OULUN SEUDUN
AMMATTIKORKEAKOULU

Jenna Moisejeff

ANDROID-KARTTASOVELLUS AJONEUVOSEURANTAAN, TYÖAJAN RAPORTOINTIIN JA TYÖMÄÄRÄINTEN VASTAANOTTAMISEEN

ANDROID-KARTTASOVELLUS AJONEUVOSEURANTAAN, TYÖAJAN RAPORTOINTIIN JA TYÖMÄÄRÄINTEN VASTAANOTTAMISEEN

Jenna Moisejeff
Opinnäytetyö
Syksy 2011
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu

Tietotekniikan koulutusohjelma, Ohjelmistojen tuotanto

Tekijä: Jenna Moisejeff

Opinnäytetyön nimi: Android-karttasovellus ajoneuvoseurantaan, työajan raportointiin ja työmääräinten vastaanottamiseen

Työn ohjaaja(t): Markus Ebeling, Max Technologies Oy; Kari Jyrkkä, Oulun seudun ammattikorkeakoulu

Työn valmistumislukukausi ja -vuosi: syksy 2011 Sivumäärä: 97

TIIVISTELMÄ

Tässä projektissa toteutettiin Max Technologies Oy:n Track-My-Work mobile -projektiin Android-pohjainen Google Maps -karttatekniikkaa käyttävä karttasovellus työmääräinten vastaanottamiseen, hallintaan sekä kevyeen navigointiin.

Työn tilaajana toimiva Max Technologies Oy on yli kolmen vuoden ajan toteuttanut useilla eri tekniikoilla töiden ja työajan hallintaratkaisuja. Asiakaskunnassa kotimaassa ja ulkomailla on esiintynyt tarvetta Android-alustalla toimiville työhallintaratkaisuille. Track-My-Work mobile -projektissa on tarkoituksena kehittää monipuolisia ohjelmistoja eri päätelaitteille, joissa ajetaan Android-, Windows CE-, Windows- ja Java-sovelluksia.

Sovellus toteutettiin Java-kielellä ja se on suunniteltu käytettäväksi erityisesti Android-tablet laitteissa.

Opinnäytetyön tuloksena syntyvää sovellusta, sen lähdekoodia ja työn dokumentaatiota hyödynnetään tuotteena ja mahdollisesti seuraavassa Android-projektissa. Tämä on yrityksen ensikosketus Android-alustaan.

Asiasanat: Android-ohjelmointi, Java, JSON, työajanhallinta, Google Maps Api

ABSTRACT

Oulu University of Applied Sciences

Information Technology, Software Development

Author: Jenna Moisejeff

Title of thesis: Android- based map application for tracking vehicles, reporting working hours and receiving work descriptions

Supervisor(s): Markus Ebeling, Max Technologies Oy; Kari Jyrkkä, Oulu University of Applied Sciences

Term and year when the thesis was submitted: Autumn 2011 Pages: 97

ABSTRACT

In this project, the goal was to develop an Android- based Google Maps application for Max Technologies' project Track-My-Work. Application was planned to consist functions such as accepting and managing work descriptions and tasks and lite navigation between current location and destination address.

The subscriber, Max Technologies has for three years carried out many work and time management related solutions using various techniques. There has been increasing demand for Android based applications among company's clientele home and abroad. Track-My-Work- project is intended for developing versatile software for different platforms, mainly Android-, Windows CE-, Windows- and Java- devices.The application is developed using Java- programming language and it is designed to be use especially in Android- tablet devices. The documentation and results of this thesis are used as a commercial product and in the next Android-software project. This is the first Android- project of Max Technologies.

Keywords: Android software development, Java, JSON, work management, Google Maps Api

ALKULAUSE

Tämä opinnäytetyö on toteutettu ja dokumentoitu kevään ja kesän 2011 aikana.

Työn ohjaavana opettajana on toiminut Kari Jyrkkä Oulun seudun ammattikorkeakoulusta ja työn valvojana on toiminut Max Technologies Oy:n toimitusjohtaja Markus Ebeling.

Haluaisin kiittää Max Technologies Oy:tä mahdollisuudesta toteuttaa opinnäyte yrityksessään sekä Max Technologies Oy:n työntekijöitä heidän antamastaan avusta ja tuesta opinnäytetyön teossa.

Oulussa 31.8.2011

Jenna Moisejeff

SISÄLTÖ

KÄYTETYT LYHENTEET JA TERMINOLOGIA

3G – 3rd Generation, kolmannen sukupolven matkapuhelinteknologiat.

ADSL – Asymmetric Digital Subscriber Line, verkkokytkentätekniikka.

A-GPS – Assisted GPS, verkkoavusteinen satelliittipaikannus.

Client/Server – Asiakas-palvelinarkkitehtuuri.

Android OS – Googlen kehittämä käyttöjärjestelmä mobiililaitteille.

AVD – Android Virtual Device, emulaattori, jossa simuloidaan laitteen toimintaa.

C# – Microsoftin kehittämä ohjelmointikieli.

C++ – Bjarne Stroustrupin kehittämä ohjelmointikieli.

Debuggaus – Virheiden etsintä ja korjaus sovelluksesta.

Dialogi – Sovelluksessa ikkuna, jolla sovellus kommunikoi käyttäjän kanssa.

Google Maps Api – Julkinen rajapinta, jonka avulla sovellukseen voidaan hakea Googlen karttatietoja.

GPS – Global Positioning System, satelliittipaikannusjärjestelmä.

HTTP – Hypertext Transfer Protocol, protokolla, jonka avulla selain ja WWW-palvelin kommunikoivat.

HTTPS – Salattu versio HTTP-protokollasta.

IDE – Integrated development environment, kehitysympäristö.

IMEI – International Mobile Equipment Identity, yksilöivä laitetunnus mobiililaitteille.

IntelliJ – Java-ohjelmointiympäristö.

IP – Internet Protocol, verkkokerroksen protokolla.

Java – Sun Microsystemsin kehittämä ohjelmistoalusta ja oliopohjainen ohjelmointikieli

JDK – Java Development Kit, kehittämistyökalut Java-sovelluksille.

JSON – JavaScript Object Notation, helppokäyttöinen formaatti tiedonsiirtoon.

Lokalisointi – Sovelluksen kansainvälistäminen (kieli, valuutta, mittayksiköt jne.).

MD5 – Message-Digest-algoritmi, käytetään tiedon kryptaamiseen.

Netbeans – Ohjelmointiympäristö mm. Java-, PHP-, C- ja C++-kielille.

NFC – Near Field Communication, etätunnistustekniikka.

Objekti, olio – Luokan ominaisuudet toteuttava instanssi eli esiintymä.

PHP – Hypertext Preprocessor, web-palveluiden tuottamiseen tarkoitettu ohjelmointikieli.

Plugin – Sovellusliitännäinen, joka liitetään isäntäohjelmaan halutun toiminnon tarjoamiseksi.

Protokolla – Tapa, jolla laite kommunikoi palvelimen kanssa.

Putty – Sovellus Telnet/SSH-yhteyden muodostamiseen.

SDK – Software Development Kit, kehittämistyökalut tietylle alustalle.

TCP – Transmission Control Protocol, luotettava tietoliikenneprotokolla.

UI – User Interface, käyttöliittymä. Näkymä käyttäjälle.

USB – Universal Serial Bus, sarjaväyläarkkitehtuuri oheislaitteiden liittämiseksi PC:hen.

Widget – Käyttöliittymäkomponentti.

WinCE – Windows-käyttöjärjestelmä kämmentietokoneille ym. pienitehoisille laitteille.

WLAN, Wi-Fi – Wireless Local Area Network, langaton lähiverkko.

XML – EXtensible Markup Language, merkintäkieli.

1 JOHDANTO

Työn tavoitteena oli toteuttaa Android-pohjainen mobiilikäyttöliittymä työmääräysten vastaanottamiseen ja suorittamiseen sekä kevyeen navigointiin.

Työn tilaaja Max Technologies Oy on yli kolmen vuoden ajan tehnyt monipuolisilla tekniikoilla toimivia töiden sekä työajan hallintaratkaisuja. Asiakaskunnassa kotimaassa ja ulkomaillla on esiintynyt tarvetta monipuolisille ja helppokäyttöisille ajan- ja paikanhallintaratkaisuille. Tämä Android-alustalle tehtävä karttasovellus on osa tilaajan Track-My-Work mobile -projektia, jonka tarkoituksena on kehittää ohjelmistoja eri päätelaitteille, joissa ajetaan Android-, Windows CE-, Windows- ja Java-sovelluksia. Ohjelmistot toteutetaan lähtökohtaisesti Java-, C++-, C#- ja Object Pascal -ohjelmointikielillä. Sovelluksilla voi kenttätyössä raportoida ja hallita töihin liittyviä asioita, kuten vastaanottaa ja suorittaa työmääräyksiä ja työtehtäviä. Lisäksi sovelluksilla voi hallita ja raportoida esimerkiksi ajoneuvon tai objektin sijaintiin liittyviä toimintoja. Sovellus on pääasiassa tarkoitettu ammattialoille, joissa pääosa työajasta ollaan tien päällä ja suoritetaan erilaisia työtehtäviä, esimerkiksi kiinteistöhuoltoyritykset, lähetit tms. Tärkeä vaatimus sovelluksille on hyvä käytettävyys.

Track-My-Work mobile -projektin myötä on myös tarkoitus yhtenäistää laitteiden käyttämää protokollaa ja näin yksinkertaista järjestelmää ja tehdä siitä joustavampi, kevyempi ja helpompi ylläpitää. Track-My-Work mobile -projekti käynnistettiin huhtikuussa 2011. Android-sovelluksen teko aloitettiin heti, ja tarkoituksena oli, että sovelluksen ensimmäinen versio olisi sisäisessä testauksessa saman vuoden toukokuussa.

Projektin tuloksina syntyviä sovelluksia pyritään hyödyntämään yrityksen liiketoiminnassa heti kun ne ovat valmiita ja ne on julkaistu. Koska tämä on ensimmäinen Android-projekti yrityksessä, tämän työn dokumentointia on tarkoitus käyttää pohjana yrityksen seuraavassa Android-sovelluskehitys projektissa eräänlaisena ohjeena ja tämän työn tuloksena syntyneen sovelluk-

sen jatkokehityksessä. Todennäköisesti sovelluksesta julkaistaan useita räätälöityjä versioita, jotta asiakkaat saisivat juuri heille sopivan ratkaisun. Projektin myötä yrityksen tavoitteena on myös laajentaa yrityksen ohjelmisto-osaamista Android-alustalle.

Insinöörityön tekijälle pääasiallinen tavoite on oppia Android-ohjelmistokehityksen perusteet ja tuoda näin yritykseen osaamista uudelle alustalle. Insinöörityö sisältää paljon ohjelmointiin liittyvää terminologiaa, ja täten lukijan olisi suotavaa osata perusteet Java-ohjelmoinnista ja sen syntaksista.

2 KOHDEJÄRJESTELMÄN KUVAUS

2.1 Kohdejärjestelmä

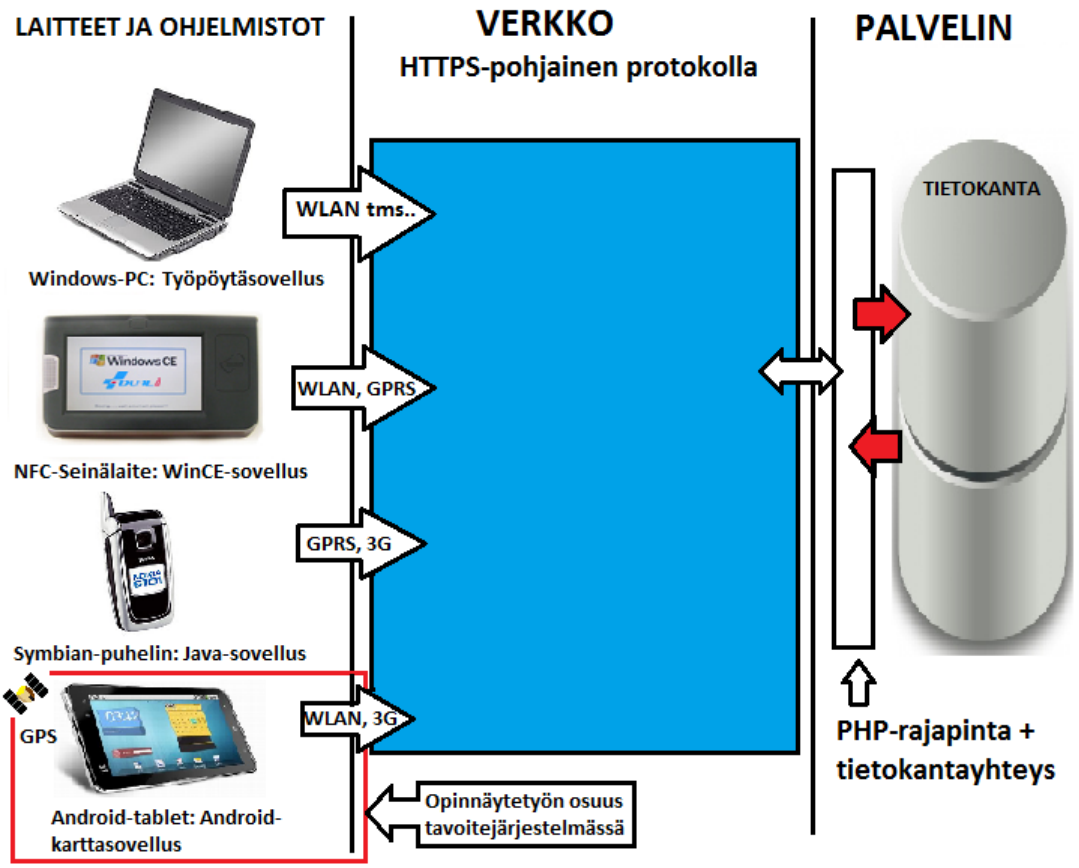
Android-sovellus on siis osa suurempaa Track-My-Work mobile -projektia, jossa toteutetaan työajan- ja paikanhallintaratkaisuja useille alustoille ja uudistetaan protokollaa eli laitteiden ja palvelimen välistä kommunikointia. PC:lle on jo olemassa versio työpöytäsovelluksesta: Desktop Reporting Tool. Se on helppokäyttöinen työkalu työajanseurantaan työpaikoissa, joissa työpiste ei juurikaan muutu. Tämän opinnäytetyön aikana työaikaa on seurattu tällä työkalulla. (1.)

Yhdessä projektin vaiheessa toteutetaan käyttöliittymä NFC-seinälaitteeseen, joka on tarkoitettu asennettavaksi kiinteästi kohteeseen ja jossa tunnistautuminen ja toiminnot hoidetaan henkilökohtaisella NFC-tunnisteella. Projektin aikana toteutetaan myös Java-ohjelmointikielellä mobiilikäyttöliittymä Symbian-älypuhelimeen. (1.)

Opinnäytetyön osuus tässä projektissa on Android-tablet- laitteeseen toteutettava karttasovellus työmääräinten vastaanottamiseen, käsittelyyn ja hallintaan sekä kevyeen navigointiin oman ja työmääräimen mukaisen kohdemääränpään välillä. Kaikki laitteet ovat yhteydessä palvelimeen Internet-yhteyden välityksellä, joka eri laitteissa muodostetaan eri tekniikoilla kuten WLAN, 3G-verkko, ADSL tms. Tiedonsiirrossa käytetään suojattua HTTPS-pohjaista protokollaa. (1.)

Projektin myötä on tarkoitus yhtenäistää yrityksen tiedonsiirtoprotokollat ja tehdä niistä helposti laajennettavia ja ylläpidettäviä. Aiemmin kaikki laitteet kommunikoivat palvelimen ja sillä olevan tietokannan kanssa oman PHP-rajapintansa kautta. Kaikki laitteet kommunikoisivat palvelimen kanssa suurelta osin samalla tavalla. Esimerkiksi kaikissa sovelluksissa pystyy lähettämään ”aloita työ”-, ”keskeytä työ”- ja ”lopeta työ” -komennot. Aiemmin käytetyt rajapinnat on tarkoitus yhdistää yhdeksi suureksi rajapinnaksi, jota kaikki

sovellukset käyttäisivät. Rajapinnan sisällä olisivat kaikki sovelluksien tarvitsemat yhteiset funktiot ja laitekohtaisesti yksilöidyt funktiot, joita muut laitteet eivät tarvitse. Android-sovelluksessa yksi tällainen funktio on paikkatiedon reaaliaikainen lähettäminen palvelimelle. Projektin staattisten päätelaitteiden, kuten PC:n ja WinCe-seinälaitteen, ei tarvitse lähettää paikkatietoaan palvelimelle. Kuvassa 1 esitetään Track-My-Work mobile -projektin kuvaus. (1.)



KUVA 1.Track-My-Work- mobile-projektin tavoitejärjestelmä

2.2 Sovelluksen toiminta ja vaatimukset

Ennen kuin sovelluksen toimintoja käsitellään tarkemmin, on syytä määritellä työssä useasti käytetty termi työmääräin. Työmääräin on sisältää työtehtävään liittyvät tiedot. Työmääräin luodaan, tallennetaan ja lähetetään laitteeseen web-käyttöliittymästä. Se on eräänlainen työtehtävän kuvaus, ja sen mukana tulee tehtävään ja sen hallintaan liittyviä tietoja, kuten identifiointi-

koodi (sijainti tietokannassa), saapumisaika, osoite, työtehtävän tarkempi kuvaus, paikkatieto karttaa varten, prioriteetti ja työtehtävän kesto. Määräimellä on myös status-ominaisuus, josta lähettäjä näkee, mikä on kyseisen työtehtävän tila.

2.2.1 Tarkka toiminnan kuvaus

Työntekijä on tallennettu tietokantaan, jossa hänelle on määritetty käyttäjänimi, PIN-koodi ja laite. Laitteen määrittämiseen käytetään IMEI-koodia, koska jokaisella laitteella se on yksilöllinen eikä samaa koodia voi olla kahdella eri laitteella. Samalla laitteella voi olla useampi käyttäjä. Tämän takia sovelluksen tulisi tallentaa viisi viimeksi kirjautunutta henkilöä.

Työntekijä on avaa sovelluksen ja kirjautunut sisään järjestelmään. Käyttöliittymä hakee automaattisesti käyttäjään liittyvät työmääräimet ja mahdollisesti käynnissä olevan tehtävän palvelimelta ja asettaa ne määräinlistaan. Autentikointi eli käyttäjän tunnistus suoritetaan vertaamalla käyttäjänimeä ja PIN-koodia keskenään, ja lopuksi katsotaan, onko laite rekisteröity ko. käyttäjälle.

Esimies luo määräimen ja lähettää sen tietylle työntekijälle Android-tablet -laitteeseen. Käyttöliittymä herää määräimen saapumiseen, ilmoittaa käyttäjälle, että määräin on saapunut, samalla tavalla kuin tekstiviesti saapuu matkapuhelimeen, ja määräin sijoitetaan käyttöliittymässä olevaan määräinlistaan. Määräin näkyy lippuna kartalla osoitteessa, joka tuli määräimen mukana.

Kun määräin tulee, käyttäjä näkee määräimeen liittyvät tiedot, kuten osoitteen, työtehtävän kuvauksen ja niin edelleen, ja voi aluksi hyväksyä tai hylätä sen. Tieto kummastakin toiminnosta menee määräimen lähettäjälle. Jos määräin hylätään, se poistetaan käyttöliittymässä olevasta määräinlistasta. Jos määräin hyväksytään, se jää näkyville määräinlistaan ja esim. väri indikoi määräimen tilaa. Kun määräin on uusi, väri vilkkuu keltaisena. Kun määräin on hyväksytty mutta sille ei ole tehty toimintoa, väri on harmaa.

Määräimen voi avata suoraan painamalla sitä määräinlistasta. Kun määräin avataan, aukeaa ikkuna, jossa näkyvät määräimen mukana tulleet tiedot ja määräimeen liittyvät toiminnot painikkeina: ”Aloita/Keskeytä”, ”Näytä reitti” ja ”Merkitse suoritetuksi”. Ennen määräimen aloittamista käyttäjä voi halutessaan painaa ”Näytä reitti” -painiketta, jolloin kartalla näkyy reitti käyttäjän senhetkisestä paikasta kohdeosoitteeseen. Kun käyttäjä painaa ”Aloita”-painiketta, lähetetään määräimen ID-koodi ja aloituspyyntö palvelimelle. Palvelimella työ merkataan aloitetuksi ja esimies näkee, milloin työ on aloitettu ja kuka sen on aloittanut.

Yhdellä työntekijällä voi olla käynnissä vain yksi työ kerrallaan. Kun työntekijä lähettää aloituspyynnön ja palvelimella huomataan, että työntekijällä on jo työ käynnissä, työntekijä saa virheilmoituksen. Kun aloituspyyntö menee läpi, työ merkataan aloitetuksi. ”Aloita”-painike muuttuu ”Keskeytä”-painikkeeksi. Kun sitä painetaan (esim. kahvitauon takia), lähetetään keskeytyspyyntö palvelimelle ja työ merkitään keskeytyneeksi. Keskeytystä indikoidaan määräinlistassa oranssilla värillä. Kun työntekijä on suorittanut työn kohteessa, hän painaa ”Merkitse suoritetuksi” -painiketta, palvelimelle lähetetään tieto siitä, että työ on suoritettu, ja määräinlistassa näkyy ”Check”-ikoni määräimen vieressä. Käyttöliittymässä käytetyt komponentit ja niiden asettelu ja käytettävyys katselmoidaan ja hyväksytetään erillisellä henkilöllä.

Kun mikä tahansa komento lähetetään palvelimelle, sen mukana lähetetään aina myös käyttäjän kirjautumistiedot (käyttäjätunnus, PIN-koodi ja laitteen IMEI-koodi).

Sovellus on koko ajan yhdistettynä Internetiin (Wi-Fi, 3G tms.), ja se lähettää reaaliajassa omaa paikkatietoaan palvelimelle. Esimies näkee laitteen sijainnin kartalla. Paikkatieto tulee sovellukseen laitteen omalta sisäänrakennetulta GPS-vastaanottimelta. Oma paikkatieto näkyy käyttöliittymässä kartalla autosymbolina.

Määräimeen voi myös liittää kuvan, joka otetaan tabletin omalla kameralla. Otettu kuva näytetään määräimen tiedoissa ja voidaan lähettää palvelimelle binäärisessä muodossa.

2.2.2 Sovelluksen vaatimukset

Sovelluksen vaatimukset koostuvat yleisistä ja toiminnallisista vaatimuksista.

Sovelluksen yleiset vaatimukset

Yleisiin vaatimuksiin kuuluvat sellaiset vaatimukset, jotka ovat tärkeitä mm. käytettävyyden kannalta:

- **Keveys:** Sovellus on toiminnaltaan mahdollisimman yksinkertainen, yhtäaikaisia toimintoja tapahtuu vähän. Palvelimelle lähtee sisällöltään pieniä HTTP-paketteja. Sovellus ei ”jäädy” tai mene jumiin.

- **Käytettävyys:** Käyttäjä on normaalihenkilö, hänen ei tarvitse tietää mitään esim. GPS:n toiminnasta tai omata laajaa teknistä tietämystä. Käyttöliittymä on miellyttävän näköinen, selkeä ja helppo käyttää ja näppäinpainalluksiin reagoidaan nopeasti. Käyttäjää opastetaan tarvittaessa. Lopullinen ulkoasu ja käytettävyys katselmoidaan ja hyväksytään, ennen kuin sovellus julkaistaan.

- **Virrankäyttö:** Laite on pääasiassa tarkoitettu auton tai muun ajoneuvon paikantamiseen. Se on siis paikallaan telineessä ja saa virtaa virta-adapterista, joka on kytketty auton tupakansytyttimeen.

