Metropolia Ammattikorkeakoulu
Tietotekniikan koulutusohjelma

Mikko Ketonen

Mobiilisovellus paikkatietodatan tietokantaan tallentamista varten

Insinöörityö 7.12.2010

Ohjaava opettaja: yliopettaja Kirsti Äystö

Metropolia Ammattikorkeakoulu **Insinöörityön tiivistelmä**

Tekijä Otsikko Sivumäärä Aika	Mikko Ketonen Mobiilisovellus paikkatietodatan tietokantaan tallentamista varten 34 sivua 7.12.2010
Koulutusohjelma	tietotekniikka
Tutkinto	insinööri (AMK)
Ohjaava opettaja	yliopettaja Kirsti Äystö

Tässä insinöörityössä kehitettiin paikkatietoja keräävä ja tietokantaan tallentava mobiilisovellus osana Samfinder Oy:n pilottiprojektia. Projektin tavoite on antaa maansiirtoyrityksille työkalu työnseurantaa, niiden optimoimista ja työtuntien seurantaa varten. Työhön kuului myös itse tietokannan rakenteen suunnittelu.

Sovelluksen tuli kyetä lukemaan puhelimen Geolocation API:lta paikannustietoja ja tallentamaan ne SQL-tietokantaan.

Sovelluksen on tarkoitus olla käynnissä matkapuhelimessa noin 8–10 tuntia päivässä.

Sovellus suunniteltiin Omni Graffle Professional suuunnittelutyökalulla ja toteutettiin Aptana Studio 2:lla johon oli asennettu Nokian WRT-liitännäinen.

Työn tuloksena on toimiva ja tarpeeseen sopiva mobiilisovellus, jonka voi räätälöidä uudestaan tarvittaviin uusiin tarpeisiin. Lähtöajatus oli, että tehdään sovellus, jolla pystytään antamaan sovellus samoilla ominaisuuksilla valtaosalle Nokian puhelimista, jotka olivat jo toteutettu Androidille ja iPhonelle projektin aikaisemmissa vaiheissa.

Hakusanat	paikannus, seuranta, puhelin, SQL, JavaScript, tietokanta, ohjelmointi

Helsinki Metropolia University of Applied Sciences **Abstract**

Author Title Number of Pages Date	Mikko Ketonen Mobile application for saving location data into a database 34 (including appendices) 7.12.2010
Degree Programme	Information Technology
Degree	Bachelor of Engineering
Supervisor	Principal Lecturer Kirsti Äystö

In the thesis project an application for saving location data into a database was developed for Metropolia UAS, to be used in the Samfinder Oy pilot. The goal was to create a tool for optimizing the workflow of construction sites. The database structure design was also a part of this thesis project.

The UML diagrams were drawn with Omni Graffle Professional, and the code was written and emulated with the Aptana Studio 2 editor with Nokia WRT plugin installed. The code was written in the JavaScript programming language. The database structure was drawn with the Omni Graffle Professional diagram editor.

The developed application suits the need and fulfills its purpose. It is easily customizable for further re-implementations. The basic idea was to develop an application which can provide similar functionalities for Nokia's mobile phones as were earlier developed for iPhone and Android devices.

Keywords	localization, tracking, mobile phone, SQL, JavaScript, database, programming

Sisällys

Tiivistelmä
Abstract
Lyhenteet

1 Johdanto 6

2 Käytetyt tekniikat 7

3 Järjestelmän kuvaus 8

3.1 Vaatimukset 8
3.2 Järjestelmän hahmotus 9
3.3 Sovelluksen käyttöliittymä 11
3.4 Ohjelmalliset rajoitteet 12
3.5 Laitteistorajoitteet 13
3.6 Toimintojen suunnittelu 13
3.7 Käyttötapaukset 14

4 Sovelluksen tuottaminen puhelimelle 17

4.1 Suunnitelu- ja toteutusohjelmistot 17
4.2 JavaScriptin lisäominaisuudet WebRuntimessa 17
4.3 Pysyvän datan tallennus 19
4.4 Paikannuksen toteutus 21
4.5 Sovelluksen tietoturva 23
4.6 Tietokannan rakenne 24
4.7 Työn kulku 28

5 Yhteenveto 29

Lähteet 31

Liitteet

Liite 1: Paikannuksen käyttöönotto ja siinä käytettävät funktiot 32

Lyhenteet

AJAX	*Asynchronous JavaScript and XML*; Internet-sivuissa käytetty tekniikka. Tekniikalla luodaan dynaamisia web-sivuja, jotka siirtävät dataa asynkronisesti järjestelmän taustalla.
HTML	*HyperText Markup Language*; Internet-sivujen kuvauskieli. Kielestä on haarautunut useita eri variaatioita ensi version jälkeen, mutta tässä dokumentissa lyhenne on yleiskäsite kaikille variaatioille.
HTTP	*Hypertext Transfer Protocol*; hypertekstin siirtoprotokolla jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon.
JSON	*JavaScript Object Notation*; web-ohjelmointi kielissä käytetty yksinkertainen tiedonsiirtomuoto.
MD5	*Message-Digest algorithm 5*; mm. kryptografiassa käytetty sekoitus funktio. Yksisuuntainen algoritmi, jonka purkaminen hyvin vaikeaa. Tämän takia käytetään datan salaamiseen.
PHP	*PHP: Hypertext Preprocessor*; Web-palvelinympäritöjen ohjelmointikieli. Käytetään erityisesti dynaamisten Internet-sivujen ohjelmointiin.
SQL	*Structured Query Language*; relaatiotietokantojen yhteinen komentokieli. Kielestä löydettävissä pieniä eroja SQL-tietokantaohjelmistojen välillä.

1 Johdanto

Samfinder Oy, jolle tämä työ on tehty, tuottaa rakennusalan ja kiinteistöalan yrityksille sovelluksen, joka seuraa yritysten käytössä olevien ajoneuvojen paikkatietoja. Eri kohderyhmiä ovat maansiirtoalan, kiinteistöalan ja muiden vastaavien alojen esimiehet, joita kiinnostavat erityisesti työskentelyn kustannustehokkuus.

Tässä työssä suunniteltiin ja toteutettiin sovellus paikkatietojen ja työtapahtumien seuraamiseen ja niiden tallentamiseen SQL-tietokantaa. Sovellus kehitettiin ajettavaksi Nokian matkapuhelimilla, joissa on Nokian WRT 1.1 tai uudempi. Sovellus kehitettiin JavaScript-kielellä [2].

Sovellus luotiin Samfinder Oy:n ja useamman maansiirtoalan yrityksen yhteisenä pilottihankkeena. Pilottihankkeessa oli mukana useampi maansiirto- ja kiinteistöalan yritys. Hankkeen päätavoite on tarjota asiakkaille tietoja, joilla työn tekoa voidaan tehostaa ja saavuttaa sitä kautta kustannussäästöjä, sekä mahdollisesti tarjota tietojen avulla mahdollisuuden tarkempaan työn laskuttamiseen.

Pilottiin lähtivät mukaan Lemminkäinen Oy ja Karjaan KTK, joissa kuljettajien oli määrä käyttää pilottisovellusta erillisissä projekteissa. Palvelu on toiminnassa sen julkaisusta eteenpäin aina siihen asti kuin kysyntää riittää, ja sitä tulee ylläpitämään vähintään yksi vakinainen työntekijä.

2 Käytetyt tekniikat

JavaScriptin lisäksi työssä käytettiin jQuery JavaScript-kirjastoa, joka mahdollistaa huomattavasti yksinkertaisemman ja nopeamman JavaScript ohjelmoimisen [1]. jQueryn kanssa käytettiin Guarana-liitännäistä, joka mahdollistaa Nokian WRT:ssä erilaisten graafisten elementtien käytön nopeasti ja tehokkaasti [11].

HTTP-pyynnön toteuttamisessa käytettiin jQuery-kirjaston Ajax-toteutusta, joka on helppolukuisempi ja yksinkertaisemmin toteutettavissa oleva ratkaisu verrattuna HTTP-pyyntöön [8; 9; 14].

