OULUN SEUDUN
AMMATTIKORKEAKOULU

Jarno Niemelä

GOOGLE PLACES -PAIKANNUSSOVELLUS JA KAVERIPYYNNÖT SOSIAALISEN MEDIAN JÄRJESTELMIIN

GOOGLE PLACES -PAIKANNUSSOVELLUS JA KAVERIPYYNNÖT SOSIAALISEN MEDIAN JÄRJESTELMIIN

Jarno Niemelä
Opinnäytetyö
17.11.2011
Tietotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietotekniikan koulutusohjelma, ohjelmistojen tuotanto

Tekijä: Jarno Niemelä
Opinnäytetyön nimi: Google Places -paikannussovellus ja kaveripyynnöt sosiaalisen median järjestelmiin
Työn ohjaajat: Mikko Kerttula, SoloCEM Systems Oy; Pertti Heikkilä, Oulun seudun ammatti-korkeakoulu
Työn valmistumisajankohta: syksy 2011 Sivumäärä: 47 + 2 liitesivua

---

SoloCEM Systems Oy on uusi, ns. startup-yritys, joka on perustettu syksyllä 2010. Yritys kehittää mobiilia sosiaalista palvelua nimeltään 6Starz. 6Starz-palvelu on suunnattu Android- ja Symbian-mobiililaitteille.

Tämän opinnäytetyön aiheena oli kehittää kohteiden paikannussovellus 6Starz-palveluun sekä kaverikutsu Facebook-, Twitter- ja LinkedIn-palveluihin mobiililaitteen käyttöliittymästä. Työn sovellusten kehitysalustana oli Google Android.

Opinnäytetyössä saatiin toteutettua toimiva Google Places APIa hyödyntävä Android-paikannussovellus, jolla voidaan etsiä käyttäjän lähellä olevia kohteita, kuten ravintoloita, yökerhoja, baareja ja kahviloita, sekä näyttää näistä tietoja ja lisätä näitä tietoja yrityksen 6Starz-palveluun käyttäen sähköpostiviestiä tiedon lähetykseen. Työssä toteutettiin myös yksinkertaiset kaveripyynnöt Twitteriin, Facebookiin ja LinkedIniin Android-mobiililaitteen käyttöliittymästä.

Työssä käytetyt APIt ja palvelut tarjoavat monipuolisia ominaisuuksia, ja ne soveltuvat hyvin pienimuotoisiin avoimen lähdekoodin projekteihin. Suurempiin kaupallisiin projekteihin niitä on kuitenkin melko vaikea suositella, sillä esimerkiksi niiden rajoitukset ja vaatimukset saattavat muuttua hyvinkin nopeasti.

Asiasanat: sosiaalinen media, API, ohjelmointi, Android, Google Maps, Google Places, 6Starz

SISÄLLYS

SANASTO

6Starz	SoloCEM Systems Oy:n kehittämä ja mobiililaitteille suunnattu sosiaalinen palvelu.
Android	Googlen kehittämä mobiililaitteisiin suunnattu käyttöjärjestelmä.
API	Application Programming Interface, kokoelma ohjelmoitavia kirjastoja.
Facebook	Internetissä toimiva yhteisöpalvelu.
Foursquare	Mobiililaitteille suunnattu ja paikannukseen perustuva yhteisöpalvelu.
Google Maps	Googlen kehittämä ilmainen karttapalvelu.
Google Places	Googlen kehittämä paikkatietopalvelu.
JSON	JavaScript Object Notation, yksinkertainen tiedonsiirtomuoto, jota voidaan käyttää monilla ohjelmointikielillä.
LinkedIn	Ammattilaisille suunnattu yhteisöpalvelu.
NFC	Near Field Communication, radiotaajuinen hyvin lyhyiden etäisyyksien etätunnistustekniikka.
Qt	Nokian omistama alustariippumaton ohjelmistokehitysympäristö.
SDK	Software Development Kit, tuotekehitystyökalu, joka sisältää koodia ja on tarkoitettu yleensä tietylle kehitysalustalle.
SSL	Secure Sockets Layer, salausprotokolla, jolla voidaan suojata internet-sovellusten tietoliikenne IP-verkkojen yli.
Symbian	Symbian ltd:n kehittämä mobiililaitteisiin suunnattu käyttöjärjestelmä.

Twitter Internetissä toimiva yhteisöpalvelu.

1 JOHDANTO

SoloCEM Systems Oy on uusi ns. startup-yritys, joka on perustettu syksyllä 2010. Yritys kehittää mobiilia sosiaalista palvelua nimeltään 6Starz, joka on julkaistu Monacossa WIMA-messuilla huhtikuussa 2011. 6Starz-palvelu on suunnattu mobiililaitteille. Tällä hetkellä kehitysalustoina ovat Symbian ja Android. Myöhemmin palvelu tulee mahdollisesti myös Applen iOS-alustalle.

Yritys tarvitsi Qt- tai Android-osaajaa kehittämään paikannuspalvelua ja yhteyksiä sosiaalisiin järjestelmiin omaan 6Starz-pilottiinsa. Olin tehnyt Qt-projekteja Oulun seudun ammattikorkeakoulun UbiGo-hankkeelle ja sain tiedon opinnäytetyön aiheesta sitä kautta.

Sain yritykseltä opinnäytetyön aiheen, jossa oli tarkoituksena kehittää kohteiden paikannussovellus 6Starz-palveluun sekä kaverikutsu Facebook-, Twitter- ja LinkedIn-palveluihin mobiililaitteen käyttöliittymästä. Kehitysalustoissa oli kaksi vaihtoehtoa, joko Symbian Qt tai Android. Päädyimme siihen, että tekisin sovellukset Android-alustalle, sillä halusin kokemusta Android-sovelluskehityksestä.

Paikannussovelluksessa käytettiin hyväksi Google Maps -karttapalvelua kohteiden esittämiseen sekä Google Places APIa kohteiden tietojen hakemiseen. Koska työ oli ns. protoilua, sovelluksen käyttöliittymän ulkoasu ei ollut tärkeä, vaan yleinen toimivuus ja helppokäyttöisyys olivat tärkeämpiä prioriteetteja. Kaveripyynnöissä tarkoituksena oli vain saada aikaan toimiva pyyntö ja siksi käyttöliittymistä tuli vieläkin pelkistetympiä.

2 KÄYTETYT YMPÄRISTÖT JA TEKNIIKAT

2.1 Google Places API

Google Places on palvelu, joka tarjoaa tietoa kohteista kuten liikkeistä, kaupoista, ravintoloista, yökerhoista sekä muista maantieteellisistä kohteista ja yleisistä paikoista. Tiedon hakuun Google Places APIsta käytetään HTTPS-pyyntöjä ja vastaus saadaan joko JSON- tai XML-muodossa. Kohteista saadaan uniikit pituus- ja leveyskoordinaatit (latitude, longitude), joita voidaan käyttää kohteen ilmaisemiseen karttanäkymässä. (The Google Places API. 2011.) Palvelu julkaistiin syyskuussa 2009 nimellä Local Business Center ja nimettiin uudelleen huhtikuussa 2010 Google Places -nimiseksi (Hanke 2010).

2.1.1 Places APIn rajoitukset ja vaatimukset

Google Places API käyttää yksilöllistä API-avainta eri sovellusten tunnistamiseen. Avaimet luodaan Google API -konsolissa käyttäen Google Mail -tiliä.

Google Places API sisältää seuraavia tiedon pyyntöä rajoittavia tekijöitä:

- API-avainta käyttävä sovellus voi tehdä 1000 pyyntöä joka 24 tunnin ajan.

- Avainta käyttävä sovellus, joka on tunnistettu API-konsolissa luottokorttilla, voi tehdä 100 000 pyyntöä joka 24 tunnin ajan. Luottokorttia ei kuitenkaan laskuteta mitenkään.

- Jos sovellus näyttää Places-tietoa muulla kuin Google Maps -kartalla, samassa yhteydessä tulee näyttää ”Powered by Google” -logoa. (The Google Places API. 2011.)

2.1.2 Places-tietotyypit

Place Searches

Place Searches -ominaisuutta käytetään, kun halutaan listata käyttäjän lähellä olevat kohteet. Kohteet etsitään lähettämällä HTTPS-pyyntö osoitteeseen

https://maps.googleapis.com/maps/api/place/search/output?parameters. Output-kohtaan tulee joko xml tai json sen mukaan, kummalla tavalla halutaan käsitellä palautettavaa tietoa. Parameters-kohdassa määritellään pyynnön mukana lähetettävät tiedot, joilla rajataan hakualue. Näitä tietoja ovat location, radius, language, name, sensor ja key. Location-elementtiin lisätään hakualueen pituus- ja leveyskoordinaatit, joiden perusteella kohteet etsitään. Koordinaattien tulee olla google.maps.LatLng-muodossa. Radius-elementtissä ilmoitetaan hakusäde metreissä, jonka sisältä etsintä suoritetaan. Types-elementillä voidaan rajata etsittävät kohteet tiettyihin tyyppeihin. Language-elementillä voidaan määrittää kieli, jolla tiedot palautetaan. Name-elementillä voidaan rajata tuloksia sisältämään vain tietynnimisiä kohteita. Sensor-elementillä määritetään, tuliko pyyntö laitteesta, jossa on käytössä GPS-paikannin (true tai false). Key-elementissä ilmoitetaan yksilöllinen API-avain, jolla tunnistetaan käytettävä tili ja sen sisältävä pyyntömäärä. (The Google Places API. 2011.)

Palvelin palauttaa tiedot joko JSON- tai XML-muodossa, kolmena elementtinä (status, results ja html_attributions). Status-elementissä ilmoitetaan, onnistuiko pyyntö. Results-elementissä palautetaan enintään 20 kohteen lista ja niissä yksittäisten kohteiden tiedot. Näitä tietoja ovat kohteen nimi, kohteen lähellä oleva merkittävä kohde (useimmiten katu), kohteen tyyppi, kohteen sijainti (pituus- ja leveyskoordinaatit), ikoni, jota voidaan käyttää karttanäkymässä, kohteen viite (reference, uniikki tieto, jota käytetään, kun kohteesta haetaan tarkempaa tietoa) ja kohteen id (kohteen uniikki id-numero). Html_attributions-elementissä palautetaan mahdolliset palveluntarjoajan lisätiedot. (The Google Places API. 2011.)

Place Details

Place searches -pyynnön tuloksista saatua yksittäisen kohteen viitettä käytetään Place details -pyynnössä ilmaisemaan haluttu kohde. Details-haussa saadaan tietoina mm. kohteen nimi, osoite, puhelinnumero, käyttäjien tekemä kohteen arvostelu ja kohteen Places url -osoite. Pyyntö lähetetään osoitteeseen https://maps.googleapis.com/maps/api/place/details/output?parameters. Pyynnössä ilmoitetaan haluttu palautustyyppi (JSON tai XML) ja parametreinä kohteen reference, sensor, API-avain ja valinnaisena palautuskieli. (The Google Places API. 2011.)

Palvelin palauttaa tiedon halutussa muodossa, kolmena elementtinä (status, result ja html_attributions). Status ilmoittaa, onnistuiko pyyntö. Result-elementissä palautetaan koh-

teen tiedot ja html_attributions-elementissä mahdolliset palveluntarjoajan lisätiedot. (The Google Places API. 2011.)

Place Check-Ins

Place-kohteeseen voidaan kirjautua, ja tällöin kohde sijoittuu korkeammalle hakutuloksissa. Jos hakutuloksia on enemmän kuin sallitut 20, suosituimmat kohteet listataan ensimmäisinä. Kirjautumiseen tarvitaan kohteen viite, sensor ja API-avain. Check-in-pyyntö on HTTPS POST -tyyppinen pyyntö, ja vastaus saadaan joko JSON- tai XML-muodossa. Tässä tapauksessa saadaan vastauksena status-viesti, joka on joko ”OK” tai virheilmoitus. (The Google Places API. 2011.)