Sovelluksen toiminnalliset vaatimukset

Toiminnalliset vaatimukset määrittävät minkälaisia toimintoja sovelluksen tulisi pystyä suorittamaan. Tässä sovelluksessa toiminnalliset vaatimukset ovat seuraavat:

- **Tietojen tallennus laitteeseen:** Sovelluksen tilaa ja käyttäjän tietoja, esimerkiksi kirjautumistietoja, pitää pystyä tallentamaan.

- **Lokalisointi:** Sovellus käyttää alkuvaiheessa suomen kieltä. Tuki lokalisoinnille toteutetaan projektin myöhemmässä vaiheessa, jolloin kielituki siirretään osittain palvelimelle. Silloin merkkijonot tallennetaan resurssitiedostoon ja luetaan sieltä. Alkuvaiheessa pääasia on, että päivämääräformaatit ja luvut esitetään oikein, toimintojen nimeämisessä käytetään yhtenäistä linjaa ja umlautit eli ”ääkköset” näkyvät oikein.

- **Jatkuva Internet-yhteys:** Sovelluksen on oltava jatkuvasti yhteydessä verkkoon, jotta määräimet pääsevät laitteeseen. Kartan päivittyminen ja reittitietojen hankkiminen vaativat myös Internet-yhteyden. Sovelluksen toiminnoista suurin osa vaatii yhteyden.

- **HTTPS:** Suojattu tiedonsiirto palvelimelle.

- **Kartta:** Sovelluksen pakollinen vaatimus. Kartalle on myös pystyttävä piirtämään erilaisia objekteja ja reittejä. Kartan avulla pystyy navigoimaan kohteeseen, mutta reittiä ei tarvitse opastaa käyttäjälle. Riittää kun käyttäjä näkee reitin kartalla.

- **Määräinten vastaanotto:** Pakollinen vaatimus. Sovelluksella pystytään vastaanottamaan työmääräimiä palvelimelta.

- **GPS-datan luku:** Pakollinen vaatimus, pitää pystyä lukemaan dataa ja lähettämään sitä palvelimelle reaaliajassa. Myös reitit piirretään oman GPS-datan avulla.

- **Autentikointi:** Koska laitteella seurataan esim. yrityksessä olevan henkilön työaikaa ja työsuoritteita, on tärkeää, että käyttäjä tunnistetaan. Sovellukseen täytyy tehdä mm. ikkuna sisäänkirjautumista var-

ten. Käyttäjä ja laite tunnistetaan jokaisen palvelinpyynnön yhteydessä.

- **Versiointi ja päivitys:** Koska sovelluksesta tehdään monta versiota projektin aikana, on tärkeää, että käyttäjälle voidaan tarjota sovelluksesta päivitystä, kun sellainen on saatavilla. Tämä toteutetaan esim. käyttöliittymään ilmestyvällä painikkeella tai dialogilla, josta käyttäjä ohjataan latauslinkkiin, josta uusin versio ladataan.

2.2.3 Laitteen vaatimukset

Laitteessa täytyy olla vähintään Android 2.2 -käyttöjärjestelmä ja sisäänrakennettu GPS-vastaanotin. Lisäksi sillä on päästävä kiinteästi Internetiin joko langattoman verkon avulla tai 3G-dataliittymän välityksellä.

3 TOTEUTUKSEN SUUNNITTELU

3.1 Kohdelaitteen ja ohjelmointiympäristön valinta

Suunnittelu alkoi kohdelaitteen valinnalla. Aluksi suunniteltiin kehittämistä ZTE Light -tablettiin, jossa oli Android 2.1 -käyttöjärjestelmä. Se kuitenkin vaihdettiin ViewSonicin VPAD 7 -tablettiin, koska haluttiin kehittää sovellusta uudempaan, ominaisuuksiltaan parempaan ja eniten käytettyyn Android 2.2 -järjestelmään. (2.)

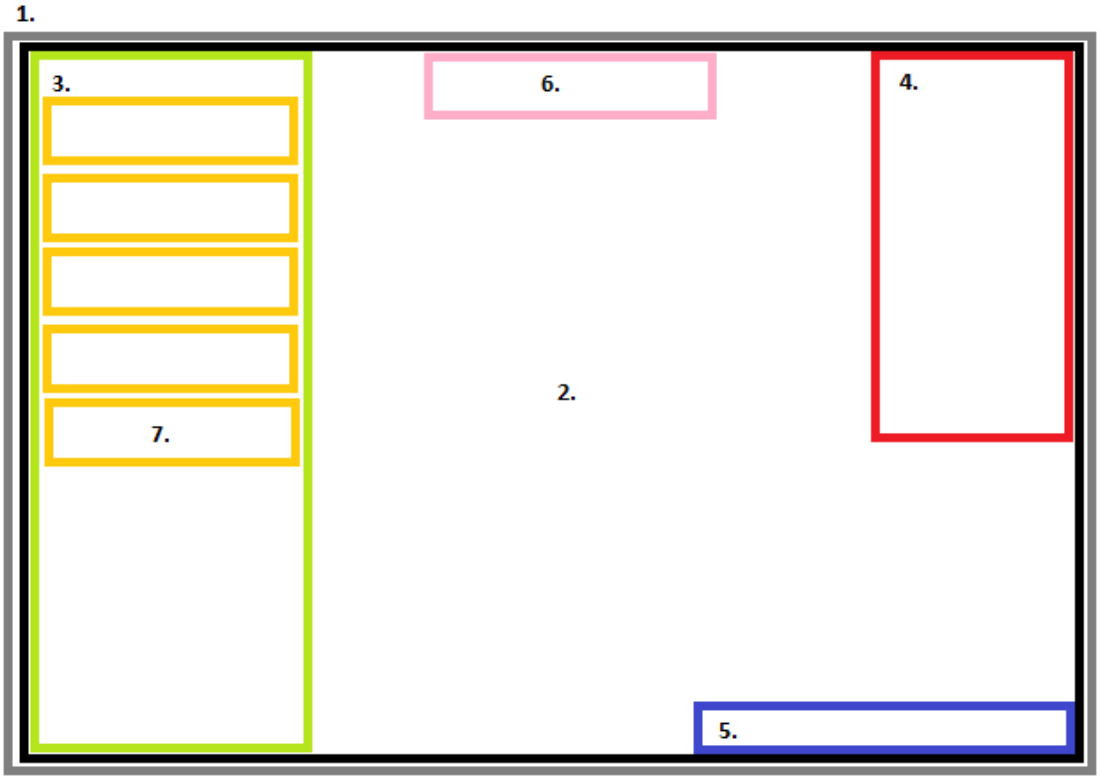
Android-koodia kirjoitetaan Java-ohjelmointikielellä ja Android-sovelluksia voi ohjelmoida useissa eri ohjelmointiympäristöissä, esimerkiksi Eclipse IDE:ssä, NetBeansissä ja IntelliJ:ssä. Ohjelmointiympäristön valinnassa ei ollut perusteena niiden hyvät ominaisuudet toisiinsa nähden, vaan Eclipse IDE valittiin yksinkertaisesti siitä syystä, että siitä oli eniten kokemusta ja esimerkiksi virallinen Android Developers -sivusto käyttää useimmissa tutoriaaleissaan Eclipse IDE:tä. Lisäksi Eclipse on ilmainen kehitysympäristö. (3.)

Sovellusta kehitettäessä päätettiin debugata eli korjata virheet suoraan kohdelaitteessa emulaattorin sijaan. Näin nähdään suoraan sovelluksen käyttöliittymän käyttäytyminen kohdelaitteessa. Samalla pystytään myös suorittamaan lohkotestausta suoraan oikeassa laitteessa ja ympäristössä.

3.2 Käyttöliittymän suunnittelu

Android-sovelluksen suunnittelu aloitettiin käyttöliittymäsuunnittelusta. Vaatimuksina oli, että käyttöliittymä olisi yksinkertainen, miellyttävän näköinen ja kaikin puolin käyttäjäystävällinen. Koska sovellus on tarkoitettu määräinten vastaanottamiseen ja navigointiin, suurimman osan näytön koosta veisi taustalla oleva kartta sekä määräinlista. Lisäksi käyttöliittymään tulisi mahdollisesti tilaikkuna, josta käyttäjä näkee erilaisia tiedotteita ja esim. GPS-vastaanottimen tilan. Koska laite on lähtökohtaisesti paikallaan autossa,

otetaan käyttöön sivuttaislukitus, ettei ruutu pääse kääntymään pystyasentoon. Kuvassa 2 näkyy sovittu komponenttien asemointi ruudulla.



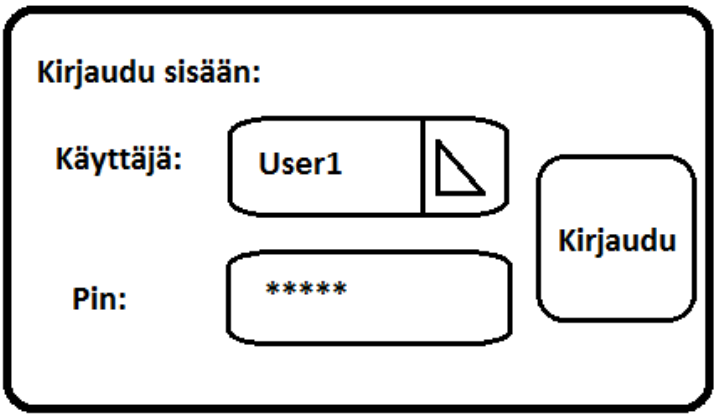
KUVA 2. Käyttöliittymän pohja: 1. Pää-layout, jonka päälle rakennetaan muut layoutit 2. Kartta-layout 3. Määräinlista 4. Toteutuksen aikana tarvittavat testi-painikkeet (poistetaan kun sovellus on valmis). 5. Tilaikkuna 6. Kartan suurennus- ja pienennyskontrollit 7. Määräimet

3.3 Dialogit

Käyttöliittymäpohjan suunnittelun jälkeen siirryttiin suunnittelemaan tarvittavia dialoginäkymiä. Samalla kartoitettiin toimintoja ja muuttujia kokonaisvaltaisesti.

3.3.1 Sisäänkirjautuminen

Sovelluksen käyttäminen vaatii, että käyttäjä kirjautuu sisään eli lähettää palvelimelle pyynnön tarkistaa kirjautumistiedot. Dialogin ulkoasun tulisi olla yksinkertainen. Viisi viimeistä käyttäjää tallentuu tiputusvalikkoon. Kirjautumisdialogin ulkoasu tulisi olemaan kuvan 3 mukainen.



KUVA3. Kirjautumisikkunan suunnitelma

3.3.2 Määräimen hyväksyminen

Ennen kuin määräin lisätään määräinlistaan, käyttäjän on ensin hyväksyttävä se. Tämä dialogi näyttää saapuneen määräimen ja sen perustiedot. Käyttäjä voi joko hyväksyä tai hylätä määräimen. Määräimen hyväksyttävä dialogi näytetään, kun käyttäjä avaa saapuneen määräimen. Dialogin suunnitelma kuvassa 4.

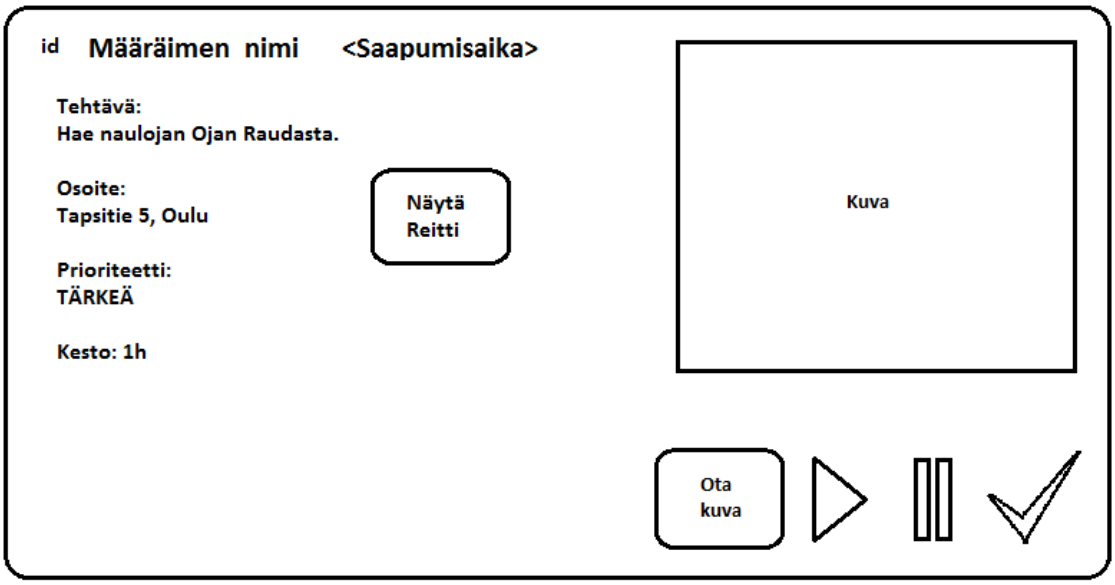
KUVA 4. Määräimen hyväksyttämisikkunan suunnitelma

3.3.3 Määräimen tiedot ja toiminnot dialogi

Dialogi avautuu silloin, kun käyttäjä on hyväksynyt määräimen tai kun määräintä klikataan määräinlistasta. Ikkunassa voi suorittaa seuraavat toiminnot:

- ”Play”-ikoni: Aloita työ

- ”Pause”-ikoni: Keskeytä työ
- ”Check”-ikoni: Määräin suoritettu
- ”Näytä reitti” -näppäin: Näytetään kartalla reitti omasta sijainnista määräimen osoittamaan osoitteeseen
- ”Ota kuva” -näppäin: Avaa kameran ja asettaa käyttäjän ottaman kuvan ikkunan oikeassa reunassa olevaan ruutuun.



KUVA 5. Määräindialogin suunnitelma

3.4 Toiminnallisuus

Seuraavaksi suunniteltiin, minkälaisia toimintoja sovellukseen tulisi.

3.4.1 Kartta

Sovelluksessa näkyvä kartta päätettiin toteuttaa Google Maps -pohjaisena, koska projektin toteuttajalla oli siitä eniten kokemusta muihin vaihtoehtoihin nähden. Muita vaihtoehtoja ei juurikaan edes tutkittu. Kartalle pitäisi pystyä piirtämään objekteja ja reittejä esimerkiksi kahden koordinaattipisteen tai osoitteiden välillä.

3.4.2 GPS-data

Laitteessa on sisäänrakennettu GPS-vastaanotin, josta data pitäisi lukea jotenkin.

3.4.3 Käyttäjän autentikointi

Järjestelmä käyttää asiakastietoja ja vaatii autentikoinnin. Tämä näkyy käyttäjälle esim. sisäänkirjautumisikkunana. Käyttäjä tunnistetaan jokaisen palvelinpyynnön yhteydessä.

3.4.4 Palvelinpyynnöt ja data

Palvelimelle lähetetään kevyt ja helposti purettava datapaketti. Käytetään jotain helposti parseroitavaa merkintätapaa (XML, JSON tai muu). Lähetyksessä ja vastaanotossa tulisi pääsääntöisesti käyttää yhtenevää protokollaa.

3.4.5 Paikallisten tietojen tallennus ja luku

Paikallisella tiedolla tarkoitetaan sellaista dataa, joka täytyy tallentaa laitteeseen, esimerkiksi käyttäjäkohtainen data kuten käyttäjätunnukset sekä sovelluksen muut asetukset ja tila. Käytännössä kaikki data, jota ei haeta palvelimelta, tallennetaan laitteen muistikortille.

3.4.6 Muut vaatimukset

Muita vaatimuksia sovellukselle olivat mm. keveys, käytettävyys ja jatkuva Internet-yhteys. Keveyteen päästään niin, että palvelimelle lähetetään mahdollisimman lyhyitä, datamäärältään pieniä viestejä. Käytettävä protokolla on lähinnä palvelinpään tehtävä. Sovellus huolehtii lähinnä datamääristä siten,, että tietoa ei lähetellä tarpeetonta määrää.

Käytettävyydestä huolehditaan julkaisun katselmoinnissa. Sovelluksen on läpäistävä laatutarkistukset ennen julkaisua. Sovelluksen on noudatettava tiettyjä sääntöjä ulkoasussa niin, että se noudattaa yrityksen yleistä linjaa grafiikoissa ja käyttää samoja logoja ja ääniä. Linjan yhdenmukaistaminen on myös yksi suuremman Track-My-Work mobile -projektin tarkoituksista.

Jatkuva Internet-yhteys hoidetaan sopivalla dataliittymällä. Palveluntarjoajalta tilataan sopiva 3G-dataliittymä. Liittymän hintaa kilpailutetaan.

3.5 Arvioitu aikataulu

Insinöörityön laajuus on yhteensä 400 tuntia. Suunnitelmana on, että puolet siitä (200 tuntia) on opinnäytetyön toteutusosa ja loput 200 tuntia käytetään insinöörityön kirjalliseen osaan, jossa tehdään insinöörityöhön liittyvät dokumentit ja raportointi, kuten projektisuunnitelma, aineiston ja lähtötietojen kerääminen ja koko työn kirjallinen raportti.

Tarkka aikataulun suunnittelu on tässä projektissa kohtalaisen haastavaa, koska esimerkiksi protokollaa, jonka mukaan sovellus kommunikoi palvelimen kanssa, uudistetaan Track-My-Work mobile -projektin yhteydessä. Android-sovelluksen eteneminen toivotussa aikataulussa riippuu suurelta osin palvelinohjelmiston etenemisestä. Track-My-Work mobile -projekti onkin laaja, kokonaisuudessaan yli vuoden (kesään 2012 asti) kestävä kokonaisuus. Tavoitteena Android-sovelluksen osalta on, että se saataisiin sisäiseen testaukseen jo toukokuun lopulla 2011, eli suunnitteluun ja toteutukseen aikaa olisi vain yli kuukausi. Toukokuussa sisäisessä testauksessa oleva sovellus ei ole kuitenkaan vielä kaupallinen tuote, vaan tämä demoversio pystyy vaadittuihin perustoimintoihin eli näyttämään kartan, kuuntelemaan ja käsittelemään GPS-dataa, lähettämään omaa paikkatietoa palvelimelle sekä vastaanottamaan ja käsittelemään työmääräimiä palvelimelta, ja lisäksi se pystyy perustyömääräimen hallintaan (työn aloitus, lopetus ja keskeytys).

Insinöörityössä syntyviä tuloksia hyödynnetään myöhemmin jatkokehityksessä. Todennäköisesti tulevaa tuotetta kustomoidaan asiakkaiden tarpeiden mukaan yhä monimutkaisemmaksi kokonaisuudeksi. Track-My-Work mobile -projekti kestää siis kesäkuulle 2012, mutta tämän opinnäytetyön osuus kestää kesäkuulle 2011.

Kevään ja alkukesän 2011 aikana ollaan yhteydessä opinnäytetyön ohjaajaan (OAMK:n edustaja) ja raportoidaan etenemisestä. Varsinainen insinöörityö kirjoitetaan kesän ja syksyn 2011 aikana. Syyslukukauden aikana suoritetaan sisällön ohjaus, kielenhuolto ja viimeistely.

4 TOTEUTUS

Tässä projektissa toteutettavaa ohjelmaa ajettiin koko ajan kohdelaitteeseen, joten suurin osa sovelluksen testauksesta on myös sisällytetty toteutuksen yhteyteen. Tätä sanotaan ns. lohkotason testaamiseksi, koska toteutus eteni lohko kerrallaan. Useimmiten ennen siirtymistä rakennetusta lohkosta seuraavaan edellisen lohkon toiminta testattiin kohdelaitteessa. Toteutus-osio kirjoitetaan yksityiskohtaisesti, jotta tästä dokumentista olisi hyötyä seuraavassa Android-ohjelmistokehitysprojektissa. Tätä dokumenttia voidaan jatkossa käyttää eräänlaisena Android-sovelluskehitysoppaana, koska se sisältää ohjeet ohjelmointiympäristön käyttöönottoon, projektin luomiseen, käyttöliittymän toteutukseen, sokettikäsittelyyn, säikeiden käytön perusteisiin ja niin edelleen.

4.1 Ohjelmointiympäristön asennus ja käyttöönotto

Ennen Eclipsen ja Android SDK:n asennusta asennetaan JDK eli Java Development Kit (tämän yhteydessä myös JRE:n asennus). Sitten ladataan varsinainen ohjelmointiympäristö, Eclipse Classic 3.6.1 -sovellus. Apuna käytettiin erään harrastelijan tekemää videotutoriaalia aiheesta (4).

Ladataan Android SDK Android-Developers -sivustolta. Ladataan sopiva versio omalle käyttöjärjestelmälle. Tässä projektissa käytettiin Windows 7 -käyttöjärjestelmää. Puretaan ladattu .zip-tiedosto haluttuun polkuun. Hakemistossa oleva SDKSetup.exe-ohjelma avaa SDK- ja AVD-Managerin, jonka kautta ladataan ja asennetaan halutut alustat eri Android-versioille (1.5, 1.6, 2.2, Google APIs, …). Tässä projektissa käytetään Android 2.2 -alustaa ja GoogleAPIs-kirjastoja. Seuraavaksi määritetään Android SDK:n polku järjestelmämuuttujiin. Mennään (Windows-ympäristössä) Käynnistä-valikko → Ohjauspaneeli → Järjestelmä → Järjestelmän lisäasetukset → Ympäristömuuttujat. Järjestelmämuuttujat taulukosta etsitään Path-muuttuja ja painetaan ”Muokkaa”-painiketta. Muuttujan arvoon lisätään ”;”-merkki ja

Android SDK:n tools-kansion polku sekä samalla tavalla JDK:n \bin-polku (esimerkiksi C:\Program Files\Java\jdk1.6.0_24\bin) ja tallennetaan. Seuraavaksi luodaan Androidin virtuaalilaitteet AVD-Managerilla.

Avataan Eclipse. Alussa määritetään käytettävä Android-työtila (engl. workspace), jonne kaikki kehitettävät ohjelmat tallennetaan. Sitten mennään valikkoon Help → Install New Software. ”Work with” -ruutuun kirjoitetaan osoite https://dl-ssl.google.com/android/eclipse/, josta asennetaan ADT-Plugin Eclipseen (Android Development Tools -plugin). Kun painetaan ”Add”-painiketta, taulukkoon ilmestyy laatikko Developer Tools, asetetaan se valituksi ja painetaan ”Next”-painiketta. Tällöin ohjelma lataa ja asentaa tarvittavat osat automaattisesti. Tämän jälkeen käynnistetään Eclipse uudelleen.

Android SDK- and AVD-Manager -sovellus ilmestyy Eclipsen Window-valikkoon, kun liitännäinen (plugin) on asennettu.

Seuraavaksi avataan Eclipsessä Window → Preferences. Vasemmasta valikosta löytyy Android-valintaruutu. Valitaan se ja määritetään SDK:n polku (<asema>:\android-sdk-r07-windows) Eclipseen. Lopuksi painetaan ”Apply”-painiketta. Nyt ruudussa pitäisi näkyä kaikki aiemmin asennetut Android-käyttöjärjestelmäversiot.

Tässä projektissa debuggauksessa eli sovelluksen virheenkorjauksessa ei käytetä ollenkaan emulaattoria, vaan ohjelma ajetaan suoraan kohdelaitteeseen (suoradebuggaus), koska se on paljon nopeampaa ja tulokset nähdään heti. Näin ohjelman käytettävyyttä ja toimintaa testataan koko ajan suoraan kohdelaitteessa.