PHP:tä käytettiin palvelinpuolella käyttäjien todentamiseen sekä paikkatietojen vastaanottamiseen ja tietokantaan tallentamiseen [6; 7].

JSON:a käytettiin JavaScriptin ja PHP:n välisen tiedonsiirron muotona, jolloin molemman ohjelmointikielet pystyivät lukemaan samaa dataa vaivattomasti [4; 5].

3 Järjestelmän kuvaus

3.1 Vaatimukset

Toiminnalliset vaatimukset määrittelevät, mitä palveluja ohjelmiston on tarjottava, miten se reagoi käyttäjän syötteisiin ja miten se käyttäytyy tietyissä tilanteissa.

Sovelluksen perustarkoitus on hakea WRT:n Geolocation rajapinnasta saatava paikkatietodata ja tallentaa se Internetin välityksellä tietokantapalvelimelle. Sovelluksen on pystyttävä hakemaan rajapinnasta data, joka saattaa osittain vaihdella puhelinkohtaisesti.

Sovellus ei saa kadottaa paikkatietoja, vaikka Internet-yhteys ei aina olisikaan saatavilla.

Käyttäjän on kyettävä hahmottamaan sovelluksen näppäimet ja keskeisimmät toiminnan noin puolen metrin päästä ja kyettävä käyttämään sovellusta ilman päätelaitteen käteen ottamista.

Sovellus ei saa jähmettyä jonkin yhden toiminnan takia, vaan sen on kyettävä vastaamaan käyttäjän toimintoihin. Sovelluksen on myös kyettävä lukemaan jokainen kerta käyttäjän painallukset.

Koska sovellusta käyttävä puhelin on liikkeessä ja sovelluksen toiminta perustuu Internet-yhteyden ja paikkatietojen varaan, on todennäköistä, että jossain vaiheessa tapahtuu jotakin epäideaalista, johon järjestelmän on varauduttava parhaansa mukaan.

Sovellus suunniteltiin käytettäväksi Nokian kosketusnäyttöpuhelimissa joissa on vähintään kolmen tuuman kosketusnäyttö. Erityisesti huomiota vaativat puhelimien näyttöjen koot ja se, miten kaikki tarvittava tieto pystytään näyttämään selkeästi ja helppolukuisesti, kun käytettävien näyttöjen koko on rajallinen.

Puhelinten tekniikan kehittyessä laitteissa on huomattavia teknisiä eroja. Suurimmat erot sovelluksen kannalta ovat paikkatiedon saannin varmuus sekä paikkatietodatasta saatavat lisätiedot.

Ei-toiminnalliset vaatimukset määrittelevät rajoitukset toiminnallisille vaatimuksille. Ne kertovat, mitä ehtoja järjestelmän on täytettävä, jotta toiminnalliset vaatimukset voidaan toteuttaa.

Sovellus ei saa kadottaa paikkatietodataa, minkä vuoksi data olisi pidettävä muistissa, kunnes se on varmasti tallennettu etätietokantaan. Lisäksi sovelluksen on kyettävä reagoimaan Internet-yhteyden tilan muutoksiin, jotta se kykenee määrittelemään, missä vaiheessa paikkatiedot on tallennettava väliaikaisesti puhelimen muistiin vai pitääkö ne lähettää palvelimelle pysyvää tallentamista varten.

Jos jokin toiminta kestää noin sekuntia pidempään, on järjestelmän näytettävä jokin indikaattori käyttäjälle.

Jokainen paikkatietoa lähettävä Ajax-pyyntö on suoritettava asynkronisesti, jotta vaikka jokin näistä pyynnöistä kestäisikin jostain syystä tavallista pidempään, ei tämä kuitenkaan vaikuttaisi sovelluksen muuhun toimintaan.

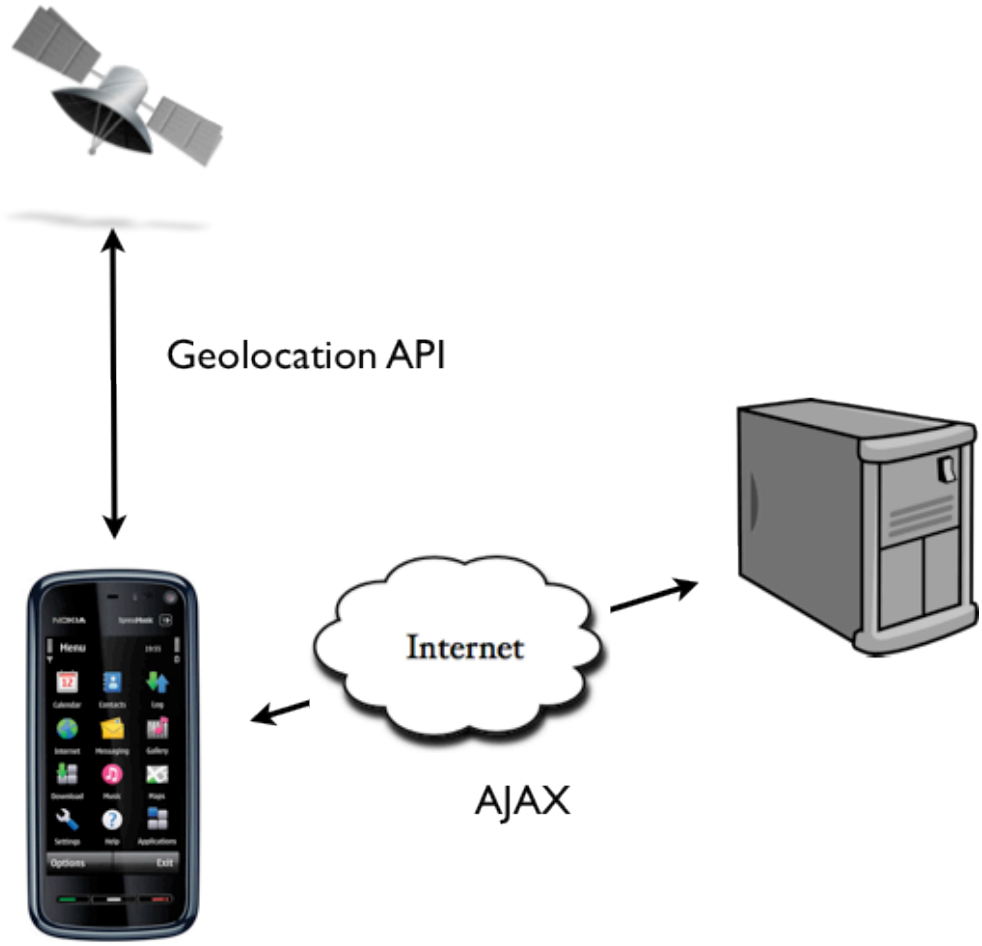
3.2 Järjestelmän hahmotus

Järjestelmään kuuluu kolme tärkeää järjestelmää, jotka ovat avainasemassa sovellusta ajettaessa. Nämä ovat GPS, itse sovellus ja tietokanta. Kuvassa 1 on havainnollistettu järjestelmän vaatimia järjestelmiä ja toiminnallisuuksia.

Jotta järjestelmä toimisi oikeaoppisesti, on satelliittiyhteyden toimittava lähes koko ajan. GPS:n toimintaa ei saa tulla muutamaa minuuttia pidempää viivettä missään tilanteessa. Tämän takia käytettävän puhelimen valinta on järjestelmän toiminnan

kannalta erityisen tärkeää, koska eri puhelinmallien GPS-sirujen luotettavuus ja tarkkuus vaihtelevat huomattavasti ja vaikuttavat sitä kautta sovelluksen toimintaan.

Sovellus sietää Internet-yhteyden väliaikaiset yhteysongelmat, koska järjestelmä kykenee tallentamaan paikkatiedot, joiden lähettäminen ei onnistunut puhelimen pysyvään muistiin siihen saakka, kunnes Internet-yhteys on taas saatavilla.



Kuva 1. Hahmotelma sovelluksen käyttämistä järjestelmistä.