User Place Reports

Place Report -toimintoa käytetään uusien Places-kohteiden lisäämiseen ja vanhojen poistamiseen. Uudet lisätyt kohteet näkyvät välittömästi ne lisänneessä sovelluksessa, mutta ne näkyvät muille sovelluksille vasta, kun Google on moderoinut ne. Hyväksytyn moderoinnin jälkeen kohdetta ei voi enää poistaa. Jos kohde halutaan poistaa, se tulee tehdä samasta sovelluksesta kuin se on luotu ja ennen hyväksyttyä moderointia. Place report -pyyntö on HTTPS POST -pyyntö ja siinä lähetetään sensor- ja API-avain-tietojen lisäksi location-tieto, johon tulee kohteen pituus- ja leveys-koordinaatit sekä accuracy-tieto, jossa ilmaistaan kohteen etäisyys location-tiedosta metreissä. Näiden lisäksi tulee ilmoittaa vielä kohteen nimi, tyyppi ja kieli, jolla kohteen nimi ilmoitetaan. Vastauksena Place report -pyynnölle saadaan JSON- tai XML-vastaus, jossa ilmoitetaan status-tieto, joka on joko ”OK” tai virheilmoitus, sekä uuden kohteen viite ja id. Jos tämä kohde halutaan poistaa, lähetetään uusi HTTPS POST -pyyntö, jossa ilmoitetaan sensor- ja API-avaimen lisäksi tämän kohteen viite. (The Google Places API. 2011.)

2.2 Google Maps

Google Maps (aiemmin nimeltään Google Local) on yhdysvaltalaisen Googlen tuottama karttapalvelu. Se tarjoaa katseltavaksi katu- ja satelliittikarttoja sekä mahdollisuuden matkareitin suunnitteluun ja paikallisten yritysten etsimiseen. (Google Maps. 2011.)

2.2.1 Google Maps External Library

Google tarjoaa ulkoisen Android-kirjaston Maps-palveluun. Kirjasto sisältää com.google.android.maps-paketin, jonka luokat tarjoavat sisäänrakennettuja ominaisuuksia karttaruutujen lataamiseen, renderöintiin ja puskurointiin sekä useita näyttöasetuksia ja kontrolleja. Maps-paketin pääluokka on nimeltään MapView. MapView näyttää Google Maps -palvelusta saatua tietoa. MapView sisältää näppäinpainellusten ja eleiden tulkintaa sekä käyttää niitä esimerkiksi kartan zoomaamiseen ja uusien karttaruutujen lataamiseen. Luokka sisältää myös tarpeellisia käyttöliittymäelementtejä kartan hallintaan. MapView-elementtiä voidaan hallita ohjelmallisesti, ja sen päälle voidaan piirtää Overlay-tasoja. (Location and Maps. 2011.)

2.2.2 Maps APIn rajoitukset ja vaatimukset

Google Maps APIn käytössä on muutama ehto, jotka täytyy huomioida sovellusta suunniteltaessa. Sovelluksen tulee olla ilmainen käyttää. Mobiilisovellus voi kuitenkin olla ladattavissa maksua vastaan esimerkiksi Android Marketista. (Google Maps/Google Earth APIs Terms of Service. 2011. Kohta 9.1.2.b.) Ei-kaupallisessa sovelluksessa ei varsinaisessa karttojen selailussa ole rajoituksia. Kaupallisessa sovelluksessa karttoja voidaan ladata 25 000 kappaletta vuorokaudessa, jonka jälkeen niistä täytyy maksaa tai hankkia Google Maps API Premier -lisenssi. (Sign Up for the Google Maps API. 2011.) Maps API Premier -lisenssi ei kuitenkaan työn kirjoitushetkellä sisältänyt Google Maps External Library -palvelua. (Frequently Asked Questions. 2011.)

2.3 Android

Android on matkapuhelimille sekä muille mobiililaitteille suunniteltu ohjelmistopino, joka sisältää käyttöjärjestelmän, väliohjelmistoja ja käyttäjän perusohjelmia. Siinä käytetään avoimen lähdekoodin GPLv2-lisenssoitua Linux-käyttöjärjestelmäydintä. Androidia kehitti alun perin Android Inc., jonka Google myöhemmin osti, ja nykyisin sen kehittämisestä vastaa Open Handset Alliance. Android-laitteisiin tarkoitettua koodia kirjoitetaan Java-ohjelmointikielellä ja se käyttää Googlen kehittämiä Java-kirjastoja. Android julkistettiin 5. marraskuuta 2007 Open Handset Alliancen perustamisen yhteydessä. Open Handset Alliance koostuu 71 laitteisto- ja

ohjelmistovalmistajasta sekä teleoperaattorista. Google julkisti suurimman osan Androidin koodista avoimen koodin ja vapaan ohjelmiston Apache-lisenssillä. (Android. 2011.)

2.4 Java

Java-ohjelmointikielen kehittivät Bill Joy ja James Gosling kollegoineen Sun Microsystemsillä 1990-luvun alussa. Java-kielen 1990-luvun lopulla saavuttaman suuren suosion takana ovat laitteistoriippumattomuuden lisäksi kielen C++-kieltä läheisesti muistuttava, mutta helpommin omaksuttavaksi suunniteltu kielioppi, oliopohjaisuus ja virtuaalikoneen mukana tuleva, erittäin kattava standardikirjasto. Java sisältää runsaasti ominaisuuksia, kuten graafisen käyttöliittymäkirjaston, rinnakkaisuuden hallinnan, verkko-ominaisuudet ja rikkaat rajapinnat. (Java. 2011.)

2.5 Eclipse

Eclipse on ohjelmointiympäristö, joka tukee muun muassa Java-ohjelmointikieltä. Muita tuettuja ohjelmointikieliä ovat esimerkiksi C, C++ ja PHP. Ympäristöä kehitetään avoimen lähdekoodin lisenssillä. Eclipsen on alunperin kehittänyt IBM, joka julkaisi ohjelman 2001 avoimen lähdekoodin lisenssille. Vuodesta 2004 ohjelman kehityksestä on vastannut säätiö. Eclipse on toteutettu pääasiassa Java-ohjelmointikiellä. Eclipse-kehitysympäristöstä on useita eri jakelupaketteja, jotka on räätälöity eri kehittäjien tarpeisiin. (Eclipse (IDE). 2011.)

2.6 Facebook

Facebook on internetissä toimiva yhteisöpalvelu. Sivusto tarjoaa käyttäjille mahdollisuuden kuvallisen käyttäjäprofiilin luomiseen sekä yhteydenpitoon ystäviensä kanssa. Facebookissa on myös mahdollista liittyä erilaisiin yhteisöihin ja saada tietoa tulevista tapahtumista. Palvelua ylläpitää ja sen omistaa samaista nimeä kantava yritys, jonka kotipaikka on Palo Altossa Kalifornian osavaltiossa Yhdysvalloissa. (Facebook. 2011.)

Facebook Places julkaistiin elokuussa 2010. Se oli alun perin Foursquaren kaltainen mobiilipalvelu, jossa käyttäjä pystyi kirjautumaan johonkin kohteeseen ja näkemään, mihin hänen ystävänsä ovat kirjautuneet. Elokuussa 2011 Facebook kuitenkin muutti Places palveluaan

siten, että käyttäjät eivät enää pelkästään kirjaudu johonkin kohteeseen mobiililaitteellaan, vaan paikkatieto voidaan ilmoittaa myös internet-selaimella ja tieto on näkyvissä kaikessa Facebookissa näkyvässä tiedossa. Paikannuspalvelussa näytetään, missä käyttäjä on ollut, missä hän on nyt ja mihin hän on menossa. (Protalinski 2011.)

2.7 Twitter

Twitter on yhteisö- ja mikroblogipalvelu, jonka käyttäjät pystyvät lähettämään ja lukemaan toistensa päivityksiä internetissä. Tekstipohjaiset viestit eli twiitit, voivat sisältää enintään 140 merkkiä. Käyttäjien on mahdollista lähettää ja vastaanottaa päivityksiä Twitter-nettisivuston kautta, tekstiviesteinä, RSS-syötteenä tai erilaisten sovellusten kautta. Palvelun käyttö internetissä on maksutonta. Twitterin kehitys aloitettiin keväällä 2006 ja palvelu julkaistiin heinäkuussa 2006. (Twitter. 2011.)

2.8 LinkedIn

LinkedIn on verkkoyhteisöpalvelu, verkostoitumisväline, ns. professional tool. Käyttäjät voivat laittaa sinne CV:n, harrastukset ja kiinnostukset. He voivat saada suosituksia entisiltä pomoilta ja työkavereilta sekä suositella muita. LinkedIn on hyvä väline laajentaa omaa verkostoa ja sitä kautta saada mahdollisia työtarjouksia sekä löytää hakemiansa työntekijöitä. (LinkedIn. 2011.)

2.9 6Starz

6Starz on SoloCEM Systems Oy:n kehittämä sosiaalinen paikannuspalvelu Android- ja Symbian-matkapuhelimille, jossa käyttäjät näkevät, ketä muita käyttäjiä on heidän kanssaan samassa kohteessa, ja voivat seurustella heidän kanssaan sekä solmia uusia ystävyyksiä. Kohteen omistajat voivat tarjota 6Starz-käyttäjille mainoksia ja tarjouksia palveluun kuuluvan hallintatyökalun kautta.

6Starz hyödyntää NFC-lähilukuteknologiaa (Near Field Communication) paikkoihin ja tapahtumiin sisäänkirjautumisessa, uusien ystävälinkkien muodostamisessa ja mobiilietukuponkien lunastamisessa. Palvelu koostuu matkapuhelinsovelluksesta, joka toimii yleisimmissä puhe-

linalustoissa, webbipalvelusta, jonka kautta käyttäjät voivat hallita profiiliaan ja ystäväkontaktejaan, sekä taustajärjestelmästä, joka tarjoaa mm. työkalut paikan omistajille. Palvelussa on linkitys myös muihin yhteisöpalveluihin, kuten Facebookiin ja Twitteriin. (6Starz tuo NFC:n ja yhteisöpalvelut ravintoloihin ja tapahtumiin. 2011.)

6Starz Biz on palvelu, joka tarjoaa yritykselle uuden tavan palvella asiakkaitaan. Aluksi yritys rekisteröi kohteen 6Starz Biz -palveluun ja tulostaa 6Starz-julisteen, jossa on kohteen uniikki QR-koodi ja NFC-tägi. Käyttäjät vovat sitten kirjautua kohteeseen lukemalla joko QR-koodin tai NFC-tägin ja aloittaa seurustelun muiden käyttäjien kesken. Yritys voi sitten lähettää mainoksia käyttäjille ja seurata heidän toimintaansa kohteessa. Käyttäjät voivat saada myös erikoistarjouksia, jos he ovat aktiivisia kohteessa ja jos he kutsuvat ystäviään yrityksen uusiksi asiakkaiksi. (Tehosta liiketoimintaasi uudella 6Starz Biz -palvelulla. 2011.)

3 OPINNÄYTETYÖN TOTEUTUS

3.1 Työn kuvaus

Työn teknisessä osassa tuotettiin Android-paikannussovellus, jolla pystyttiin etsimään käyttäjän lähellä olevia kohteita kuten ravintoloita, yökerhoja ja baareja, sekä Android-sovellukset Facebook-, LinkedIn- ja Twitter-kaveripyyntöjä varten. Paikannussovelluksen kohteet näytettiin mobiililaiteessa Google Maps -karttanäkymässä, kuvakkeina. Kohteiden tiedot saatiin käyttämällä Google Places APIa. Tiedonhakuun olisi voitu käyttää myös Foursquare APIa tai Facebook APIn Places-osiota, mutta päädyimme yrityksen IT-arkkitehdin kanssa pidetyssä palaverissa siihen, että emme käytä näitä kumpaakaan paikannussovelluksessa niiden huonon dokumentoinnin ja haastavuuden vuoksi. Työssä myös haettiin ainoastaan lähellä olevat paikat sekä näytettiin niistä tietoja, ja tähän käyttöön Google Places API soveltui riittävän hyvin. Sovelluksessa tuli myös toteuttaa mahdollisuus siirtää tiedot yrityksen omalle palvelimelle. Yritys ei kuitenkaan saanut käyttöönsä palvelinta, jota olisin voinut käyttää työssä, joten kohteiden lähettäminen toteutettiin sähköpostiviestiä hyödyntäen.

Kaveripyyntösovelluksissa pyyntö kohdistettiin palvelun yksittäisen käyttäjän tiliin. Kaveripyyntösovelluksissa ei tarvinnut toteuttaa muuta toiminnallisuutta kuin onnistunut pyynnön lähetys.

3.2 Työn eteneminen

Työ alkoi aloituspalaverilla, jossa kirjoitettiin lähtötietomuistio yhdessä yrityksen edustajan sekä valvojana toimineen opettajan kanssa. Varsinainen työn tekninen osuus alkoi projektisuunnitelman kirjoittamisella, jossa jaoin työn kolmeen osaan eli sprinttiin. Työn alku oli suunnittelua sekä aiheeseen tutustumista ja kehitysympäristöjen asennusta. Työn toinen sprintti sisälsi koodin kirjoitusta ja testaamista. Kolmas sprintti pyhitettiin opinnäytetyön kirjalliselle osuudelle. Opinnäytetyön aikataulun näkee kuvasta 1.