Jos kuitenkin halutaan käyttää emulaattoria sovelluksen debuggauksessa, pitää luoda uusi virtuaalilaite eli AVD. Avataan Android SDK- and AVD-manager Eclipsen Window-valikosta. Luodaan uusi virtuaalilaite. Ensin valitaan kohdejärjestelmä eli target (Android 2.2). Annetaan luotavalle AVD:lle jokin nimi ja painetaan Create AVD. Emulaattori eli virtuaalilaite on nyt luotu ja se voidaan käynnistää painamalla ”Start”-painiketta Android SDK- and

AVD Manager -ikkunassa. Emulaattori käynnistyy hetken kuluttua. Emulaattoria ei tarvitse sulkea buildausten välillä, vaan se voi olla koko ajan päällä. Buildaus tarkoittaa kirjoitetun ohjelmakoodin koostamista ajettavaksi kokonaisuudeksi.

Seuraavaksi kokeillaan kehitysympäristöä yksinkertaisella ”Hello, World” -oppaalla, joka löytyy Android Developers -sivustolta osoitteesta http://developer.android.com/resources/tutorials/hello-world.html. Aluksi luodaan Eclipsessä valikosta uusi projekti painamalla File → New → Android project. New Android Project- ikkunassa määritetään projektille

- **Projektin nimi,** joka näkyy työtilassa kansion nimenä. Tänne tallentuvat kaikki ohjelman lähdekooditiedostot ja resurssit.
- **Kohdekäyttöjärjestelmä,** valitaan aiemmmin asennetuista alustoista.
- **Ohjelman nimi,** käyttäjälle näkyvä ohjelman nimi.
- **Package** eli ”Namespace”, Java ohjelmoinnissa käytetty perusrakenne. Näihin kansioihin lähdekoodi järjestyy.
- **Create Activity,** pääluokka joka ladataan ensimmäisenä kun ohjelmaa ajetaan.
- **MinSDK-versio,** määritetään pienin API-level, joka laitteessa on oltava. (5.)

Lopuksi painetaan ”Finish”-painiketta, jolloin projektin tiedostot ilmestyvät Package Explorer -ikkunaan. Avataan sieltä muokkaukseen HelloAndroid.java-tiedosto ja tehdään sinne pieni muutos, jotta nähdään, toimiiko kääntäminen ja linkittäminen. Kun lähdekoodia on muokattu, painetaan valikosta Run → Run. Android Device Chooser -ikkunasta valitaan käynnissä oleva Android-laite, joka on joko virtuaalinen laite eli emulaattori tai koneeseen kiinnitetty, fyysinen Android-laite. Tässä tapauksessa käytetään oikeaa Android-laitetta. Valitaan se listasta ja painetaan ”Ok”-painiketta, jolloin ohjelma ilmestyy laitteen ruutuun. Android-laitteesta on muutettava pari asetusta ennen ajamista. (5.)

Android-laitteessa avataan Applications → Settings → Applications → Development-valikko ja sieltä asetetaan ”USB debugging” -valintaruutu aktiiviseksi. Jos esimerkiksi halutaan muuttaa GPS-dataa käsin, rastitaan myös ”Allow Mock Locations”. Laite kiinnitetään PC:hen yleensä miniUSB- tai microUSB-kaapelilla. Kun laitteen ajurit on asennettu PC:hen, sen pitäisi näkyä Eclipsen laitelistassa. (5.)

4.2 Projektin luonti

Projekti luotiin edellä mainitulla tavalla. Projektin tiedot:

- **Project Name:** MapsDemo
- **Build Target:** Google APIs 2.2 API Level 8
- **Application Name:** MapsDemo
- **Package Name:** com.example.android.apis
- **Create Activity:** MapsDemo.

Ympäristöön luodaan automaattisesti

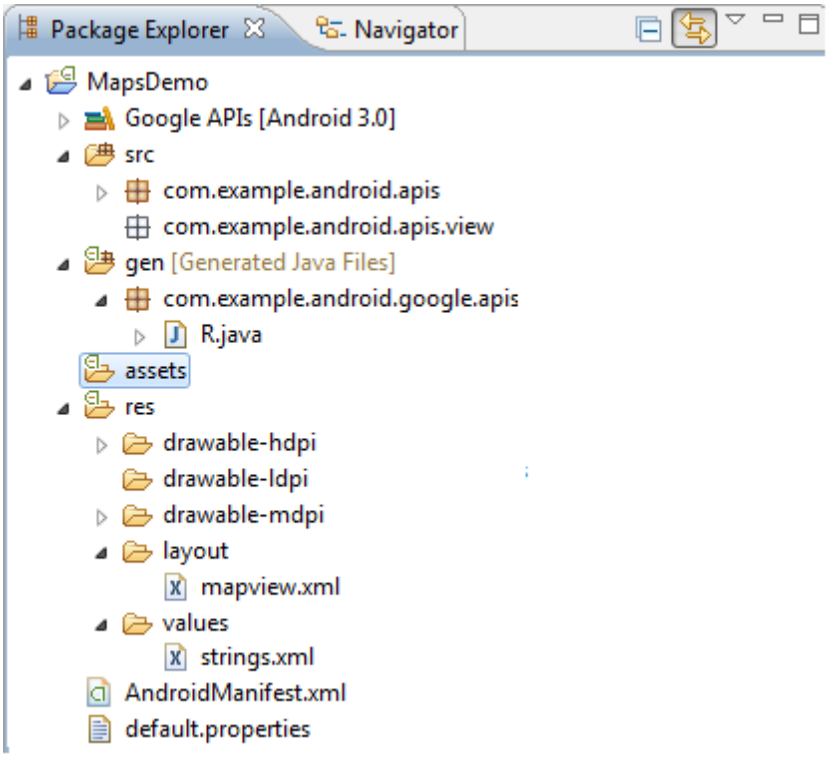
- MapsDemo- luokka ja lähdekooditiedosto MapsDemo.java
- Package-nimen mukainen hakemistorakenne \src\com\example\android\apis, tänne kerätään kaikki lähdekoodi
- Main.xml-ulkoasutiedosto, jossa oletuksena yksi TextView-elementti, ”Hello World, MapsDemo!”
- Strings.xml, tiedosto johon kerätään kaikki projektissa käytetyt resurssit
- R.java-tiedosto, joka sisältää viittaukset kaikkiin resursseihin ja komponentteihin. Sisältö aluksi:

```
package com.example.android.apis;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
```

```
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

- AndroidManifest.xml-tiedosto, johon määritellään mm. sovelluksella olevat oikeudet ja versiointiasiat.

Projektin luonnissa syntynyt hakemistorakenne tiedostoineen näkyy Package Explorer-ikkunassa (kuva 6).



KUVA 6. Luodun projektin hakemistorakenne

4.3 Ulkoasupohjan toteutus

Sovelluksen toteutus aloitettiin projektin luonnin jälkeen käyttöliittymän ulkoasupohjan toteutuksella.

4.3.1 Layout ja Layout-parametrit

Androidissa Layout on eräänlainen näkymäpohja, johon käyttöliittymäkomponentit kuten painikkeet ja tekstikentät asetellaan. Layouteja on useita erilaisia, useimmin käytettyjä lienevät LinearLayout ja RelativeLayout. Tässä sovelluksessa käytetään myös TableLayout- ja ScrollView-näkymää. LinearLayoutissa layoutin ”lapset” (engl. child) eli siinä kiinni olevat käyttöliittymäkomponentit tai muut layoutit asettuvat näkyviin vierekkäin joko pysty- tai vaakasuunnassa. RelativeLayoutissa jokaiselle siinä kiinni olevalle käyttöliittymäkomponentille voi määrittää erikseen paikan joko suhteessa layoutiin tai toisiin siinä kiinni oleviin komponentteihin. TableLayoutiin kiinnitetyt komponentit sijoitellaan taulukkoon, riveille ja sarakkeisiin. ScrollView on vieritettävä (engl. scrollable) näkymä. Tätä näkymää on pakko käyttää, jos kaikki komponentit eivät mahdu ruutuun yhtä aikaa.

Jokaiselle layoutille voi määrittää erikseen parametrit, joilla komponentit asemoidaan. Näitä kutsutaan LayoutParams-nimellä. Parametreissa voidaan eri layouteissa määritellä mihin kohtaan layoutia komponentti asettuu.

Yleisimpiä arvoja komponenttien lisäämisessä ovat FILL_PARENT ja WRAP_CONTENT. Jos esimerkiksi Button-widgetin leveydeksi annetaan FILL_PARENT, sen koko leveyssuunnassa on sama kuin sen omistavan layoutin leveys. Jos layoutin korkeus on WRAP_CONTENT, se on yhtä korkea kuin siinä olevien komponenttien korkeus yhteensä.

4.3.2 Ulkoasupohja

Käyttöliittymän pohjalle tulee mapLayout, johon kartta kiinnitetään. Kaikki muut layoutit ja komponentit tulevat näkyville kartan päälle. Näkymä luodaan dynaamisesti, eli suunnitteluun ei käytetä xml-tiedostoa. Näkymän luomista .xml-tiedoston avulla käytetään tässä projektissa lähinnä dialogi-ikkunoiden luonnissa, mutta siitä lisää myöhemmin.

Käyttämällä edellisen kappaleen tietoja layouteista luodaan pohja käyttöliittymälle. Lisätään layoutit eri värisinä selkeyden vuoksi. Esimerkkinä ulkoasupohjan toteutuksesta luodaan alimmainen näkymä, mapLayout, johon kiinnitetään ja asemoidaan oikealla käyttöliittymässä näkyvä buttonsLayout. (6.)

```
final RelativeLayout mapLayout = new RelativeLayout(this);
mapLayout.setBackgroundColor(Color.YELLOW);
```

Oikealle kartan päälle lisätään buttonsLayout ja määritellään asemointiparametrit. ButtonsLayout on LinearLayout-tyyppinen. Asetetaan komponenttien järjestäytyminen (engl. orientation) pystysuuntaiseksi ja taustaväriksi vihreä. (6.)

```
final LinearLayout buttonsLayout = new LinearLayout(this);
buttonsLayout.setOrientation(LinearLayout.VERTICAL);
buttonsLayout.setBackgroundColor(Color.GREEN);
```

Seuraavaksi määritellään parametrit, joiden perusteella buttonsLayout kiinnitetään pohjalayoutiin (6).

```
// leveys = 100 px , korkeus = FILL_PARENT
RelativeLayout.LayoutParams buttonsLayoutParams =
       new RelativeLayout.LayoutParams(100,
       RelativeLayout.LayoutParams.FILL_PARENT);

// asemointi oikealla suhteessa karttanäkymään
buttonsLayoutParams.addRule(RelativeLayout.ALIGN_PARENT_RIGHT);
```

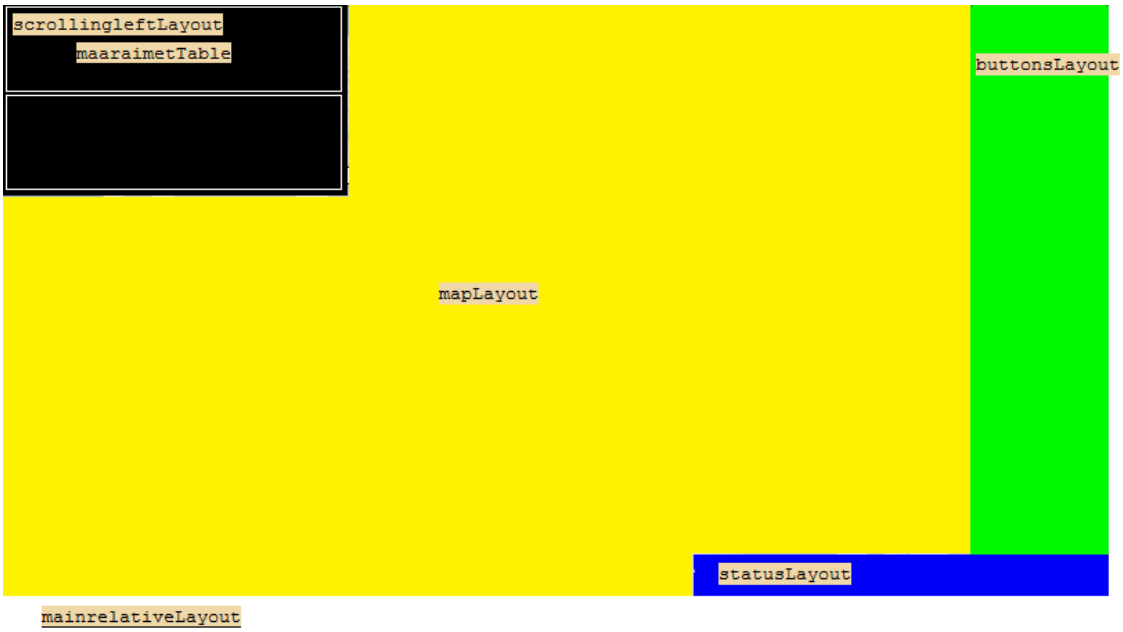
Lopuksi kiinnitetään buttonsLayout isäntäänsä mapLayoutiin. AddView()-metodille annetaan parametreina kiinnitettävä buttonsLayout ja sen asemointiparametrit (6.).

```
// buttonsLayout kiinni karttaan
mapLayout.addView(buttonsLayout, buttonsLayoutParams);
```

Muut näkymään tulevat layoutit määritellään samalla tavalla ja kiinnitetään mapLayoutiin. Lopuksi asetetaan sovelluksen päänäkymäksi mapLayout, joka vie mukanaan näkyviin muutkin sisältämänsä layoutit (6).

```
super.setContentView(mapLayout);
```

Kuvassa 7 kuvataan näkymä, kun kaikki layoutit on luotu ja asetettu näkyviin.



KUVA 7. Layoutit käyttöliittymässä

4.4 Käyttöliittymäkomponentit

Kun käyttöliittymän pohja on rakennettu, voidaan alkaa lisätä sinne painikkeita ja tekstikenttiä ja muita komponentteja. Android-käyttöliittymän rakentamiseen on käytettävissä hyvät peruskomponentit. Tässäkin projektissa käytetään muutamia yksinkertaisia käyttöliittymäkomponentteja eli UI-widgetejä.

4.4.1 Label

Yksinkertaisin käyttöliittymäkomponentti on ei-muokattava tekstikenttä eli label. Android-ohjelmoinnissa labelia kutsutaan TextView-komponentiksi. Luodaan sovelluksen käyttöliittymään statuslabel-niminen tekstikenttä, annetaan sille ominaisuudet ja kiinnitetään se ulkoasupohjaan:

```
 TextView statuslabel = new TextView(this); //luodaan TextView-
objekti
```

```
 statuslabel.setTextSize(12); //tekstin koko
statuslabel.setText("STATUS"); //teksti
statuslabel.setTextColor(Color.WHITE); //tekstin väri

statusLayout.addView(statuslabel);//kiinnitys layoutiin
```

HUOM! Jos komponenttia ei kiinnitetä mihinkään käyttöliittymän layoutiin, se ei näy käytöliittymässä. Komponentti kiinnitetään layoutiin kutsumalla layoutin addView()-metodia. (7.)

4.4.2 Button

Button on käyttöliittymässä oleva painike. Luodaan ”Sulje”-button, tehdään sille toiminnallisuutta ja kiinnitetään se käyttöliittymäpohjaan. Kun painiketta painetaan, sovellus suljetaan.

```
Button exitButton = new Button(this);// Luodaan button-objekti
exitButton.setText("Sulje");// asetetään painikkeen teksti
buttonsLayout.addView(exitButton);// kiinnitys layoutiin
```

Liitetään painikkeeseen kosketuksenkuuntelija, joka signaloituu, kun painiketta kosketaan. Sen sisälle voidaan määrittää ohjelmakoodi, joka suoritetaan, kun nappia kosketaan.

```
exitButton.setOnTouchListener(new OnTouchListener() {
public boolean onTouch(android.view.View v, MotionEvent event)
{
        // Suljetaan sovellus
android.os.Process.killProcess(android.os.Process.myPid());
return false;
}
});
```

Jatkossa tähän komponenttiin viitataan tässä dokumentissa nimellä Button-widget. Button-widgetistä on myös olemassa variaatio, jossa painike on kuva. Tämän komponentin nimi on ImageButton. (7.)

4.4.3 CheckBox

Valintaruutu-widget voi olla valittuna tai ei. Toteutetaan paikannus-valintaruutu. Paikannus-valintaruudulla hallitaan paikannuslippua, jota tarvitaan myöhemmin. Kun valintaruutu on valittuna, paikannuslippu menee arvoon ”true”, jos valintaruutu ei ole valittuna, paikannusarvo on ”false”. Lippuarvot ovat yleisiä ohjelmoinnissa. Yleensä niihin viitataan termillä boolean-arvo.

```
boolean paikannus = true;// Paikannuslippu, alussa true arvoon

paikannuscb = new CheckBox(this);// Luodaan checkbox
paikannuscb.setTextSize(13);     // Tekstin koko
paikannuscb.setText("Paikannus");// Teksti
paikannuscb.setTextColor(Color.BLACK);// Tekstin väri
paikannuscb.setChecked(true);   // asetetaan valituksi (koska
paikannus- lippu on true)
buttonsLayout.addView(paikannuscb);// kiinnitys layoutiin
```

Liitetään valintaruutuun kuuntelija, joka signaloituu, kun valinta ruutu vaihtaa tilaansa, esimerkiksi kun valinta poistetaan. Valinnan vaihtuessa tarkistetaan tila ja muutetaan lipun arvoa.

```
paikannuscb.setOnCheckedChangeListener(new OnCheckedChangeLis-
tener()
{
 public void onCheckedChanged(CompoundButton arg0, boolean
arg1) {
       // checbox on valittuna kun isChecked() palauttaa true
       if ( paikannuscb.isChecked() == true )
       {
          paikannus = true;
       }
       // muuten isChecked() palauttaa false
       else
       {//focus pois
          paikannus = false;
       }
       });
```

Jatkossa tähän komponenttiin viitataan tässä dokumentissa nimellä Check-Box-widget tai pelkkä checkbox. (7.)

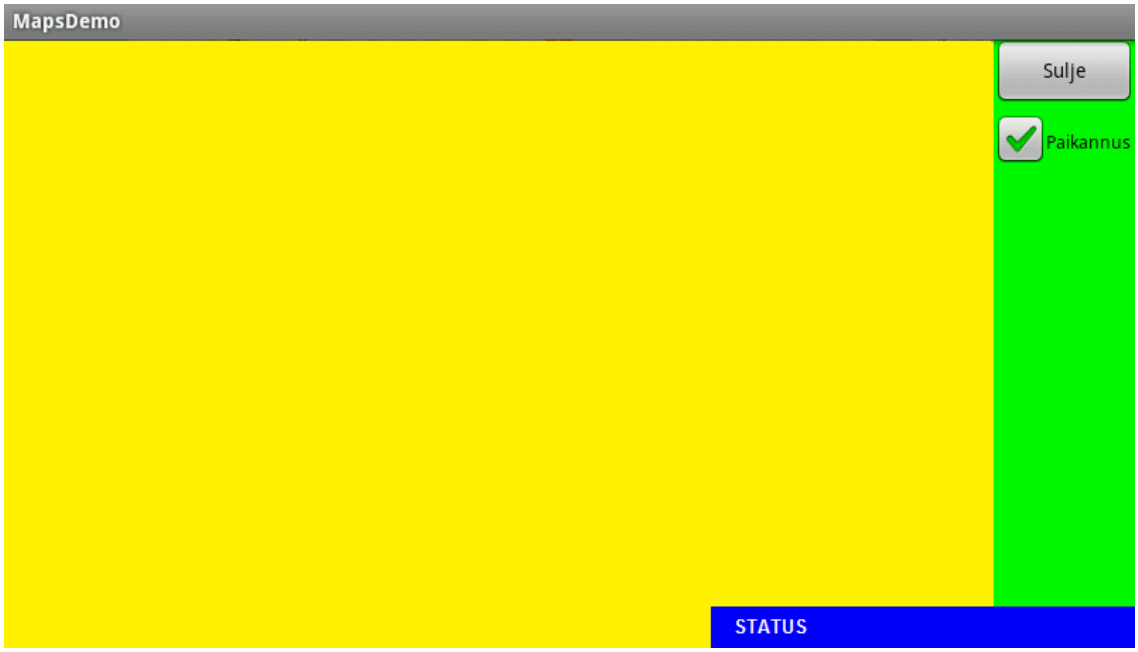
4.4.4 EditText

EditText on yksinkertainen tekstikenttä, johon käyttäjä voi syöttää merkkijonoja. Tässä sovelluksessa tätä komponenttia käytetään mm. kirjautumisdialogin yhteydessä, kun käyttäjä syöttää käyttäjätunnuksensa ja salasanansa sovellukseen. Androidissa näppäimistö aktivoituu automaattisesti, kun käyttäjä koskee EditText-komponenttia, eikä sille tarvitse erikseen kirjoittaa kuuntelijaa kuten Buttonille. (7.)

4.4.5 ImageView

ImageView-widgetin avulla käyttöliittymään voi lisätä kuvan (7). Tätä komponenttia ei esitellä sen tarkemmin, koska sitä ei käytetä luotavassa näkymässä.

Kuvassa 8 on näkymä komponenttien ja layoutien luonnin jälkeen.



KUVA 8. Näkymä, kun layoutit ja komponentit on luotu.

4.5 Palvelinpyynnöt

Sovellus tarvitsee useassa toiminnossaan tietoa palvelimelta. Sovelluksen pitää mm. ilmoittautua ja lähettää omaa paikkatietoaan palvelimelle, hoitaa autentikointi ja päivitykset sekä suorittaa määräintenhallintaa palvelimen avulla. Palvelinpyynnöissä käytetään JSON-formaattia. Rakennettavia lohkoja testattiin lähettelemällä sovelluksesta pyyntöjä ja käsittelemällä niiden palauttamia arvoja.

4.5.1 JSON (JavaScript Objecy Notation)

JSON on Douglas Crockfordin kehittämä kevyt tiedonsiirtostandardi. Lausekieliset avain-arvoparit voidaan koostaa JSON-objekteiksi, joita voidaan lähetellä palvelimen ja asiakasohjelman välille. JSON ymmärtää perusdatatyyppejä, kuten numeroita, merkkijonoja, boolean-arvoja, taulukkoja ja objekteja. JSON tarjoaa kevyen vaihtoehdon XML-merkintäkielelle (8).

Alla on JSON-syntaksiesimerkki henkilöobjektista, jossa etu- ja sukunimi ovat merkkijonoja, ikä on numero, osoite on objekti ja puhelinnumerot ovat taulukossa olevia objekteja. (8.)