3.3 Sovelluksen käyttöliittymä

Sovelluksen käyttöliittymä oli kyettävä pitämään mahdollisimman yksinkertaisena ja selkeänä, jotta se olisi mahdollisimman helppolukuinen. Lisäksi näppäinten oli oltava kooltaan ja väreiltään erottuvat eivätkä ne saaneet jäädä taustan hämärtämiksi. Lisäksi auringonvalon heikentäessä näytön näkyvyyttä käyttäjän on kuitenkin kyettävä hahmottamaan tärkeimmän näppäimet.

Sovelluksen käyttöliittymä sisältää kuvallisia näppäimiä ja listavalikoita. Kuvassa 2 olevat vihreä ja punainen näppäin on tarkoitettu tilanteisiin, joissa ajoneuvon kuljettaja ottaa tai luovuttaa kuorman. Edellä mainittujen näppäimien alla on kaksi alasvetovalikkoa, jotka ovat tarkoitettu kuorman laadun ja määrän valintaan.

Kuva 2. Kuva sovelluksen käyttöliittymästä, käyttäjän kirjautuneena järjestelmään.

3.4 Ohjelmalliset rajoitteet

Ohjelmallisia rajoitteita järjestelmän kehityksessä ei paljon ole. On yksi asia, joka on hyvä ottaa huomioon sovellusta kehittäessä. Tämä on se, että järjestelmä kykenee suorittamaan sovellusta myös taustalla, mikä tarkoittaa, että GPS-seuranta ja Internet-yhteys toimivat, vaikka laitetta, jossa sovellusta ajetaan, käytettäisiin välillä johonkin muuhunkin.

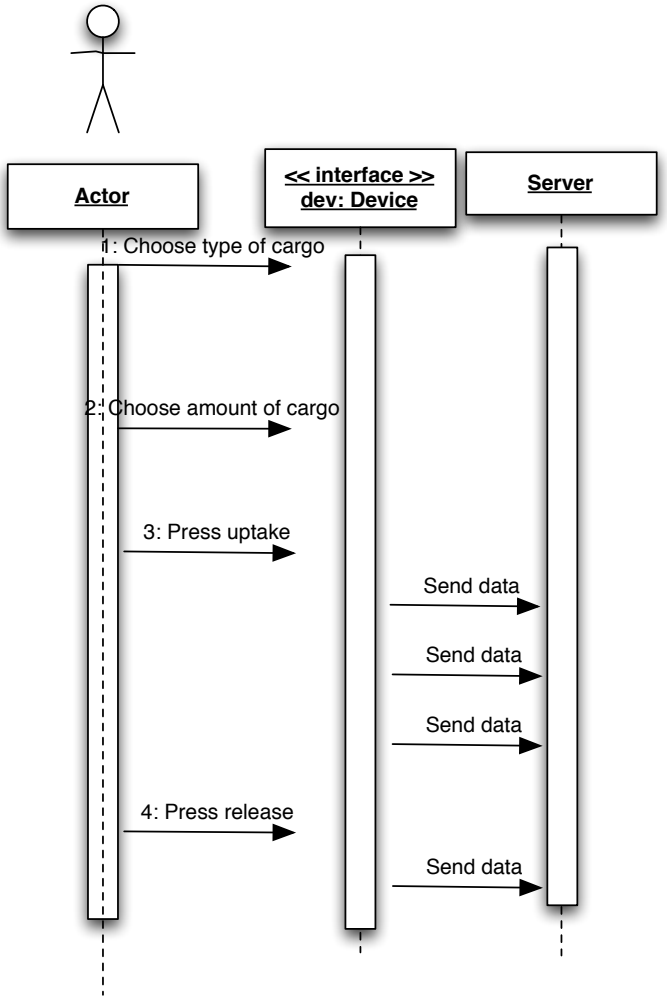
3.5 Laitteistorajoitteet

Koska sovellusta tullaan käyttämään erilaisissa puhelinmalleissa, myös laitteiston tehot vaihtelevat suhteessa paljon. Lisäksi eri puhelinmallien näyttökoot asettavat haasteita sovellukselle, jonka on oltava käyttökelpoinen myös pienemmillä näytöillä. Myös muut puhelinmallikohtaiset ominaisuudet, kuten GPS-piirin tarkkuus, vaihtelevat voimakkaasti. Puhelimet tulevat olemaan suurimman osan ajasta auton latureissa, joten akun riittävyys ei ole sovelluksen kannalta ongelma.

3.6 Toimintojen suunnittelu

Sovellus lähettää paikkatietoja tietyin väliajoin, kun laite on liikkunut tietyn määritellyn metrimäärän, sekä tilanteissa, joissa käyttäjä on painanut kuorman otto- tai luovutusnäppäintä. Ideaalitilanteessa käyttäjä valitsee ensin ottamansa kuorman laadun ja määrän ja painaa tämän jälkeen kuormanottonäppäintä. Tämän jälkeen sovellus lukee käyttäjän tekemän valinnat ja lähettää nämä tiedot paikkatiedon kanssa eteenpäin palvelimelle. Kun ajoneuvo liikkuu, sovellus laskee koko ajan kuljetun matkan pituutta. Kun edellisen lähetetyn paikkatiedon etäisyys on saavuttanut tietyn määritetyn metrimäärän, sovellus lähettää paikkatiedon ja informoi palvelimelle, onko autossa kuorma vai ei.

Kun kuljettaja luovuttaa aikaisemmin ottamansa kuorman, sovellus lähettää palvelimelle paikkatiedon ja tiedon siitä, että kuorma on luovutettu. Tämän jälkeen kierros alkaa alusta.



Kuva 3. Järjestelmien välinen tapahtumaketju ideaalitilanteessa.

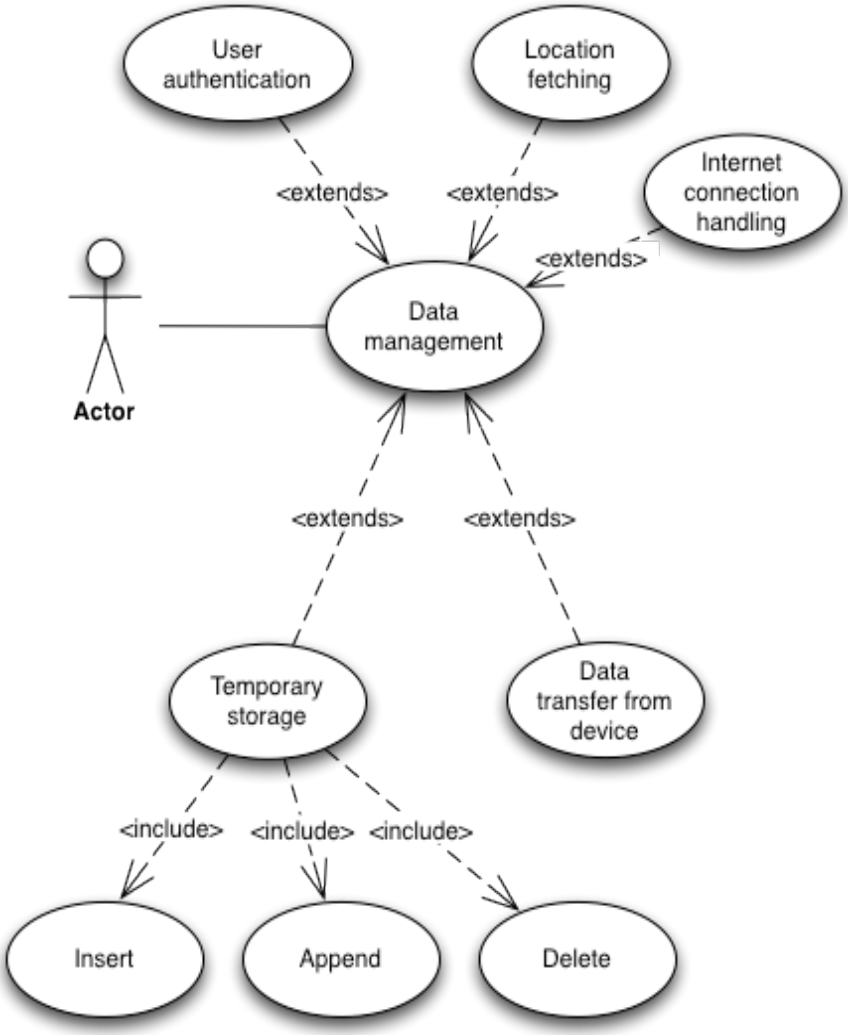
3.7 Käyttötapaukset

Käyttötapauksia ovat järjestelmän palveluidean ympärille suunnitellut toiminnot ja tavat käyttää järjestelmää. Käyttötapauskaaviossa esitetään järjestelmän toiminnot ulkopuolisen tarkkailijan näkökulmasta. Kaavioon on suositeltavaa kuvata pikemminkin vähän kuin paljon. Funktiot ja tapahtumaketjut toimintojen takana esitellään muissa kaavioissa. Käyttötapauskaavioita sekä muita UML-kaavioita piirretään sovelluksen rakenteen ja toiminnan havainnollistamiseksi. Kaikkia kaavioita ei suinkaan ole pakko luoda jokaisen sovelluksen kohdalla [12; 13].

