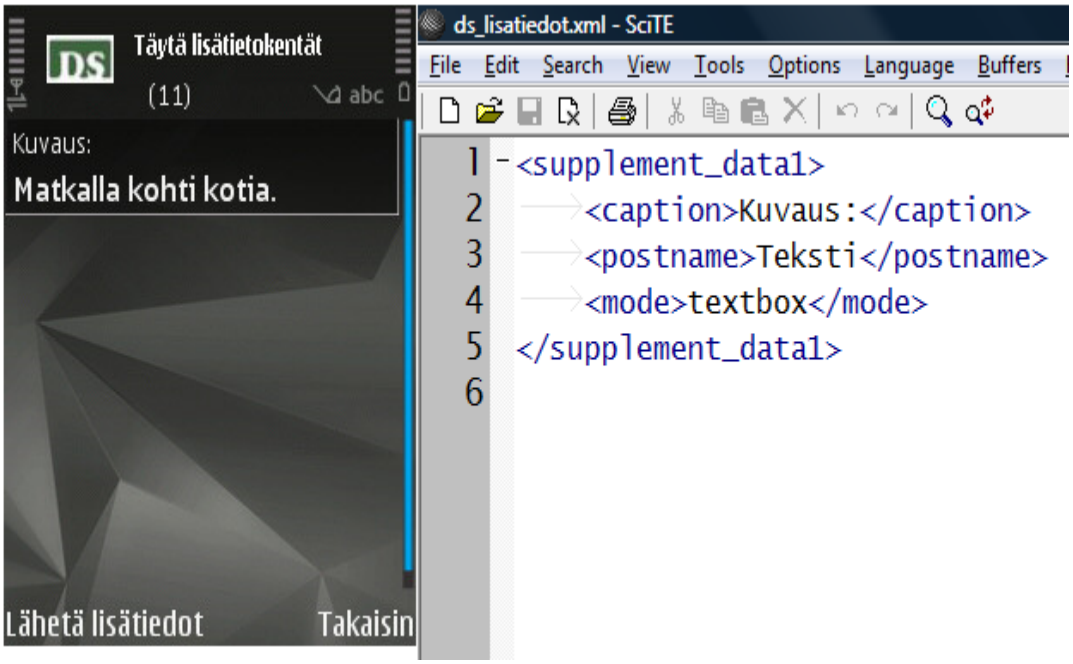
Ennen reitin lähettämistä sovellus tarkistaa vektoriin tallennettujen reittipisteiden lukumäärän, ja varmistaa samalla että niiden määrä on suurempi kuin nolla. Näin vältytään sellaisen reitin lähettämiseltä, missä ei ole ollenkaan reittipisteitä. Reitin lähettäminen tapahtuu omassa säikeessään, jossa käydään läpi vektori, johon reitin nauhoittamisessa tallennettiin GPS-koordinaatit, ja lähetetään ne sitten verkon yli vastaanottavalle palvelimelle. Reitin nauhoittamista ja lähettämistä on kuvattu kuvion 11 vuokaaviossa.



KUVIO 11. Reitinotto ja lähettäminen

5.8 Lähetettävät lisätiedot

Mobiilisovellus tukee myös lisätietojen lähettämistä, mikäli asetustiedostossa on yhteydelle sellainen asetus määrätty. Käytännössä nämä lisätiedot ovat ylimääräisiä tekstikenttiä ja niiden otsikoita, joiden sisältö lähetetään varsinaisen kuva tai paikkatiedon lähetyksen lisänä. Lisätietoja voidaan käyttää vastaanottavalla palvelimella esimerkiksi vastaanotetun kuvan kuvaustekstinä. Lisätiedot välitetään palvelimelle HTTP POST -tyyppisinä parametreina. Lisätietokentät luodaan XML-muotoisen lisätietoasetustekstin perusteella, jonka osoite on määritetty asetustiedostossa. Tiedostossa määritellään tekstikentän otsikko sekä postname eli nimi, jolla kuvausteksti lähetetään HTTP POST -tapahtumassa ja jonka avulla lähetetty tieto osataan vastaanottaa palvelinpäässä. Jatkokehittelymielessä XML-tiedostoon on lisätty myös mode-muuttuja, johon on laitettu tekstikenttäkontrollin tyyppi. Toistaiseksi mobiilisovellus tukee vain Textbox-tyyppisiä tekstikenttiä, mutta ohjelma olisi esimerkiksi laajennettavissa monivalintoihin, kuviin tai vaikkapa ohjeistusteksteihin. Kuviossa 12 on esitetty ruutukaappaus mobiilisovelluksesta, jossa on nähtävissä lisätietokenttä, sekä XML-muotoinen teksti, jonka perusteella lisätietokenttä on luotu.