Yritys:	6starz
Projekti:	opinnäytetyö
Projektin aikataulu:	13.6.-30.9.2011
Release 1 suunnitelma	
projektin viikot	16
tunnit per viikko	35
projektin tunnit yhteensä	400
projektin sprintit	3
sprint 1 aikataulu	viikko24-25
sprint 1 viikot	2
sprint 1 tunnit	40
sprint 1 tavoite	*projektisuunnitelman kirjoittaminen*
sprint 2 aikataulu	viikko26-33
sprint 2 viikot	8
sprint 2 tunnit	210
sprint 2 tavoite	*perehtyminen aiheeseen ja sovelluksen rungon tuottaminen*
sprint 3 aikataulu	viikko34-39
sprint 3 viikot	6
sprint 3 tunnit	150
sprint 3 tavoite	*sovelluksen lopullisen version tuottaminen, ja loppuraportin kirjoittaminen*
release 1 tavoite	*sovellus testattu ja hyväksytty, loppuraportti kirjoitettu*

KUVA 1. Opinnäytetyön aikataulu

Opinnäytetyön teknisessä osassa oli tarkoitus toteuttaa alunperin vain yksi sovellus, kohteiden paikannussovellus. Kuitenkin työn toteutusvaiheeseen jäi aikaa, koska paikannussovelluksen palvelintoiminnallisuudet viivästyivät ja lopulta jäädytettiin. Tämä jäljelle jäänyt aika päätettiin käyttää kaveripyyntöjen toteuttamiseen.

Kehitysympäristönä työssä toimi Ubuntu Linux 10.10 -PC, jossa oli asennettuna Eclipse 3.6.2 sekä Android SDK r11. Kohdelaitteena käytettiin ZTE Blade -mobiililaitetta, johon oli asennettu Android-käyttöjärjestelmän versio 2.2. Projektissa tuotetun sovelluksen tuli kuitenkin toimia myös vanhemmalla Android 2.1 -versiolla.

Opinnäytetyön aikana pidettiin palavereita tasaisin väliajoin yhdessä yrityksen IT-arkkitehdin kanssa. Palavereissa tarkasteltiin työn etenemistä ja määriteltiin uusia tehtäviä. Palavereiden keskeisimmät asiat dokumentoitiin ja lisättiin opinnäytetyötä varten perustetulle nettisivulle, josta ohjaava opettaja pystyi ne näkemään. Opinnäytetyön tekninen osuus päätettiin loppupalaveriin, jossa esiteltiin työn tulokset ohjaavalle opettajalle sekä yrityksen edustajalle.

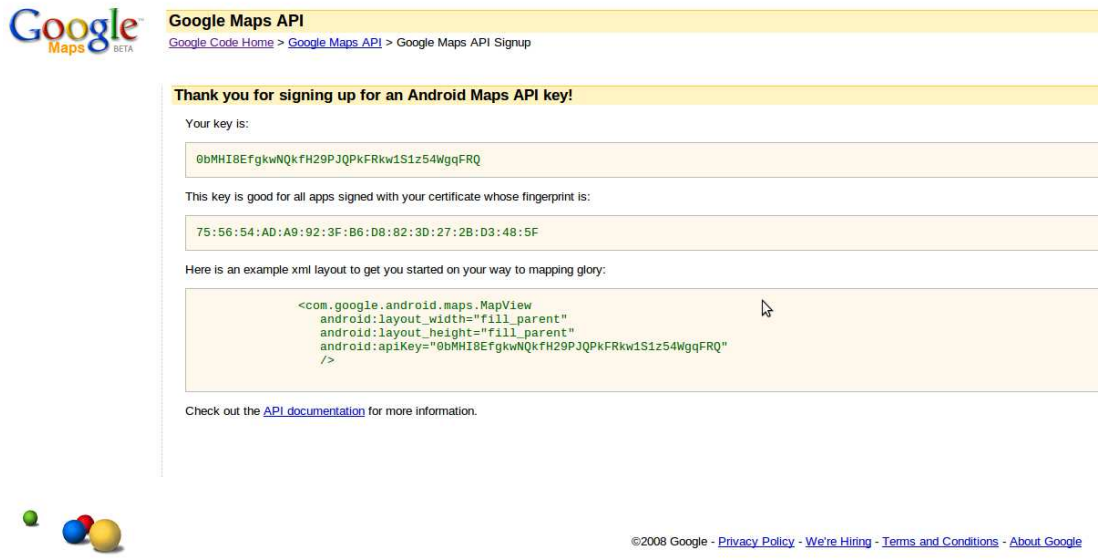
3.3 Google Maps- ja Places API -avaimet

Työvaiheen alussa luotiin uusi Google Maps API-avain, jotta Maps-karttanäkymää voitiin käyttää kehitettävässä sovelluksessa. Google tunnistaa käytetyn avaimen sen sormenjäljen perusteella. Jokainen kääntäjä luo oman sormenjälkensä, joten toisen kääntämää sovellusta voi käyttää vain sille tarkoitetulla avaimella. Avaimen luonti aloitettiin kirjoittamalla kuvassa 2 näkyvä komento PC:n käyttöjärjestelmän konsoliin, jolloin saatiin MD5-koodi.

```
jn@jn-Aspire-5742G:~/.android$ keytool -list -alias androiddebugkey -keystore de
bug.keystore -storepass android -keypass android
androiddebugkey, Jun 20, 2011, PrivateKeyEntry,
Certificate fingerprint (MD5): 75:56:54:AD:A9:92:3F:B6:D8:82:3D:27:2B:D3:48:5F
jn@jn-Aspire-5742G:~/.android$
```

KUVA 2. MD5-koodin luonti

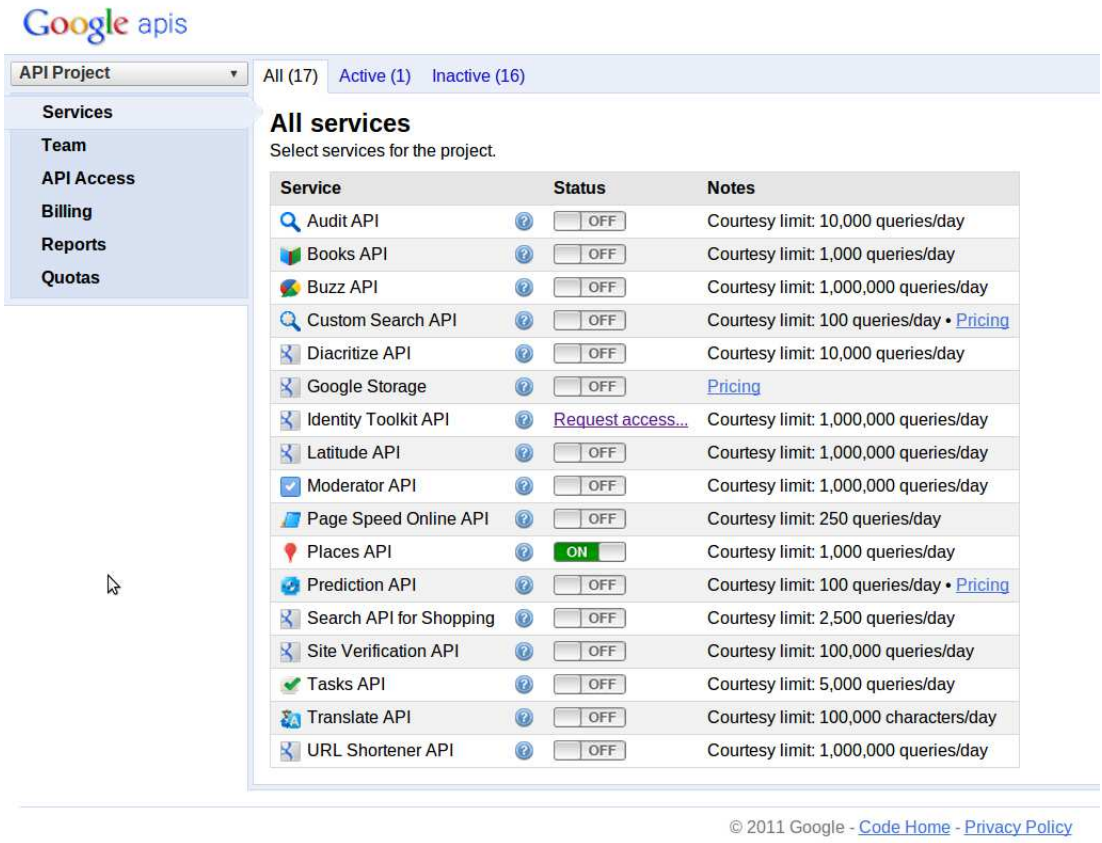
Saatua koodia käytettiin Maps API -avaimen luontiin menemällä avaimen luontiin tarkoitetulle sivulle (http://code.google.com/intl/fi-FI/android/maps-api-signup.html), kirjautumalla sisään omalla Google-tilillä ja syöttämällä koodi sille tarkoitettuun kenttään ja hyväksymällä lisenssiehdot. Kuvassa 3 näkyy onnistunut Maps-avaimen luonti.



KUVA 3. Maps API -avain

Työssä käytettiin Google Places APIa kohteiden etsimiseen, joten sillekin täytyi luoda oma API-avain. Places-avaimen luonti oli hieman erilainen, sillä siinä ei tarvittu erillistä hash-koodia, vaan jokaiselle Places-avaimelle luodaan oma projekti Google API -konsolin kautta

osoitteessa https://code.google.com/apis/console. API-konsoliin kirjaudutaan omalla Google mail -tunnuksella. Haluttu avain valitaan sieltä ja siellä näkee myös tietoja jäljellä olevasta kyselymäärästä ja muista tiedoista. Kuvassa 4 näkyy Google API -konsoli.



KUVA 4. Google API -konsoli

3.4 Facebook API -avain

Facebook APIn käyttöön tarvitaan Application ID- ja Application secret -koodit. Ne saadaan osoitteesta https://developers.facebook.com/apps. Osoitteeseen kirjaudutaan omilla Facebook-tunnuksilla, ja tämän jälkeen valitaan ”+ Luo uusi sovellus”. Sovelluksen luomiseen tarvitaan tunnistettu Facebook-tili. Tunnistuksen voi tehdä myös avainta luodessa matkapuhelimella tai luottokortilla.

Facebook APIn dokumentaatio ja koodiesimerkit ovat avointa lähdekoodia, ja ne on lisensoitu Creative Commons Attribution-ShareAlike 3.0 -lisenssillä. Yksittäiset koodiesimerkit on lisensoitu vielä erikseen Apache-lisenssin versiolla 2.0. (Documentation and Code Licensing. 2011.)

3.5 LinkedIn API -avain

LinkedIn APIn käyttöön tarvittavat avain ja secret-koodi luodaan osoitteessa https://developer.linkedin.com/. Osoitteessa kirjaudutaan omilla LinkedIn-tunnuksilla klikkaamalla ”Sign in” ja tämän jälkeen valitaan samasta kohtaan ”API Keys”. Tämän jälkeen valitaan ”+ Add New Application” jolloin sovelluksen rekisteröintiin tarkoitettu kaavake avautuu. Kaavakkeessa kysytään mm. yritys, sovelluksen nimi ja kuvaus, sovelluksen tyyppi, käyttötarkoitus sekä kehittäjän sähköpostiosoite ja puhelinnumero. Tämän jälkeen hyväksytään lisenssiehdot ja klikataan ”Done”. Seuraavaksi avautuu uusi sivu jossa näkyy APIn käyttöön tarvittavat ”API key” ja ”Secret Key”.

3.6 Twitter API -avain

Twitter-avain luodaan osoitteessa https://dev.twitter.com/. Osoitteessa kirjaudutaan sisään Twitter-tunnuksilla valitsemalla ”Sign in”. Tämän jälkeen valitaan ”My Applications” ja ”Create a new application”. Tämän jälkeen täytetään kaavake, jossa kysytään sovelluksen nimeä, kuvausta, nettisivua ja autentikointia varten tarvittavaa Callback url -osoitetta. Callback-osoitteen voi jättää tyhjäksi, jos sovelluksen autentikointivaiheessa saatavat request-token- ja token-secret-koodit käsitellään suoraan niitä pyytäneessä laitteessa. Seuraavaksi hyväksytään lisenssiehdot ja valitaan ”Create your Twitter application”. APin käyttöön tarvittavat avaimet ”Consumer key” ja ”Consumer secret” saadaan seuraavalta sivulta kohdasta ”Oauth settings”.