Käyttötapaukset, joihin viitataan avainsanalla include, tarkoittavat tilanteita, joissa toiminnot toimivat osana sitä käyttötapausta, josta niihin viitataan. Usein tällaiset toiminnot ovat osa useampaa käyttötapausta, jolloin näihin viitataan niistä kaikista. Havainnollistaminen on myös yksi syy erottaa toiminto käyttötapauksesta joissain tilanteissa.

Käyttötapaukset, joissa ilmenee avainsana extend, laajentavat niiden osoittamaan käyttötapausta.

Ulkopuolisen toimijan eli aktorin ja järjestelmän vuorovaikutuksen aloittaa tässä sovelluksessa Actor. Actor on laitteen varsinainen käyttäjä, jossa sovellusta ajetaan. Lähtökohtaisesti Actor on järjestelmän ulkopuolinen käyttäjä tai toinen järjestelmä.



Kuva 4. Sovelluksen käyttötapaukset.

Aktorin kirjautuessa järjestelmään *User authentication* ottaa käyttäjän antamat tiedot talteen ja lähettää ne salattuna *Data transfer from devicen* avulla palvelimelle. Palvelin lähettää vastauksen kirjautumisyrityksestä ja *User authentication* reagoi sen mukaan. Tämä jälkeen aktorin tai järjestelmän itsensä toimesta Location fetching noutaa laitteen paikkatiedot. Tämän jälkeen *Internet connection handling* tarkistaa, onko Internet-yhteys saatavilla. Jos Internet-yhteys on saatavilla *Data transfer from device* lähettää paikkatiedot internetin yli tietokantaa. Jos Internet-yhteyttä ei ole saatavilla, *Temporary storage* tallentaa lähettämättömät paikkatiedot puhelimen pysyvään muistiin siksi aikaa, kunnes Internet-yhteys on taas saatavilla. Kuva 4 hahmottaa näitä eri järjestelmän toimintoja.

4 Sovelluksen tuottaminen puhelimelle

4.1 Suunnitelu- ja toteutusohjelmistot

Sovelluksen UML-kaaviot ja tietokantarakenteet luotiin Omni Graffle-kaavionluontisovelluksella. Itse ohjelman koodi kirjoitettiin ja emuloitiin Aptana Studio 2:lla, johon oli asennettu Nokian WRT-liitännäinen.

Aptana Studio 2:n ja Nokian WRT-liitännäisen pystyy asentamaan kaikkiin yleisimpiin Windows-, Linux- tai Mac-järjestelmiin. Tämä työkalu mahdollista muun muassa puhelimen emuloimisen tietokoneella sekä sovelluksen pakkaamisen .wgz muotoon, joka pystytään asentamaan kohdelaitteeseen monella eri tavalla.

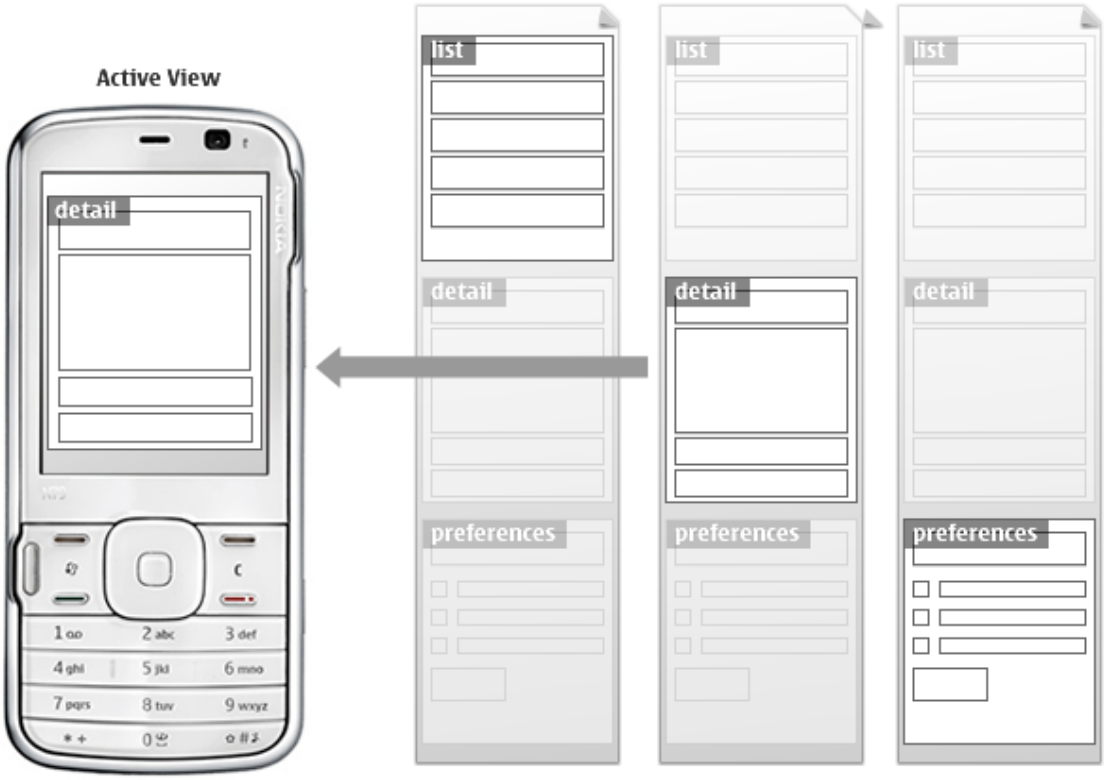
Vaikka JavaScript on ohjelmointikielenä varsin vikasietoinen, haluttiin kuitenkin taata, että ohjelma ei lopettaisi toimintaansa vaikka jokin virhetilanne syntyisikin. Yleisimmät ohjelmointivirheet johtuvat määrittelemättömistä muuttujista ja niiden ennenaikaisista käyttöyrityksistä. Koska kyse on web-ohjelmointikielestä, mahdollisia virhetilanteita voidaan yrittää löytää esimerkiksi Firefox-selaimen Firebug-liitännäisellä.

4.2 JavaScriptin lisäominaisuudet WebRuntimessa

Koska WRT-sovelluksissa koko ohjelman toimintaa ohjaa JavaScript koodi, eroaa se joissain tapauksissa web-ohjelmoinnissa käytetyistä JavaScript-tekniikoista, joissa JavaScript toimii vain osana sivua. JavaScriptin rooli WRT ohjelmoinnissa on niin paljon keskeisempi, että uusien ominaisuuksien ja toiminnallisuuksien tuominen JavaScriptiin on ollut perusteltua [3; 10].