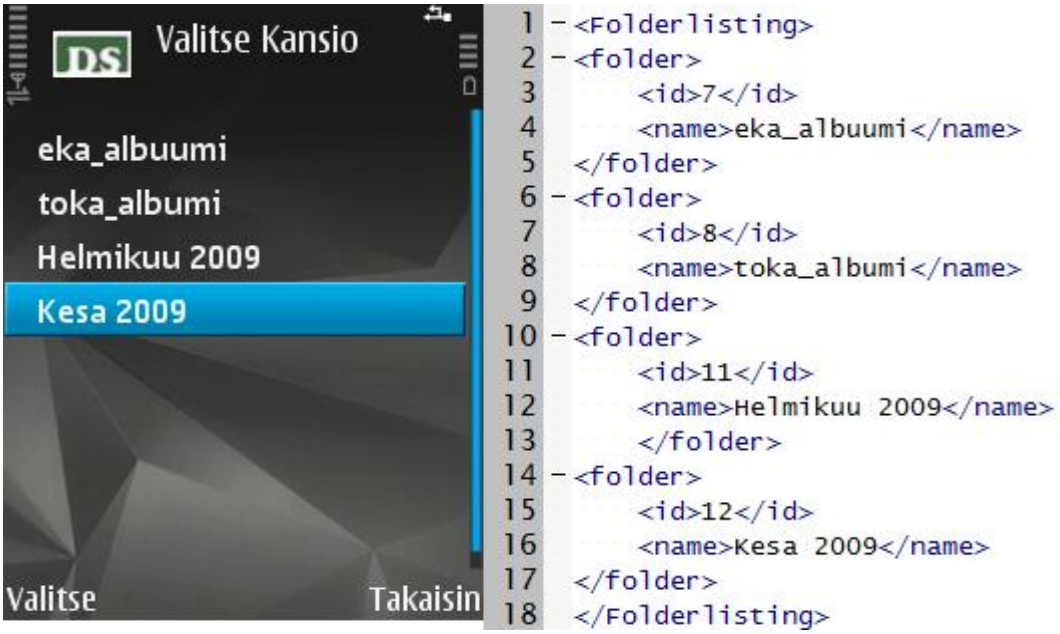


KUVIO 12. XML-muotoisen tiedoston perusteella luotu lisätietokenttä-tekstikenttä

5.9 Lähetettävät kansiotiedot

Idea kansiotietojen liittämisestä lähetyksiin tuli Coppermine Photo Gallerystä. Copperminessä käyttäjien kuvat ovat jaettavissa useisiin albumeihin, ja tällaisen ominaisuuden tukemista varten muokattiin mobiilisovellusta, siten että se voisi vastaanottaa listan yhteyden "albumeista" tai kansioista ja valita niistä, mihin esimerkiksi kuvatieto lähetettäisiin.

Tiedot palvelimella olevista kansioista tai albumeista välitetään XML-muotoisena tekstinä, jonka perusteella mobiilisovellus kykenee luomaan ne valinnoiksi listalle. Kansiolistan XML tiedosto ladataan mobiililaitteeseen osoitteesta, joka on määritetty asetustiedostossa. Valinnat saavat XML-muotoisen tekstin perusteella nimen, sekä tunnistenumeron. Lähetyksen yhteydessä välitetään valitun kansion tunnistenumero. Tämän tunnistenumeron perusteella lähetyksen vastaanottava palvelin voi luokitella vastaanotetun tiedoston esimerkiksi tiettyyn kansioon palvelimella tai ryhmään tietokannassa. Valitun kansiolistan tunnistenumero välitetään muiden lähetettävien tietojen joukossa lähetys vaiheessa. Kuviossa 13 on esillä ruutukaappaus mobiilisovelluksen kansionvalinnasta, sekä XML-muotoinen teksti jonka perusteella kansiovalintaruutu on luotu.



KUVIO 13. XML-muotoisen tiedoston perusteella luotu kansiolistaus.

5.10 Esimerkki kuva ja paikkatiedon vastaanottamisesta PHP:llä

Kuviossa 14 esitetty PHP-koodi vastaanottaa mobiilisovelluksen lähettämän kuvadatan, jonka jälkeen se luo siitä tiedoston ja tallentaa sen tietokantaan. Kuva tallennetaan kahdella eri resoluutiolla, ensin alkuperäisessä koossa 640 x 480 - resoluutiolla, ja sitten pienemmällä resoluutiolla 160 x 120. Pienempää kuvaa käytetään thumbnail-kuvana.

```php
<?PHP
//Vastaanotetaan kuva
if(($file=fopen($_FILES['img_data']['tmp_name'],"r")) == FALSE) die("3");  // palauttaa virhekoodin 3
if(($img_data=fread($file,filesize($_FILES['img_data']['tmp_name']))) == FALSE) die('4');   // palauttaa virhekoodin 4

//Luodaan kuva vastaanotetusta kuvadatasta.
$oSourceImage = imagecreatefromstring(base64_decode($img_data));
$nWidth = imagesx($oSourceImage);
$nHeight = imagesy($oSourceImage);
$nDestinationWidth = 640;
$nDestinationHeight = 480;
$oDestinationImage = imagecreatetruecolor($nDestinationWidth, $nDestinationHeight);
imagecopyresampled( $oDestinationImage, $oSourceImage, 0, 0, 0, 0, $nDestinationWidth, $nDestinationHeight, $nWidth, $nHeight);

ob_start();
imageJPEG($oDestinationImage);
$sBinaryThumbnail = ob_get_contents();
ob_end_clean();
$img_data=base64_encode($sBinaryThumbnail);

$oSourceImage = imagecreatefromstring(base64_decode($img_data));
$nWidth = imagesx($oSourceImage);
$nHeight = imagesy($oSourceImage);
$nDestinationWidth = 160; $nDestinationHeight = 120;
$oDestinationImage = imagecreatetruecolor($nDestinationWidth, $nDestinationHeight);
imagecopyresampled( $oDestinationImage, $oSourceImage, 0, 0, 0, 0, $nDestinationWidth, $nDestinationHeight, $nWidth, $nHeight);


ob_start();
imageJPEG($oDestinationImage);
$sBinaryThumbnail = ob_get_contents();
ob_end_clean();
$sBinaryThumbnail=base64_encode($sBinaryThumbnail);


$sql = "Insert Into DataSender(Text, Date, Data, Thumbnail) Values ('".$text."','".$date."','".$img_data."','".$sBinaryThumbnail."')";
//$yhteys muuttujassa on mysql_query funktion vaatimat kirjautumistunnukset palvelimelle, jätän ne näyttämättä tässä esimerkissä.
$query = mysql_query($sql,$yhteys);
$lastInsertedID = mysql_insert_id();
mysql_close($yhteys);
echo "0";
?>
```