APIn dokumentaatio ja koodiesimerkit ovat avointa lähdekoodia. Dokumentaatio on lisensoitu pääasiassa Creative Commons Attribution-ShareAlike 3.0 -lisenssillä. Yksittäiset koodiesimerkit on lisensoitu vielä erikseen Apache-lisenssin versiolla 2.0. (Code and Documentation Licensing.)

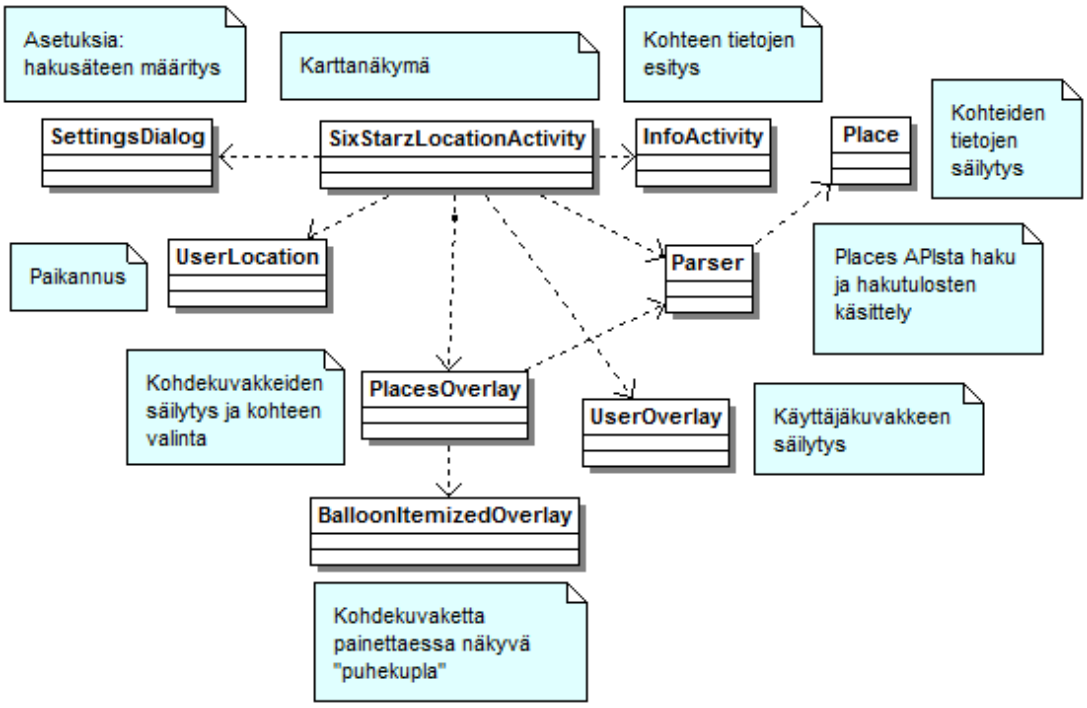
4 SOVELLUKSIEN RAKENNE JA TOIMINTA

Esittelen tässä luvussa opinnäytetyössä kehitettyjä sovelluksia ja niiden rakennetta sekä toimintaa. Käyn läpi pääkohdat koodeista, mutta en käsittele suuremmin käyttöliittymää, sillä se ei ollut tärkeässä osassa tässä työssä.

4.1 Paikannussovellus

Sovellus on jaettu luokkiin, joilla on omat tehtävänsä. SixstarzLocationActivity-luokassa näytetään ja käsitellään mapview-karttanäkymää. UserLocation-luokassa hoidetaan paikannukseen liittyvät asiat. PlacesOverlay-luokassa säilytetään karttanäkymässä näkyvät, kohteita esittävät kuvakkeet. UserOverlay hoitaa käyttäjän sijainnin näyttämisen kartalla. Parser hoitaa tiedonhaun Google Places APIsta. Place-luokassa säilytetään kohteiden tiedot. InfoActivityssä näytetään tietoja yksittäisestä kohteesta. SettingsDialogissa suoritetaan asetusten muuttaminen.

PlacesOverlay-luokka on periytetty BalloonItemizedOverlay-luokasta. BalloonItemizedOverlay-luokka on ReadyState Software Ltd:n kehittämä luokka, jolla karttakohteisiin saadaan painettaessa avautuva, puhekuplalta näyttävä taso, johon voidaan lisätä tekstiä. (Android Mapview Balloons. 2011.) Kun tätä avautunutta tasoa painetaan, saadaan uusi tapahtumakäsittelijä, jota voidaan käyttää esimerkiksi uuden Activityn avaamiseen. Luokka lisää siis yhden tason lisää verrattuna normaalisti käytettävään Google Maps -kirjaston com.google.android.maps.itemizedoverlay<item>-luokkaan. Kuvassa 5 näkyy paikannussovelluksen rakenne.



KUVA 5. Paikannusovelluksen luokat

Sovelluksen toiminnallisuudet näkyvät viestiyhteyskaavioista 1–2 (liitteet 1–2). Viestiyhteyskaaviossa 1 esitetään kohteiden etsintä Places APIsta ja karttanäkymään piirto. Yksittäisen kohteen tietojen haku ja esitys tapahtuvat viestiyhteyskaaviossa 2 esitetyllä tavalla.

4.1.1 Karttanäkymä

Google Maps -kirjasto ja tarvittavat tiedonsiirto oikeudet lisätään karttanäkymää käyttävän sovelluksen AndroidManifest.xml-tiedostoon näin:

```
<uses-library android:name="com.google.android.maps" />
<uses-permission android:name="android.permission.INTERNET"/>
```

Maps-karttanäkymää käyttävän luokan layout-tiedostoon lisätään com.google.android.maps.mapview-elementti ja siihen Maps API -avain.

```
<com.google.android.maps.MapView
android:id="@+id/map_view"
android:clickable="true"
```

```
android:enabled="true"
android:apiKey="abcd12245..." />
```

Karttanäkymää käyttävä luokka periytetään MapActivity-luokasta. Karttanäkymä ja siihen mahdollisesti lisätyt tasot (kartan päällä näytettävät kuvat) ladataan luokan OnCreate()-metodissa. Kartan zoomauspainikkeet voidaan myös lisätä tässä kohtaan.

```
public class SixStarzLocationActivity extends MapActivity
private MapView mapView;
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.main);
mapView = (MapView) findViewById(R.id.map_view);
mMapOverlays = mapView.getOverlays();
mapView.setBuiltInZoomControls(true);
mapView.displayZoomControls(true);}
```

4.1.2 Kohteiden haku Places APIsta

Kohteiden haku Google Places APIsta suoritetaan tekemällä HTTP-pyyntö Places Search -osoitteeseen, jossa ilmoitetaan API-avaimen lisäksi hakupisteen koordinaatit, hakusäde sekä haluttujen kohteiden tyyppi. Haluttu tietojen palautusmuoto on JSON. Kohteet haetaan listoihin yksi kerrallaan. Kohteiden haku suoritetaan seuraavasti:

```
private static final String PLACES_SEARCH_URL =
"https://maps.googleapis.com/maps/api/place/search/json?";
private String type1 = "bar"; private String type2 = "cafe"; private
String type3 = "night_club"; private String type4 = "restaurant";
HttpRequestFactory httpRequestFactory = createRequestFacto-
ry(transport);

HttpRequest request = httpRequestFactory.buildGetRequest(new Generi-
cUrl(PLACES_SEARCH_URL));
request.url.put("key", API_KEY);
request.url.put("location", latitude + "," + longitude);
```

```
request.url.put("radius", radius);
request.url.put("types", type1 + "|" + type2 +"|" + type3 + "|" +
type4);
request.url.put("sensor", "true");
HttpResponse response = request.execute();
String parse = response.parseAsString();
jObject = new JSONObject(parse);
JSONArray resultsArray = jObject.getJSONArray("results");
pname = resultsArray.getJSONObject(i).getString("name").toString();
pref = resultsAr-
ray.getJSONObject(i).getString("reference").toString();
pid =  resultsArray.getJSONObject(i).getString("id").toString();
JSONObject geoObject =  resultsAr-
ray.getJSONObject(i).getJSONObject("geometry");
JSONObject locObject = geoObject.getJSONObject("location");
plat = locObject.getDouble("lat");
plng = locObject.getDouble("lng");
JSONArray attributions = jObject.getJSONArray("html_attributions");
```

Yksittäisen Places-kohteen tarkemmat tiedot saadaan, kun lähetetään pyyntö Places Details -osoitteeseen yhdessä API-avaimen ja kohteen viitteen kanssa. Haku suoritetaan näin:

```
private static final String PLACES_DETAILS_URL =
"https://maps.googleapis.com/maps/api/place/details/json?";
HttpRequestFactory httpRequestFactory = createRequestFacto-
ry(transport);
HttpRequest request = httpRequestFactory.buildGetRequest(new Generi-
cUrl(PLACES_DETAILS_URL));
request.url.put("key", API_KEY);
request.url.put("reference", reference);
request.url.put("sensor", "true");
HttpResponse response = request.execute();
String parse = response.parseAsString();
jObject = new JSONObject(parse);
JSONArray attributions = jObject.getJSONArray("html_attributions");
JSONObject results = jObject.getJSONObject("result");
mTargetName = results.getString("name");
mTargetTypes = results.getString("types");
mTargetPhone = results.getString("formatted_phone_number");
```

```
mTargetAddress = results.getString("formatted_address");
mTargetRating= results.getDouble("rating");
mTargetUrl = results.getString( "url");
```

4.1.3 Kohteiden lisäys karttanäkymään

Karttakohteiden piirtoon tarvitaan luokka, joka on periytetty ItemizedOverlay-luokasta. Käytin sovelluksessa BalloonItemizedOverlay-luokkaa, joka toimii muuten samoin kuin tavallinen ItemizedOverlay-luokka, mutta lisää luokkaan yhden tason lisää, jota käytin kohteen nimen ja tyypin näyttämiseen kohdetta painettaessa. Tämä ominaisuus voitaisiin toteuttaa myös dialogia käyttäen, mutta se huonontaisi sovelluksen käytettävyyttä.

Periytetty luokka sisältää listan, johon overlayitemit säilötään. Luokka sisältää myös toisen listan jota käytetään kohteiden tietojen säilömiseen. Yksittäinen overlayitem sisältää Geopoint-koordinaatit, nimen ja tyypin.

Luokan onBalloonTap-metodi käynnistyy, kun kohdetta painetaan toisen kerran. Metodi käynnistää kohteen tarkempien tietojen haun Places APIsta ja niiden näyttämisen InfoActivityssä. Tiedot välitetään InfoActivityyn startActivity(Intent)-metodilla ja käyttäen Bundle-objektia, joka säilöö merkki-arvoparit.

```
public class PlacesOverlay extends BalloonItemizedO-
verlay<OverlayItem>{
private ArrayList<OverlayItem> mOverlays;
public PlacesOverlay(Drawable defaultMarker, MapView mapView) {
super(boundCenter(defaultMarker), mapView);
        c = mapView.getContext();
        mOverlays = new ArrayList<OverlayItem>();
        places = new ArrayList<Place>();}
public void addOverlay(OverlayItem overlay) {
mOverlays.add(overlay);
populate();}
@Override
protected OverlayItem createItem(int i) {
return mOverlays.get(i);}
```

```
@Override
public int size() {
return mOverlays.size();}
public void addToList(Place place) {
places.add(place); }}
@Override
protected boolean onBalloonTap(int index, OverlayItem item)
{       mIndex = index;
        reference = places.get(mIndex).getReference();
        parser.performDetails(reference);
        Intent myIntent = new Intent(c, InfoActivity.class);
        Bundle bundle = new Bundle();
        mName = parser.mTargetName;
        ...
        bundle.putString("NAME",mName);
        ...
        myIntent.putExtras(bundle);
        c.startActivity(myIntent);
return true; }
```

Kohteet lisätään kartalle SixStarzLocation-luokan DrawThem-metodissa. Kohteena käytettävät kuvat lisätään projektin res/drawable-kansioon.

```
void DrawThem()
{Drawable drawable =
this.getResources().getDrawable(R.drawable.img_unregistered_place_ma
rker);Drawable drawable2 =
this.getResources().getDrawable(R.drawable.img_6starz_place_marker);
mapOverlayPlaces = new PlacesOverlay(drawable, mapView);
for (int i = 0; i < mItems2.size(); i++)
{if (mPlaces2.get(i).isRegisteredPlace() == true )     {
// tutkitaan onko kohde 6Starz-paikka vai ei, ja lisätään kohteen
kuvake sen mukaan
        OverlayItem item;
        item = mItems2.get(i);
        drawable2.setBounds(0,0, 56, 56);
        item.setMarker(drawable2);
        mItems2.set(i, item);  }
```

```
        else{   OverlayItem item;
                item = mItems2.get(i);
                drawable.setBounds(0,0, 56, 56);
                item.setMarker(drawable);
                mItems2.set(i, item); } }
// lisätään kohteet ja niiden tiedot listoihin
        for (int i = 0; i < mItems2.size(); i++)
         {      mapOverlayPlaces.addOverlay(mItems2.get(i));
                mapOverlayPlaces.addToList(mPlaces2.get(i));
         }
// lisätään overlayt mapviewiin
                mMapOverlays.add(mapOverlayPlaces);
                mapView.invalidate();  }
```

4.1.4 Paikannus

Paikannusta varten tarvitaan sovelluksen AndroidManifest.xml-tiedostoon seuraavat tiedot:

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

ACCES_COARSE_LOCATION sallii sovellukselle tukiasemapaikannuksen käytön. ACCESS_FINE_LOCATION sallii GPS-satelliittipaikannuksen.