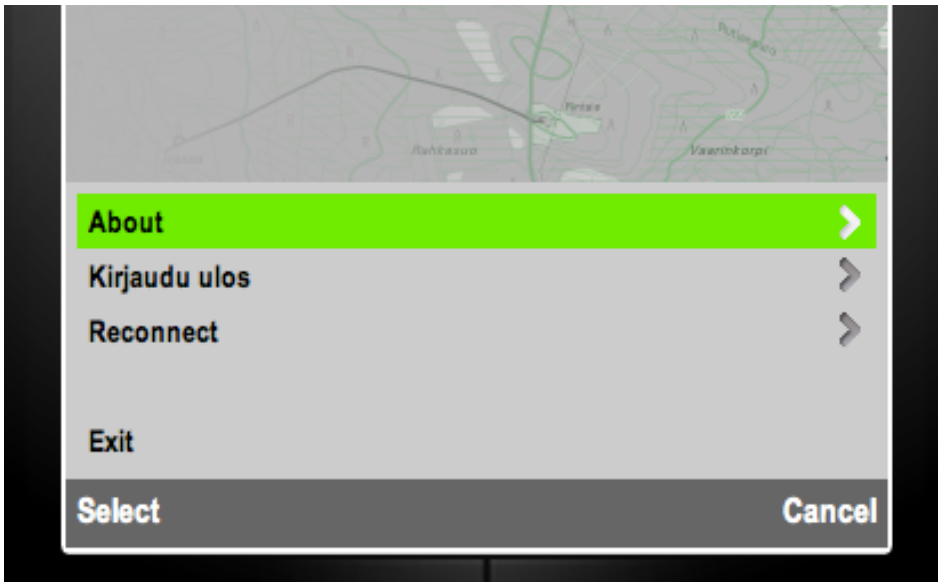
WRT-ohjelmoinnissa on suositeltavaa käyttää eri näkymissä siirtymiseen erillisiä div-tageihin sijoitettuja sisältöjä, joita hallitaan JavaScriptillä. Tällöin koko sovellus on mahdollista toteuttaa vain yhdellä HTML-sivulla, joka sisältää kaikki tarvittavat näkymät. Kuten kuvasta 5 käy ilmi, tämä toteutus mahdollistaa useamman erilaisen

näkymän käytön samassa sovelluksessa ilman ylimääräisiä sivun uudelleenlatauksia. Tämä on mahdollista joko div-tageilla tai vaihtoehtoisesti suoraan JavaScriptillä [3; 10].



Kuva 5. Hahmotelma WRT:n käyttämästä näkymien vaihdosta (10)

Lisäksi puhelimen valikkoihin sijoitettavat komennot mahdollistavat harvemmin käytettävien komentojen, kuten järjestelmästä uloskirjautumisen sijoittamisen siten, että se ei häiritse sovelluksen varsinaista käyttöä. Koska päätelaitteiden näytöt ovat kohtuullisen pieniä, on tärkeää, että tätä ominaisuutta käytetään hyväksi. Vaikka sovelluksen käyttäjä joutuukin näitä valikkonäppäimiä tarvittaessa painamaankin muutaman näppäimen ylimääräistä, valjastaa se kuitenkin näytön itse tärkeämmille ominaisuuksille ja toiminnoille. Kuva 6 hahmottaa valikkorakennetta.



Kuva 6. Esimerkki valikkonäppäimiin sijoitetuista toiminnosta

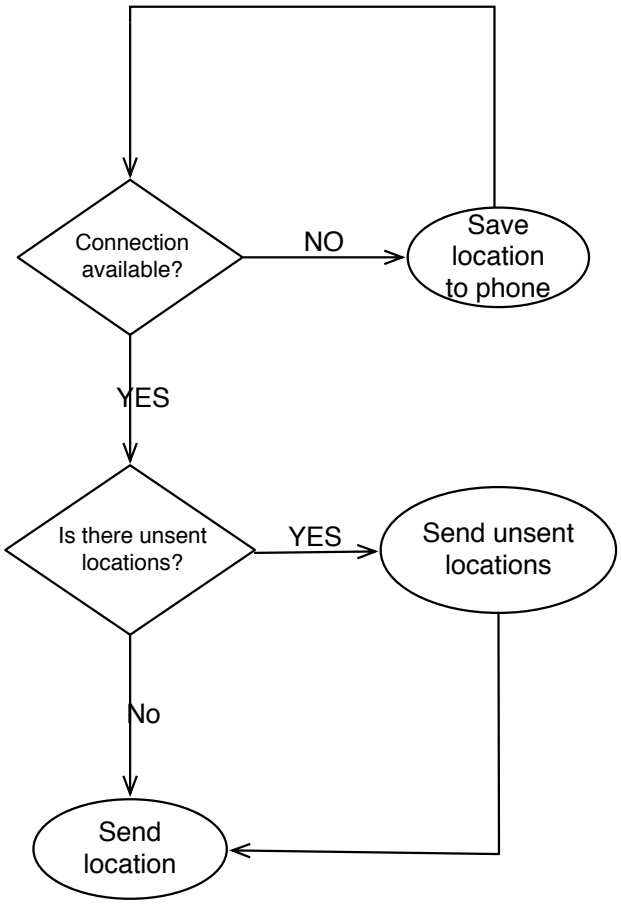
4.3 Pysyvän datan tallennus

Nokia WRT, mahdollistaa pysyvän datan tallentamisen puhelimen muistiin JavaScriptillä. Tämä data pysyy puhelimessa niin kauan kuin sovellus on puhelimessa asennettuna [3; 10]. Tämä mahdollistaa erilaisten tietojen tallentamisen ja uudelleenkäyttämisen tilanteissa, joissa puhelin olisi sammutettunakin.

Projektin sovelluksessa tästä ominaisuudesta oli erityisesti hyötyä esimerkiksi silloin, kun ei haluttu, että sovelluksen käyttäjän tarvitsisi joka kerta sovellusta käynnistäessä kirjautua sisään palveluun vaan pystyisi käyttämään sovellusta välittömästi. Lisäksi erikoistilanne, joissa Internet-yhteys ei ole saatavilla, tämä ominaisuus on ehdoton, koska silloin lähettämättömät paikkatiedot pystytään tallentamaan väliaikaisesti puhelimen muistiin myöhempää lähetystä varten.

Sovellus tarkistaa ennen jokaista paikkatiedon lähetystä Internet-yhteyden tilan. Jos Internet-yhteys on saatavilla ja käyttäjä on hyväksynyt puhelimen pyynnön Internet yhteyden luomiseen, lähetään paikkatieto. Muissa tapauksissa järjestelmä tallentaa paikkatiedot väliaikaisesti puhelimen muistiin, kunnes Internet-yhteys on jälleen saatavilla, jolloin aikaisemmat lähettämättömät paikkatiedot lähetetään ennen uusinta

paikkatietoa. Tällä tavoin paikkatiedot pysyvät järjestyksessä eikä uusin paikkatieto lähde palvelimelle ennen kuin aikaisemmat paikkatiedot on lähetty onnistuneesti. Kuva 7 hahmottaa tätä prosessia.



Kuva 7. Intenet-yhteyden tarkistus

4.4 Paikannuksen toteutus

WRT määrittelee Geolocation-objektin, jonka avulla sovellus pääsee käsiksi laitteen mahdollisiin paikkatietoihin. Kolme keskeisintä toimintoa Geolocation-objektissa ovat funktiot *watchPosition*, *getCurrentPosition* ja *clearWatch*. Näiden avulla pystytään toteuttamaan sovelluksen paikannustoiminnot. Kun Geolocation-objekti on luotu, pystytään kutsumaan *watchPosition*-funktiota, jonka tarkoitus on seurata laitteessa tapahtuneiden paikkatietojen muutoksia ja kutsua muutoksen tapahtuessa käyttäjän määrittelemää funktiota [10; Liite 1].

WatchPosition-funktio saa kolme parametria, jotka ovat kutsuttavan funktion nimi, virhetilanteessa kutsuttavan funktion nimi ja asetukset. Kun laitteen paikkatiedot ovat muuttuneet ja *watchPosition*-funktio kutsuu määritettyä funktiota, se antaa Position-objektin kutsuttavalle funktiolle. *Position*-objekti sisältää paikkatietodatan. Paikkatietodatasta ainoastaan latitude- ja longitude -muuttujat annetaan varmasti. Kaikki muu data on optionaalista ja annetaan vain silloin, kun ne on saatavilla. Tämän takia sovelluksen on reagoitava tilanteisiin, joissa jotain tarvittavaa tietoa ei olekaan saatavilla [10; Liite 1]. Kuva 9 havainnoillistaa *Position*-objektia.