KUVIO 14. Lähdekoodi lähetyksen vastaanottamisesta PHP-ohjelmointikielellä

5.11 Esimerkki kuvan ja paikkatiedon vastaanottamisesta C#:lla

Kuvioissa 15 ja 16 esitelty C# koodi vastaanottaa ensin URL- parametreista mobiilisovelluksesta lähetetyt leveysaste- ja pituusastekoordinaatit sekä lisätietotekstin. Tämän jälkeen vastaanotetaan lähetetty kuva, ja muutetaan se käsiteltävään muotoon ja tallennetaan se palvelimelle määrättyyn kansioon siten, että sille generoidaan päivämäärän ja kellonajan perusteella tiedostonimi.

```
1           //Vastaanotetaan paikkatietokoordinaatit ja lisätietoteksti
2           latitude = Request.Params["loc_latitude"];
3           longitude = Request.Params["loc_longitude"];
4           description = Request.Params["Description"];
5
6           //Vastaanotetaan kuvatiedostodata
7           HttpFileCollection uploadedFiles = Request.Files;
8           if (uploadedFiles.Count > 0)
9           {
10              HttpPostedFile userPostedFile = uploadedFiles[0];
11
12              const int BUFFER_SIZE = 255;
13              int nBytesRead = 0;
14              Byte[] Buffer = new Byte[BUFFER_SIZE];
15              StringBuilder strUploadedContent = new StringBuilder("");
16
17              //Luetaan kuvatiedoston data streamiksi
18              Stream theStream = userPostedFile.InputStream;
19              nBytesRead = theStream.Read(Buffer, 0, BUFFER_SIZE);
20
21              while (0 != nBytesRead)
22              {
23                  strUploadedContent.Append(
24                  Encoding.ASCII.GetString(Buffer, 0, nBytesRead));
25                  nBytesRead = theStream.Read(Buffer, 0, BUFFER_SIZE);
26              }
27              String StringImageDataBytes = Server.HtmlEncode(strUploadedContent.ToString());
28              //Tallennetaan kuva bitteinä byte array muotoon.
29              byte[] image = System.Convert.FromBase64String(StringImageDataBytes);
30              //Generoidaan kuvalle "timestamp" tiedostonimi.
31              String FileName = getNewFileName();
32              writeFile2Disk("C:/PalvelimenEsimerkkiKansio/Images/" + FileName + ".jpg", image);
33              //Kuvan tiedot ja sijainti tallennetaan tekstitiedostoon myöhempää esittämistä varten.
34              StreamWriter sw = new StreamWriter("C:/EsimerkkiKansio/DemoData.txt");
35             //Mode muuttujassa ilmoitetaan esitettävän asian tyyppi (1 = kuva, 2 = paikkatieto, 3 = reitti)
36             //"#" - merkkiä käytetään myöhemmin tekstitiedoston sisällön parsimisessa erottamaan tiedot toisistaan.
37              sw.Write(Mode + "#" + latitude + "#" + longitude + "#" + description + "#" + "Images/" + "#" + FileName + ".jpg");
38
39              sw.Close();
40          }
```

KUVIO 15. Lähdekoodi lähetyksen vastaanottamisesta C#-ohjelmointikielellä (1/2)

```
41
42      /* Funktio tallentaa byte array muodossa olevan kuvan parametrissä määrättyyn sijaintiin palvelimella.*/
43      public bool writeFile2Disk(string strFilePath, byte[] strBinaryFile)
44      {
45          bool writtenFile = true;
46
47          try
48          {
49              using (FileStream objFileSystem = new FileStream(strFilePath, FileMode.CreateNew))
50              {
51                  using (BinaryWriter objBinaryWriter = new BinaryWriter(objFileSystem))
52                  {
53                      objBinaryWriter.Write(strBinaryFile);
54                  }
55              }
56          }
57          catch (IOException e)
58          {
59              writtenFile = false;
60          }
61
62          return writtenFile;
63      }
64      /*Funktio generoi kellonajan ja päivämäärän perusteella nimen tiedostolle ja palauttaa sen.*/
65      public string getNewFileName()
66      {
67          DateTime dt = DateTime.Now;
68
69          String millisecond = dt.Millisecond.ToString();
70          String second = dt.Second.ToString();
71          String minute = dt.Minute.ToString();
72          String hour = dt.Hour.ToString();
73          String day = dt.Day.ToString();
74          String month = dt.Month.ToString();
75          String year = dt.Year.ToString();
76
77          String timestamp = ""+millisecond+""+second+""+minute+""+hour+""+day+""+month+""+year;
78
79          return timestamp;
80      }
81
```

KUVIO 16. Lähdekoodi lähetyksen vastaanottamisesta C#-ohjelmointikielellä (2/2)

5.12 Esimerkki XML-parseroinnista J2ME:ssä

XML muotoisen asetustiedoston parsimisessa mobiililaitteessa käytettiin apuna Xparse-J nimistä luokkaa. Xparse-J luokan avulla saadaan J2ME ympäristöön XML-tekstin parseroinnin mahdollistavat funktiot. Kuviossa 17 on esillä funktio, jossa XML-muotoinen teksti jäsennetään Xparse-J:tä käyttäen.

```
    /**
     * Parseroidaan xml ja luodaan sen sisällön perusteella kansiot image_folder vektoriin,
     * näitä kansioita käytetään myöhemmin kuvan lähetyksessä päättämään että
     * mihin kansioon mikäkin kuva lähetetään.
     * @param xml
     */
    public void parseImage_folder_vektori_xml(String xml)
    {
        //System.out.println(xml);
        this.vektori_image_folders = new Vector();

        if(xml.length()>5)
        {
            Node root = new Xparse().parse(xml);
            Node child;
            int i=1;
            do{
               int occur[] = {i};
               child = root.find("folder", occur);
               if(child!=null)
               {
                   int occur2[] = {1};
                   Node child_id = child.find("id", occur2);
                   Node child_name = child.find("name", occur2);

                   String ID =  child_id.getCharacters();
                   String Name = child_name.getCharacters();

                   this.vektori_image_folders.addElement(new Folder(ID,Name));
               }
               i++;
            }while (child!=null);
        }

    }
```