Paikannus suoritetaan UserLocation-luokassa. Luokassa käytetään Handler-objektia viestin lähettämiseen pääluokkaan, kun sijainti muuttuu tai paikannuslaskuri nollautuu.

```
public UserLocation(Handler parentMsgHandler,Context mycontext){
                this.mycontext = mycontext;
                mParentHandler = parentMsgHandler;}
```

UserLocation-luokan StartPositioning-metodi käynnistää paikannuksen. Luokassa käytetään kahta UserLocationListener-objektia, jotta GPS-paikannus ja tukiasemapaikannus voidaan käsitellä erikseen. GPS- tai tukiasemapaikannus on käytössä sen mukaan, onko GPS-paikannus tuettu laitteessa ja onko se kytketty päälle laitteen asetuksista. Tätä asetusta tutkitaan .isProviderEnabled(LocationManager.GPS_PROVIDER)-metodilla.

```
public void StartPositioning() {
        isStarted = true;
        GPSListener = new UserLocationListener();
        BSListener = new UserLocationListener();
        GPSLocation = (LocationManager) mycon-
text.getSystemService(Context.LOCATION_SERVICE);
        BSLocation = (LocationManager) mycon-
text.getSystemService(Context.LOCATION_SERVICE);
GPSEnabled = GPSLoca-
tion.isProviderEnabled(LocationManager.GPS_PROVIDER);
if (GPSEnabled) {
        GPSLocation.requestLocationUpdates(
        LocationManager.GPS_PROVIDER, 300000,
                                100, GPSListener);
        GPSLocation.addGpsStatusListener(GPSListener);
        } else {
        BSLocation.requestLocationUpdates(
        LocationManager.NETWORK_PROVIDER, 300000,
                                100, BSListener); return;}}
```

Paikannus suoritetaan 5 minuutin välein tai jos siirrytään 100 metriä edellisestä paikasta. Paikannus lopetetaan luokan StopPositioning-metodilla.

```
public void StopPositioning() {
        isStarted = false;
if (BSListener != null) {
                BSLocation.removeUpdates(BSListener);}

        if (GPSListener != null) {
                GPSLocation.removeUpdates(GPSListener);
```

```
                GPSLocation.removeGpsStatusListener(GPSListener);}}
```

UserLocationListener-luokka perii LocationListener-luokan. Luokan onLocationChanged-metodi kutsutaan, kun siirrytään tarpeeksi edellisestä paikasta tai laskuri nollautuu. Location-muuttujasta saadaan paikan koordinaatit. Koordinaatit täytyy muuttaa GeoPoint-muuttujaksi, jos niitä halutaan käyttää kohteen esittämiseen mapviewissa. Metodista lähetetään viesti pää-luokkaan (SixStarzLocationActivity) aina, kun metodia kutsutaan.

```
public class UserLocationListener implements LocationListener  {
@Override
public void onLocationChanged(Location loc) {
if (loc != null) {
                currentLatitude  =  loc.getLatitude();
                currentLongitude =  loc.getLongitude();
                userpoint = new GeoPoint(
                    (int) (currentLatitude * 1E6),
                    (int) (currentLongitude * 1E6));
                 Message msgToMain = mParentHandler.obtainMessage();
          msgToMain.what = 1;
          mParentHandler.sendMessage(msgToMain);}}}
```

SixStarzLocationActivity-luokan onCreate-metodissa käynnistetään paikannus ja näytetään viesti käyttäjälle.

```
@Override
public void onCreate(Bundle savedInstanceState) {
        userloc = new UserLocation(mHandler,this);
        userloc.StartPositioning();
       Toast.makeText(getBaseContext(),"Positioning ON",
       Toast.LENGTH_SHORT).show();            }
```

Paikannus lopetetaan luokan onDestroy-metodissa. Metodia kutsutaan, kun sovellus sulje-taan.

```
@Override
        protected void onDestroy() {
        super.onDestroy();
        userloc.StopPositioning();
        Toast.makeText(getBaseContext(),"Positioning OFF",
        Toast.LENGTH_SHORT).show();     }
```

UserLocation-luokasta tullut viesti käynnistää getPosition-metodin.

```
Handler mHandler = new Handler()
{       @Override
        public void handleMessage(Message msg)
        {       super.handleMessage(msg);
                switch (msg.what) {
                case 1:
                        getPosition();
                        break;  }} };
```

Käyttäjän sijainnin piirto kartalle tapahtuu getPosition-metodissa. Karttanäkymässä siirrytään pisteeseen, jossa käyttäjä sijaitsee, ja tähän kohtaan lisätään käyttäjää kuvaava kuvake. Muut kuvakkeet siivotaan karttanäkymästä aina, kun sijainti vaihtuu.

```
void getPosition(){
mUserpoint = userloc.userpoint;
if(mUserpoint != null) {
        mMapController.animateTo(mUserpoint);
        Drawable drawable = this.getResources().getDrawable(
R.drawable.img_current_location_marker);
        mapOverlayUser = new UserOverlay(drawable,
mapView.getContext());
        mMapOverlays.clear();
        mapOverlayUser.addOverlay(new OverlayItem(mUserpoint, "-" +
mUserpoint.getLatitudeE6(), "-"+ mUserpoint.getLongitudeE6()));

mMapOverlays.add(mapOverlayUser);
        mapView.invalidate();
```

```
                }else {return; }}
```

4.1.5 Kohteen tietojen näyttäminen ja lähetys

Kohteen tiedot avataan Bundle-objektista InfoActivityn onCreate-metodissa seuraavalla tavalla:

```
@Override
public void onCreate(Bundle savedInstanceState) {
        bundle = getIntent().getExtras();
        name =  bundle.getString("NAME");
```

Kohteen Places-nettisivu näytetään käyttämällä Intent-objektin ACTION_VIEW-toimintoa ja antamalla sille parametrina kohteen internet-osoite.

```
Uri uri = Uri.parse(urli);
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);
```

Sähköpostiviesti lähetetään käynnistämällä dialogi ja tämän jälkeen luomalla uusi Intent-objekti, joka käyttää ACTION_SEND-toimintoa. Viestiin lisätään osoite, otsikko ja haluttu teksti käyttämällä toiminnon .putExtra-ominaisuutta.

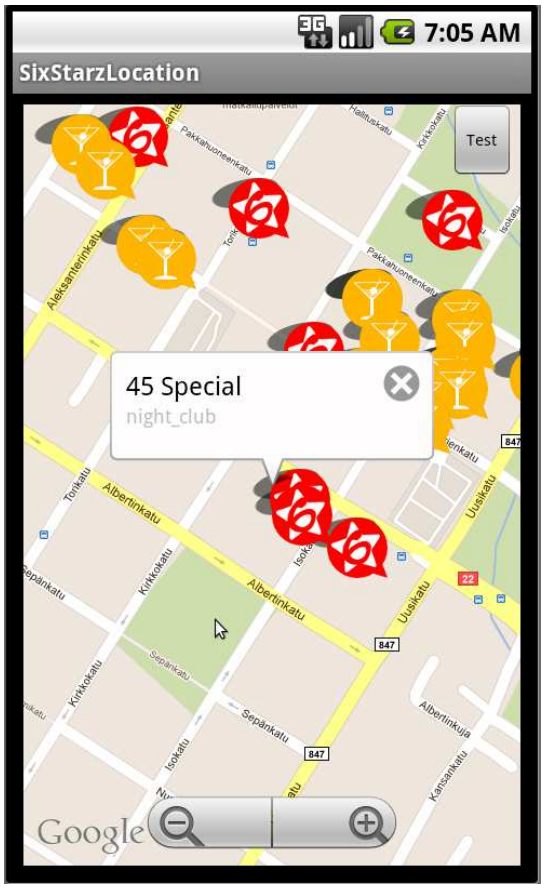
```
AlertDialog.Builder builder = new AlertDia-

log.Builder(v.getContext());
        builder.setTitle("Suggest a new 6Starz place");
        builder.setMessage("Send e-mail about a new place: " + name)
        .setCancelable(false)
        .setPositiveButton("Yes", new DialogInter-
face.OnClickListener() {
@Override
public void onClick(DialogInterface dialog, int id) {
String body = "Name: "+name ...;
```

```
final Intent emailIntent = new Intent( an-
droid.content.Intent.ACTION_SEND);
emailIntent.setType("message/rfc822"); //ehdottaa vain sähköpostioh-
jelmia
emailIntent.putExtra(android.content.Intent.EXTRA_EMAIL,
new String[] { "joku@osoite.fi" });
emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT,
"A new 6Starz place");
emailIntent.putExtra(android.content.Intent.EXTRA_TEXT,
body);
startActivity(Intent.createChooser(
emailIntent, "Send mail..."));
        }})
  .setNegativeButton("No", new DialogInterface.OnClickListener() {
@Override
public void onClick(DialogInterface dialog, int id) {
dialog.cancel();}});
AlertDialog alert = builder.create();
alert.show();
```

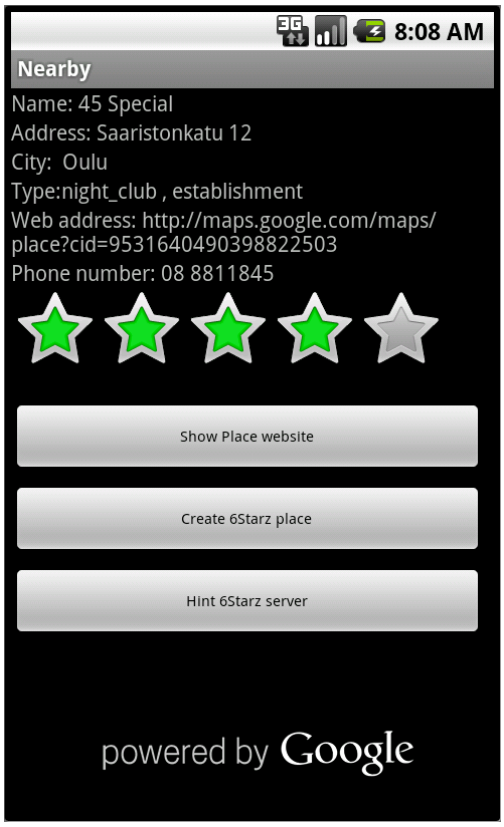
4.1.6 Paikannussovelluksen toiminta

Sovelluksen käyttöliittymä on varsin yksinkertainen. Siinä on vain karttanäkymä, joka toimii Googlen Maps -sovelluksen tavoin eli siinä on käytössä kartan vieritys sormella painamalla ja vetämällä sekä kartan zoomaus painikkeilla. Sovelluksessa on käytössä paikannus, joka paikantaa käyttäjän sijainnin ja näyttää sen karttanäkymässä kuvakkeella. Paikannus toimii joko GPS- tai tukiasemapaikannuksena sen mukaan, onko GPS käytössä vai ei. Kun paikannus on suoritettu, käyttäjän lähellä olevat Places-kohteet etsitään ja näytetään kartalla kahdella erityyppisellä kuvakkeella. Kuvakkeet arvotaan tässä versiossa sovellusta. Kuvassa 6 näkyy sovelluksen karttanäkymä ja kohteita (kuvassa näkyvät kohteet eivät ole siis 6Starz kohteita, vaikka jotkin niistä onkin merkitty 6Starz-kuvakkeella).



KUVA 6. Karttanäkymä

Kun halutaan tietoa yksittäisestä ruudulla näkyvästä kohteesta, valitaan kohde, jolloin sen nimi ja tyyppi tulee näkyviin, ja sen jälkeen painetaan esiin tulevaa valkoista laatikkoa, jolloin näkyviin tulee uusi ruutu, jossa tiedot esitetään. Näytettävät tiedot ovat kohteen nimi, osoite, puhelinnumero, kohteen arviointi (0–5), tyyppi sekä kohteen Google Places -nettisivu, josta löytyy lisätietoa kohteesta. Nettisivu voidaan näyttää painamalla sille tarkoitettua nappia, jolloin Android-käyttöjärjestelmän oma internet-selain käynnistyy ja aukaisee sivun. Kuvassa 7 näkyy yksittäisestä kohteesta saatavia tietoja.