```
{       "etuNimi": "Teppo",
        "sukuNimi": "Testaaja",
        "ika": 25,
        "osoite":
        {
                "katuOsoite": "Ouluntie 7",
                "postiNumero": "90520"
      "kaupunki": "Oulu",

        },
        "puhelinNumero":
        [
                {
                   "tyyppi": "koti",
                   "numero": "088123456"
                },
                {
                   "tyyppi": "kanny",
                   "numero": "0501234567"
                }
        ]
 }
```

JSON-objektien koostamiseen ja purkamiseen (engl. parsing) on olemassa luokkia ja funktioita useilla eri ohjelmointikielillä. Koostamisella tarkoitetaan esimerkiksi, että JSON-syntaksia noudattava merkkijono muunnetaan suoraan JSON-objektiksi, ja purkaminen sitä, että JSON-luokan funktioilla irrotetaan haluttu arvo sen vastinavaimen perusteella. (8.)

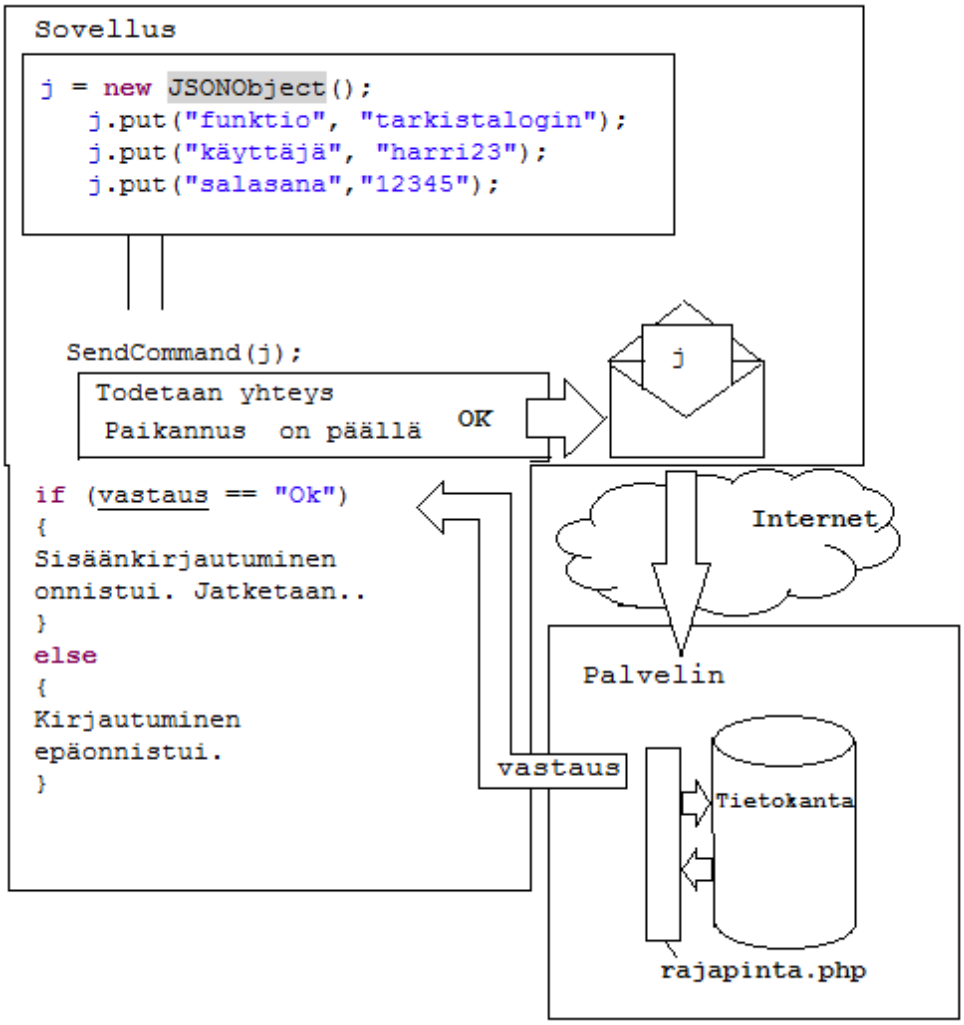
4.5.2 Palvelimelle lähetettävä tieto ja vastauksen käsittely

Palvelinpyyntöjen lähettämiseen toteutettiin SendMyCommand()-funktio, joka tarkistaa ennen lähetystä, onko paikannus (oman paikkatiedon lähetys) päällä ja onko laitteesta yhteyttä palvelimeen. Jos on, lähetettävät tiedot koostetaan JSON-objektiksi ja lähetetään HTTP-POST-funktion avulla palvelimelle ja palvelin lähettää vastauksen String-muodossa ohjelmaan takaisin. Alla esimerkki SendMyCommand()-funktion toiminnasta:

Esimerkissä on kyse kirjautumisen tarkistamisesta. Luodaan JSON-objekti, joka sisältää kolme avain-arvoparia. Objekti annetaan parametrina SendMyCommand()-funktioon. Funktiossa tarkistetaan, onko lähetys päällä ja onko

laite yhteydessä Internetiin. Jos ehdot täyttyvät, lähetetään objektin sisältävä pyyntö palvelimelle, jossa tarkistetaan, onko käyttäjä-salasanapari tietokannassa eli onnistuuko kirjautuminen. Palvelin lähettää vastauksena ”Ok”-merkkijonon, jos kirjautuminen onnistuu.

SendMyCommand()- funktio lukee vastauksen ja muuttaa sen String-muotoon. Vastauksen perusteella tehdään jatkotoimenpiteet ohjelmassa (kuva 9).



KUVA 9. Kirjautumisen tarkistaminen ja käsittely

Kun pyyntöjä lähettävä SendMyCommand()-funktio oli rakennettu ja sen toiminta testattu opinnäytetyössä, voitiin siirtyä rakentamaan JSON-parseri-funktiota.

4.5.3 JSON-parseri funktio

Sovellus vastaanottaa osan tiedosta JSON-stringinä eli merkkijonona, joka noudattaa JSON-lausekielen syntaksia. Esimerkiksi:

```
String maarainjsonstring = {"id":"1","status":"TÄRKEÄ"};
```

Jotta sovellus voisi irrottaa JSON-stringeistä tietoa, toteutettiin myös ParseJsonString()-funktio. Funktiolle annetaan parametrina vastaanotettu JSON-stringi ja avain, jonka arvo halutaan. Funktio koostaa JSON-stringin JSON-objektiksi ja irrottaa objektista halutun avaimen arvon ja palauttaa sen. Tämän jälkeen se voidaan tallentaa muuttujaan.

Esimerkiksi: Sokettiin tulee määräin JSON-stringinä. Irrotetaan määräimen id muuttujaan ParseJsonString()-funktion avulla. ParseJsonString() on itse rakennettu funktio, jolla avaimen arvo saadaan irti, ja funktio sisältää myös virheen kiinniottavan lohkon, eli jos haluttua arvoa ei löydy, virhe saadaan kiinni.

```
String maaraimen_id = ParseJsonString(maarainjsonstring, "id");
```

4.6 Paikallisten tietojen tallennus

Melkein kaikissa sovelluksissa on pakko tallentaa käyttäjäkohtaisia tietoja, tunnuksia ja muita sellaisia tietoja, jotka halutaan säilyvän senkin jälkeen, kun sovellus suljetaan (sovelluksen tila, kirjautunut käyttäjä jne.). Dataa voi tallentaa kohdelaitteeseen useilla eri tavoilla, esimerkiksi sisäiseen tietokantaan, muistikortille erilliseen tiedostoon tai laitteen muistiin.

Tässä sovelluksessa käytetään Androidin SharedPreferences-luokkaa, jonka avulla voi tallentaa ja ladata erityyppisiä avain-arvopareja. Tiedot säilyvät vaikka sovellus suljettaisiin. Lähdekoodi ja tiedot on saatu pääosin Android Developers -sivustolta (9).

SharedPreferences otettiin käyttöön seuraavalla tavalla. Ensin määritellään luokan sisälle asetusten tiedoston nimi:

```
public static final String PREFS_NAME = "Settings";
```

Sitten haetaan Settings-objektiin tiedoston mukaisen muokattavat asetukset.

```
SharedPreferences settings = getSharedPreferences(PREFS_NAME,
0);
```

Tiedoston nimeä ei tarvitse määrittää, jos käytetään vain yhtä tiedostoa, vaan silloin settings-objektin arvo haetaan getPreferences()-funktiolta. Jos esimerkiksi on mahdollista, että sovellukseen tulee usean käyttäjän tuki, silloin kaikille käyttäjille tulee oma asetus-tiedosto, ja asetukset haetaan käyttäjäkohtaisesti:

```
SharedPreferences settings = getSharedPreferences("user1", 0);
```

Tähän projektiin valittiin getSharedPreferences()-funktio, koska jatkossa saatetaan tarvita useita asetustiedostoja eri käyttäjille.

Seuraavaksi haetaan SharedPreferences Editor-objekti, jolla tietoja voi jatkossa muokata. Edit()-funktio palauttaa editorin.

```
SharedPreferences.Editor editor = settings.edit();
```

Seuraavaksi toteutettiin yksinkertaiset asetusten luku- ja kirjoitusfunktiot:

```
private String ReadSettings(String key) {
        String value = "-1";
        // hakee asetuksista avainta (key) vastaavan arvon (va-
lue)
        value = settings.getString(key, value);
        // jos arvoa ei löydy palautetaan ”-1”
return value;

        }
```

```
private void WriteSettings(String key ,String value) {
       // kirjoitetaan editor-muokkaajalla avain-arvo-pari ase-
tuksiin
         editor.putString(key, value);
       // muutosten vahvistus commit()-funktiolla
         editor.commit();
       }
```

Jatkossa asetuksia voi lukea ja kirjoittaa milloin tahansa kutsumalla:

```
String myname = ReadSettings("username"); //luku
WriteSettings("username","jenna"); //username-arvoksi ”jenna”
```

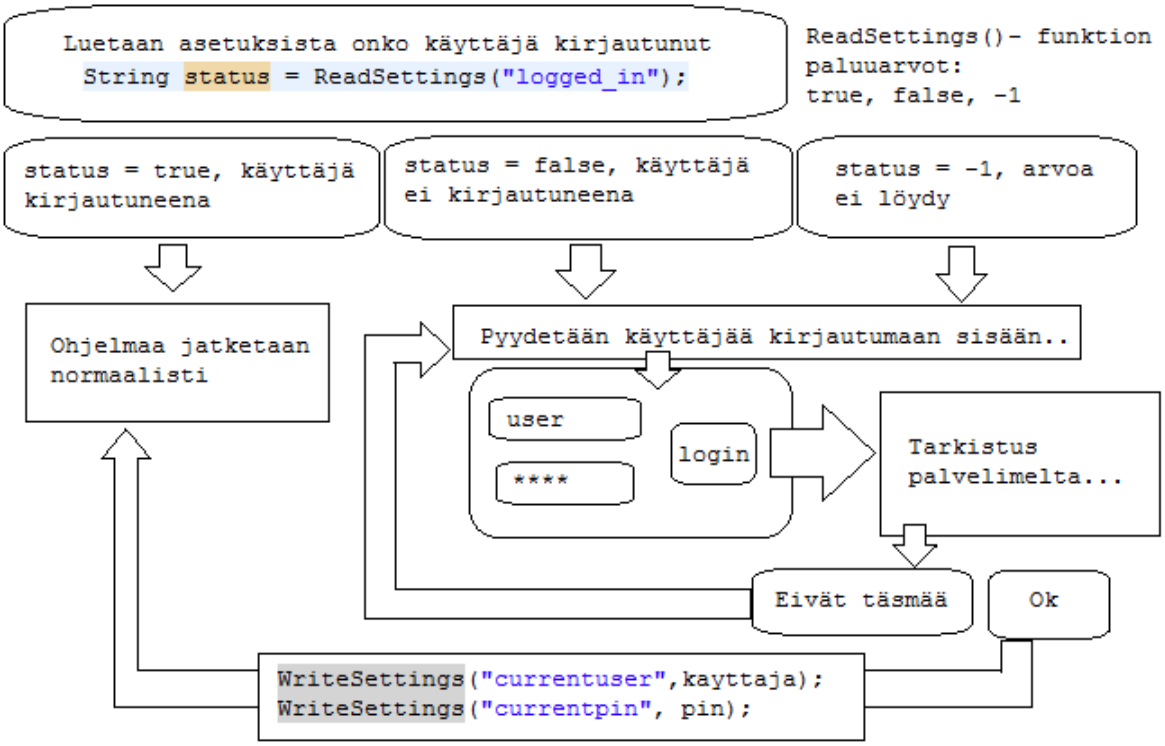
4.7 Käyttäjän autentikointi

Sovelluksen alussa on tarkoitus autentikoida eli tunnistaa käyttäjä. Sovelluksen alkuun luotiin seuraavanlainen tarkistusfunktio ja ensimmäinen dialogi eli vuoropuhelu käyttäjän kanssa.

Funktio lukee paikallisista asetuksista, onko kukaan kirjautunut käyttämään sovellusta. Kirjautumisen tarkistavassa funktiossa käytetään ns. lippuarvoa (engl. boolean). Lippuarvo voi saada kaksi erilaista arvoa, ”true” tai ”false” eli tosi tai epätosi. Jos logged_in-arvon kohdalle on asetustiedostoon kirjoitettu ”true”, käyttäjä on kirjautuneena, jos ”false”, käyttäjä ei ole kirjautuneena. Kun arvo on ”false”, näytetään käyttäjälle kirjautumisdialogi, jossa kysytään käyttäjältä käyttäjänimi ja salasana. Sitten käyttäjä tunnistetaan palvelimelle lähetettävällä pyynnöllä.

Palvelimelle lähetetään käyttäjän käyttäjänimi ja salasana, ja sitten tarkistetaan, löytyykö kyseinen käyttäjä-salasanapari tietokannasta. Jos löytyy, kirjautuminen onnistuu ja ohjelmaa jatketaan normaalisti. Mikäli kyseessä on ensimmäinen kirjautuminen laitteessa, logged_in-arvoa ei ole vielä asetustiedostossa. Silloin asetusten luku-funktio palauttaa arvon ”-1”. Tämä pitää käsitellä samalla tavoin kuin ”false”-arvo. Kun käyttäjä-dialogi näytetään, asetustiedostoon kirjoitetaan ensimmäistä kertaa, ja arvot alustuvat. Logged_in-lippuarvo alustetaan viimeistään, kun kirjaudutaan sisään. Silloin asetustiedostoon kirjoitetaan logged_in-arvoksi (asetetaan käsin) ”true”. Kun

käyttäjä kirjautuu ulos kirjoitetaan logged_in-arvoksi ”false”. Näin luetaan ja tarkistetaan oikea arvo asetustiedostosta (kuva 10).



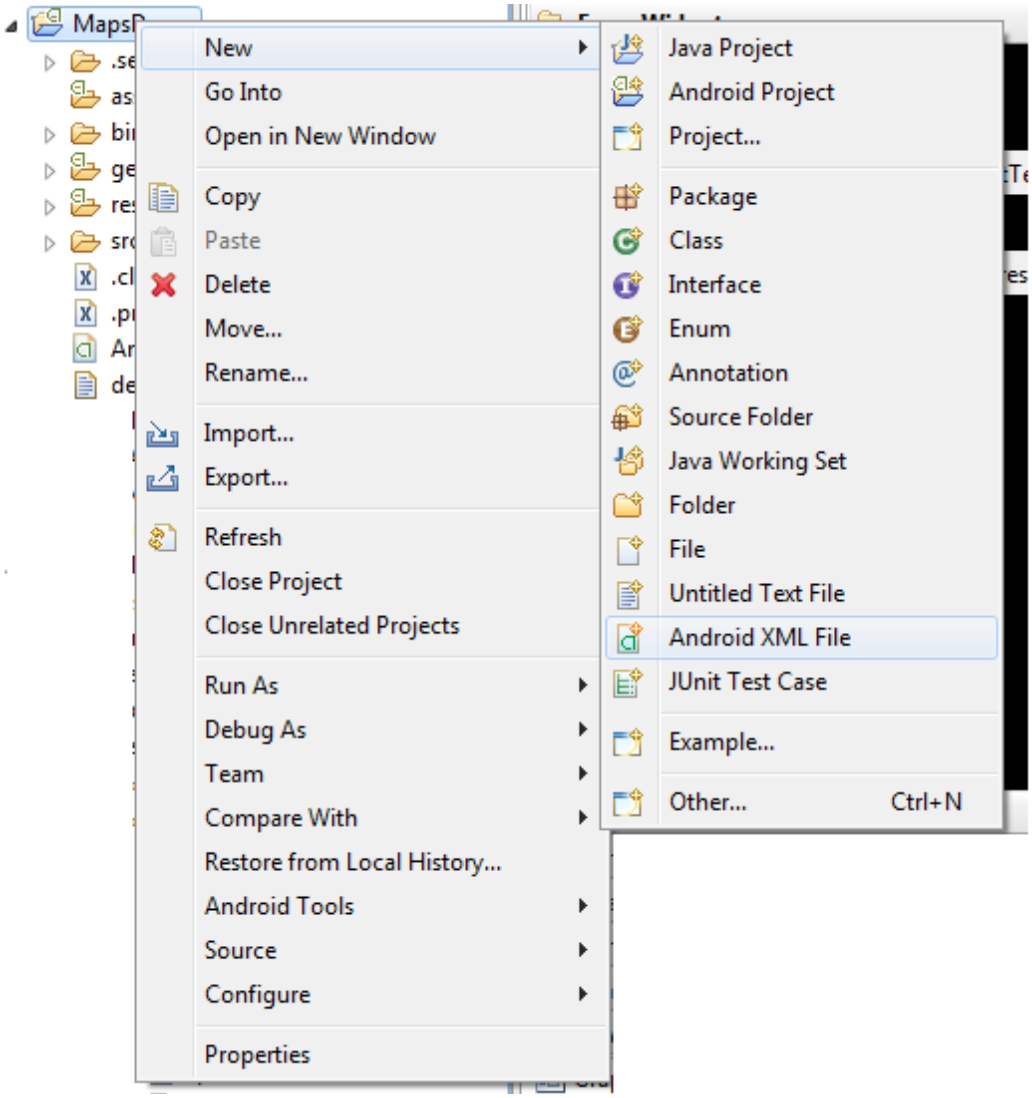
KUVA 10. Käyttäjän kirjautumistilan tarkistus ja autentikointi

4.7.1 Dialogin luonti ja näyttäminen

Tässä sovelluksessa käytetään useita eri dialogeja. Dialogeja voi luoda joko dynaamisesti tai luomalla dialogille oma ulkoasutiedosto. Dialogi on ikkuna, jonka avulla sovellus pyytää käyttäjää syöttämään tietoja, tai sillä ilmoitetaan, varoitetaan tai muuten kerrotaan käyttäjälle tapahtumista sovelluksessa. Muokatut lähdekoodit ja ohjeet ovat Android Developers -sivustolta (10.).

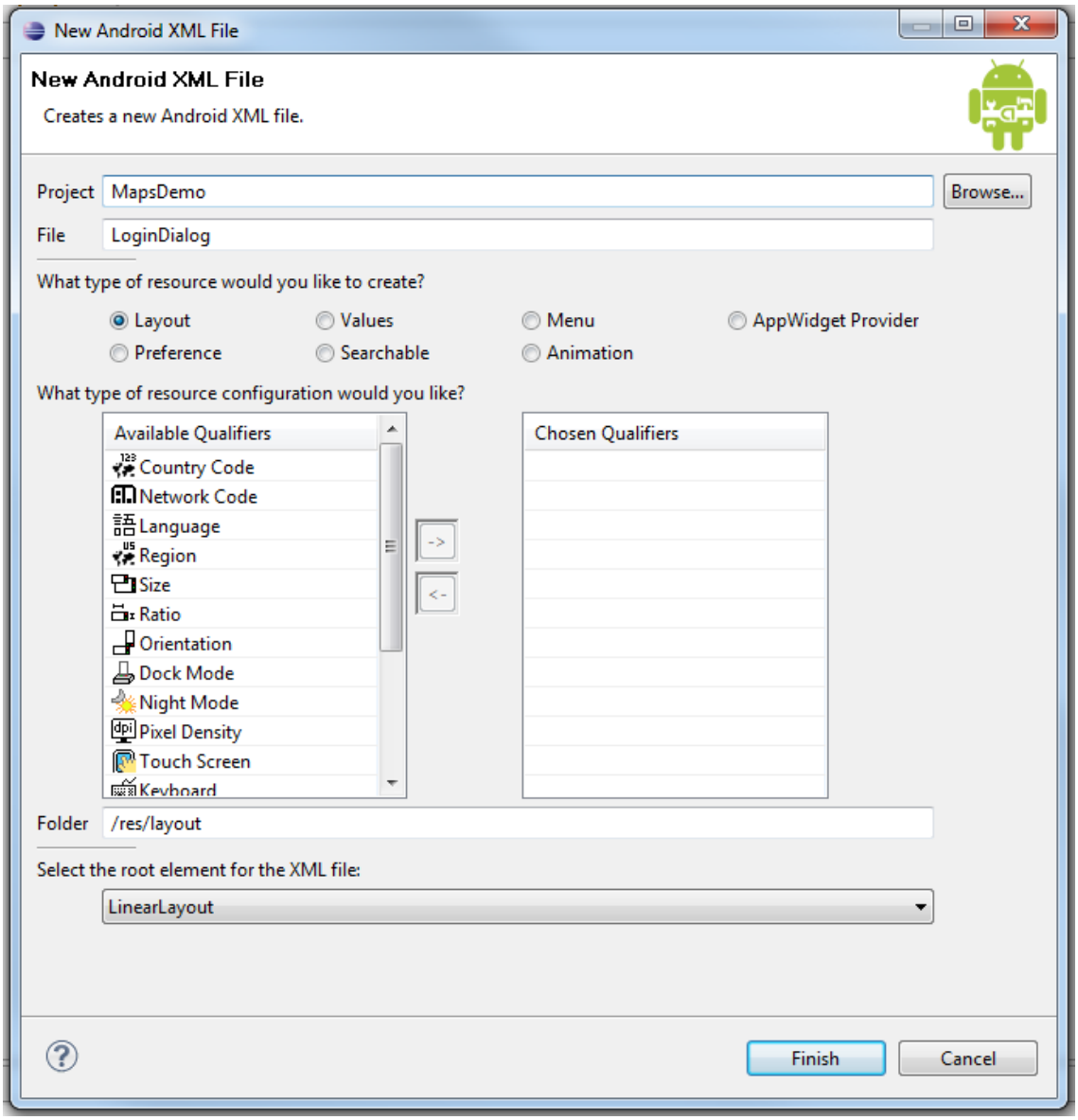
Dialogin luonti ulkoasutiedoston avulla

Oma ulkoasutiedosto luodaan painamalla hiiren oikealla näppäimellä projektitiedostoa ilmestyvästä tiputusvalikosta New... → Android XML File (kuva 11).



KUVA 11. Uuden ulkoasutiedoston luominen

Ilmestyvässä dialogissa annetaan tiedostolle nimi ja resurssityypiksi Layout. Luodaan esimerkkinä kirjautumisdialogi (kuva 12).



KUVA 12. Kirjautumisdialogin luominen Eclipsessä.

Muokataan .xml-tiedostoa. Lisätään widgetit omina lohkoinaan .xml-tiedostoon. Niille voidaan antaa eri parametreja, kuten korkeus ja leveys, asemointiparametrit, teksti ja niin edelleen. Tärkein kuitenkin on ID-parametri, jolla komponentti yhdistetään ohjelmakoodiin. Alla oleva .xml-tiedosto kuvaa kirjautumisdialogia, jossa pohja-layout on LinearLayout-tyyppinen. Sen sisällä on RelativeLayout, jonka avulla komponentit asemoidaan toisiinsa ja isäntäänsä (engl. parent) nähden. Parent on tässä tapauksessa koko ruudun täyttävä RelativeLayout ja esim. login-button asemoidaan

pystysuunnassa oikealle keskelle. LinearLayout määrää, miten komponentit asettuvat luodessa ruudulle. Jos orientation-parametri on ”vertical” komponentit asemoituvat pystysuunnassa ruudulle (kuva 13).

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
android:id="@+id/widget30"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<RelativeLayout
android:id="@+id/widget35"
android:layout_width="316px"
android:layout_height="128px"
>
<Button
android:id="@+id/LoginButton"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Kirjaudu"
android:layout_centerVertical="true"
android:layout_alignParentRight="true"
>
</Button>
<Spinner
android:id="@+id/userSpinner"
android:layout_width="148px"
android:layout_height="wrap_content"
android:layout_above="@+id/pinedit"
android:layout_alignParentLeft="true"
>
</Spinner>
<EditText android:id="@+id/pinedit" android:layout_width="148px"
android:layout_height="wrap_content" android:textSize="18sp"
android:password="true"
android:text="1234"
android:layout_alignParentBottom="true" android:layout_alignParentLeft="true">
</EditText>

</RelativeLayout>
</LinearLayout>
```

KUVA 13. Muokattu XML-tiedosto

Seuraavaksi luodaan ShowMyDialog()-funktio, jossa resurssit yhdistetään ohjelmakoodiin ja niille lisätään käsittelijät. Käsittelijässä määritellään, mitä tapahtuu, kun nappia painetaan.