KUVIO 17. Lähdekoodi XML-muotoisen tiedoston jäsentämisestä mobiilisovelluksessa

6 KÄYTETYT OHJELMOINTIKIELET JA TYÖKALUT

Tässä luvussa selvitetään opinnäytetyön tekemisessä käytettyjä työkaluja, ohjelmia ja ohjelmointikieliä, sekä sitä mihin niitä on työssä käytetty.

Käytetyt ohjelmointikielet ja sovellusympäristöt:

- Java (J2ME)
- PHP
- MySQL(SQL)
- C#
- ASP.net
- XML

Java (J2ME) on Java-ohjelmointikielen mobiiliversio, joka on Java-ohjelmointikieleen verrattuna huomattavasti rajatumpi luokkakirjastoiltaan. Se on tarkoitettu suorituskyvyltään huomattavasti heikompien matkapuhelimille ja PDA-laitteille tehtävien sovellusten kehittämiseen. Tässä opinnäytetyössä mobiilisovellus koodattiin J2ME Javaa käyttäen.

PHP (PHP: Hypertext Preprocessor) on ohjelmointikieli dynaamisten palvelinsivujen ohjelmoimiseen. PHP-koodia voidaan kirjoittaa suoraan HTML- koodin sisään. PHP:n syntaksi on ottanut mallia Javasta, C:stä ja Perlistä. PHP:ssä on valtavasti valmiina olevia funktioita, mikä tekeekin PHP- sivujen tekemisestä suhteellisen nopeaa. PHP:ta käytettiin tässä opinnäytetyössä palvelinpuolen testisivujen toteuttamiseen.

MySQL on yksi suosituimmista SQL- tietokannan hallintajärjestelmistä. Se perustuu vapaaseen lähdekoodiin ja on saatavissa vapaalla GNU GPL -lisenssillä tai kaupallisella lisenssillä, jos GPL ei ole sopiva. MySQL:ssä ei ole itsessään minkäänlaista graafista käyttöliittymää, mutta siihen on saatavilla erikseen esimerkiksi phpMyAdmin-sovellus, jonka avulla SQL-serveriä voidaan hallinnoida web-selaimen avulla. MySQL-tietokantaa ja SQL-kieltä käytettiin tässä opinnäytetyössä

vastaanotettujen kuvien ja lisätietotekstien tallentamisessa.

C# eli "C Sharp" on Microsoftin kehittämänä ohjelmointikieli ASP.net- ohjelmointia varten. C#:n kehittämisen tavoitteena oli yhdistää C++ -kielen tehokkuus ja Javan helppokäyttöisyys. C# julkaistiin kesäkuussa 2000. Opinnäytetyössäni käytin C#-kieltä Visual Studion kanssa toteuttaessani ASP.net testisivun toiminnallisuutta.

Asp (Active Server Pages) on Microsoftin luoma ja kehittämä sovelluskehys. Sen avulla voidaan luoda dynaamisia webbsivustoja. ASP.net-sivuja tehdessä voidaan valita, käytetäänkö ohjelmointikielenä Visual Basicia vai C#- ohjelmointikieltä. ASP.net- sivut muistuttavat rakenteeltaan HTML-sivuja, mutta niissä on usein mukana tiettyjä ASP.netille tyypillisiä tageja. ASPin avulla tehtiin mobiilisovelluksen lähetyksiä vastaanottava testisivu.

XML eli eXtensible Markup Language on eräänlainen merkintäkieli ja standardi. XML suunniteltiin tiedon säilyttämistä ja siirtämistä varten, kun taas HTML (Hypertext Markup Language) suunniteltiin tiedon näyttämistä varten. (World Wide Web Consortium 2009.) Tässä opinnäytetyössä XML:ää käytettiin runsaasti mobiilisovelluksen ja palvelimen välisessä kommunikoinnissa.

Käytetyt työkalut ja liitännäiset:

- NetBeans IDE 6.5 & 6.7.1
- Wireless Toolkit 2.5.1
- Visual Studio 2005 & 2008
- Scite
- Reimers Map
- Xparse-J
- Kaappain (Screenshot)

NetBeans IDE (Integrated Development Environment) on ilmainen, avoimeen lähdekoodiin perustuva integroitu kehitysympäristö ohjelmistojen kehittäjille. Se on saatavilla Windowsille, Linuxille, Mac OS X:lle sekä Solarikselle. NetBeans sisältää työkaluja monenlaisten sovellusten kehittämiseen työpöytä-, yritys-, web- tai

mobiilisovellus käyttöön. NetBeans IDE tukee suurinta osaa yleisimmistä ohjelmointikielistä, ja se on laajennettavissa ilmaisilla liitännäisille jotka lisäävät siihen puuttuvia ominaisuuksia. Tässä opinnäytetyössä käytin NetBeans IDE - työkalua mobiilisovelluksen koodaamiseen.

Wireless Toolkit on eräänlainen liitännäinen, joka lisää mahdollisuuksia kehittää sovelluksia mobiililaitteille. Wireless Toolkit koostuu käännöstyökaluista, apuohjelmista ja emulaattorista. Wireless Toolkitiä voidaan käyttää NetBeansin kanssa yhdessä, jolloin NetBeansillä pystyy käyttämään Wireless Toolkitin emulaattoria ja ohjelmia mobiilisovelluksen kehittämiseen ja testaamiseen. Wireless Toolkitiä käytettiin NetBeansin kanssa mobiilisovelluksen kehittämisessä, ja virheiden jäljittämisessä sekä niiden korjaamisessa.