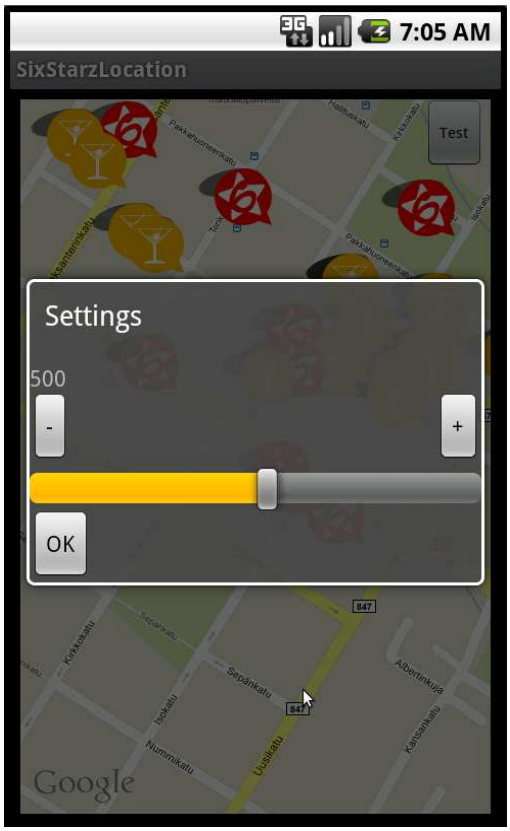


KUVA 7. Kohteen tietoja

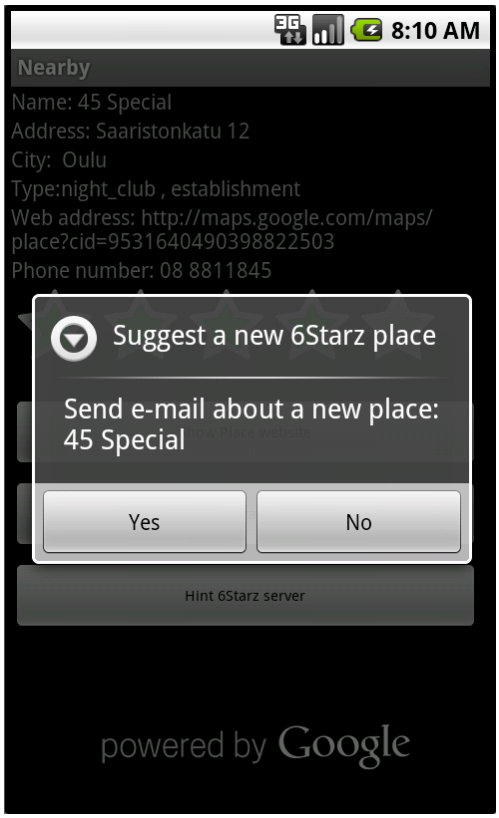
Sovelluksella voidaan ehdottaa uusia 6Starz-kohteita valitsemalla ”Hint 6Starz Server”, jolloin uusi dialogi avautuu ja kysyy käyttäjältä, haluaako hän lähettää sähköpostiviestin, jossa ilmoitetaan uudesta mahdollisesta kohteesta. Jos käyttäjä suostuu, käyttöjärjestelmän oma sähköpostisovellus tai joku käyttäjän itse asentama sähköpostisovellus avautuu ja viesti näkyy siinä. Sähköpostiviesti sisältää valmiiksi kaiken tarvittavan tiedon, mutta käyttäjän täytyy itse lähettää se. Kuvassa 8 näytetään kohteen Places-nettisivu.

KUVA 8. Kohteen Places-nettisivu

Places-kohteiden etsintä perustuu käyttäjän sijaintiin ja hakusäteeseen sekä kohdetyyppeihin. Tässä sovelluksessa voidaan hakutulokseen vaikuttaa vain säteen perusteella, joten se on ainut varsinainen asetus, jota käyttäjä voi muuttaa. Säde voidaan määrittää 10 metristä 1000 metriin. Sädettä muutetaan joko liukusäädintä vetämällä tai nappeja painamalla (kuvassa 9 näytetään asetusten muuttaminen). Jos hakutuloksia on yli 20, joka on maksimi jonka palvelin voi palauttaa, silloin säde puolitetaan ja haku suoritetaan uudelleen, jolloin saadaan lisää tuloksia lähempänä käyttäjää. Kohteet luokitellaan palvelimella sen perusteella, mitkä kohteet ovat suosituimmat, ja palvelin palauttaa aina kohteita koko hakusäteen alueelta eli isolla säteellä palautetaan vain alueen suosituimmat kohteet. Tästä seuraa se, että varsinaiset käyttäjän lähellä olevat kohteet eivät kaikki näy tuloksissa, ja siksi sädettä täytyy pienentää ja haku suorittaa uudelleen, että vähemmän suositutkin kohteet saadaan näkyviin.



KUVA 9. Asetukset dialogi



KUVA 10. Ehdota paikkaa

4.2 Kaveripyynnöt

Opinnäytetyössä toteutettiin kaveripyyntö Facebook- ja LinkedIn-palveluihin sekä seuraatoiminto Twitteriin. Kaikkiin kolmeen palveluun on olemassa kattava API palvelun nettisivuilla. Näiden APIen käytöstä on kuitenkin olemassa jo valmiiksi Androidille tehtyjä kirjastoja, joita hyödynsin tässä työssä, sillä minulla ei ollut enää riittävästi työaikaa jäljellä APIen tarkempaan tutkimiseen ja kokeiluun.

Kaikissa kolmessa sovelluksessa käytetään OAuth-autentikointia, jossa käyttäjä kirjautuu soveluksessa omilla tunnuksilla ja tämän jälkeen sovellus saa käyttöoikeudet APIin ja käyttäjän henkilökohtaiseen tiliin kyseisessä palvelussa.

Facebook käyttää uudempaa OAuth 2.0 -protokollaa, Twitter ja LinkedIn vanhempaa OAuth 1.0a -versiota. OAuth 1.0a versiossa API-asiakas ja palvelin jakavat access-tokenin ja token-secretin, jotka ovat kuin käyttäjänimi ja salasana. Asiakkaan täytyy luoda allekirjoitus (signature) joka API pyynnöllä käyttäen token-secretiä. Palvelin luo saman allekirjoituksen ja sallii yhteyden, jos molemmat allekirjoitukset ovat yhtenevät. OAuth 2.0 yksinkertaistaa tätä prosessia, sillä kaikkeen tokenin luomisessa tehtävään tiedonsiirtoon käytetään SSL:ää. Täten allekirjoituksia ei enää tarvita. API-pyynnöissä ei myöskään tarvita allekirjoituksia, kun token on luotu. (Brail 2010.)

Kaveripyynnöt eri palveluihin syntyivät melko helposti, sillä käyttämäni kirjastot tarjosivat valmiiksi palveluiden tärkeimmät ominaisuudet eikä niitä tarvinnut itse toteuttaa. Suurimmat ongelmat olivat Twitter- ja LinkedIn APIen OAuth-autentikoinnin callback-osiossa, sillä ne käyttivät esimerkeissään OOB (Out-Of-Band Authentication) -tapaa, eli käyttäjä kirjautuu sovelluksessa ja saa PIN-koodin, jonka hän sitten syöttää sovellukseen saadakseen access-token-koodin, jolla saa pääsyn APIin. Facebook Android Sdk sisälsi myös joitakin koodivirheitä, jotka täytyi korjata.

4.2.1 Facebook-sovellus

Facebook-sovelluksessa käytin Facebook APIn omaa Facebook Android Sdk:ta, joka on avointa lähdekoodia ja lisensoitu Apache-lisenssin versiolla 2.0 (Facebook-android-sdk. 2011). Facebook Android Sdk:ssa on valmiiksi dialogeja joilla voidaan tehdä erilaisia toimin-

toja kuten status-päivityksen tekeminen ja kaveripyyntö. Kaveripyynnössä täytyy ilmoittaa kohdetilin ID-numero tai käyttäjänimi.

Facebook APIn käyttöön tarvitaan Application-ID, jolla tunnistetaan sovellus (sama asia kuin muiden palveluiden API-avain) sekä sovelluksen tarvitsemat Facebook-tilin käyttöoikeudet.

```
public static final String APP_ID = "";
    private static final String[] PERMISSIONS =
    new String[] {"publish_stream", "read_stream", "offline_access",
"read_requests"};
```

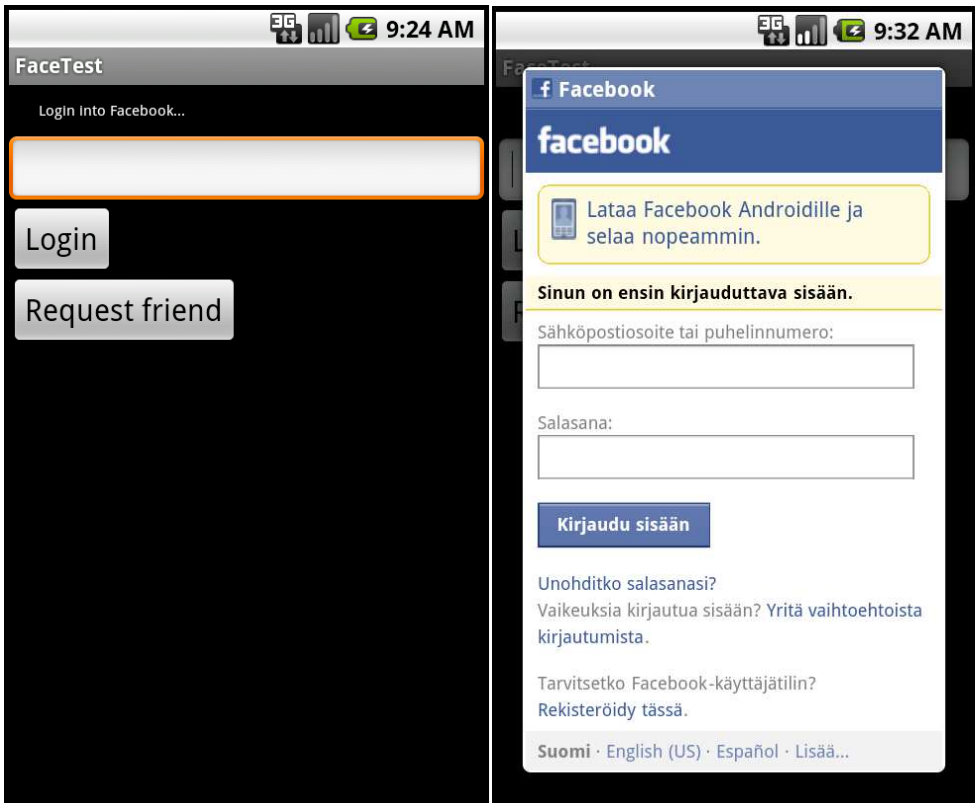
Kirjautuminen suoritetaan seuraavalla koodilla:

```
mFacebook.authorize(this, PERMISSIONS, new LoginDialogListener());
private final class LoginDialogListener implements
com.facebook.android.Facebook.DialogListener {
public void onComplete(Bundle values) {
    // Palvelimen vastaus  }}
```

Kaveripyyntö tehdään näin:

```
params.putString("id", id);
mFacebook.dialog(FaceTestActivity.this, "friends",params, new Re-
questFriendDialogListener());
public class RequestFriendDialogListener implements
com.facebook.android.Facebook.DialogListener {
public void onComplete(Bundle values) {
final String action = values.getString("action");
        if (action.startsWith("1")) { /*Pyyntö onnistui*/ }
        else { /*Tuli virhe tai käyttäjä peruutti pyynnön*/   }}}
```

Facebook-sovelluksen käyttöliittymässä on EditText-komponentti, johon kirjoitetaan kaveripyynnön kohteen ID-numero tai alias-nimi. Login-napista sovelluksen käyttäjä kirjautuu Facebookiin tunnuksillaan. Reguest friend -napista lähetetään kaveripyyntö. Kuvassa 11 näkyy sovelluksen käyttöliittymä ja kirjautuminen ja kuvassa 12 kaveripyyntö.