WatchPosition-funktioon annettavat asetukset ovat *PositionOptions*-objektissa (Kuva 8). Tässä objektissa määriteltävä *enableHighAccuracy* asettaa ehdon, saako paikkatiedon tarkkuus olla löyhempi vai ei. Tämän määrittely ei välttämättä kuitenkaan takaa tarkempaa paikkatietoa, koska jotkin puhelinmallit eivät kuitenkaan pysty tarkempaa paikkatietoarviota antamaan, vaikka sovellus niin haluaisikin. *Timeout*-parametrilla määritellään kuinka monta millisekuntia järjestelmällä saa kestää paikkatiedon saanti. Jos määritelty aikaraja menee umpeen, kutsutaan virhetilanteissa kutsuttavaa funktiota, johon annetaan parametrina *PositonError*-objekti [10; Liite 1].

```
PositionOptions
{
    enableHighAccuracy : Boolean,
    timeout : Number,
    maximumAge : Number
}
```

Kuva 8. Kuvaus PositionOption objektista (10)

Kun kuljettaja painaa kuorman otto- tai luovutusnäppäintä sovelluksessa, käytetään edellisen watchPosition funktiosta saatua paikkatietoa. *GetCurrentPosition*-funktiota ei tässä sovelluksessa tarvita koska tarvittavat paikkatiedot saadaan *watchPosition*-funktiosta.

Kun paikkatietojen lähetys halutaan lopettaa, kutsutaan *clearWatch*-funktiota, johon annetaan parametriksi aikaisemmin luotu *Position*-objekti.

```
Position
{
    coords : {
        latitude : Number,
        longitude : Number,
        altitude : Number,
        accuracy : Number,
        altitudeAccuracy : Number,
        heading : Number,
        speed : Number
    },
    timestamp : Date
}
```

Kuva 9. Kuvaus Position objektista (10)

4.5 Sovelluksen tietoturva

Koska kaikki data lähetetään Internetin välityksellä, on datan lukeminen sovelluksen ulkopuolisille tahoille mahdollista, jos dataa ei millään tavalla salata.

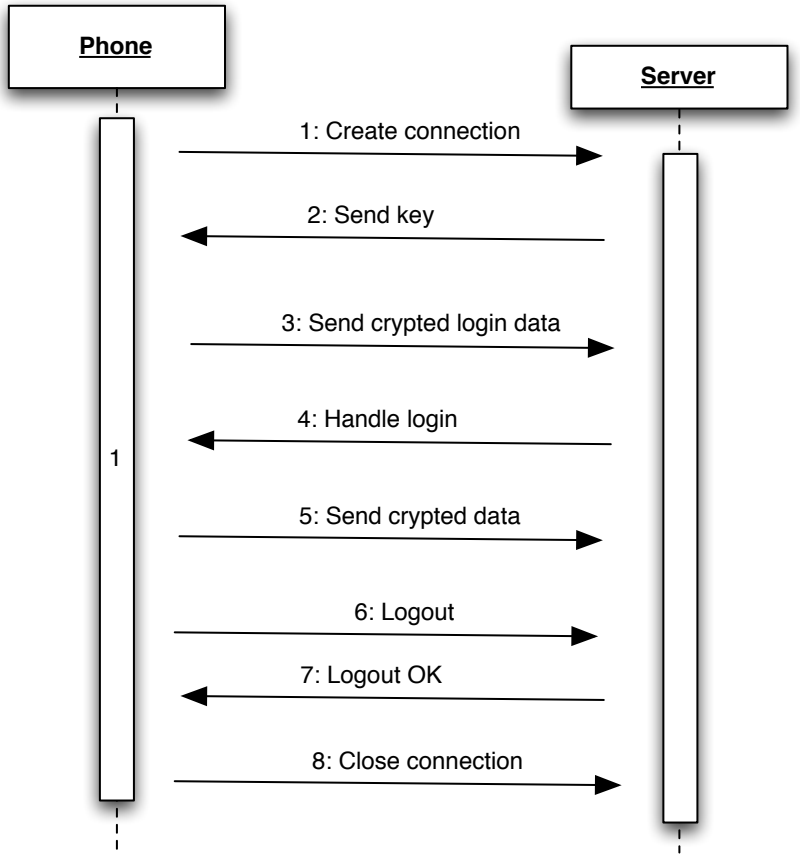
Sovelluksen ja palvelimen välillä kulkeva data päätettiin salata md5-salauksella ja serverin generoimalla satunnaisella avaimella, joka alustetaan jokaisen yhteyden alussa.

Kun yhteys luodaan, PHP-skripti generoi satunnaisluvun, jossa on vähintään neljä numeroa. Tämän jälkeen generoitu avain tallennetaan SESSION-muuttujaan ja lähetetään asiakassovellukselle. Ennen kuin asiakassovellus lähettää datan Internetin välityksellä palvelimelle, se yhdistää salattavaan dataan saamansa avainluvun ja generoi tämän jälkeen tästä md5-merkkijonon.

Kun palvelin lukee salatun viestin, se tietää, mikä avain on ollut kyseessä. Avaimen tarkistamisen lisäksi palvelin tarkistaa, että vastauksen SESSION id on sama kuin se, missä itse salausavain luotiin. Tämä varmistaa, että salattua avainta ei voida käyttää toisen yhteyden aikana.

Kaikkea dataa ei ole tarpeellista salata. Paikkatiedot ja näiden aikaleimat voivat olla puhtaassa tekstimuodossa, kunhan datan lähettämän asiakkaan henkilökohtaisia tietoja, kuten salasanaa, ei ulkopuolinen voi lukea.

Salausmenetelmä mahdollistaa sen, että vaikka järjestelmän ulkopuolinen taho pystyisikin lukemaan osan siirrettävästä datasta, kuten salausavaimen, ei siitä ole kuitenkaan vahinkoa. Yksittäisellä avaimella ei tee mitään, jos ei tiedä, mitä muuta dataa md5 salaukseen on sisällytetty. Kuva 10 hahmottaa asiakkaan ja palvelimen välistä yhteyttä.



Kuva 10. Yhteyden avaaminen, datan siirtäminen ja yhteyden sulkeminen.

4.6 Tietokannan rakenne

4.6.1 Tietokannan tarkoitus

Tiedot tallennetaan tietokannassa tauluihin, joissa tiedot ovat yksittäisinä yksilöllisinä ja siten tunnistettavissa olevina riveinä eli tietueina. Tiedoille suunnitellaan annetun yhtenäisen joukon piirteitä mahdollisimman hyvin vastaava rakenne [15]. Yleensä tauluissa on

- vähintään yksi tietueen yksilöivä tieto
- tieto joka sitoo (yhteys) tietueen tauluhierarkiassa ylempänä olevaan tai varsinaista dataa täydentävään tietueeseen
- varsinainen data, kuten paikkatieto sekä

- ajankohta, jolloin paikkatieto on saatu, tallennettu, viimeksi päivitetty tai mikä tahansa vastaava.

Järjestelmän relaatiotietokannan tiedot on tallennettu MySql-tietokantaan, joita ei kenenkään kannata lähteä muokkaamaan, vaan niiden sisältöä käsitellään asiaan tarkoitetuilla työkaluilla.

4.6.2 Palvelun vaatimuksen tietokannan suhteen

Palvelussa halutaan tarkastella kohdelaitteista saatuja paikkatietoja sekä niiden pohjalta laskettuja eri tietoja. Tärkeintä on tallentaa järjestelmästä saatava paikkatietodata. Tarpeen on myös tietää, mistä laitteesta ja milloin paikkatieto on saatu. Myös muut paikannustiedot, kuten paikkatiedon tarkkuus, ovat hyödyllisiä, jos ne vain ovat saatavilla. Kun edellä mainitut tiedot ovat saatavilla, on näistä tiedoista mahdollista luoda lähes reaaliaikaisen seurannan ajoneuvoihin, minkä ansiosta selkeän kuvan saaminen kokonaisista projekteista helpottuu huomattavasti.