Visual Studio on Microsoftin kehittämä IDE, jossa on käytettävissä useita ohjelmointikieliä kuten Visual Basic, C++, C# ja J#. Sen avulla voi tehdä webbsivuja ASP.net- tekniikkaa hyödyntäen, sekä myös työpöytäsovelluksia. Visual Studion avulla on erityisen helppoa tehdä käyttöliittymiä, minkä ansiosta se onkin niin suosittu. Visual Studiota käytettiin ASP.net - testisivun toteuttamisessa.

Scintilla Text Editor (Scite) on kevyt ja näppärä tekstieditori. Sen ohjelmointikielikohtainen syntaksin värjäys tekee sen varsin näppäräksi ohjelmointityökaluksi pienille koodinpätkille. Tässä opinnäytetyössä käytin sitä PHP- tiedostojen koodaamiseen testisivuja varten.

Reimers Map on Jacob Reimerin tekemä ilmainen Open Source -liitännäinen. NET ympäristöön. Sen avulla on mahdollista päästä käsiksi Google Mapsiin C#-kieltä käyttämällä (Reimers, 2009.) Tässä opinnäytetyössä tätä liitännäistä käytettiin ASP.net-testisivuston toteuttamisessa.

Xparse-J on ilmainen XML-jäsennin J2ME-sovelluksille. Se on Java-versio JavaScriptin xParsesta. Xparse-J yrittää omien sanojensa mukaan olla pienin XML-parseri koko maailmassa. Se ei jäsennä aivan kaikenlaista XML:ää eikä tue aivan kaikkia XML:n ominaisuuksia, mutta sopii yksinkertaisen XML:n parseroimiseen (Classen 2009.) Käytin Xparse-J:tä opinnäytetyössäni jäsentämään mobiilisovellukseen palvelimelta ladattua XML-muotoista tekstiä.

Kaappain (Screenshot) on Antony Pranatan kehittämä ilmainen J2ME-sovellus jonka avulla on mahdollista ottaa kuvakaappauksia matkapuhelimessa ajetettavista ohjelmista (Pranata 2009.) Tätä ohjelmaa käytettiin ottamaan kuvia kännykän käyttöliittymästä tätä opinnäytetyötä varten.

7 TESTAUS

Mobiilisovellusta testattiin enimmäkseen emulaattoria käyttämällä. Kun virhetilanne huomattiin, käynnistettiin debuggeri ja tutkailtiin, mistä ongelma johtui ja miten sen pystyi korjaamaan. Sovellusta testattiin aina uuden ominaisuuden tai isomman koodinpätkän lisäämisen jälkeen.

Myöhemmin kun sovellus alkoi olla siinä kunnossa, että sitä pystyi testaamaan oikealla laitteella, siirryttiin ulos testaamaan sovelluksen toimintaa, koska työpaikan paksut betoniseinät estivät GPS-vastaanotinta vastaanottamasta signaalia. Pihalla pyöriessä kuvattiin ja lähetettiin runsaasti testiaineistoa testausta varten ja sitten palattiin takaisin toimistoon tutkailemaan vastaanottiko palvelin lähetettyä dataa. Testauksen kannalta asiaa ei helpottanut, se että toimisto sijaitsi rakennuksen ylimmässä kerroksessa, mutta tällainen pienimuotoinen väliaikaliikunta oli mukavaa vaihtelua koodailulle.

7.1 Testauksessa käytetyt mobiililaitteet

Testauksessa käytin työpaikalta lainaan saamaani Nokia 6110 Navigator puhelinta. Myöhemmin sovellusta tuli testattua omalla Nokia N95 8GB -puhelimella. Molemmat puhelimet sisälsivät sisäänrakennetun GPS-vastaanottimen, ja ainakin N95:ssä pystyi käyttämään myös tehostettuja GPS-palveluita, esimerkiksi Internet-yhteyden vaativaa A-GPS:ää (Assisted GPS). Lisäpalvelut ilmeisesti mahdollistaisivat nopeamman lukituksen GPS-satelliitteihin, mutta A-GPS:ää lukuun ottamatta en niitä käyttänyt, koska osa niistä on maksullisia. N95 8GB:llä testatessa käytössä oli DNA:n datapaketti kiinteällä kuukausimaksulla. Navigatorissa oli muistaakseni Elisan liittymä.

7.2 Testisivut

Mobiilisovelluksen toiminnan testaamista varten tehtiin kolme erilaista testisivua, joiden avulla pystyttiin testaamaan kaikkia mobiilisovelluksen toimintoja.
Jokaista luotua testisivua varten luotiin omat XML-muotoiset asetustiedostot, joiden perusteella mobiilisovellukseen luotiin yhteydet näiden testisivujen käyttämistä varten. Testisivuista kaksi ensimmäistä toteutettiin samalle Linux-palvelimelle. Kolmas ja viimeinen testisivu toteutettiin erilliselle Windows-palvelimelle. Seuraavissa luvuissa esitellään tarkemmin jokaista luotua testisivua.