KUVA 11. Facebook-sovelluksen kirjautumisnäkymät

KUVA 12. Kaveripyyntönäkymä

4.2.2 LinkedIn-sovellus

LinkedIn-sovelluksessa käytin Code.google.com-yhteisön kehittämää Linkedin-j-javakirjastoa. Linkedin-j on avointa lähdekoodia ja lisensoitu Apache-lisenssillä, versiolla 2.0. (Linkedin-j. 2011.) Kaveripyyntöön tarvitaan kohdekäyttäjän nimi sekä sähköpostiosoite. Sovelluksen autentikointi suoritetaan seuraavasti:

```
//luodaan Oauth palvelu, tarvitaan API-avaimet
final LinkedInOAuthService oAuthService = LinkedInOAuthServiceFacto-
ry.getInstance().createLinkedInOAuthService(API_key , API_secret );
//määritetään callback-osoite josta saadaan request token, sama
osoite kuin intent-filtterissä
LinkedInRequestToken requesttoken =
oAuthService.getOAuthRequestToken("callback://testi");
//määritettän uusi intentti, joka avaa selaimen ja siirtyy autenti-
kointi-osoitteeseen
intent = new Intent(Intent.ACTION_VIEW,
Uri.parse(requesttoken.getAuthorizationUrl()));
//käynnistetään intentti
startActivity(intent);
```

Callback-osoitteena voidaan käyttää oikeaa url-osoitetta erillisellä palvelimella, jossa sitten hoidetaan acces-tokenin tarkistus ja säilöminen tai lisätään intent-filter sovelluksen Android-Manifest.xml-tiedostoon, jolloin tiedot selaimesta saadaan suoraan sovellukseen.

```
 <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
       <category android:name="android.intent.category.BROWSABLE"/>
       <data android:host="testi" android:scheme="callback"></data>
</intent-filter>
```

Kun käyttäjä on kirjautunut LinkedIn-palveluun, selain uudelleenohjataan callback-osoitteeseen access-token- ja verifier-koodien kanssa. Nämä tarvitaan API-käskyjen käyttöä varten. Uudelleenohjauksen jälkeen palataan takaisin pää-Activityyn ja tarkistetaan, onnistuiko autentikointi, sekä tallennetaan access-token.

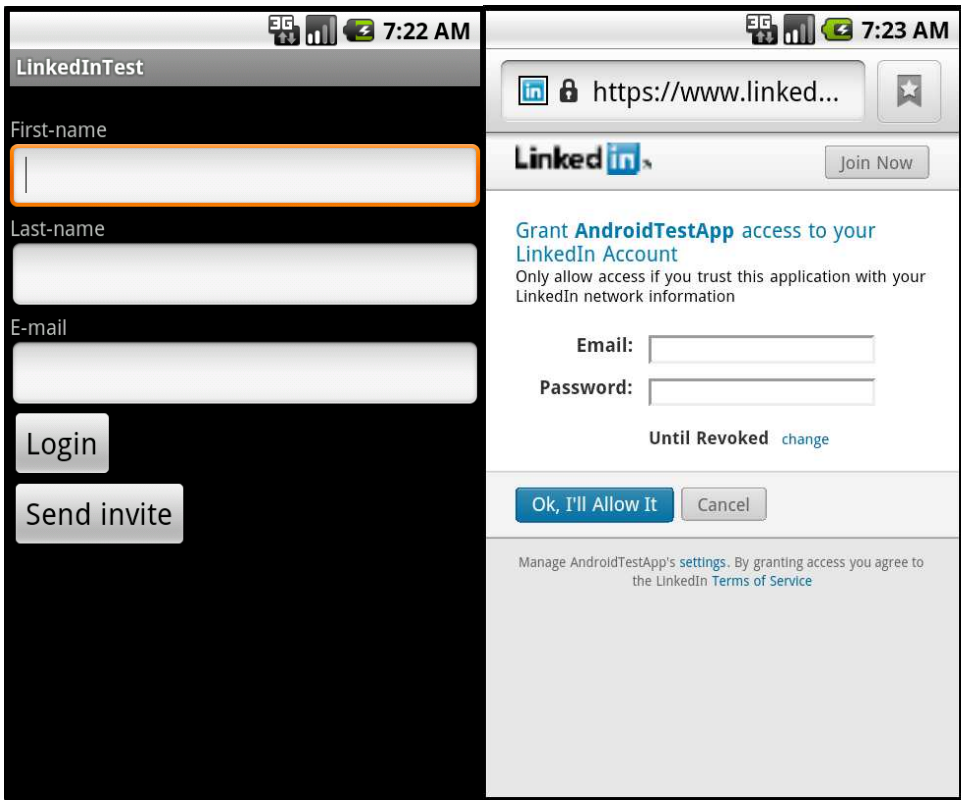
```
@Override
public void onNewIntent(Intent intent) {
super.onNewIntent(intent);   // tämä metodi käynnistyy kun selain uudelleenohjataan
Uri uri = intent.getData();    //napataan selaimen osoiterivin tiedot
// jos uri ei ole tyhjä ja scheme täsmää
if (uri != null && uri.getScheme().equals("callback")) {
String verifier = uri.getQueryParameter("oauth_verifier");
//napataan parametri kohdasta “oauth_verifier”
//saadaan access token
LinkedInAccessToken accessToken = oAuthService.getOAuthAccessToken(requesttoken, verifier);  } }
```

Kaveripyyntö Linkedin-j-kirjastoa käyttäen suoritetaan näin:

```
final LinkedInApiClientFactory factory = LinkedInApiClientFactory.newInstance(API_key , API_secret);
 LinkedInApiClient client = factory.createLinkedInApiClient(accessToken);
client.sendInviteByEmail(email, fname, lname, "Test invite, ignore this if not expected!", "This is a test.");
```

Sovellus lähettää sähköpostiviestin kohdehenkilölle, joka näkyy hänen LinkedIn-profiilissaan sekä sähköpostissa.

LinkedIn-sovelluksessa käyttöliittymä muodostuu EditText-komponenteista, joihin syötetään kohdehenkilön etunimi, sukunimi ja sähköpostiosoite, sekä napeista ”Login” ja ”Send invite”, joilla kirjaudutaan käyttäjän LinkedIn-tiliin ja lähetetään kaveripyyntö. Käyttöliittymä ja kirjautuminen näkyvät kuvassa 13.



KUVA 13. LinkedIn-sovelluksen kirjautumisnäkymä

4.2.3 Twitter-sovellus

Twitter-sovelluksessa käytin Winterwell Associates Ltd:n kehittämää JTwitter-javakirjastoa. JTwitter on avointa lähdekoodia ja lisensoitu LGPL-lisenssillä. (JTwitter - the Java library for the Twitter API. 2008.) Twitterissä seuraa-pyyntö kohdistetaan käyttäjän nimimerkkiin. JTwitter-kirjaston kanssa käytetään OAuth Signpost -kirjastoa, joka on Code.google.com-yhteisön kehittämä javakirjasto ja jonka tarkoituksena on helpottaa OAuth-autentikoinnin suorittamista. (Oauth-signpost. 2011.) OAuth Signpost -kirjasto on avointa lähdekoodia ja lisensoitu Apache-lisenssin versiolla 2.0.

Twitter-sovelluksessa autentikointi suoritetaan seuraavasti:

```
//määritetään callback-osoite
private static final String OAUTH_CALLBACK_SCHEME = "testi";
private static final String OAUTH_CALLBACK_URL =
OAUTH_CALLBACK_SCHEME + "://callback";
```

```
private OAuthSignpostClient oauthClient; private OAuthConsumer mCon-
sumer; private OAuthProvider mProvider;
```

```
mConsumer = new CommonsHttpOAuthConsumer(Consumer_key, Consum-
er_secret);

//määritettän autentikointiin tarvittavat osoitteet

mProvider = new DefaultOAuthProvi-
der("http://api.twitter.com/oauth/request_token",
"http://api.twitter.com/oauth/access_token",
"http://api.twitter.com/oauth/authorize");

// luodaan uusi intentti, joka avaa selaimen ja osoitteen jossa au-
tentikointi suoritetaan

String authUrl = mProvider.retrieveRequestToken(mConsumer,
OAUTH_CALLBACK_URL);

Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(authUrl));

startActivity(intent);
```

AndroidManifest.xml-tiedostoon lisätään intent-filter, jossa määritetään callback-osoite.

```
<intent-filter>
     <action  android:name="android.intent.action.VIEW" />
     <category android:name="android.intent.category.DEFAULT" />
     <category android:name="android.intent.category.BROWSABLE" />
     <data  android:scheme="testi" android:host="callback" />
</intent-filter>
```

Twitterissä access-token ja token-secret saadaan näin:

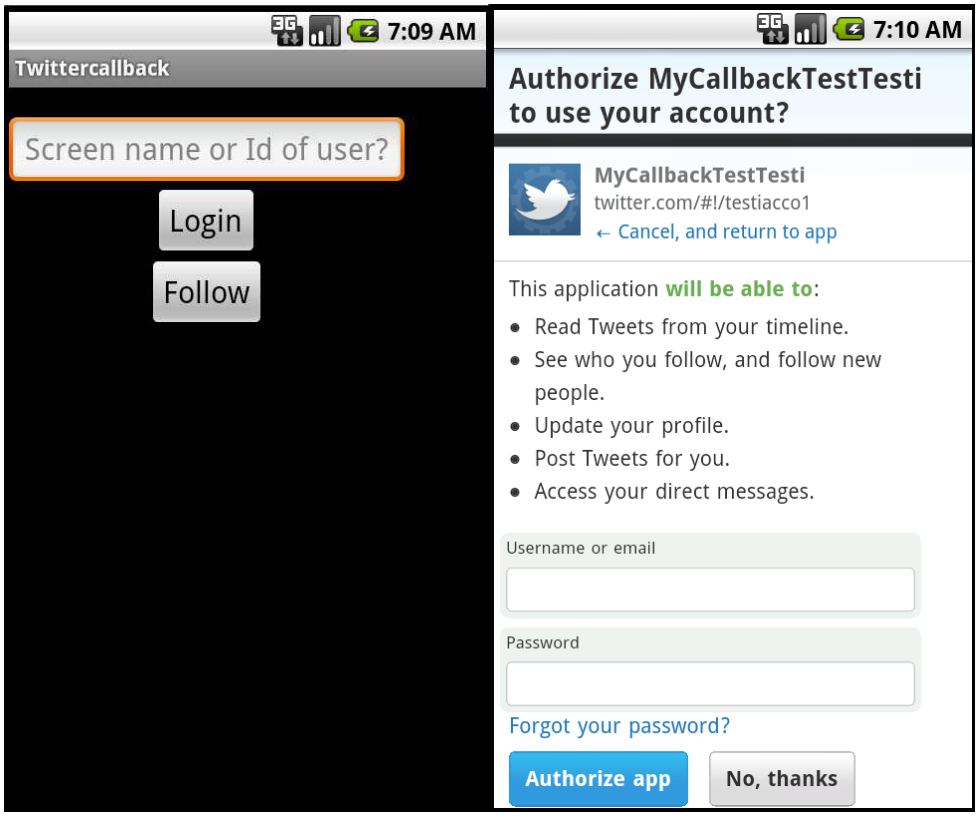
```
@Override
public void onNewIntent(Intent intent) {
 super.onNewIntent(intent);
 Uri uri = intent.getData();
 if (uri != null && uri.getScheme().equals(OAUTH_CALLBACK_SCHEME)) {
    String verifier = uri.getQueryParameter(OAuth.OAUTH_VERIFIER);
    mProvider.retrieveAccessToken(mConsumer, verifier);
    String token = mConsumer.getToken();
    String tokenSecret = mConsumer.getTokenSecret();
```

```
    mConsumer.setTokenWithSecret(token, tokenSecret);
    oauthClient = new OAuthSignpostClient(Consumer_key, Consum-
er_secret, token, tokenSecret);  }}
```

Seuraa-pyyntö suoritetaan näin:

```
twitter = new Twitter("", oauthClient);
twitter.follow(“Screen-name”);
```

Twitter-sovelluksen käyttöliittymässä on EditText-komponentti, johon syötetään kohdehenkilön Twitter-käyttäjänimi, sekä napit, joilla käyttäjä kirjautuu tiliinsä ja lähettää seuraa-pyynnön. Kuvassa 14 näkyy Twitter-sovelluksen käyttöliittymä ja kirjautuminen.