Luodaan dialog-objekti, ja asetetaan setContentView()-funktiolla äsken luotu logindialog.xml dialogin ulkoasuksi. SetTitle()-funktiolla asetetaan dialogille otsikkoteksti, ja setCancelable(false) tarkoittaa, että dialogia ei voi perua laitteen back-buttonilla.

```
Dialog logindialog = new Dialog(this);
logindialog.setContentView(R.layout.logindialog);
logindialog.setTitle("Kirjaudu sisään:");
logindialog.setCancelable(false);
```

Luodaan login-button ja käsittelijä.

```
Button loginbutton =(Button)   logindia-
log.findViewById(R.id.LoginButton);

loginbutton.setOnTouchListener(new OnTouchListener() {

public boolean onTouch(View v, MotionEvent event) {

//Tänne ohjelmakoodi, joka ajetaan kun login-buttonia painetaan
//eli tässä tehtäisiin pyyntö palvelimella jolla käyttäjä au-
tentikoidaan


        return false;
        }
    });
```

Kun kaikki komponentit on esitelty ja käsittelijät luotu, voidaan näyttää dialogi.

```
logindialog.show();
```

Kun kirjautuminen on todettu oinnistuneeksi, voidaan palata ohjelmaan sulkemalla dialogi.

```
logindialog.cancel();
```

Dynaamisesti luotu dialogi

Dialogi voidaan myös luoda ilman .xml tiedostoa. Tällaista luontia kannattaa käyttää silloin, kun kyseessä on dialogi, jossa on vain pari komponenttia. Komponenttien luonnissa ei viitata .xml-tiedostoon vaan parametrit asetetaan dynaamisesti. Myös Layoutit pitää luoda käsin. Esimerkkinä on dynaamisesti luotu ”Lisää käyttäjä” -dialogi, jossa on vain tila nimen kirjoittamiseen ja ”Tallenna”-button:

```
Dialog adduserdialog = new Dialog(MapsDemo.this);
adduserdialog.setTitle("Uusi käyttäjä:");
adduserdialog.setCancelable(true);

    // Pohjalayout luodaan dynaamisesti
    final LinearLayout dlgLayout = new LinearLayout(this);

  // Nimi-EditText, oletustekstinä ”Uusi”
    final EditText name = new EditText(this);
    name.setText("Uusi");

// tallenna button ja kuuntelija
    Button okbutt = new Button(this);
    okbutt.setText("Tallenna");

// komponenttien kiinnitys layoutiin
    dlgLayout.addView(name);
    dlgLayout.addView(okbutt);

    okbutt.setOnTouchListener(new OnTouchListener() {

public boolean onTouch(android.view.View v,MotionEvent event) {
// ….....
 });

// !Kiinnitetään layout dialogiin ja näytetään dialogi!
 adduserdialog.setContentView(dlgLayout);
 adduserdialog.show();
```

Toast

Toast on Androidissa käytettävä pieni ilmoitusdialogi-widget. Toasteja voi näyttää Android sovelluksessa, kun omaan luokkaan tuodaan Toast-luokka.

```
import android.widget.Toast;
```

Toast näytetään seuraavasti. Toastin pitää tietää konteksti, näytettävä teksti ja näkymän kesto (10).

```
Toast.makeText(getBaseContext(),
    "Käyttäjätunnus ja salasana eivät täsmää",
        Toast.LENGTH_SHORT).show();
```

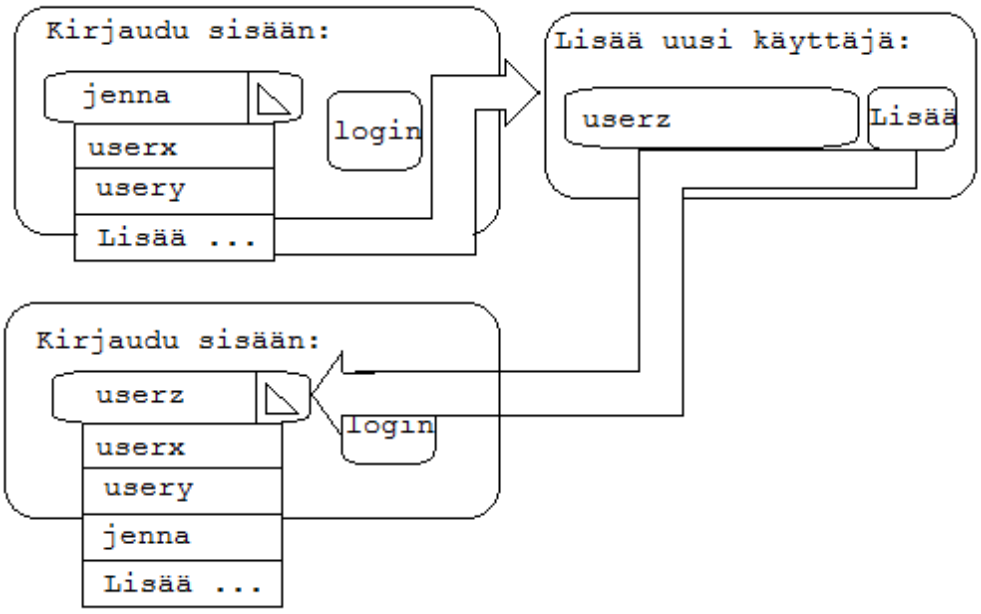
Kuvassa 14 näkyy luotu Toast.

KUVA 14. Toast

4.7.2 Käyttäjien tallennus

Sovelluksen vaatimuksena oli myös, että sovellukseen tallentuu viisi viimeistä käyttäjää. Tehtiin funktio, joka kirjautuessa tallentaa käyttäjänimet ja ne näkyvät spinnerissä. Spinner on yksi Androidin peruskäyttöliittymäkomponenteista. Spinner on Android-tyylinen tiputusvalikko. Käyttäjiä pystyy myös lisäämään.

Kun kirjautumisdialogi näytetään, edelliset käyttäjät ovat tiputusvalikossa, jossa näytetään viisi edellistä käyttäjää. Jos edellisiä käyttäjiä ei ole, tiputusvalikossa on vain oletus-alkio ”Lisää...”. Kun se valitaan tiputusvalikosta, ilmestyy ”Lisää uusi käyttäjä” -dialogi. Kun käyttäjä lisätään, kirjautumisdialogin tiputusvalikko päivittyy ja lisätty käyttäjänimi on nyt valittuna. Uusia käyttäjiä tallentuu viisi niin, että vanhin pyyhkiytyy pois listasta (kuva 15).

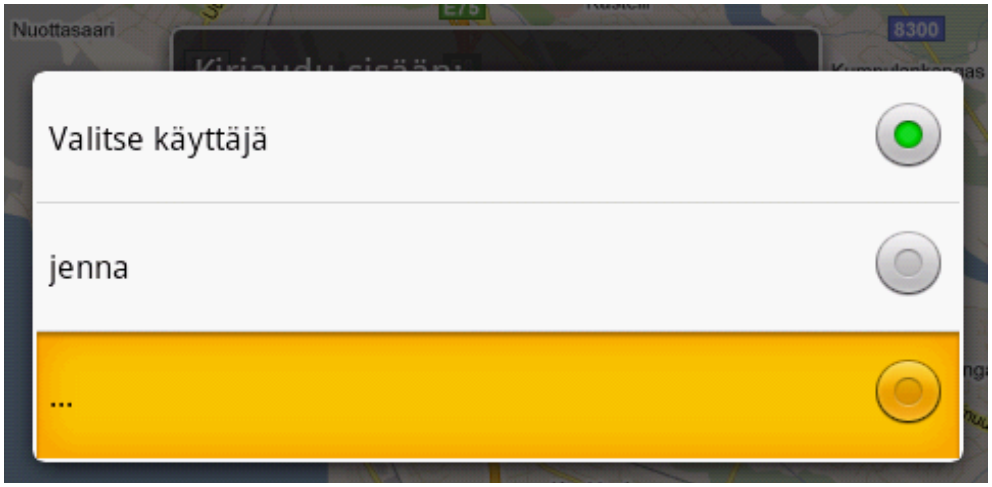


KUVA 15. Uuden käyttäjän tallennus

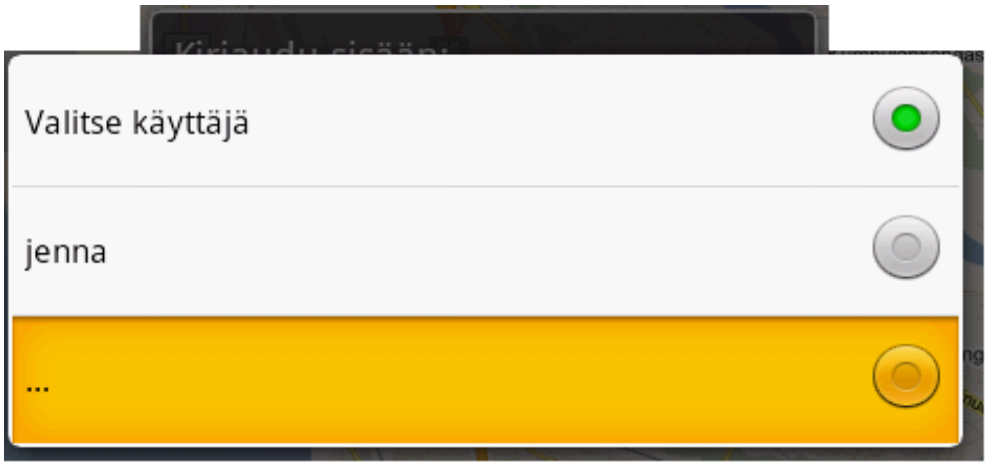
4.7.3 Näkymä

Käyttöliittymässä uuden käyttäjän tallennus näyttää seuraavalta. Kuva kun kirjautumisdialogi on auki (kuva 16), avataan tiputusvalikko (kuva 17), valitaan ”...”-alkio (kuva 18), näin avataan uusi käyttäjä dialogi (kuva 19), lisätään käyttäjä painamalla ”Tallenna”- painiketta. Kirjoitetaan kirjautumisdialogissa PIN-koodi ja kirjaudutaan sisään (kuva 20). Kirjautumisen onnistumisesta ja epäonnistumisesta ilmoitetaan käyttäjälle toastilla (kuva 21).

KUVA 16. Xml-tiedoston avulla luotu kirjautumisdialogi

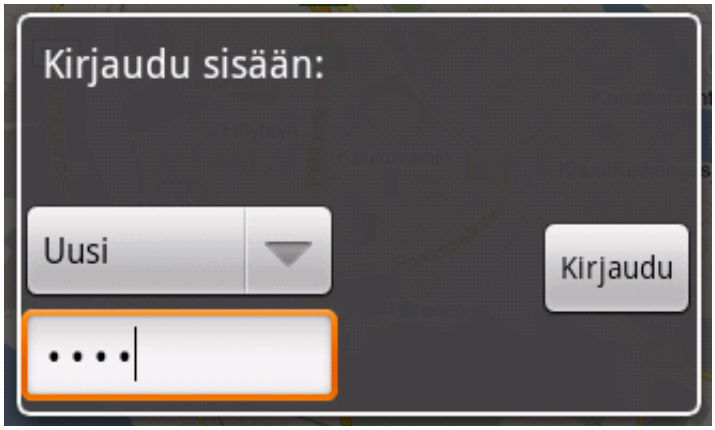


KUVA 17. Tiputusvalikko auki



KUVA 18. Uusi käyttäjä dialogi avautuu "…"-alkiota painamalla

KUVA 19. Uusi käyttäjä -ikkuna



KUVA 20. Kirjautuminen

KUVA 21. Sisäänkirjautumisen epäonnistuminen

4.7.4 Autentikointi palvelinpyyntöjen yhteydessä

Käyttäjä autentikoidaan myös jokaisen palvelinpyynnön yhteydessä, koska täytyy tietää, mikä käyttäjä ja mikä laite lähettää pyynnön palvelimelle. Esimerkiksi paikkatiedon mukana käyttäjä lähettää oman käyttäjänimensä, salasanansa ja laitteen IMEI-koodin.

4.8 Päivityksen tarjoaminen ja versiointi

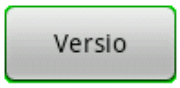
Samoin kuin kirjautumisen tila, myös saatavilla olevat päivitykset tarkistetaan alussa. Sovelluksen versionumero päivitetään AndroidManifest-tiedostoon aina, kun uusi versio julkaistaan. AndroidManifest-tiedostossa määritellään versionumero ja version nimi seuraavasti.

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.android.google.apis"
android:versionCode="1" android:versionName="versio 1.1">
```

Versionumero saadaan AndroidManifest-tiedostosta seuraavanlaisella funktiolla getVersionCode().

```
private String getVersionCode(){

String strVersionCode;
PackageInfo packageInfo;
   try {
packageInfo = getPackage-
Manager().getPackageInfo(getPackageName(), 0);
strVersionCode = String.valueOf(packageInfo.versionCode);
          } catch (NameNotFoundException e) {
           e.printStackTrace();
           strVersionCode = "Cannot load Version!";
          }
    return strVersionCode;}
```

Funktio palauttaa version koodin string-muuttujana. Palvelimelta pyydetään uusimman version versionumero ja latauslinkki. Sovelluksen versionumeroa ja palvelimelta pyydettyä versionumeroa verrataan toisiinsa, ja jos palvelimella oleva versionumero on suurempi, näytölle ilmestyy ”Versio”-button (kuva 21), jota painamalla käyttäjä voi ladata uusimman ohjelmaversion laitteeseensa.



KUVA 21. ”Versio”-button käyttöliittymässä.

Version päivitysdialogi on kuvan 22 mukainen. Osoite on piilotettu kuvasta yksityisyyssyistä.

KUVA 22. Päivitysdialogi

Kun ”Aloita päivitys” -painiketta painetaan, .apk-tiedosto (Android-sovelluspaketti) ladataan osoitteesta ja asennuspaketti tallentuu laitteen muistikortille download-kansioon. Pakettia painamalla asennus alkaa, sovellus asentuu edellisen sovelluksen päälle. Asennus ei muuta asetustiedostoja, joten käyttäjäkohtaiset asetukset säilyvät.

HUOM! Jotta laitteeseen voisi ladata ja asentaa sovelluksia jostain muualta kuin Android Marketista, laitteen Settings → Applications → Unknown sources-asetus on oltava aktiivisena.

4.9 GPS-datan kuuntelu

GPS-datan kuuntelua varten kohdelaitteeseen ja itse sovellukseen täytyy määritellä erinaisiä asetuksia.

4.9.1 Kohdeaite ja asetukset

GPS-datan kuuntelua varten Android-laitteessa täytyy olla GPS-vastaanotin paikkatiedon kuuntelua varten. Lisäksi laitteen asetukset täytyy olla oikein. Paikkatiedon vastaanottamiseen voi käyttää:

- Langattomia verkkoja,

  Settings → Location&Security → Use Wireless Networks

- GPS-satelliitteja,

  Settings → Location&Security → Use GPS satellites

- AGPS, GPS--tietoa palvelimelta,

  Settings → Location&Security → Use Assisted GPS

4.9.2 Sovellus

AndroidManifest-tiedostoon lisätään oikeus GPS-datan lukemiseen ja Internet-yhteyden muodostamiseen laitteesta. ACCESS_COARSE_LOCATION-oikeus tarkoittaa, että laite saa lukea paikkatietoa mobiililiittymän tai langattoman verkon kautta. ACCESS_FINE_LOCATION-oikeudella tarkoitetaan oikeutta lukea dataa laitteen omasta GPS-vastaanottimesta. Toteutus on tehty Android Developers -sivustolta löytyvän ohjeen mukaisesti (11).

```
<uses-permission an-
droid:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission an-
droid:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.INTERNET" />
```

Pääluokkaan lisätään Location-tyyppinen muuttuja, johon uusi paikkatieto tallennetaan, jotta sitä voidaan hyödyntää muissa sovelluksen funktioissa:

```
private Location mLocation;
```

Pääluokan, MapsDemo.java, OnCreate()-funktiossa luodaan ja alustetaan LocationManager-luokan objekti. LocationManager-luokka tarjoaa paikkatietopalvelut Androidissa. Paikkatietopalvelut haetaan objektiin getSystemService(Context.*LOCATION_SERVICE*)-funktiolla.

```
LocationManager locManager = (LocationManager) getSystem-
Service(Context.LOCATION_SERVICE);
```

Pyydetään paikkatiedon päivitystä GPS-tarjoajalta getLocationUpdates()-funktiolla. Tarjoajalla tarkoitetaan tekniikkaa jonka avulla paikkatieto haetaan, esim. AGPS:n tai GPS-satelliittien kautta haettava paikkatieto. Parametreina ilmoitetaan paikkatiedon päivitysvälit. Jos aikaa on kulunut 5000 millisekuntia, eli käytännössä paikkatieto päivitetään 5 sekunnin välein tai jos etäisyys edellisestä paikasta on 5 metriä, paikkatieto päivitetään. LocationListener-objekti annetaan myös parametrina. Siten LocationManager-objekti

tietää, että juuri kyseinen LocationListener-objekti hoitaa paikkatiedon päivittymiseen liittyvät toiminnot.

```
locManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
5000, 5, locListener);
```

Alustetaan LocationListener-luokan objekti, joka reagoi paikkatiedon muuttumiseen.

```
LocationListener locListener = new MyLocationListener();
```

MyLocationListener on luokka, joka perii Androidin LocationListener-luokan. Alla näkyvät LocationListener-luokan metodit

```
private class MyLocationListener implements LocationListener
{
   public MyLocationListener() {
       //konstruktori
        }
 // Signaloituu automaattisesti, kun uusi paikkatieto vastaan-
otetaan
   public void onLocationChanged(Location location) {

       mLocation = location;
         }
       // Signaloituu kun käyttäjä deaktivoi GPS:n
        public void onProviderDisabled(String provider) {
            // TODO Auto-generated method stub
        }
       // Signaloituu kun käyttäjä aktivoi GPS:n
        public void onProviderEnabled(String provider) {
            // TODO Auto-generated method stub
        }
       // GPS-tarjoajan status muuttuu

    public void onStatusChanged(String provider,
                   int status,Bundle extras) {
            // TODO Auto-generated method stub
        }
    }
```

Kun edellä mainitut muutokset on tehty laitteeseen ja ohjelmakoodiin, sovellus kuuntelee GPS-dataa. Tämä ilmenee yläpalkkiin ilmestyvänä GPS-ikonina (kuva 23). Kun GPS-ikoni vilkkuu, paikkatietoa haetaan, kun se palaa kiinteästi, yhteys paikkatiedon tarjoajaan on muodostettu ja paikkatieto päivittyy reaaliajassa.

KUVA 23. GPS-ikoni yläpalkissa.

4.10 Paikkatiedon lähettäminen palvelimelle

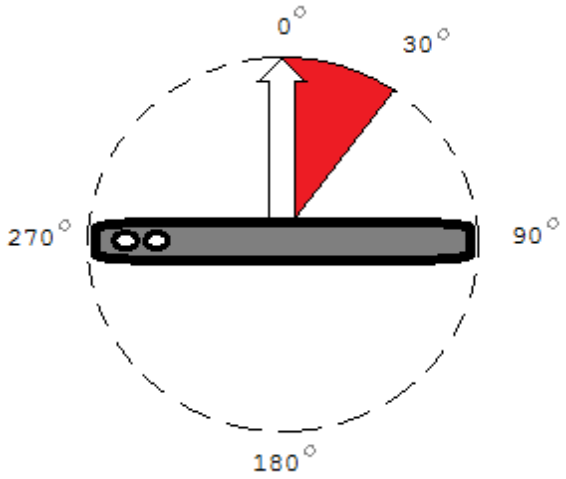
Laitteen on lähetettävä omaa paikkatietoaan palvelimelle. Ennen kuin paikkatieto lähetetään, sen on käytettävän protokollan mukaan täytettävä kolme ehtoa jotta päästäisiin vaadittavaan tarkkuuteen paikkatiedon seuraamisessa.

4.10.1 Lähetysehdot

Kun yksi tai useampi lähetysehto on tosi, lähetys tehdään.

Ehto 1: Heading

Laitteen heading on muuttunut 30 astetta. Heading on laitteen ”rintamasuunta”. Kun käyttäjä kulkee suoraan kääntymättä, heading on 0. Jos käyttäjä kääntyy esim. risteyksestä, laitteen heading muuttuu. Laite on kuvattuna ylhäältäpäin kuvassa 24. Laitteen kääntymiskulma luetaan laitteen sisäisestä sensorista. Heading saa arvonsa väliltä 0–360 astetta.



KUVA 24. Laitteen heading

Sensoreita kuunnellaan laitteesta niin, että ensin pääluokkaan lisätään SensorEventListener-rajapinta. Sensoreiden kuuntelun toteuttamisessa on käytetty apuna Android Developers -sivuston Sensor Manager -tutoriaalia (12.).

```
public class MapsDemo extends MapActivity implements SensorEv-
entListener{
```

Lisätään SensorManager-objekti, jolla päästään kuuntelemaan laitteen sensoreita.

```
SensorManager sensorManager;

  sensorManager = (SensorManager) getSystemService (SEN-
SOR_SERVICE);

sensorManager.registerListener(this,
sensorManager.getDefaultSensor(Sensor.TYPE_ORIENTATION),
                SensorManager.SENSOR_DELAY_NORMAL);
```

Sensoreiden kuuntelu hoidetaan rajapinnan onSensorChanged()-funktiolla.