Sovelluksen käyttäjätunnusten on myös sijaittava tietokannassa. Käyttäjällä on oltava käyttäjätunnus ja salasana, jotta järjestelmään kirjautuminen on mahdollista. Ilman näitä tietoja käyttäjää ei voida tunnistaa. Näiden tietojen avulla jokainen paikkatieto voidaan yhdistää tiettyyn käyttäjään.

4.6.3 Vaatimuksiin sopiva rakenne

Sovelluksen käyttäjien tunnukset sijoitetaan erilliseen tauluun, joka sisältää muun muassa käyttäjätunnukset (nimi), salatussa muodossa olevat salasanat (passwd) ja käyttäjien automaattisesti generoituvat id:t (id).

Koska järjestelmän lähettämä tieto on on aina samassa formaatissa riippumatta käytetystä puhelimesta, on tietokannan rakenne melko yksinkertainen paikkatietojen tallennuksen osalta.

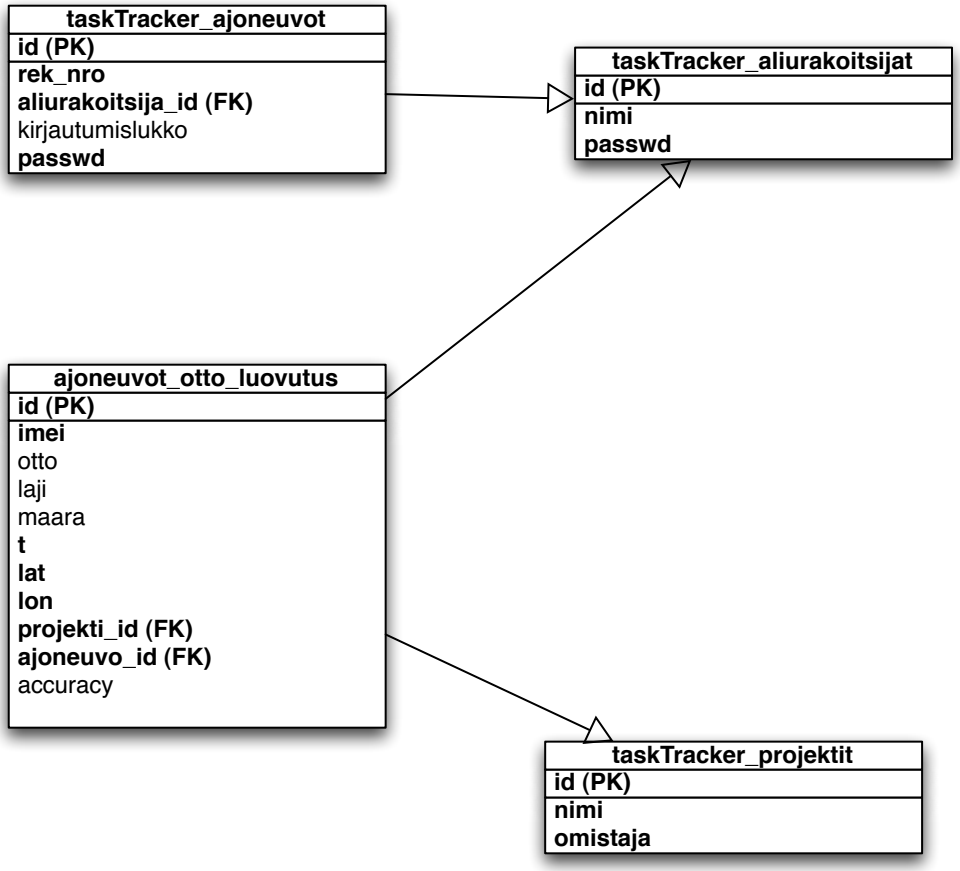
Luonnollisesti jokaiselle paikkatietomerkinnälle annetaan automaattisesti juokseva tunnistenumero (id). Aikaleima (t) on oltava paikkatietomerkinnässä, koska muuten ei ole mitään keinoa saada selville saadun paikkatiedon tapahtuma-aikaa, jolloin paikkatieto on useimmissa tapauksissa hyödytön. Lisäksi leveysasete (lat) ja pituusaste (lon) on asetettava, jotta paikkatiedon piirtäminen kartalle olisi mahdollista. Paikkatiedon tarkkuus (acc) on optionaalinen tieto, joka voidaan antaa tilanteissa, joissa se on saatavilla. Erilaisista puhelinmalleista riippuen tämä tieto ei aina ole saatavilla. Tarkkuustiedolla voidaan suodattaa pois mahdolliset virhemerkinnät, joissa paikkatiedon laatu ei ole välttämättä luotettava.

Tapahtuman tyyppi (otto) on pakollinen tieto, sillä se kertoo, millaisesta paikkatiedosta on kyse. Kuten, onko tämä paikkatieto tullut liikkeestä eli onko etäisyys on kasvanut tarpeeksi suureksi edellisestä tallennetusta paikkatiedosta vai onko sovelluksen käyttäjä painanut kuorman luovutus- tai ottonäppäintä.

Koska ajetun matkan (ajettu) lähettäminen jokaisen paikkatiedon yhteydessä ei ole mielekästä, päätettiin, että yhteenlaskettu ajettu matka lähetetään jokaisen kuorman luovutuksen yhteydessä. Muissa tapauksissa tämä kenttä jätetään tyhjäksi. Tällä tavalla puhelin pystyy tekemään laskentatyön valmiiksi, ilman että järjestelmän pitäisi jokaisen paikkatiedon yhteydessä laskea edellisen ja nykyisen paikkatietomerkinnän matkan paikkatietoja jatkokäsiteltäessä.

Lopuksi jokainen paikkatieto linkitetään tiettyyn käyttäjään (aliurakoitsija_id), jotta tiedetään, keneltä käyttäjältä paikkatieto on ylipäätään saatu.

Kuvassa 11 näkyvät merkinnät PK, FK1 kertovat kyseisen kentän arvojen olevan ensisijaisia ja uniikkeja sekä toimivan indeksinä (PK, Primary Key) tai viittaavan toisen taulun vastaavaan kenttään (FK, Foreign Key). Lihavointi merkitsee, ettei kenttään hyväksytä tyhjää arvoa.



Kuva 11. Hahmotus toimivasta tietokannan rakenteesta.

4.7 Työn kulku

Ohjelmointivirheet saatiin lopulta korjattua siihen pisteeseen asti, että virhetilanteita sattui hyvin harvoin. Tämä virhemarginaali riitti sovelluksen lopulliseen versioon, ja tarvittaessa ohjelmointivirheitä paikkaillaan, jos siihen tulee jatkossa tarvetta.

Sovelluksen päänäkymä, jossa käyttäjä on jo kirjautunut järjestelmään, tehtiin pelkästään div-tagien avulla. Kirjautumisprosessissa käytettävien näkymien näppäimet ja tekstikentät tehtiin sovelluksen alussa JavaScriptillä.

Näppäimet, joita ei käytetä kuin harvoin sovelluksen ajon aikana, kuten uloskirjautuminen, sijoitettiin menu-valikoihin, joissa ne eivät häiritse sovelluksen varsinaista käyttöä.

5 Yhteenveto

Insinöörityössä suunniteltiin ja toteutettiin toimiva sovellus, joka lukee puhelimen paikkatiedot ja lähettää sen Internetin välityksellä Samfinderin Oy:n SQL-tietokantaan. Tietyissä erikoistilanteissa laite tallentaa paikkatiedot välimuistiin, kunnes se onnistuu lähettämään paikkatiedot tietokantaan. Sovelluksen tarkoitus on olla sovellus, joka tarjoaa paikkatietoja Samfinder Oy:n muille järjestelmille, jotka käyttävät hyödykseen saatuja paikkatietoja.

Sovellus suunniteltiin Omni Graffle Pro -suunnittelutyökalulla, kirjoitettiin Aptana Studio 2 –ohjelmalla, jossa oli asennettuna Nokian WRT -liitännäinen, ja toteutettiin JavaScript-ohjelmointikielellä ja sen liitännäisillä.