KUVA 14. Twitter-sovelluksen kirjautumisnäkymä

5 POHDINTA

Opinnäytetyön teknisessä osassa oli tarkoitus luoda paikannussovellus, jolla pystyi etsimään käyttäjän lähellä olevia kohteita ja näyttämään niistä tietoa käyttäjälle sekä lähettämään tiedot erilliselle palvelimelle. Tiedot kohteista piti hakea Google Places APIa käyttäen ja näyttää ne Google Maps -karttanäkymässä. Sovelluksen kohteena oli mobiililaite ja alustana Android. Lopulta en kuitenkaan pystynyt toteuttamaan tiedonsiirtoa palvelimelle, sillä en saanut palvelinta käyttööni määräajassa. Tiedon lähetys toteutettiin sen sijaan sähköpostiviestiä käyttäen. Paikannussovellus toimii muilta osin hyvin ja ainakin osia siitä voidaan käyttää osana yrityksen 6Starz-palvelua. Paikannussovellus on myös melko helposti päivitettävissä käyttämään jotakin toista APIa tiedon hakuun, jos sille tulee tarvetta.

Opinnäytetyön teknisen osan lopussa luotiin myös yksinkertaiset kaveripyynnöt Facebookiin, LinkedIniin ja Twitteriin. Niiden tarkoituksena oli vain osoittaa, että toiminnot voidaan suorittaa onnistuneesti suoraan mobiililaitteesta. Tämä tavoite täyttyi, ja jokaisella kaveripyyntösovelluksella voidaan suorittaa onnistunut pyyntö.

Opinnäytetyön tuloksena syntyneissä sovelluksissa on vielä jatkokehitettävää. Paikannussovellukseen täytyisi toteuttaa kohteiden lisäys palvelimelle. Siihen täytyisi toteuttaa myös jo siellä olevien kohteiden haku, jotta karttanäkymään piirrettävät kuvakkeet voidaan näyttää oikein, kun uusia kohteita haetaan. Kaveripyynnöt eri palveluihin eivät sovellu sellaisenaan kovin hyvin käytettäväksi osana 6Starz-palvelua. Niiden APIen käsittely tulisi toteuttaa itse eikä käyttää valmiita kirjastoja. Näin sovellusten kääntäminen esimerkiksi Qt:lle olisi helpompaa. Kaveripyynnöissä esiintyvä kirjautuminen tulisi myös kiertää jotenkin, jotta pyynnöistä saadaan mahdollisimman yksinkertaiset ja huomaamattomat. Nykyisessä muodossaan ne vaativat käyttäjän kirjautumista erikseen joka palveluun kaveripyynnön lähettämistä varten.

Työssä käytetyt APIt ja palvelut tarjoavat monipuolisia ominaisuuksia, ja ne soveltuvat hyvin pienimuotoisiin avoimen lähdekoodin projekteihin. Suurempiin kaupallisiin projekteihin niitä on kuitenkin melko vaikea suositella, sillä esimerkiksi niiden rajoitukset ja vaatimukset saattavat muuttua hyvinkin nopeasti. Itse sain huomata tämän opinnäytetyön loppuvaiheessa, kun kirjoitin työn kirjallista osuutta, sillä Google Maps -palvelun rajoitukset olivat muuttuneet siitä,

mitä ne olivat, kun tein varsinaista teknistä osuutta. Tämä ei kuitenkaan onneksi vaikuttanut työn lopputulokseen.

Minulla ei ollut aikaisempaa kokemusta Android-sovelluskehityksestä ja tämän opinnäytetyön myötä tuli mahdollisuus kerätä tätä kokemusta. Java-kielestäkään minulla ei ollut lähtökohaisesti kokemusta kuin parin koulussa pidetyn Java-kurssin verran. Suuri osa itse työtä olikin erilaisten tutoriaalien ja esimerkkiohjelmien tutkimista. En myöskään saanut varsinaista opastusta Android-ohjelmoinnista, vaan opiskelin asiat itse. Sain kuitenkin selkeän käsityksen Android-sovelluskehityksestä sekä erilaisten APIen ja SDK:iden käytöstä.

Työn tekninen puoli oli mielenkiintoinen, mutta ei lopulta niin haastava. Sain paikannussovelluksen perustoiminnot tehtyä jo melko varhaisessa vaiheessa. Suurin osa paikannussovellukseen käytetystä ajasta kului eri toiminnallisuuksien pilkkomiseen omiin luokkiinsa, sillä halusin opetella tiedon välitystä Java-luokkien välillä ja Android-kontekstien ymmärtämistä. Opinnäytetyön kirjoittaminen oli puolestaan hyvinkin haastavaa, sillä aineistoa oli paljon ja siitä oli vaikea tehdä eheää kokonaisuutta. Olisikin ollut parempi, jos projektissa olisi toteutettu pelkästään paikannussovellus kokonaisuudessaan ja jätetty kaveripyynnöt pois.

LÄHTEET

6Starz tuo NFC:n ja yhteisöpalvelut ravintoloihin ja tapahtumiin. 2011. Saatavissa: http://www.digibusiness.fi/uploads/attachments/1304080491_6Starz-Lehdist%C3%B6tiedote-Julkaisuvapaa-19Huhti2011.pdf. Hakupäivä 18.10.2011.

Android. 2011. Saatavissa: http://fi.wikipedia.org/wiki/Android. Hakupäivä 18.10.2011.

Android MapView Balloons. 2011. Saatavissa: https://github.com/jgilfelt/android-mapviewballoons. Hakupäivä 18.10.2011.

Brail, Greg 2010. Top Differences between OAuth 1.0 and OAuth 2.0 for API Calls. Saatavissa: http://blog.apigee.com/detail/oauth_differences/. Hakupäivä 20.10.2011.

Code and Documentation Licensing. Saatavissa: https://dev.twitter.com/opensource. Hakupäivä 20.10.2011.

Documentation and Code Licensing. 2011. Saatavissa: http://developers.facebook.com/licensing/. Hakupäivä 20.10.2011.

Eclipse (IDE). 2011. Saatavissa: http://fi.wikipedia.org/wiki/Eclipse_(IDE). Hakupäivä 18.10.2011.

Facebook. 2011. Saatavissa: http://fi.wikipedia.org/wiki/Facebook. Hakupäivä 20.10.2011.

Facebook-android-sdk. 2011. Saatavissa: https://github.com/facebook/facebook-android-sdk. Hakupäivä 20.10.2011.

Frequently Asked Questions. 2011. Saatavissa: http://code.google.com/intl/fi-FI/apis/maps/documentation/premier/faq.html. Hakupäivä 20.10.2011.

Google Maps. 2011. Saatavissa: http://fi.wikipedia.org/wiki/Google_Maps. Hakupäivä 18.10.2011.

Google Maps/Google Earth APIs Terms of Service. 2011. Saatavissa: http://code.google.com/intl/fi-FI/apis/maps/terms.html. Hakupäivä 20.10.2011.

Hanke, John 2010. Introducing Google Places. Saatavissa: http://googleblog.blogspot.com/2010/04/introducing-google-places.html. Hakupäivä 18.10.2011.

Java. 2011. Saatavissa: http://fi.wikipedia.org/wiki/Java. Hakupäivä 18.10.2011.

JTwitter - the Java library for the Twitter API. 2008. Saatavissa: http://www.winterwell.com/software/jtwitter.php. Hakupäivä 20.10.2011

LinkedIn. 2011. Saatavissa: http://fi.wikipedia.org/wiki/LinkedIn. Hakupäivä 20.10.2011.

Linkedin-j. 2011. Saatavissa: http://code.google.com/p/linkedin-j/. Hakupäivä 20.10.2011.

Location and Maps. 2011. Saatavissa: http://developer.android.com/guide/topics/location/index.html. Hakupäivä 18.10.2011.

Oauth-signpost. 2011. Saatavissa: http://code.google.com/p/oauth-signpost/. Hakupäivä 20.10.2011.

Protalinski, Emil 2011. Facebook kills Places, but emphasizes location sharing more. Saatavissa: http://www.zdnet.com/blog/facebook/facebook-kills-places-but-emphasizes-location-sharing-more/2972. Hakupäivä 20.10.2011.

Sign Up for the Android Maps API. 2011. Saatavissa: http://code.google.com/intl/fi-FI/android/maps-api-signup.html. Hakupäivä 20.10.2011.

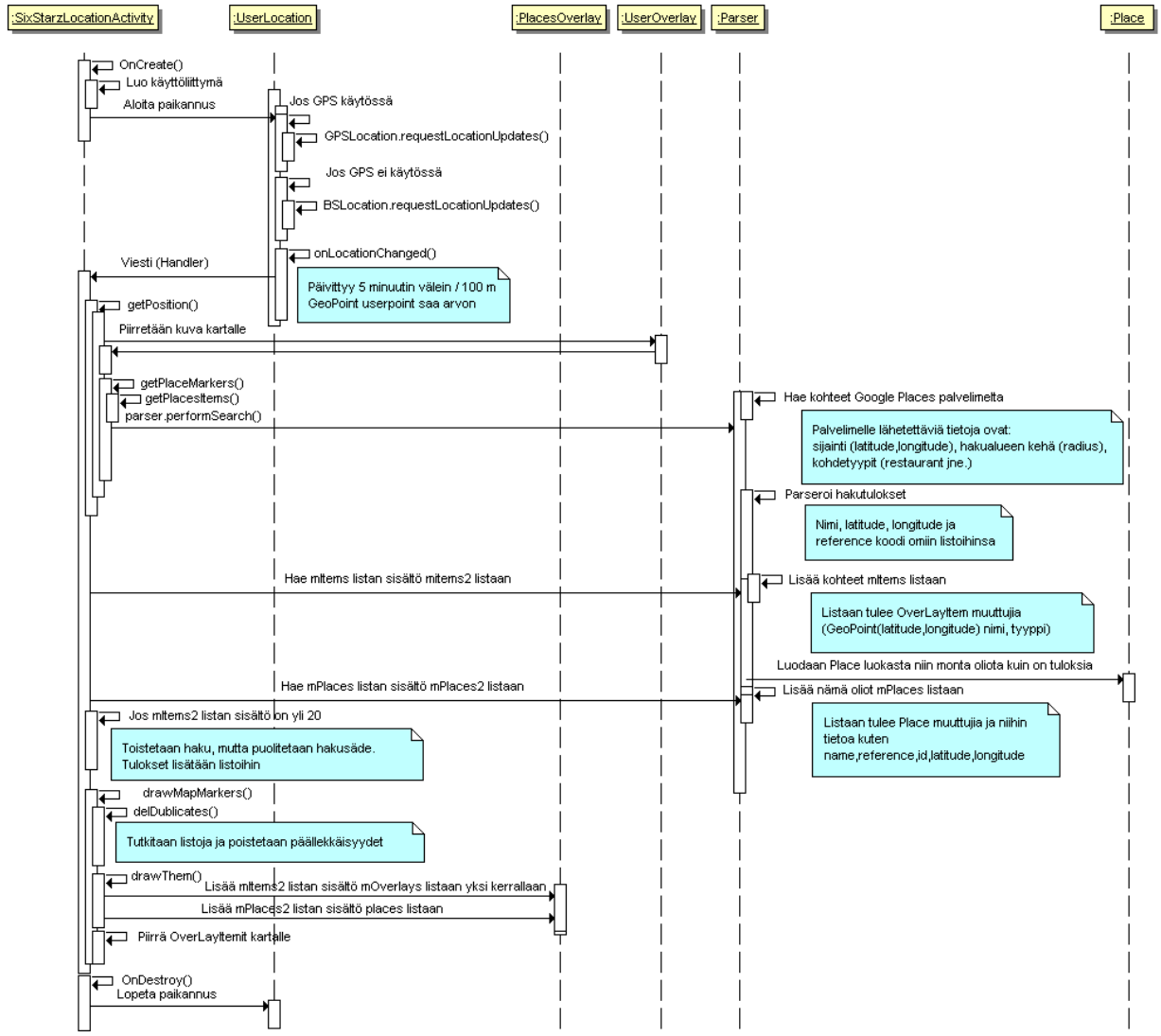
Sign Up for the Google Maps API. 2011. Saatavissa: http://code.google.com/intl/fi-FI/apis/maps/signup.html. Hakupäivä 18.10.2011.

Tehosta liiketoimintaasi uudella 6Starz Biz -palvelulla. 2011. Saatavissa: https://6starz.com/pilot/owner/content/about. 20.10.2011.

The Google Places API. 2011. Saatavissa: http://code.google.com/intl/fi-FI/apis/maps/documentation/places/index.html. Hakupäivä 18.10.2011.

Twitter. 2011. Saatavissa: http://fi.wikipedia.org/wiki/Twitter. Hakupäivä 20.10.2011.

VIESTIYHTEYSKAAVIO, NÄYTÄ KOHTEET LIITE 1



VIESTIYHTEYSKAAVIO, NÄYTÄ TIETOA KOHTEESTA LIITE 2