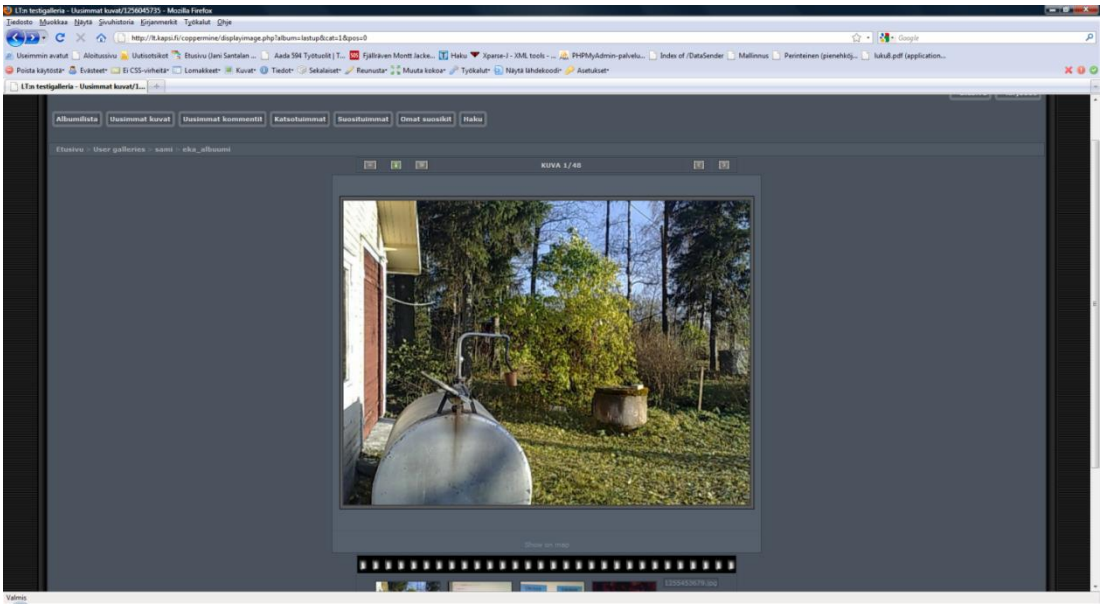
7.2.1 Coppermine – Google Maps

Ensimmäisessä testisivussa käytettiin hyväksi Coppermine Photo Gallery -kuvagalleriaa. Mobiilisovellus asetettiin lähettämään kuva- ja paikkatietoa vastaanottaville PHP-tiedostoille. PHP-tiedostot kirjoittivat vastaanotetut tiedot suoraan Copperminen omaan MySQL-tietokantaan. Tämän avulla kyettiin lisäämään lähetetty data suoraan kuvagalleriaan käyttäjien nähtäväksi. Sovelluksen lähettämien tietojen liittäminen Coppermine- kuvagalleriaan onnistui varsin hyvin juurikin Copperminen helpon muokattavuuden ja joustavuuden ansiosta.

Toisella PHP-tiedostolla saatiin listattua Copperminen käyttäjäkohtaiset albumilistaukset ja näiden tietojen avulla pystyttiin mobiilisovellusta kehittämään eteenpäin siten, että käyttäjä pystyi mobiilisovelluksessa valitsemaan albumin mihin kuvattu kuvadata lähetettiin.

Kolmannella PHP- tiedostolla saatiin mobiilisovelluksesta välitetyt paikkatiedot liitettyä Google Maps -karttaan. Linkki tähän Google Maps -karttaan pystyttiin lisäämään suoraan Copperminen tietokantaan kuvan "description" eli kuvaustekstikenttään siten, että kuvaustekstin sijaan kirjoitettiin siihen HTML-koodilla linkki karttatiedostoon, jossa oli samalla parametreina lähetetyt koordinaatit. Näiden tietojen perusteella karttatiedosto kykeni luomaan Google Maps -kartan ja esittämään sijainnin. Vastaanotettu kuva käsiteltiin siten, että palvelimelle tallennettiin vastaan-

otetusta kuvasta pienikokoinen thumbnail-esikatselukuva ja täysikokoinen kuva tallennettiin tietokantaan. Kuva tallennettiin kahdessa eri koossa sen takia, että Coppermine Photo Gallery -kuvagalleriasofta vaatii alkuperäisen kuvan lisäksi pienemmän esikatselukuvan kuvalistaustaan varten. Kuviossa 18 on esillä ruutukaappaus Coppermine Photo Gallery - kuvagalleriaa hyödyntävästä testisivustosta.