```
 public void onSensorChanged(SensorEvent event) {

        // SensorEventin values[0], kääntymiskulma pohjoiseen
nähden
            float myroll = Math.round(event.values[0]);
              ...
    }
```

Silloin kun kääntymiskulma muuttuu 30 astetta edellisestä lähetetystä paikkatiedosta, lähetys voidaan tehdä.

Ehto 2: Etäisyys

Lähetys tehdään, jos etäisyys edellisestä lähetetystä pisteestä on yli 100 metriä. Tarkistus tehdään ennen lähetystä. Dosend-lippu asetetaan”true”-arvoon, jos etäisyys on yli 100 metriä. Etäisyys saadaan metreinä Location-luokan distanceTo()-funktiolla:

```
float metres = mLocation.distanceTo(mylastLocation);
     if (metres > 100){dosend = true;}
```

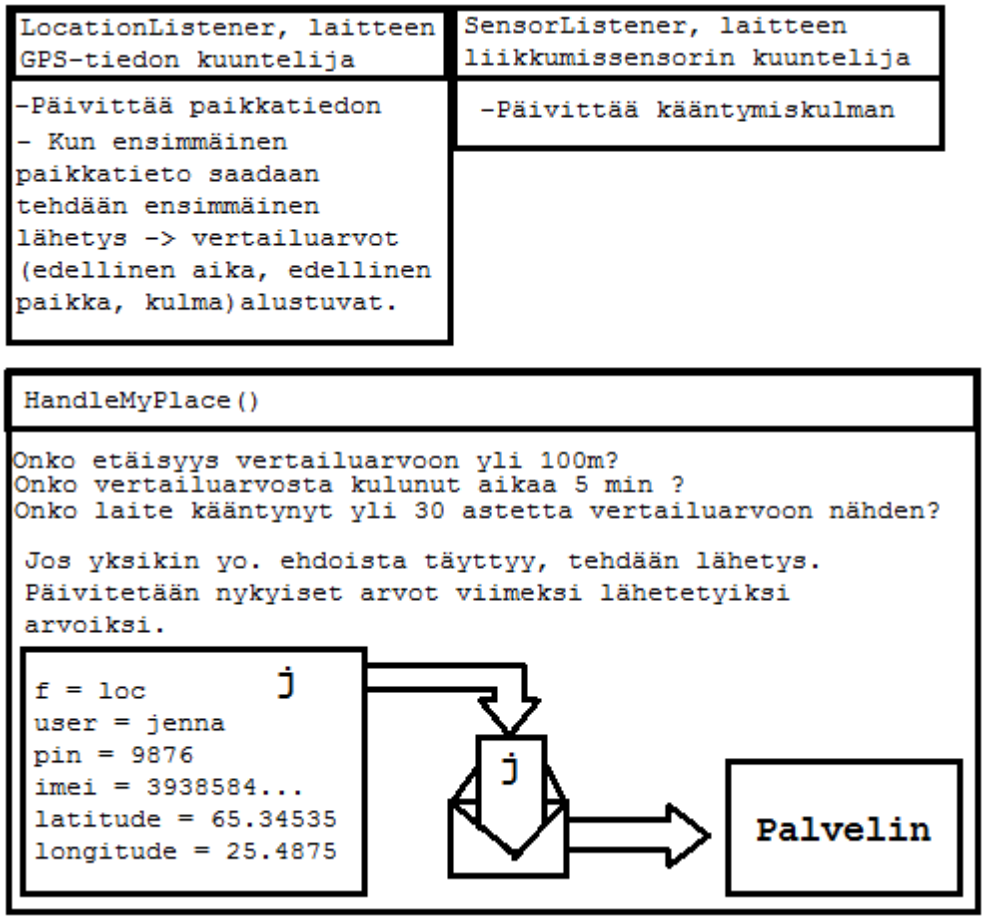
Ehto 3: Aika

Aikaa viimeisestä lähetyksestä on kulunut 5 minuuttia. Tarkistus ja erotuksen laskeminen:

```
 java.util.Date datenow = new java.util.Date(); //aika nyt
msecsnow = datenow.getTime(); //millisekunneiksi
datenow = null;

//erotuksen laskeminen
long diffms = msecsnow – lastsendmsecs;
//erotus sekunteina
long diffInSec = TimeUnit.MILLISECONDS.toSeconds(diffms);
// 300 sekuntia = 5 minuuttia
if (diffInSec > 299){dosend = true;}
```

Näiden ehtojen lisäksi laite on saanut vähintään yhden paikkatiedon. Tämä osoittaa, että ”ollaan kiinni” GPS:ssä. Silloin voidaan olettaa, että yhteys on löytynyt.

HandleMyPlace()-funktio hoitaa lähetyksen kuvan 25 mukaisesti.



KUVA 25. HandleMyPlace()-funktion toiminta

Toimintaa testattiin seuraamalla, milloin laite lähettää tietoa palvelimelle. Kun laitetta käänneltiin, se alkoi lähettää paikkatietoaan. Kun laitetta testattiin maastossa, se lähetti paikkatietoaan 100 metrin välein. Kun laite oli paikoillaan eikä sitä käännelty tai liikutettu, paikkatieto tuli tasaisesti 5 minuutin välein.

4.10.2 Säikeet

Sovelluksen säikeistys tarkoittaa sitä, että prosessoriaikaa vieviä tehtäviä ja prosesseja eriytetään erillään ajettaviksi tehtäviksi. Säikeet eivät ole itsenäisiä prosesseja, vaan eräänlaisia prosessin sisäisiä prosesseja. Prosessit suoritetaan käyttäjän näkökulmasta samaan aikaan, vaikka todellisuudessa säikeiden suoritusta vuorotellaan hyvin nopeasti. Kun useampia tehtäviä

suoritetaan näennäisesti rinnakkain, voidaan parantaa sovelluksen suorituskykyä ja käytettävyyttä. (13.)

Säikeitä kannattaa käyttää, kun sovelluksessa halutaan suorittaa prosessoriaikaa vieviä tehtäviä, esimerkiksi monimutkaisia laskentatehtäviä, tietokantatransaktioita, datan hakua verkkoyhteyden yli, levyltä lukemista tai kirjoittamista. Yleensäkin kirjoitus- ja lukutoimenpiteet saattavat aiheuttaa odottamatonta viivettä. (13.)

Laitteen tuli lähettää paikkatietoaan koko ajan palvelimelle ja muistaa ja vertailla edellisiä lähetettyjä arvoja keskenään. Päätettiin erottaa paikkatiedon vertailija ja lähetys omaksi säikeekseen.

Säie luodaan MapsDemon onCreate()-metodissa, jossa muuttujat alustetaan. Säie käynnistetään start()-metodilla.

```
Thread locationthread= new Thread(new LocationSenderThread());
locationthread.start();
```

Varsinainen säie toteutetaan omassa luokassaan. Uusi luokka toteuttaa Runnable-rajapinnan, jonka run()-metodiin ohjelmoidaan säikeen tehtävät. Säikeen tehtävänä on tässä tapauksessa suorittaa toistuvasti aiemmin luotua HandleMyPlace()-metodia, joka tarkistaa etäisyyden, ajan ja kääntymiskulman ja lopuksi tekee lähetyksen palvelimelle. Jos etäisyyttä, aikaa ja kulmaa ei tarkistettaisi, lähetystä ei rajoitettaisi. Silloin paikkatietoja lähtisi palvelimelle n. 3 sekunnissa, joka on yli tarpeellisen määrän. Nyt lähetystarkkuutta saadaan hallittua. Jos esimerkiksi olisi haluttu, että paikkatieto lähetetään palvelimelle joka viides sekunti, pakotettaisiin säie ”nukkumaan” joka kierroksella 5000 millisekuntia. Nukutus tapahtuu säikeen sleep(5000)-metodilla.

```
public class LocationSenderThread implements Runnable{
       public void run() {
       while(true)
       {
            HandleMyPlace();

       }
                   }
 }
```

Kun säikeen ajo halutaan pysäyttää, kutsutaan stop()-metodia.

```
locationthread.stop();
```

Säie pysäytetään silloin, kun sitä ei enää tarvita.

4.10.3 Paikkatiedon simulointi

Koska sovellus toteutetaan sisätiloissa olevassa työyksikössä, ei GPS-signaalia ole välttämättä saatavilla. Laitteen liikkumista kartan mukaisella reitillä on simuloitava esimerkiksi ohjelmilla, jotka väärentävät laitteen vastaanottamaa GPS-dataa. Tässä projektissa on käytetty Fake GPS -ohjelmaa. Fake GPS on ilmainen, Android Marketista ladattava sovellus, jonka avulla valitaan kartalta haluttu paikkapiste, missä laite luulee olevansa.

Jotta laite voisi vastaanottaa väärennettyä GPS-dataa, on otettava ”Allow mock locations” -ominaisuus käyttöön laitteen asetuksista. Valinta tehdään Settings → Applications → Development-valikosta.

Toinen vaihtoehto on ajaa ohjelmaa Eclipsen virtuaalilaitteessa, jossa esim. GPS-dataa, akun tyhjenemistä ja laitteen kiihtyvyysanturia voidaan hallita syöttämälllä käsin testiarvoja virtuaalilaitteeseen.

4.11 Kartta näkyviin

Kun ulkoasupohja oli valmis ja sovellus kuunteli ja lähetti GPS-dataa, alettiin tutkia sovelluksen tärkeintä käyttöliittymäkomponenttia eli karttaa. Kartta saa-

tiin näkyville internetistä löytyvien tutoriaalien avulla. Seuraavassa esitellään vaiheistus, miten kartta saatiin näkyviin.

4.11.2 Google Api key

Jotta sovelluksessa voidaan käyttää Google-pohjaista karttaa, tarkemmin sanottuna MapView-luokkaa projektissa, on kehittäjän hankittava laitekohtainen Google Api Key -autentikointikoodi. Koodin hankkija sitoutuu noudattamaan Google Apin käyttöehtoja, ja tunnistautuminen hoidetaan Google-tilin avulla (Gmail-tili). Api key generoidaan seuraavanlaisesti: Annetaan ”keytool -list -keystore debug.keystore” -komento käyttäjän Android-hakemistossa (C:\Users\<user>\.android\debug.keystore). Jos hakemistoon on määritetty salasana, se kysytään käyttäjältä. Muussa tapauksessa salasana on tyhjä. Tulostuva rivi on ns. MD5-fingerprint (näyttää tältä: 43:51:43:a1:b5:fc:8b:b7:0a:3a:a9:b1:0f:66:73:a8), joka kopioidaan leikepöydälle ja mennään osoitteeseen http://code.google.com/intl/fi-FI/android/maps-api-signup.html. Hyväksyttyään käyttöehdot käyttäjä syöttää MD5-koodin kenttään ja painaa ”Generate API Key” -painiketta. Tässä vaihessa vaaditaan kirjautuminen Google-tilille,ja sen jälkeen näytetään käyttäjän Api Key, jonka käyttäjä tallentaa ja kehitystyö voi jatkua (15).

4.11.3 AndroidManifest.xml

Jokaisella Android-sovelluksella on oltava AndroidManifest.xml-tiedosto. Tiedoston tarkoituksena on määrittää sovelluksen olennaiset tiedot, kuten sovelluksen käyttämät kirjastot, aktiviteetit, oikeudet ja kohdealustan tiedot (14). Kartan näkyminen vaatii, että sovelluksen AndroidManifest.xml-tiedostoon lisätään seuraavat tiedot.

```
//Oikeus käyttää GPS-toimintoa
<uses-permission an-
droid:name="android.permission.ACCESS_FINE_LOCATION"/>
 //Oikeus Internet-yhteyden käyttämiseen laitteesta
<uses-permission android:name="android.permission.INTERNET" />
```

```
//Määritetään Googlen maps-kirjasto
<uses-library android:name="com.google.android.maps" />
```

4.11.4 MapsDemo.java

Karttaa varten otetaan käyttöön mapView-komponentti ja mapActivity. Seuraavassa käytetty apuna kurssimateriaalia Jyväskylän Yliopiston Wikistä (16).

```
import com.google.android.maps.MapView;
import com.google.android.maps.MapActivity;
public class MapsDemo extends MapActivity{
```

Esitellään karttanäkymäkomponentti ja Googlelta saatu Map Api Key -luokassa.

```
private MapView mapView;
private static final String DEBUG_MAP_API_KEY = "gOoGle-
MaPaPIKey";
// luodaan MapView (dynaamisesti) OnCreate()- funktion sisällä
  @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

mapView= new MapView(this, DEBUG_MAP_API_KEY);
```

Kiinnitetään mapView-karttanäkymä pohjalayoutiin.

```
final RelativeLayout mapLayout = new RelativeLayout(this);
mapLayout.addView(mapView);
```

(16.)

4.12 Piirto kartalle

Googlen karttanäkymään voidaan lisätä kuvia erillisinä näkymään kiinnittyvinä ”tasoina”. Luokka, jossa tasot luodaan, on Googlen Maps-kirjastosta löy-

tyvä Overlay-luokka. Kun sovellukseen luodaan oma MyOverlay-luokka ja periytetään sille Googlen Overlay-luokka, saadaan omia, kustomoituja Overlay-tasoja sovellukseen.

Kuvan lisäys resursseihin

Kuvan voi lisätä resurssiksi sovellukseen. Ensin valitaan haluttu kuva ja kopioidaan se projektin MapsDemo/res/drawable-kansioon, joka näkyy Eclipsen Package Explorer -ikkunassa. Kopiointi tapahtuu yksinkertaisimmin niin, että haluttu kuva raahataan projekti-kansioon. Eclipse kysyy että kopioidaanko kuva projektiin vai linkitetäänkö se kuvan nykyisestä sijainnista. Kun valitaan kopiointi, kuva siirtyy automaattisesti /res/drawable- kansioon.

4.13 Oman paikan näyttäminen kartalla

Tässä projektissa tarvittava, omaa paikkaa indikoiva kuva on auton kuva (kuva 26), koska sovellusta käytetään lähtökohtaisesti ajoneuvon paikan seuraamiseen kartalla.

KUVA 26: Lisättävä kuva car_32.png.

MyLocationOverlay-luokka

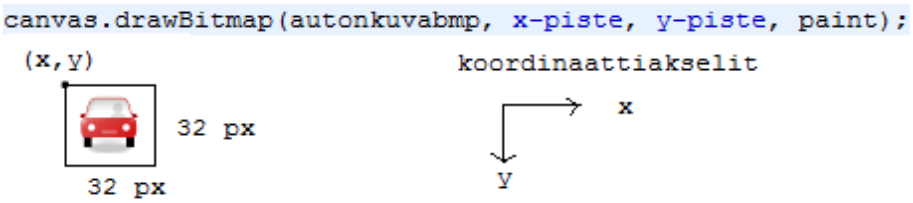
Luodaan oma Overlay-luokka, MyLocationOverlay-pääluokan sisälle.

```
 class MyLocationOverlay extends
com.google.android.maps.Overlay {
@Override
//draw() piirtää overlayn kartalle
public boolean draw(Canvas canvas,MapView mapView,boolean shad-
ow,long when) {
super.draw(canvas, mapView, shadow);

/*mLocation on pääluokan muuttuja, joka saa uuden arvon aina
kun paikkatieto päivittyy LocationListenerissä eli    se sisäl-
tää tuoreimman paikkatiedon*/

/* Irrotetaan mLocationista latitude- ja longitude-arvot*/
```

```
String latitude =  String.valueOf(mLocation.getLatitude());
String longitude =String.valueOf(mLocation .getLongitude());
       /*Luodaan uusi GeoPoint-objekti, joka on paikkapiste eli
       latitude-longitude pari*/
       p = new GeoPoint(
                   (int) (Double.parseDouble(newlat) * 1E6),
                   (int) (Double.parseDouble(newlon) * 1E6));

       Point myScreenCoords = new Point();
       /*Muutetaan GeoPoint-piste näyttöpikseleiksi*/
       mapView.getProjection().toPixels(p, myScreenCoords);

       /* Luodaan BitMap-objekti resurssitiedostosta*/
       Bitmap bmp = BitmapFacto-
ry.decodeResource(getResources(),
       R.drawable.car_32);

       /*Varsinainen piirto ja asemointi*/
       canvas.drawBitmap(bmp, myScreenCoords.x - 16, myScreen-
Coords.y - 16, paint);

return true;

 } }
```

Huomioitavaa kuvan asemoinnista

Kuva piirretään paikkapisteeseen kuvan 27 mukaisesti.

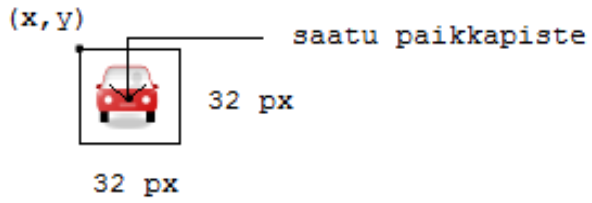


KUVA 27. Kuvan piirto.

Auto näkyy nyt siis 16 pikseliä väärässä paikassa sekä x- että y-suunnassa, koska saatu paikkapiste pitäisi olla auton keskellä eikä sen vasemmassa yläkulmassa. Kuvan piirtämisen aloituspaikkaa pitää siirtää 16 pikseliä x- ja y- suunnassa, jotta kuva piirtyy niin, että saatu paikkapiste tulee kuvan keskelle (kuva 28). (17.)

.

```
canvas.drawBitmap(autonkuvabmp, x-piste - (32/2) ,
            y-piste - (32/2), paint);
```



KUVA 28. Auton asemointi.

Overlayn lisääminen kartalle

Luodaan MyLocationOverlay-objekti.

```
MyLocationOverlay myLocationOverlay = new MyLocationOverlay();
```

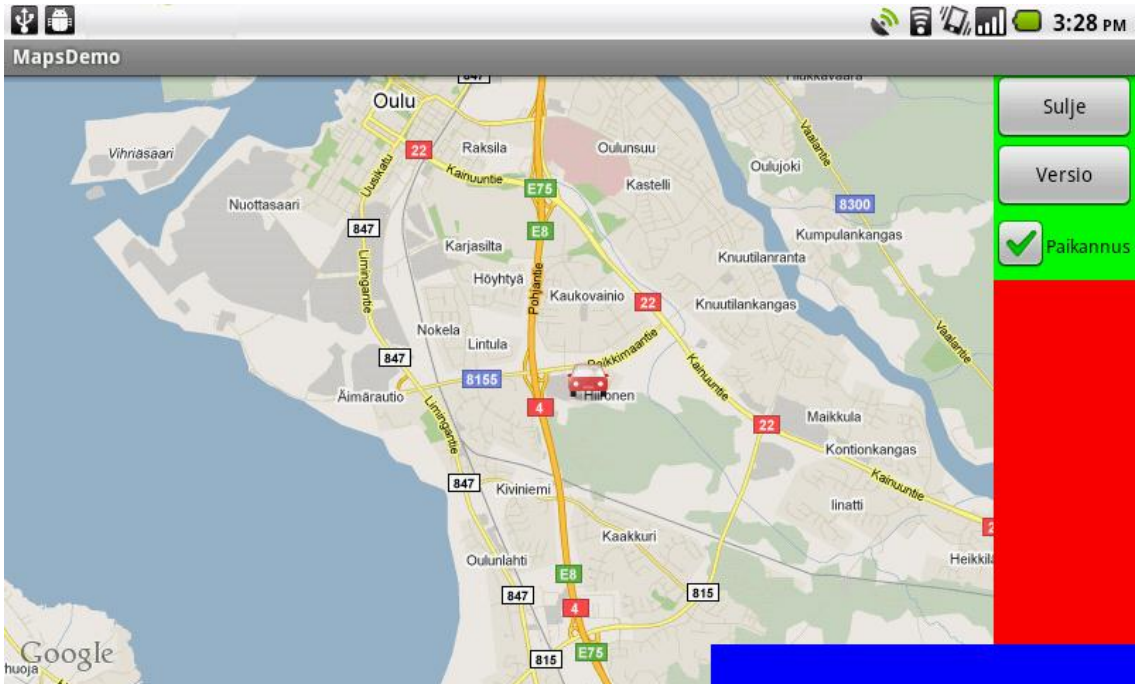
Kaikki mapView-karttanäkymään piirrettävät objektit ovat muokattavassa Overlay-listassa. Kun Overlay-listaan lisätään uusi Overlay, se piirretään automaattisesti. Haetaan mapView:n Overlay-lista ja lisätään äsken luotu Overlay listaan.

```
List<Overlay> list = mapView.getOverlays();
list.add(myLocationOverlay);
```

Aina kun Overlay-listaa muokataan, on syytä kutsua postInvalidate()-funktiota, jolloin muutokset piirtyvät näytölle, koska funktio signaloi draw()-metodin.

```
mapView.postInvalidate();
```

Oma paikka ja kartta näkyvät sovelluksessa, kuten kuvassa 29.



KUVA 29. Kartta ja käyttäjän paikka käyttöliittymässä.

4.14 Soketit

Soketti on perusmenetelmä yhteyden luomiseksi asiakas- ja palvelinohjelman välillä. Soketin välityksellä voidaan kirjoittaa ja lukea tiettyyn verkkoyhteyteen. Yhteydessä käytetään TCP-protokollaa. Verkkoyhteys kuvataan IP-osoitteella ja portilla. Yhteyden otossa vaiheet ovat seuraavanlaiset: Ensin palvelimena toimiva ohjelma luo ServerSocket-olion, joka jää odottamaan asiakkaan yhteydenottoa johonkin porttiin. Asiakasohjelma luo soketin, jolla otetaan yhteyttä palvelinohjelmassa kuuntelevaan sokettiin määrittämällä palvelimen osoite (IP) ja portti. Kun palvelinohjelman sokettiin otetaan yhteyttä asiakasohjelmasta, palvelinohjelman ServerSocket luo uuden soketin. ServerSocket siirtää asiakasohjelman luoman yhteyden äsken luomaansa sokettiin. Yhteys on muodostettu palvelinohjelman ja asiakasohjelman soketien välille. Nyt ne voivat kommunikoida keskenään kirjoittelemalla toisen ohjelman sokettiin ja lukemalla omaa sokettiaan.

4.14.2 Soketin kuuntelu ohjelmassa

Ohjelman vaatimuksena oli, että se pystyy vastaanottamaan työmääräimiä palvelimelta jollain tekniikalla. Päätettiin käyttää soketteja. Ohjelmaan luotiin yhteyksiä kuunteleva serverSocket-objekti:

```
ServerSocket serverSocket = new ServerSocket(4444);
```

Yhteyttä odottavan soketin luontiin annetaan parametrina portti, jota soketti kuuntelee. Porttiosoite on kokonaisluku välillä 1–65535. Porttiosoitteet välillä 1–1024 ovat varattuja portteja. Niitä siis käytetään jo johonkin palveluun, ja esimerkiksi porttinumero 80 on varattu WWW-palvelimelle. Lähtökohtaisesti omaan ohjelmaan kannattaa valita siis porttinumero, joka on suurempi kuin 1024. Tärkeintä kuitenkin on, että valittavaa porttinumeroa ei käytä mikään muu palvelu järjestelmässä. Tässä tapauksessa porttinumero 4444 on valittu mielivaltaisesti väliltä 1024–65535. (18.)

Jotta palvelin tietäisi, mihin sokettiin sen pitäisi yrittää yhteyttä kun määräin lähetetään, ohjelman alussa laite ilmoittautuu palvelimelle kertomalla oman IP-osoitteensa ja portin, jota se laitteessa kuuntelee. Ilmoittautuminen tapahtuu niin, että laite lähettää IP:n ja porttinumeron palvelimelle. Lisäksi käytettävän protokollan mukaisesti lähetetään myös käyttäjätunnukset ja laitteen IMEI-koodin autentikointia varten. (18.)

```
{"f":"signup","ip":"192.168.x.xxx","port":"1234","user":"Joonas1",
"pin":"1342","imei":"42983749327983744 72"}
```

Jotta laitteesta pystyttäisiin hakemaan IP-osoite ja IMEI-koodi, Android-Manifest-tiedostoon pitää lisätä oikeus, jolla päästään käsiksi puhelimen tilaan.

```
<uses-permission android:name = "an-
droid.permission.READ_PHONE_STATE"/>
```

Kun on luotu kuuntelijasoketti ja ilmoittauduttu palvelimelle, jäädään odottamaan yhteydenottoa asiakasohjelmalta. Ohjelma on ”jumissa” accept()-

metodissa niin pitkään, kunnes asiakasohjelma ottaa yhteyttä sokettiin. Accept()-metodi palauttaa viittauksen uuteen Socket-olioon. (18.).

```
try {
Socket clientSocket = serverSocket.accept();
```

Seuraavaksi luodaan PrintStream- objekti, jolla voidaan kirjoittaa asiakassokettiin, kirjoitusfunktiolla println(”teksti”) (18.).

```
PrintStream os = new PrintStream(clientSocket.getOutputStream());
os.println("Hei Client!");
```

Sitten luodaan lukuvirta, josta luetaan asiakassoketin viestit. Luku tapahtuu funktiolla readLine(). Lukuvirran parametrina annetaan myös merkistöstandardi, jonka mukaisia merkkijonot ovat (18).

```
BufferedReader in = new BufferedReader(new  InputStreamReader
(clientSocket.getInputStream(), "ISO-8859-1"););
String tekstiasoketista = in.readLine();
```

Kun soketista on luettu/kirjoitettu haluttu data suljetaan datavirrat ja asiakassoketti. Sitten ”nullataan” eli alustetaan muuttujat tyhjäksi, jotta ne eivät varaa muistia ja roskienkerääjä pääsee niihin käsiksi (18).

```
os.close();
os = null;
in.close();
in= null;
clientSocket.close()
clientSocket = null;
is.close();
is = null;
```

4.14.3 Soketinkuuntelijasäie

Soketin kuuntelu on pakko toteuttaa omassa säikeessä, koska sen suoritus jää jumiin accept()-metodiin odottamaan asiakasohjelmalta yhteyspyyntöä. Säikeistys suoritettiin samalla periaatteella kuin paikkatiedon lähettäjäsäie.

Luonti ja käynnistys tapahtuu onCreate()-metodissa.