Nokian WRT soveltui sovelluksen tarpeisiin loistavasti, ja jokainen suunniteltu ominaisuus saatiin toteutettua tällä tekniikalla. Koska WRT perustuu JavaScriptiin, on sen oppimiskynnys hyvin matala, joten sovelluksen jatkokehityskin on varsin helposti toteutettavissa tilanteesta riippumatta.

Kun kyseessä on suurille yrityksille käyttöön tuleva järjestelmä, ei voida koskaan alleviivata liikaa järjestelmän toimintavarmuutta ja tietoturvaa, varsinkin, kun järjestelmästä saatuja paikkatietoja saatetaan käyttää esimerkiksi laskutuksen perusteina.

Vaikka sovelluksen kehityksen aikana ei testauksessa ilmennytkään tilanteita joissa järjestelmä antaisi paikkatiedon jossa molemmat leveysasteen arvot olisivat 0, voisi tällaisiin tilanteisin kuitenkin jatkossa varautua. Esimerkiksi tilanteessa, jossa kuljettaja painaa otto- tai luovutusnäppäintä, ei aina välttämättä saada totuudenmukaista paikkatietoa. Puhelinten kehittyessä ja paikkatietodatan parantuessa esimerkiksi luotettavalla tarkkuusarviolla voitaisiin estää tällaisten epätarkkojen paikkatietojen lähetys heti puhelimessa, jolloin odotettaisiin, niin kauan kunnes luotettava paikkatieto olisi saatavilla ja sitten lähetettäisiin data.

Sovelluksen kustomointia asiakaskohtaisesti hakemalla käyttäjäkohtaiset asetukset tietokannasta sovelluksen käynnistyessä kannattaa myös harkita. Koska tämä toiminnallisuus on jo tuotu iPhonelle ja Androidille projektin aikaisemmissa vaiheissa, on ominaisuuksien tuominen myös tässä työssä tehtyyn sovellukseen hyvin nopeasti toteutettavissa.

Koska Nokian matkapuhelimet ovat olleet pitkään kehityksen kärjessä, varsinkin mitä tulee yksittäisiin komponentteihin ja laitteistoon, tarjoavat Nokian puhelimet hyvän kehitysalustan tälle sovellukselle muiden matkapuhelimien ja niiden käyttöjärjestelmien rinnalla. Varsinkin GPS:n kehittyminen tarkkuuden ja nopeuden suhteen Nokian laitteissa mahdollistavat jatkossa entistä varmemman ja tarkemman paikkatiedon saannin.

Lähteet

1 jQuery. (WWW-dokumentti.) <http://www.w3schools.com/jquery/default.asp>. Luettu 21.10.2010.

2 JavaScript. (WWW-dokumentti.) <http://www.w3schools.com/js/default.asp>. Luettu 21.10.2010.

3 Nokia Web Runtime. (WWW-dokumentti.) <http://wiki.forums.nokia.com>. Luettu 20.10.2010.

4 JSON. (WWW-dokumentti.) Wikipedia. <http://fi.wikipedia.org/wiki/JSON>. Luettu 19.10.2010.

5 JSON. (WWW-dokumentti.) Wikipedia. <http://en.wikipedia.org/wiki/JSON>. Luettu 19.10.2010.

6 PHP. (WWW-dokumentti.) Wikipedia. <http://fi.wikipedia.org/wiki/PHP> 21.10.2010.

7 PHP. (WWW-dokumentti.) Wikipedia. <http://www.w3schools.com/php/default.asp>. Luettu 21.10.2010.

8 Ajax. (WWW-dokumentti.) W3Schools.com http://w3schools.com/ajax/default.asp. Luettu 21.10.2010.

9 Ajax (programming). (WWW-dokumentti.) Wikipedia. <http://en.wikipedia.org/wiki/Ajax_(programming)>. Luettu 20.10.2010.

10 Nokia Library. (WWW-dokumentti.)<http://library.forum.nokia.com/>. Luettu 20.10.2010.

11 Guarana UI. (WWW-dokumentti.) <http://wiki.forum.nokia.com/index.php/Guarana_UI:_a_jQuery-Based_UI_Library_for_Nokia_WRT>. Luettu 19.10.2010.

12 UML Use Case Diagram. (WWW-dokumentti.) Wikipedia. <http://en.wikipedia.org/wiki/Use_case_diagram>. Luettu 21.10.2010.

13 Käyttötapaukset. (WWW-dokumentti.) <http://myy.helia.fi/~kalei/oliomats/kayttotapaukset.html>. Luettu 21.10.2010.

14 jQuery ajax(). (WWW-dokumentti.) <http://api.jquery.com/jQuery.ajax/.> Luettu 20.10.2010.

15 Känsälä Jussi. Sovellus ilmastointidatan tietokantaan tallentamista varten. Insinöörityö, 2010. Metropolia Ammattikorkeakoulu.

Liite 1: Paikannuksen käyttöönotto ja siinä käytettävät funktiot

```
function startMonitoring(){
     try {
          so = null;
          tid = ""

          // On exit lets just clear the ongoing watch for
location changes if any.
          window.widget.onexit = function(){

               if (tid) {
                    so.clearWatch(tid);
               }
          };


          // Lets first create geolocation object.
          so = com.nokia.device.load("",
"com.nokia.device.geolocation");
          // timeout at 60000 milliseconds (60 seconds)
          var options = {
               timeout: 0
          };
          // Lets start getting location updates.
          tid = so.watchPosition(onLocationUpdate,
onLocationError, options);

     }catch(e){
          alert("startMonitoring: "+e);
     }

}
```

Liite 1: Paikannuksen käyttöönotto ja siinä käytettävät funktiot

```
// Callback method called whenever change in location is
detected.
function onLocationUpdate( location ){
      try {
      //prevent error "Script is too big"
      setTimeout("void()",0);

      $("#locationInfo").html("lat:"+location.coords.latitude+"
lon:"+location.coords.longitude);
            if (prev_timestamp_second == undefined)
                  prev_timestamp_second = 0;
            var timestamp = new Date().getTime();

            if (prev_lat == null || prev_lon == null) {
                  prev_lat = location.coords.latitude;
                  prev_lon = location.coords.longitude;
            }
            locationObj.lon = location.coords.longitude;
            locationObj.lat = location.coords.latitude;

            locationObj.acc = location.coords.accuracy;
            if(locationObj.acc === null || locationObj.acc ===
undefined){
                  locationObj.acc = 10;
            }

            distance = geoGetDistance(prev_lat, prev_lon,
location.coords.latitude, location.coords.longitude);

      $("locationInfo").html("lat:"+location.coords.latitude+"
lon:"+location.coords.longitude);
            $("#distance").html(distance);


            if (choosenProject_id != null && choosenVehicle_id !=
null && !isNaN(distance) && distance > 0.02) {
                  onlineMode = window.navigator.onLine;
                  prev_lat = location.coords.latitude;
                  prev_lon = location.coords.longitude;
                  uiManager.showNotification(500, "warning",
"Location sent!");
```

Liite 1: Paikannuksen käyttöönotto ja siinä käytettävät funktiot

```
                if (checkConnection() === true) {
                      if (locationStack.length > 0) {
                            sendLocationStack();
                      }
                      $.ajax({
                            type: "GET",
                            url:
"http://www.samfinder.com/ajoneuvot/locationHandler_nokia.php",
                            data: "lat=" + locationObj.lat +
"&lon=" + locationObj.lon + "&acc="+ locationObj.acc
+"&ajoneuvo_id=" + choosenVehicle_id + "&projekti_id=" +
choosenProject_id
                      });
                }
                else {
                      locationStack.push(locationObj);

      $("#locationStackInfo").html(locationStack.length);
                }
                prev_timestamp_second = timestamp;
          }
      }
      catch (e) {
            alert(e);
      }
}

//Error handling callback function for watchPosition.
function onLocationError( error ){

   alert("Error getting Location Updates: " + error.message );
};
```