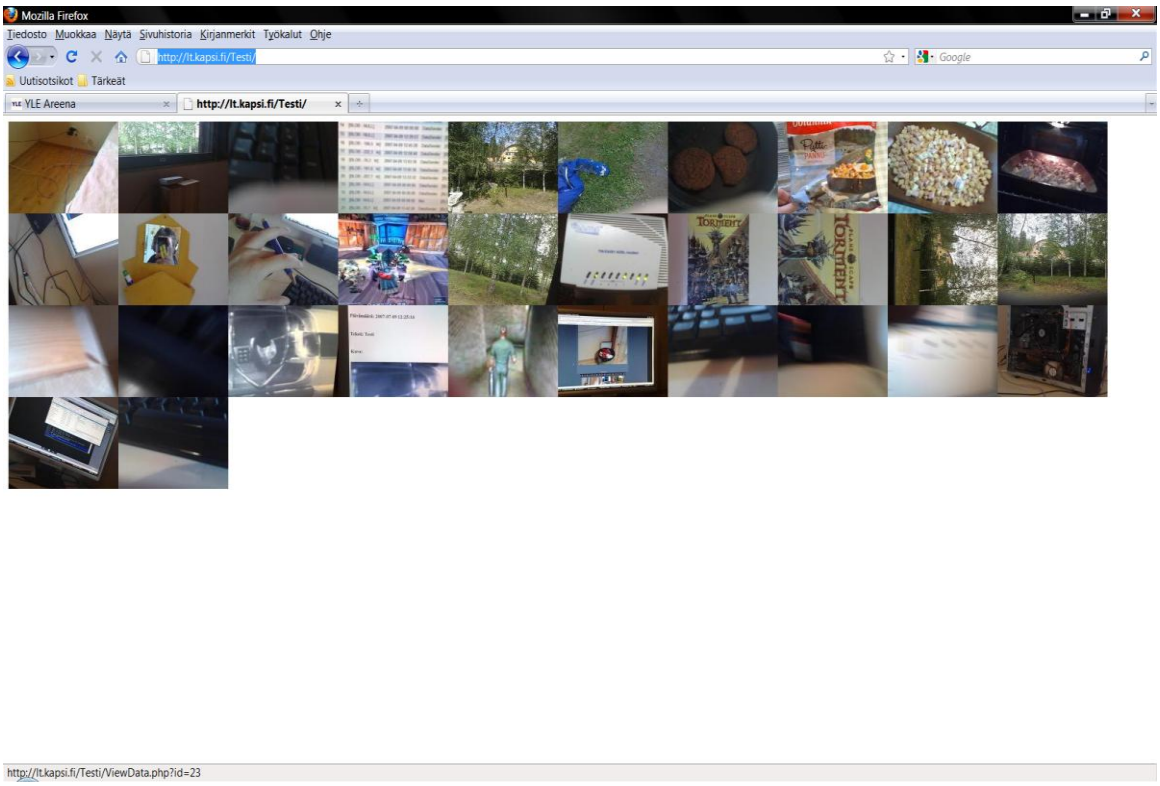


KUVIO 18. Coppermine-kuvagalleriaan lähetetty kuva

7.2.2 Kuvalista testisivu

Kuvalista oli yksinkertainen listaus lähetetyistä kuvista. Kuvalistaa varten tehtiin tietokantaan oma taulunsa, jossa säilytettiin tiedot kuvan nimestä, vastaanottopäivämäärästä ja kellonajasta sekä mahdollisesta lisätietotekstistä. Tauluun tallennettiin kuva kahteen kertaan.

Tietokannasta löytyvät kuvat tulostettiin sivulle PHP-tiedoston avulla siten, että ruudulle ilmestyi sivun latautuessa sarja pieniä esikatselukuvia, joita klikatessa pääsi näkemään alkuperäisen kuvan oikeassa koossa ja kuvaan liittyviä tietoja. Kuvalista toimi eräänlaisena nopeana testisivuna, josta voitiin nopeasti tarkastaa menikö kuva perille vai ei. Kuviossa 19 on nähtävissä ruutukaappaus testisivusta.



KUVIO 19. Testisivuston kuvalistaus

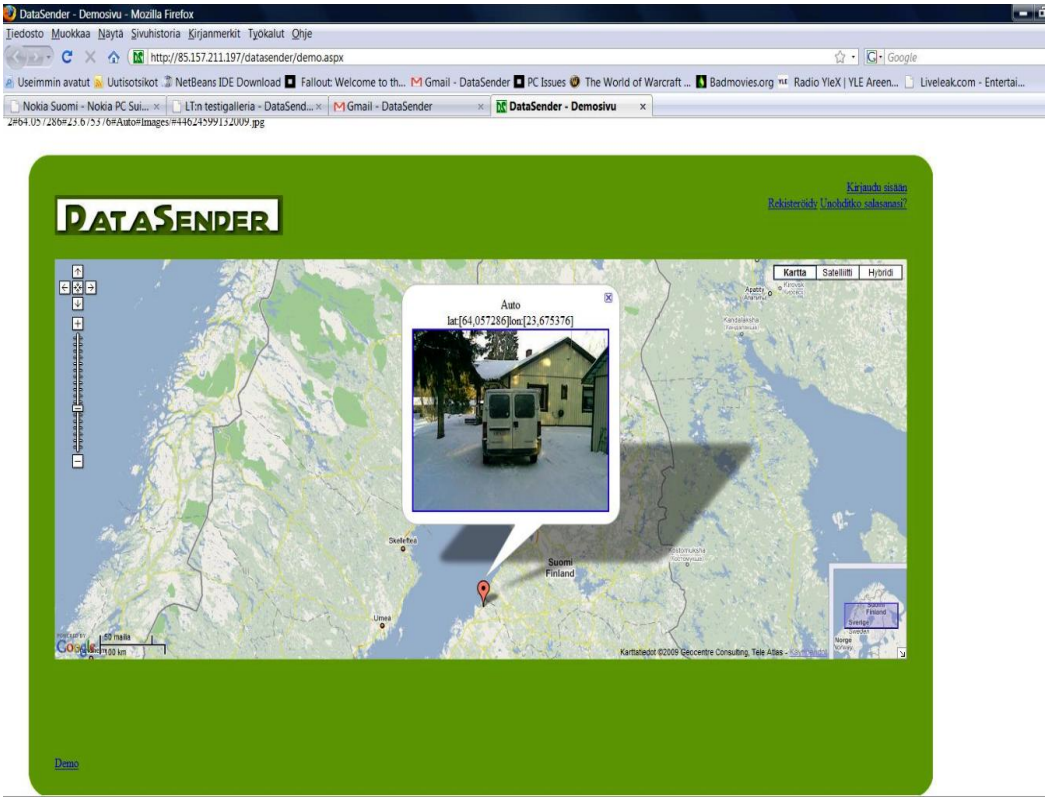
7.2.3 ASP.net Google Maps

Työnantajan toivomuksesta tehtiin testisivu myös ASP.net -tekniikalla, sillä työnantaja halusi perehdyttää työntekijää ASP.net -tekniikkaan jonka parissa on nyt myöhemmin tullut tehtyä enemmänkin töitä.