```
Thread socketthread = new Thread(new SocketReaderThread());
       socketthread.start();
```

Säieluokka SocketReaderThread toteuttaa Runnable- rajapinnan. Soketin kuuntelu suoritetaan ReadSocket()-metodissa kuten ”Soketin kuuntelu ohjelmassa” -luvussa aiemmin kuvattiin.

```
public class SocketReaderThread implements Runnable{
         public void run() {
         while(true)
         {
                     ReadSocket();

         }
         }
}
```

4.14.4 Maarain-luokka

Määräinten käsittelyä varten luotiin Maarain-luokka. Koska määräin asetetaan näkyville käyttöliittymään Button-objektina, periytetään Button-luokka Maarain-luokkaan. Tämä tarkoittaa sitä, että Maarain-luokka perii kaikki Button-luokan ominaisuudet. Luokan luonnissa on käytetty apuna Javan tutoriaalia (19).

Määräin-luokan objekti luodaan antamalla parametrina soketista luetusta datasta koottu JSONObject-olio. JSONObject-olio puretaan Maarain-objektin muuttujiksi.

Määräimiä varten ohjelmaan luodaan Maarain-tyyppinen taulukko, johon saapuvat määräimet kerätään.

```
private Maarain[] maaraimet = new Maarain[20];
vrt. kokonaislukutaulukko
private int[] kokonaisluvut = new Int[20];
```

Soketista luetaan JSON-stringi, joka koostetaan JSONObject-olioksi:

```
String tekstiasoketista = in.readLine();
try {
          JSONObject jfromsocket = new JSONOb-
ject(tekstiasoketista);
```

Tekstiasoketista string-muuttuja sisältää JSON-formaattia noudattavan string-muuttujan. Koostaminen muodostaa JSONObject-olion, joka sisältää avain-arvopareja:

```
{"f":"maarain","status":"UUSI","task":"Hae naulo-
ja","destination":"Professorintie, Oulu","id":1,"time":"21.2.2011
14:12","priority":"TÄRKEÄ","name":"Uusi maarain"}

Avain- arvo parit:
f = maarain
status = UUSI
task = Hae nauloja
jne...
```

JSONObjekti annetaan parametrina CreateAndShowNewMaarain()-metodille, joka luo uuden Maarain-olion ja asettaa sen näkyville käyttöliittymään.

```
CreateAndShowNewMaarain(jfromsocket);
```

CreateAndShowNewMaarain()-metodissa luodaan TableRow-rivi, johon luodaan ja kiinnitetään määräin-olio. TableRow kiinnitetään aiemmin luotuun maaraimetTableen. Counter-muuttuja pitää kirjaa olioiden määrästä. Kun uusi olio luodaan, counter-arvoa kasvatetaan yhdellä:

```
TableRow tr = new TableRow(MapsDemo.this);
maaraimet[counter]  = new Maarain(MapsDemo.this,jfromsocket);
```

Maarain-objektillle annetaan seuraavat muuttujat (määräimen ominaisuudet): funktio, ID, nimi, kohdeosoite, taski, status, aika, prioriteetti ja koordinaattipiste. Maarain-luokassa on haku- ja asetusfunktiot kaikille näille ominaisuuksille. Esimerkkinä on Status-muuttujan käsittely Maarain-luokassa. Määräinluokan esittely:

```
public class Maarain extends Button{
    public String statusstr;
```

Seuraavaksi asetus konstruktorissa. Määräintä luodessa annetaan parametrina JSON-stringistä koostettu JSONObject-olio. Status-muuttuja asetetaan noutamalla avaimen ”status”-arvo JSONObject-oliosta muuttujaan statusstr.

```
public Maarain(Context context, JSONObject j) {
try {
        statusstr = j.getString("status");
```

Funktio, jolla muuttujan arvo voidaan noutaa (”getteri”), palauttaa määräimen status-arvon stringinä:

```
 public String GetStatus() {
        return statusstr;
    }
```

Getterin käyttö pääohjelmassa:

```
maaraimet[counter]  = new Maarain(MapsDemo.this,jfromsocket);
String askstatus = maaraimet[counter].GetStatus();
```

Funktio, jolla asetetaan (”setteri”) määräimelle status-arvo.

```
public void setStatus(String newValue) {
           statusstr = newValue;
   }
```

Setterin käyttö pääohjelmassa, esimerkiksi kun määräimen suoritus on aloitettu. Jokaisen saapuvan määräimen status on oletusarvoisesti ”UUSI”.

```
maaraimet[counter].setStatus("KÄYNNISSÄ");
```

Kaikki luokan muuttujat voidaan käsitellä samalla tavalla.

4.14.5 Määräinten näyttäminen käyttöliittymässä

Ennen kuin määräin voidaan näyttää käyttöliittymässä Button-widgetinä, sille on annettava muutamia Button-luokan ominaisuuksia.

Tekstiksi annetaan määräimen nimi ja id. Esim: Uusimaarain 1

```
maaraimet[counter].setText(maaraimet[counter].GetName() + " " +
maaraimet[counter].GetId());
```

Buttonille voidaan myös asettaa taustakuva. Asetetaan resursseihin tallennettu kuva (kuva 30).

KUVA 30. Button- komponentin taustakuva.

```
maaraimet[counter].setBackgroundResource
(R.drawable.container_yellow);
```

Asetetaan määräimelle osoitteelle haettu latitude-longitude -pari lipun kartalle piirtoa varten.

```
maaraimet[counter].setPoint(getLatLonOfAddr (maar-
aimet[counter].GetDestination()));
```

Kyse on Button-widgetistä, joten luodaan sille kosketuskuuntelija.

```
maaraimet[counter].setOnTouchListener(new OnTouchListener() {

public boolean onTouch(android.view.View v,MotionEvent event) {

          //määräintä painetaan, se lopettaa vilkkumisen
          v.clearAnimation();
          //näytetään Uusi Määräin- Dialogi
          ShowTheDialogobj((Maarain) v);
          return false;
                                              }
            });

          // kiinnitetään määräin TableRow-olioon
          tr.addView(maaraimet[counter],220,70);
          // kiinnitetään TableRow-olio maaraimetTableLayoutiin
          maaraimetTableLayout.addView(tr);
          // kasvatetaan määräinlaskuria
          counter = counter + 1;
```

CreateAndShowNewMaarain()-metodissa siis luodaan määräin, asetetaan sille Button-luokan ominaisuuksia ja kiinnitetään se käyttöliittymäkomponenttiin. Metodia kutsutaan soketinkuuntelijasäikeestä. Koska suurinta osaa käyttöliittymäkomponenteista ei pysty muokkaamaan säikeestä, ajetaan muutokset käyttöliittymää muokkaavassa säikeessä, UI-säikeessä (käyttöliittymän pääsäie) metodilla runOnUiThread() (20).

```
 runOnUiThread(new Runnable()
{
   public void run()
   {
      CreateAndShowNewMaarain(jfromsocket);
   }
}
});
```

Määräintä indikoiva lippu kartalle

Määräintä indikoiva lippu lisätään karttanäkymään käyttämällä samaa Overlay-luokkaa, joka näyttää oman paikan kartalla. Ideana on käydä läpi Maarain-objekti taulukko ja hakea jokaisen määräimen latitude-longitude -pisteet ja piirtää punainen lippu pisteeseen. Jos määräimiä ei ole, sisempi IF-ehto ei täyty ja lippuja ei piirretä. MyLocationOverlay-luokan draw()-metodiin lisätään:

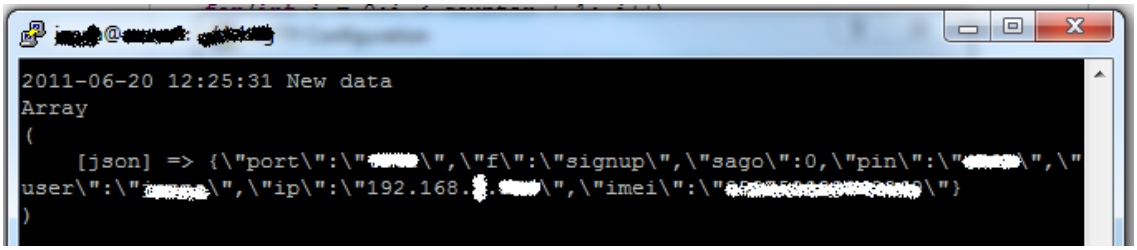
```
 // piirretään niin monta kertaa kuin määräimiä on (counterissa määräinten lukumäärä)
    for(int i = 0;i < counter + 1; i++)
    {
            // määräin objekti ei ole tyhjä
            if (maaraimet[i] != null)
            {
            Point coord = new Point();
            // muutetaan koordinaatit karttapikseleiksi
            mapView.getProjection().toPixels(
                        maaraimet[i].GetPoint(), coord);

            // luodaan bitmap-kuva resurssitiedostosta
            Bitmap bmp = BitmapFactory.decodeResource(getResources(),              R.drawable.flag_red);
            // varsinainen piirto ja asemointi
            canvas.drawBitmap(bmp, coord.x - 3, coord.y - 16,
paint);

            // lipun yläpuolelle määräinbuttonin teksti
            canvas.drawText(maaraimet[i].getText().toString(),
```

```
                        coord.x - 3, coord.y - 16, paint);
                }
        }
```

Toiminnan testaaminen

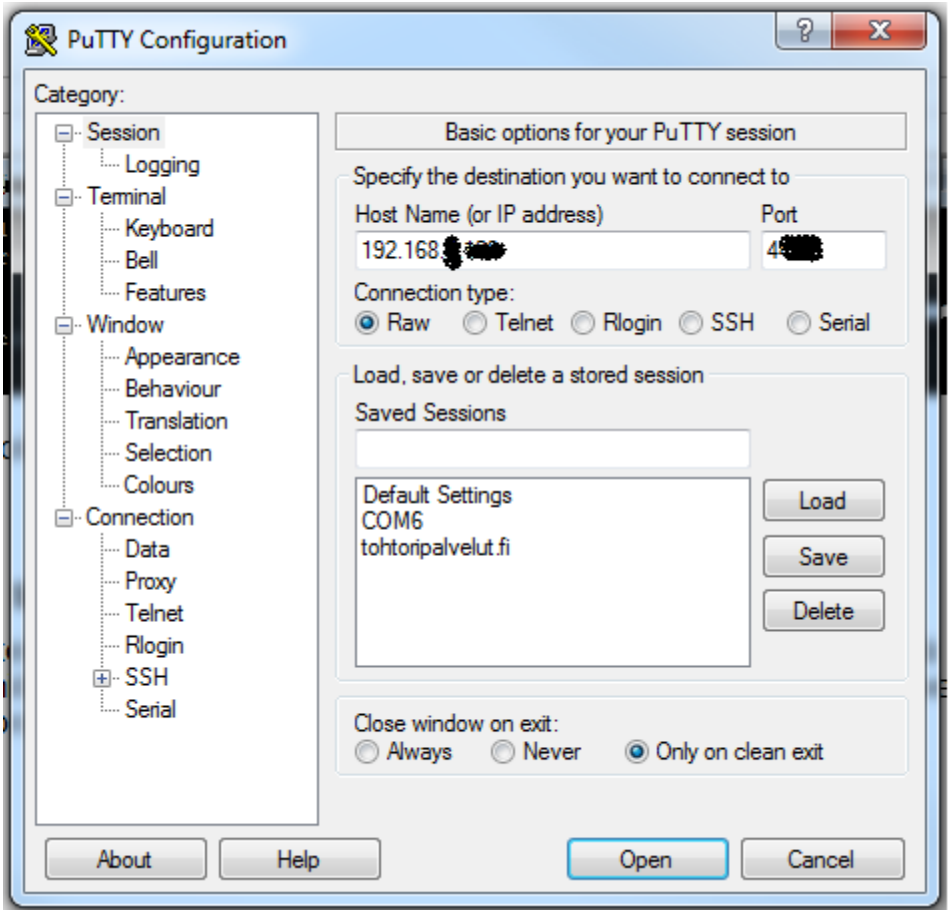
Simuloidaan kirjoittamalla Putty-ohjelmalla sokettiin, kun laite on ilmoittautunut palvelimelle.

Kuvassa 31 nähdään palvelimella vastaanotettu ilmoittautuminen. Nyt tiedetään soketin portti ja laitteen IP, (laitteen ja kirjautumisen tietoja piilotettu yksityisyyssyistä).



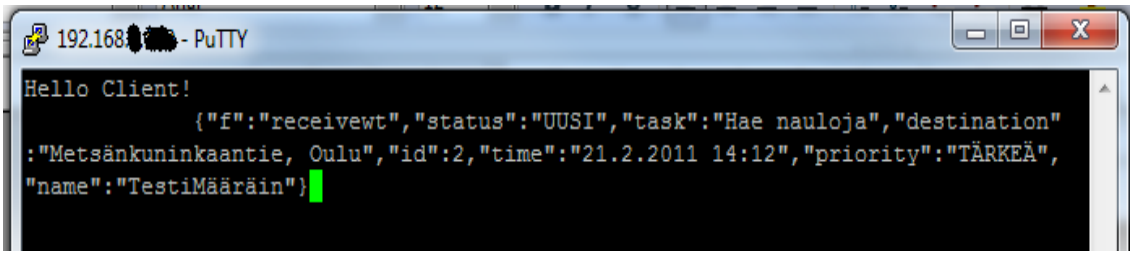
KUVA 31. Data tulee JSON-formaatissa palvelimelle.

Kuvassa 32 otetaan Puttyllä yhteys soketiin käyttämällä laitteen ilmoittavia tietoja.



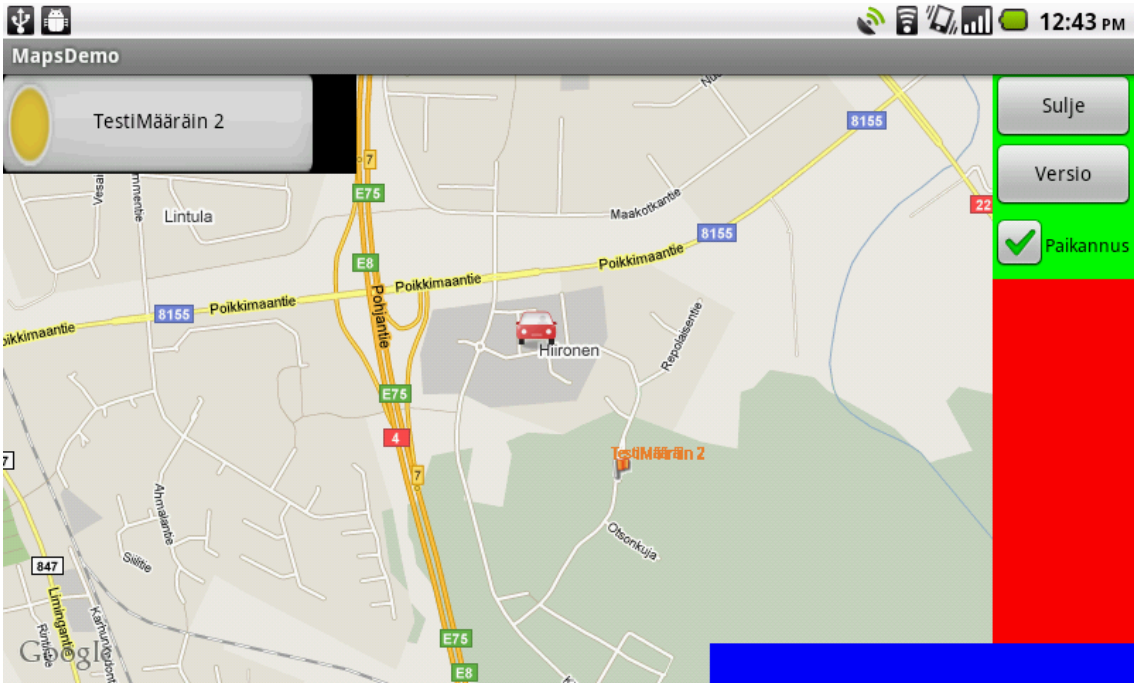
KUVA 32. Yhteyden muodostaminen client-ohjelmaan

Kun yhteys muodostettu, laite kirjoittaa tervehdyksen ”Hello Client!”. Kirjoitetaan määräin JSON-stringinä sokettiin (kuva 33).



KUVA 33. JSON-stringin kirjoittaminen sokettiin.

Maarain-objekti luodaan soketista luetusta tekstistä koostetusta JSON-objektista ja asetetaan näkyville käyttöliittymään (kuva 34).

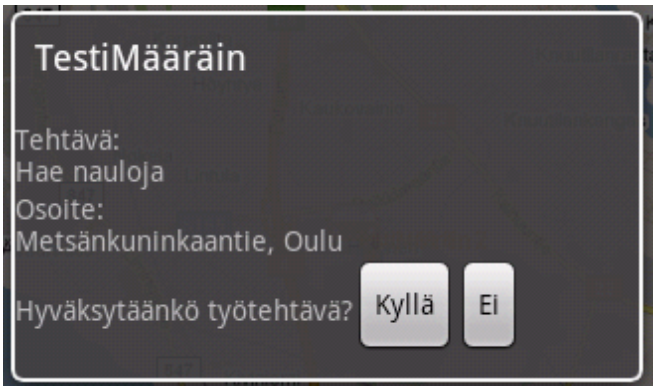


KUVA 34. Määräin vastaanotettu soketista ja asetettu näkyville käyttöliittymään

4.14.6 Määräimen hyväksyttäminen

Kun uusi määräin on saapunut, käyttäjän täytyy hyväksyä se ensin. Kun määräintä painetaan määräinlistasta, sen statusarvo tarkistetaan. Jos statusarvo on uusi, näytetään hyväksymisdialogi (kuva 35). Osoite- ja tehtäväkenttiin haetaan tiedot Maarainluokan GetDestination()- ja GetTask()-metodeilla.

```
TextView tehtava = (TextView) uusmaarainDlg.findViewById
(R.id.tehtavatextview);
tehtava.setText("Tehtävä:\n"+ maarainbtn.GetTask());

TextView osoite = (TextView) uusmaarainDlg.findViewById
(R.id.osoitetextview);
osoite.setText("Osoite:\n"+ maarainbtn.GetDestination());
```



KUVA 35. Hyväksymisdialogi

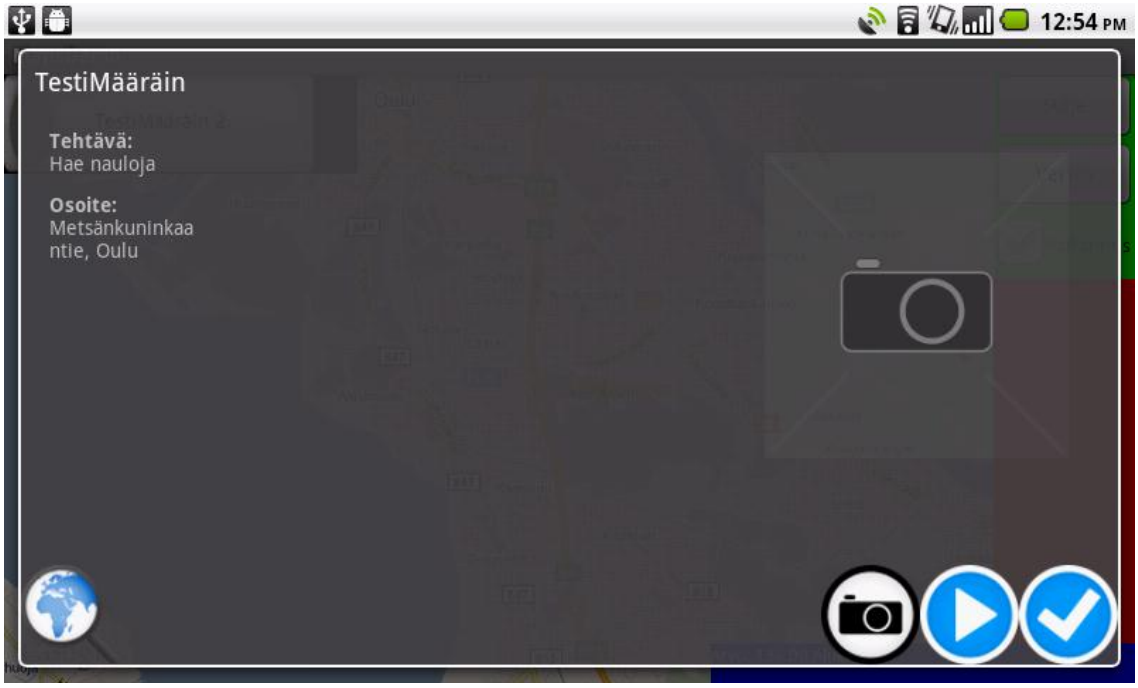
Määräimen tilan tarkistus:

```
checkstatus = maarainbtn.GetStatus();
if (checkstatus == "UUSI")
{
    UusiMaarainDialogobj(maarainbtn);
}
```

Kun käyttäjä painaa ”Ei”-painiketta, määräintä ei hyväksytä ja se poistetaan näkymästä ja määräin-taulukosta. Kun Käyttäjä painaa ”Kyllä”-painiketta, avautuu Määräin-dialogi.

4.14.7 Määräindialogi

Kuvassa 36 näkyvä määräindialogi sisältää kaikki määräimen tiedot ja toiminnot.



KUVA 36. Määräindialogi.

Määräimen toiminnot

Kaikissa toiminnoissa lähetetään palvelimelle JSON-muotoinen pyyntö, josta käy ilmi, mikä käyttäjä pyynnön on tehnyt ja mistä määräimestä on kyse. Pyyntö käsitellään palvelimella, joka vastaa onnistuuko pyyntö vai ei. Ilmoitetaan palvelimen vastaus (selkokielellä) myös käyttäjälle. Määräinten osalta protokollaa ei ole syytä käydä tarkasti läpi, koska se muuttuu jatkuvasti toteutuksen aikana. Kerrotaan vain pääasiallinen logiikka siitä, miten töidenhallinta on rakennettu.

Aloita työ

Kun käyttäjä painaa -ikonia, palvelimelle lähetetään JSON-stringinä komento, jossa määritellään määräin, josta on kyse, sekä autentikoidaan lähettäjä. Työ käynnistetään, kun palvelin vastaa työn aloituksen onnistuneen ja määräindialogi suljetaan. Määräinlistassa määräinpainikkeen tausta vaihtuu kuvan 37 mukaiseksi.

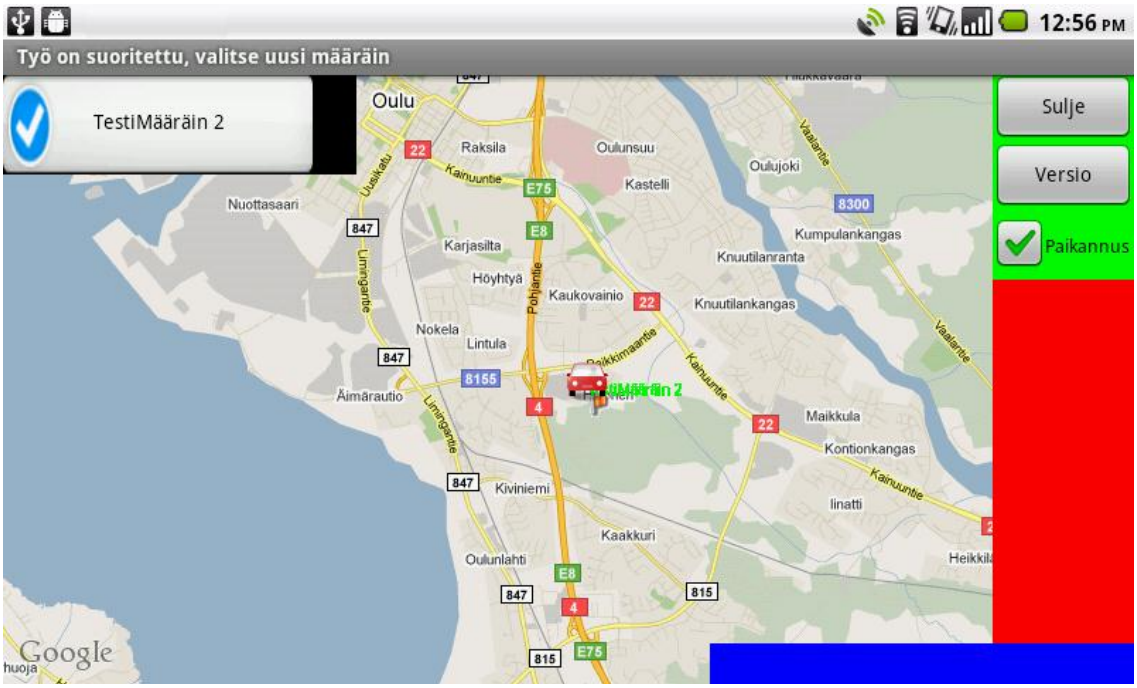
KUVA 37. Käynnissä olevaa määräintä indikoiva painikkeen taustakuva

Keskeytä työ

Kun käyttäjä painaa -ikonia, käynnissä oleva työmääräin keskeytetään samalla periaatteella ja -ikoni muuttuu -ikoniksi.

Työ on valmis

Kun käyttäjä painaa -ikonia, työ merkitään suoritetuksi. Tämä näkyy myös määräinlistassa.



KUVA 38. Käynnissä oleva määräin käyttöliittymässä

Kuvan liittäminen

Kun käyttäjä painaa -ikonia, sovelluksesta avataan laitteen kamera, ja otettu kuva lisätään määräimelle. Kamera avataan sovelluksesta Intent-

objektina eli sovelluksesta irrallisena suorituksena. Intent-objektin luonnissa on käytetty apuna Internetistä löytyvää artikkelia (21.). Kuvassa 39 kuvataan näkymä, jossa kuva on liitetty määräimeen.

Ensin esitellään pyynnön identifioiva mielivaltainen kokonaisluku.

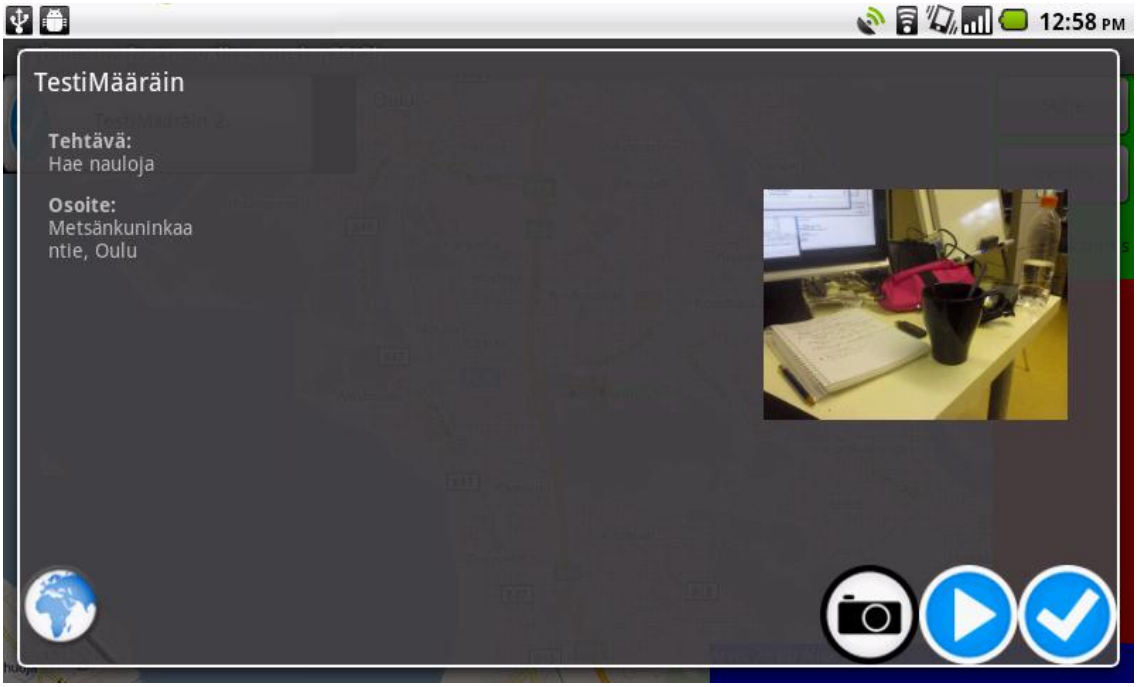
```
private static final int CAMERA_REQUEST = 1888;

Intent cameraintent= new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
```

Luodaan ”Kamera”-buttonin kuuntelija.

```
camButton.setOnTouchListener(new OnTouchListener() {

public boolean onTouch(android.view.View v, MotionEvent event) {
          startActivityForResult(cameraintent, CAMERA_REQUEST);
return false;
          }
});
```

Kun intent on suoritettu, se lähettää pääohjelmalle tiedon omasta valmistumisestaan. Tulokset käsitellään täällä:

```
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {

    if (requestCode == CAMERA_REQUEST && resultCode == RESULT_OK)
{
            Bitmap photo = (Bitmap) data.getExtras().get("data");
          //imv on dialogin imageView-objekti
            this.imv.setImageBitmap(photo);
            maaraimet[counterarvo].setBitmap(photo);
            photo = null;
      }
```



KUVA 39. Kuva liitetty määräimeen

Navigointi kohteeseen

Painamalla -ikonia käyttäjä näkee reitin oman paikkansa ja määräimen kohdeosoitteen välillä.

4.14.8 Reitin piirtäminen kartalle

Vaatimuksien mukaan sovelluksessa täytyy pystyä näyttämään reitti kahden kohteen välillä, lähtökohtaisesti oman paikkapisteen ja määräimen kohde-määränpään osoitteen välillä.

DirectionPathOverlay-luokka

Kuten oma paikkakin, reitti lisätään karttaan Overlay-objektina. DirectionPathOverlay-luokka on rakenteeltaan samankaltainen kuin MyLocationOverlay-luokka, mutta piirtometodit ovat erilaisia. Kahden reittipisteen välille piirretään viiva, ja näin saadaan aikaan reitti kartalle.

```
canvas.drawLine((float)point.x,(float) point.y,float) point2.x,
                    (float) point2.y, paint);
```

DirectionPathOverlay- objektin luontiin tarvitaan aloitus- ja lopetuspiste, joiden välille viiva piirretään.

Geocoder

Osoitteiden hakemiseen tarvittiin Googlen Maps-paketista Geocoder-luokan objekti. Geokoodaaminen itsessään on prosessi, jossa muutetaan osoitetietoja paikkakoordinaateiksi ja toisin päin. GeoCoder-luokan metodi getFromLocationName() palauttaa löytyneet osoitetiedot, joista voi irrottaa paikkakoordinaattitiedot, kun oman paikkapisteen ja osoitteella haetun määränpään koordinaattipisteet ovat tiedossa.

GetDirectionData()-metodi

GetDirectionData() on luotu metodi, jolla haetaan paikkapiste-taulukko Googlelta. GetDirectionData lähettää pyynnön Googlelle, parametreina ainakin seuraavat reittipisteiden saamiseksi. Lähtö- ja määränpään pisteet haetaan GeoCoderilla.

```
http://maps.google.com/maps?
...
&saddr=<lähtöpaikan lat,lon>
&daddr=<määränpään lat,lon>
...
&output=kml";
```

Pyyntö palauttaa kml-formaatissa olevat tiedot. Tiedosto sisältää ainakin ajo-ohjeet ja koordinaattipisteet, joiden kautta reitti piirretään kartalle. Kml-tiedoista parseroidaan koordinaattipisteet ja tallennetaan ne taulukkoon. Taulukko palautetaan ohjelmaan. Taulukon avulla piirretään DirectionPathOverlay-tasot antamalla pisteitä, joiden välille viiva piirretään. Kun kaikki taulukon koordinaattipisteiden väliviivat on piirretty kartalle, käyttäjä näkee kartalla reitin.

DrawPath()-metodi

DrawPath() on metodi, johon koko reitinpiirtoprosessi on yhdistetty. Se on vaiheistettu seuraavasti:

1. DrawPath()-metodi saa parametrina osoitteen johon reitti piirretään omasta paikasta.

2. Määritetään kohdeosoitteen latitude ja longitude arvot getFromLocationName()-funktiolta. Oma piste on jo tiedossa koska se päivittyy aina kun GPS-vastaanotin saa uuden paikkatiedon.

3. Haetaan Googlen palvelusta reittipisteet kohdeosoitteen ja oman pisteen välille.

4. Piirretään DirectionPathOverlay-luokan avulla reittiviiva haettujen reittipisteiden kautta. (17.)

Kuvassa 40 näkyy piirretty reitti ja määräimet käyttöliittymässä.

KUVA 40. Reitti ja määräimet kartalla käyttöliittymässä

5 VIIMEISTELY

Viimeistelytoimenpiteinä paranneltiin ulkoasua ottamalla layouteista värit pois ja nostamalla fonttikokoa dialogi-ikkunoissa. Lisäksi karttanäkymään lisättiin zoom-kontrollit.

Saapuneeseen määräimeen lisättiin hieman animointia, jotta käyttäjä huomaisi uuden määräimen paremmin. Keltainen ja vilkkuva määräin indikoi saapunutta, statukseltaan ”UUSI” olevaa määräintä. Kun määräin asetetaan määräinlistaan, siihen liitetään vilkkuva animaatio.

```
// Muutetaan kuvan alpha arvoa täysin näkyvän ja näkymättömän vä-
lillä, kuva vaikuttaa vilkkuvan
Animation animation = new AlphaAnimation(1, 0.5f);
// yhden animoinnin kesto 500 ms
animation.setDuration(500);
// Jatkuvasti muuttuva animointi
animation.setInterpolator(new LinearInterpolator());
// Animointia toistetaan loputtomasti
animation.setRepeatCount(Animation.INFINITE);
// Animointia toistetaan alusta-loppuun-lopusta-alkuun
animation.setRepeatMode(Animation.REVERSE);
// aloitetaan animointi
maaraimet[counter].startAnimation(animation);
```

Haluttiin vielä varmistaa, että käyttäjä huomaa määräimen saapumisen. Lisättiin vielä pieni äänimerkki resursseihin. Äänimerkki (.wav-tiedosto) soitetaan, kun määräin saapuu. Äänimerkki soitetaan Androidin MediaPlayer-luokan oliolla.

```
MediaPlayer mp = MediaPlayer.create(MapsDemo.this, R.raw.mywavfile);
mp.start();
```

Lisättiin vielä karttaan zoom-kontrollit, joilla karttanäkymää voi lähentää tai loitontaa (22).

```
RelativeLayout.LayoutParams zoomControlParams = new RelativeLay-
out.LayoutParams(RelativeLayout.LayoutParams.WRAP_CONTENT, Rela-
tiveLayout.LayoutParams.WRAP_CONTENT );
```

```
zoomControlParams.addRule(RelativeLayout.ALIGN_PARENT_TOP);
zoomControlParams.addRule(RelativeLayout.CENTER_HORIZONTAL);

mapLayout.addView(mapView.getZoomControls(),zoomControlParams);
```

6 LOPPUTESTAUS

Lopputestaus on tässä tapauksessa pidempi yhtäjaksoinen kokonaistestaus, jossa todetaan sovelluksen toimiminen realistisissa olosuhteissa ennen tuotteen mahdollista julkaisua. Suurin osa sovelluksen testauksesta on suoritettu samaan aikaan kohdelaitteessa kun sovellusta on toteutettu. Tällaisessa lohkotestauksessa ei välttämättä huomata pidemmällä aikavälillä olevaa vaikutusta sovelluksessa, esimerkiksi virrankäytön ja muistivuotojen aiheuttamaa vaikutusta.

Lopputestauksessa huomattiin ainakin muutamia muistinhallinnallisia ongelmia. Ohjelmassa oli muistivuotoja eli se ei vapauttanut varaamaansa muistia kun sitä ei enää tarvittu. Kun vapaa muisti lopulta loppui, ohjelma kaatui. (23.)

Suurin osa muistivuodoista löytyi säikeiden sisältä, joten ohjelma kaatui aika äkkiä. Muistivuotoja etsittiin testiohjelmilla ja tarkistettiin kaikki koodi, jossa muistivuotoja voisi syntyä. Muistivuodot saatiin kuriin, ja ohjelma toimi moitteettomasti.

Lopputestauksessa huomattiin myös, että jostain syystä reitit eivät näy koko pituudelta alkupisteen ja määränpään välillä, vaan reitti jää vajaaksi. Kävi ilmi, että Googlen käyttöehdot rajoittavat reitin pituutta ilmaisella lisenssillä. Kun tutkittiin käyttöehtoja tarkemmin, huomattiin, että käyttöehdot estävät sovelluksen tuotteistamisen sellaisenaan.

7 VALMIS SOVELLUS

7.1 Toteutetut ominaisuudet

Sovellus toteuttaa nyt kaikki sille asetetut vaatimukset. Se pystyy vastaanottamaan työmääräimiä ja lähettämään omaa paikkatietoaan palvelimelle. Myös kevyt navigointi kartan avulla onnistuu. Kevyellä navigoinnilla tarkoitetaan, että käyttäjä näkee vain reitin omasta paikasta kohdemääränpäähän, eikä häntä ohjata kohteeseen esimerkiksi äänikomennoin. Määräimet näkyvät ruudulla listassa, jota painamalla avautuu määräindialogi, josta käyttäjä näkee reitin kartalle ja voi aloittaa, keskeyttää ja suorittaa työn. Myös kuvan ottaminen on mahdollista. Käyttäjä tunnistetaan autentikoinnin avulla. Autentikointi suoritetaan sekä palvelinpyyntöjen yhteydessä että kirjautumisdialogin avulla.

Projektin myötä toteutettiin paljon sellaista toiminnallisuutta, josta tulevaisuudessa on hyötyä yritykselle Android-sovelluskehityksessä. Android-alustan suosion myötä tämä projekti ei varmasti jää yrityksen viimeiseksi Android-sovellukseksi.

Projektin toteutuksen myötä saatiin laaja-alaisesti aikaan ohjelmointipohjia versiointiin, karttanäkymän lisäämiseen ja piirtoon, laitteen asetusten tallennukseen, sokettikäsittelyyn, ulkoasun suunnitteluun, päivityksen tarjoamiseen ja versiointiin sekä GPS-datan kuunteluun laitteesta.

Koska työhön on kirjoitettu hyvin yksityiskohtaisesti työn toteutus, aloittelevan tai karttaprojektia jatkavan työntekijän on mahdollista käyttää sitä vaikka oppaana Android-sovelluskehityksessä.

Tämä sovellus on myös yrityksen ensimmäinen, joka käyttää kokonaisvaltaisesti Track-My-Work mobile -projektin mukaista uudistettua protokollaa. Tämän sovelluksen kehittäminen on omalta osaltaan edistänyt protokollan

suunnittelua ja toteutusta, koska se toi konkretiaa palvelinohjelmiston yksityiskohtaiseen toteuttamiseen.

7.2 Jatkokehitys

Sovelluksen ulkoasua kehitetään edelleen ja karttatekniikka vaihdetaan OpenStreetMap-tekniikkaan. Alustavan karttatekniikanvaihdoksen suunnittelun perusteella karttatekniikan vaihtaminen lennosta onnistuu eikä suurempia ongelmia todennäköisesti ilmenisi. OSM sisältää lähes identtiset komponentit kuin Googlen Maps-luokka. Piirto tapahtuu Overlay-luokalla ja reitin piirto onnistuu samalla tavalla, mutta reitin koordinaattipisteet haetaan alustavien suunnitelmien perusteella yournavigation.org-palvelusta (OpenSource). Sovellus on siis toiminnaltaan ihan samanlainen, mutta kartta on OpenStreetMap-karttanäkymä eikä Googlen.

OpenStreetMap-tekniikka tuo mukanaan muitakin suuria hyötyjä, koska se on ilmaisen lähdekoodin karttatekniikka. Tekniikalla on saatu aikaan näyttäviä kartta- ja navigointisovelluksia.

8 OMIA MIETTEITÄ

Googlen käyttöehdot estävät sovelluksen tuotteistamisen sellaisenaan. Ehtojen mukaan käyttäjältä vaaditaan voimassa oleva Google-tili, Google omistaa oikeuden mainostaa sovelluksen kartta-komponentin sisällä eikä sovellusta saa kaupallistaa, vaan se on oltava ladattavissa ilmaiseksi. Käyttöehtojen mukaan myöskään ajoneuvoa ei saa seurata eikä navigoida Google Maps API-sovelluksella. Pituudeltaan rajattu reitin piirto on kuitenkin sallittu. (24.)

Jos sovellus olisi kaupallistettu, olisi Googlen käyttöehtoja rikottu räikeästi. Päätettiin vaihtaa karttatekniikka OpenStreetMap-karttatekniikkaan (OSM), joka on avoimen lähdekoodin tekniikka ja sitä saa käyttää myös kaupallisissa sovelluksissa.

Googlen karttatekniikalla toimiva työmääräinsovellus oli täysin valmis. Aika ei kuitenkaan ihan riittänyt karttatekniikan vaihdokseen ja päätettiin rajata insinöörityön osuus Google-pohjaiseen sovellukseen ja jatkokehityksessä keskittyä OpenStreetMap-tekniikkaan.

Käyttöehtovaikeuksia lukuun ottamatta projekti oli onnistunut. Tiukaksi suunniteltu aikataulu piti hyvin, vaikka aluksi oli vaikeaa ennustaa miten projektissa käy.

Kaikkiaan projekti oli mieluisa opinnäytetyö. Aluksi pelotti lähteä tekemään sovellusta niin tiukalla aikataululla, koska Android-alustalle kehittämisestä ei juurikaan ollut kokemuksia, Java-ohjelmoinnista muutama kurssi ja yksi harjoitustyö. Aiempien Java-taitojen ansiosta projekti lähti kuitenkin nopeasti liikkeelle ja se suoritettiin loppuun asti ilman suurempia vaikeuksia. Android-sovelluskehityksestä löytyi paljon harrastelijoiden tekemiä oppaita ja Androidin oma dokumentaatio oli kattava.

Android-alustalle kehittäminen avautui yllättävän helposti. Suurin syy tähän oli, että toteutuksessa kaiken joutui tekemään johdonmukaisesti alusta loppuun työkalujen ja ympäristön asennuksesta sovelluksen luomiseen ja lopul-

ta testaukseen. Kyseessä oli kokonaisvaltainen ohjelmointiprojekti, joka sisälsi kaikki projektin osa-alueet vaatimusmäärittelystä lopullisen toteutetun sovelluksen testaukseen.

Työ oli myös hyvin itsenäinen. Koska itsellä tai kenelläkään yrityksessä ei ollut juurikaan kokemusta Android-ohjelmoinnista, tarvittavat tiedot ja ratkaisumallit piti selvittää itse. Jälkikäteen ajateltuna tulee mieleen, että jos saman sovelluksen tekisi uudelleen alusta asti kaiken oppimansa perusteella, sen saisi aikaan murto-osassa tähän projektiin kulutettuun aikaan verrattuna. Tästä huomaa, että insinöörityö oli ennen kaikkea oppimisprosessi, koska valtaosa ajasta käytettiin uusien tekniikoiden ja ratkaisujen opetteluun. Kaikkiaan Android-ohjelmointi oli ja on jatkossakin mielekästä puuhaa. Oman käden jäljen näkee nopeasti ja ratkaisut ongelmiin löytyvät suhteellisen helposti.

LÄHTEET

1. Jauhiainen, Lauri 2011. Track-My-Work mobile -projektisuunnitelma. Max Technologies Oy.

2. Android (operating system). 2011. Saatavissa: http://en.wikipedia.org/wiki/Android_(operating_system). Hakupäivä 13.5.2011.

3. Eclipse (software). 2011. Saatavissa: http://en.wikipedia.org/wiki/Eclipse_(IDE). Hakupäivä 13.5.2011.

4. How to Install Android SDK + ADT into Eclipse IDE on Windows OS. (tutorial). Hakupäivä: 16.8.2011. http://www.youtube.com/watch?v=OIL1UouA4dE.

5. Hello, World. 2011. Saatavissa: http://developer.android.com/resources/tutorials/hello-world.html. Hakupäivä 16.8.2011.

6. Common Layout Objects. 2011. Saatavissa: http://developer.android.com/guide/topics/ui/layout-objects.html. Hakupäivä: 16.8.2011.

7. Lee, Wei-Meng 2010. Undestanding User Interface in Android. Saatavissa: http://mobiforge.com/designing/story/understanding-user-interface-android-part-2-views. Hakupäivä: 16.8.2011

8. JSON. 2011. Saatavissa: http://en.wikipedia.org/wiki/JSON. Hakupäivä 16.8.2011.

9. Data Storage. 2011. Saatavissa: http://developer.android.com/guide/topics/data/data-storage.html. Hakupäivä: 16.8.2011.

10. Creating dialogs. 2011. Saatavissa: http://developer.android.com/guide/topics/ui/dialogs.html. Hakupäivä 16.8.2011.

11. Obtaining user location. 2011. Saatavissa: http://developer.android.com/guide/topics/location/obtaining-user-location.html. Hakupäivä 16.8.2011.

12. Sensor Manager. 2011. Saatavissa: http://developer.android.com/reference/android/hardware/SensorManager.html .Hakupäivä 16.8.2011.

13. Harvey, Martin 2000. Multithreading - The Delphi Way: Chapter 1: What are Threads? Why use them? Saatavissa: http://www.eonclash.com/Tutorials/Multithreading/MartinHarvey1.1/Ch1.html#Why use threads?. Hakupäivä 16.8.2011

14. The AndroidManifest.xml file. 2011. Saatavissa: http://developer.android.com/guide/topics/manifest/manifest-intro.html. Hakupäivä 16.8.2011

15. Tech Blog 2008, Getting Android emulator working with Google Maps API Key. Saatavissa: http://informationideas.com/news/2008/11/06/getting-android-emulator-working-with-google-maps-api-key/. Hakupäivä: 16.8.2011.

16. Jyväskylän Yliopisto projects Wiki 2011. Androidin ohjelmointia. Saatavilla: https://trac.cc.jyu.fi/projects/ohj2/wiki/Android. Hakupäivä: 16.8.2011

17. Lee, Wei-Meng 2010. Using Google Maps in Android. Saatavilla: http://mobiforge.com/developing/story/using-google-maps-android. Hakupäivä 16.8.2011.

18. Writing the Server Side of a Socket. 2011. Saatavissa: http://download.oracle.com/javase/tutorial/networking/sockets/clientServer.html. Hakupäivä 16.8.2011.

19. Classes. 2011. Saatavissa: http://download.oracle.com/javase/tutorial/java/javaOO/classes.html. Hakupäivä 16.8.2011

20. Activity. 2011. Saatavissa: http://developer.android.com/reference/android/app/Activity.html. Hakupäivä 16.8.2011.

21. Chorniy, Andrey 26.4.2010. How to launch android camera using intents. Saatavissa: http://achorniy.wordpress.com/2010/04/26/howto-launch-android-camera-using-intents/. Hakupäivä 16.8.2011.

22. Murphy, Mark 2008. WebView ZoomControls Issues. Kirjoitus Google groupsin keskustelupalstalla. Saatavissa: http://groups.google.com/group/android-developers/browse_thread/thread/b4a12843cd33497b. Hakupäivä 16.8.2011.

23. Muistivuoto. 2011. Saatavissa: http://fi.wikipedia.org/wiki/Muistivuoto. Hakupäivä 1.6.2011.

24. Google 2011. GoogleMaps/Google Earth APIs Terms of Service. Saatavilla: http://code.google.com/intl/fi-FI/apis/maps/terms.html. Hakupäivä 16.8.2011.