Alun perin oli suunnitelmissa tehdä tästä sivustosta suurempi kokonaisuus johon sisältyisi kirjautuminen, käyttäjäkohtaiset albumilistaukset ja kuvien hallinnointi, mutta myöhemmin päätimme rajata testisivun vain näyttämään viimeisimmän vastaanotetun kuvan tai paikkatiedon. Kokonaisen kuvagalleriasivuston toteuttaminen olisi sen verran työläs prosessi ja siinä olisi aihetta jo toiselle opinnäytetyölle.

Tehty testisivu vastaanottaa kuvia, paikkatietoja ja reittejä. Tässä testisivussa ei käytetty ollenkaan tietokantaa ja niinpä vastaanotetut kuvat tallennetaan palvelimelle kuvakansioon. Vastaanotetut paikkatiedot kirjoitetaan tekstitiedostoon. Demosivun latautuessa noudetaan tiedostoista tarvittavat tiedot ja esitetään ne

Google Maps -sovelluksen avulla käyttäjille. Tietojen esittämiseen Google Mapsissa käytettiin hyväksi Jacob Reimersin kehittämää Reimers.Map -rajapintaa, joka sisältää useita C#-kielen funktioita, joiden avulla päästään käsiksi Google Mapsiin. C#-kielen käyttäminen oli mielestäni mielekkäämpää, kuin Google Mapsin yhteydessä tavallisesti käytettävä JavaScript-kieli, koska koko testisivun toiminnallisuus voitiin koodata yhdellä ja samalla ohjelmointikielellä. Testisivusto on nähtävissä kuviossa 20.



KUVIO 20. ASP.net-tekniikalla toteutettu testisivu

8 YHTEENVETO

Opinnäytetyön aikana toteutettu mobiilisovellus onnistui mielestäni ihan hyvin. Se oli ensimmäinen suurempi sovellus, jonka toteutin itsenäisesti. Opinnäytetyön sovelluksen koodaaminen alkoi kesällä 2008, ja se prosessi kesti noin kolme kuukautta. Opinnäytetyön aikana tuli opittua valtavasti asioita ohjelmoinnista ja työkaluista. Esimerkiksi ennen opinnäytetyön tekemistä debuggerin käyttö oli tuttua vain käsitteenä, mutta nykyään en voisi edes ajatella ohjelmointia ilman sellaisen työkalun käyttämistä.

Työkiireiden takia opinnäytetyön kirjallisen osion kirjoittaminen viivästyi vajaalla vuodella, ja niinpä näin jälkeenpäin asiasta kirjoittaminen vaati vanhan koodin tutkailemista ja muistiinpanojen selailua. Samalla tämä tällainen vuoden takainen paluu opinnäytetyön pariin on antanut hieman perspektiiviä. Huomaan koodissa paljon asioita mihin ei aikaisemmin tullut niin kiinnitettyä huomiota, ja jotka näin jälkeenpäin toteuttaisin aivan eri tavalla. Kaikesta huolimatta, olen varsin tyytyväinen omaan mobiilisovellukseeni.

Jatkokehittelyn kannalta, nykyisessä mobiilisovelluksen versiossa olisi tarvetta varsinkin reittitiedon käsittelyn parantamiseen. Nykyinen versio ei huomioi GPS-laitteen ongelmatilanteita käytännössä ollenkaan. Ongelmatilanteilla tarkoitan tilanteita, joissa GPS-laite ei saa signaalia esimerkiksi tilanteissa, joissa käyttäjä kävelee perunakellariin tai vaikkapa rakennuksen sisälle. Lisäideoina jatkokehittelyyn olisi esimerkiksi puhelimeen ladattava paikkatiedon esikatselu palvelimen luoman Google Maps -kuvan perusteella, joka ladattaisiin puhelimeen ennen varsinaisen paikkatiedon lähetyksen tapahtumista, sekä monenlaiset muut ominaisuudet.

LÄHTEET

Aviation Explorer. 2009. Global Positioning System (GPS) - History, Facts and Pictures. Www-dokumentti. Saatavissa: http://www.aviationexplorer.com/Global_Positioning_System_GPS.html. Luettu: 8.12.2009.

Carnes, J. 2008. Lat/Lon Formats and Symbols. Www-dokumentti: Saatavissa: http://www.maptools.com/UsingLatLon/Formats.html. Luettu 13.joulukuuta.2009.

Classen, M. 2001. Xparse-J 1.0 User documentation. Www-dokumentti. Saatavissa: http://www.webreference.com/xml/tools/xparse-j.html. Luettu 8.12.2009.

Coppermine dev team. 2003. Coppermine Photo Gallery. Www-dokumentti. Saatavissa: http://coppermine-gallery.net/. Muutettu 2009. Luettu: 28.10.2009.

Giguère, E. 2002. J2ME Core Concepts. Www-dokumentti. Saatavissa: http://www.developer.com/article.php/1378971. Luettu 28.10.2009.

Google. 2009. Google Maps. Www-dokumentti. Saatavissa: http://maps.google.com/help/maps/tour/. Luettu 11.10.2009.

Koodikolme Oy. 2009. Koodikolme Oy:n kotisivut. Www-dokumentti. Saatavissa: http://www.koodikolme.fi/. Luettu 28.10.2009

kowoma.de. 2009. GPS explained. Www-dokumentti. Saatavissa: http://www.kowoma.de/en/gps/. Luettu: 6.12.2009.

Pranata, A. 2009. Screenshot for Symbian OS. Www-dokumentti. Saatavissa: http://www.antonypranata.com/screenshot. Luettu: 28.10.2009.

Reimers, J. 2009. Reimers Map. Www-dokumentti. Saatavissa: http://www.reimers.dk/ Luettu: 10.11.2009.

U.S. Department of Defense. 2007. News Release: DoD Permanently Discontinues Procurement Of Global Positioning System Selective Availability. Www-dokumentti.
Saatavissa: http://www.defense.gov/releases/release.aspx?releaseid=11335. Luettu: 10.12.2009.