**Mobile Project -loppudokumentaatio**

Janne Hakala

Timo Saukonoja

Joulukuu 2016

Ohjelmistotekniikan koulutusohjelma

Tekniikan ja liikenteen ala

Sisältö

[1 Johdanto 2](#_Toc468870412)

[2 Käyttötarkoitus 2](#_Toc468870413)

[3 Use Case & User Strory 2](#_Toc468870414)

[3.1 Use Case 2](#_Toc468870415)

[3.2 User story 3](#_Toc468870416)

[4 Sovelluksen toiminta 4](#_Toc468870417)

[5 REST 5](#_Toc468870418)

[6 Screenshotit 6](#_Toc468870419)

[7 UML 11](#_Toc468870420)

[8 Tietokanta 11](#_Toc468870421)

[9 Aikataulutus vs toteuma 11](#_Toc468870422)

[10 Analyysi 13](#_Toc468870423)

[10.1 Janne 13](#_Toc468870424)

[10.2 Timo 14](#_Toc468870425)

[10.3 Lopputulema 15](#_Toc468870426)

# Johdanto

Tarkoituksenamme oli toteuttaa Android 5.1 Lollipop Sdk –versiolle toimiva mobiilisovellus, joka hyödyntää Kaa IoT Platformia ja Cassandra –tietokantaa. Toteutus tehtiin Windows 10 –käyttöjärjestelmällä käyttäen Android Studio 2.2.2 –versiota.

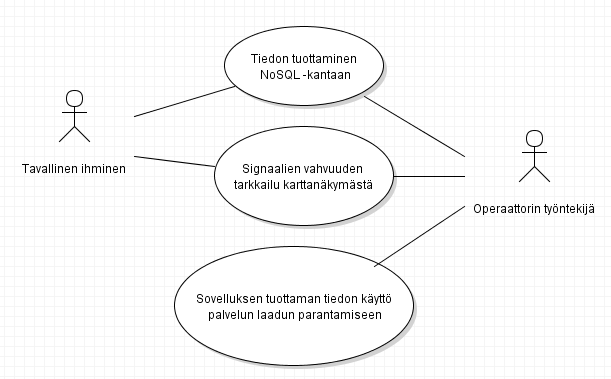
Lopputuotoksessa ei käytetty Kaa IoT Platformia ollenkaan, koska se olisi ollut turhan raskas yhden ainoan sovelluksen pyörittämiseen ja koska Kaa ei tarjoa rajapintaa tiedon hakemiselle Cassandra –kannasta. Tiedon hakemiseen ja tallentamiseen käytimme Spring Frameworkillä toteutettua REST –rajapintaa.

# Käyttötarkoitus

Sovelluksen tarkoitus on tuottaa valtavia määriä puhelinmallikohtaista tietoa suomalaisten operaattoreiden signaalien vahvuudesta eri puolilla Suomea. Operaattorit voisivat käyttää tätä tietoa palveluidensa laadun parantamiseksi. Sovellus tarjoaisi myös yleisnäkymän eri operaattoreiden signaalien vahvuudesta tavallisten ihmisten käyttöön.

# Use Case & User Strory

## Use Case



## User story

**Peruskäyttäjä, Timppa 23 opiskelija**

Timppa on kiinnostunut eri palveluntarjoajien signaalien vahvuudesta omalla kotipaikkakunnallansa. Timppa on myös hyvä ihminen ja haluaa auttaa muitakin saamaan tietoa eri palvelun tarjoajien signaalien vahvuudesta. Timppa etsii PlayStoresta mahdollisia sovelluksia kyseistä toimintaa varten ja löytää ylimmäksi rankatun sovelluksen ”Signaalin vahvuus” ja lataa sovelluksen puhelimeensa. Tämän jälkeen Timppa tutkailee eri palveluntarjoajien signaalien vahvuuksia ja voi tätä kautta mahdollisesti vaihtaa omaa puhelinliittymäänsä. Hyvänä ihmisenä hän myös jakaa tietoa sen hetkisen palveluntarjoajan signaalien vahvuudesta.

**Operaattorin työntekijä, Pekka 42**

Pekka on saanut työnantajaltaan tehtäväkseen selvittää työnantajansa tarjoaman palvelun taso. Pekalla ei ole kuitenkaan varattu tähän määrärahoja tai työkaluja ollenkaan. Pekka etsii siis PlayStoresta ilmaisia vaihtoehtoja tehtävän suorittamiseen ja löytää ylimmäksi rankatun sovelluksen ”Signaalin vahvuus” ja lataa sovelluksen puhelimeensa. Pekka saa sovellusta käyttämällä välittömästi tietoa puhelimensa näytölle työnantajan tarjoaman palvelun tasosta ja voi tätä kautta tehdä johtopäätöksiä siitä, missä on vahvin ja heikoin signaali. Pekkaa kiinnostaa myös raakadata, josta hieno näkymä on koostettu. Pekka painaa tietoja nappia, josta hän löytää linkin REST –rajapinnan käyttöön, mistä hän voi saada raakadatan kokonaan käyttöönsä.

# Sovelluksen toiminta

Sovellus kerää passiivisesti käyttäjältä paikkatiedon sekä GSM, CDMA, EVDO signaalien vahvuudet. Tiedot lähetetään JSON -olioina REST –rajapinnan kautta Cassandra –kantaan. REST –rajapinta hakee operaattoritaulun sisällön ja parsii JSON –muotoon, josta se parsitaan käyttäjän laitteessa ja esitetään karttanäkymässä (katso screenshotit). Käyttäjällä on valta valita lähettääkö tietoa palvelimelle. Käyttäjä voi sovelluksella avulla tarkkailla oman puhelimensa signaalien vahvuutta.

\\storage\homes\H3694\Dox\Downloads\SignalStateChart.png

# REST

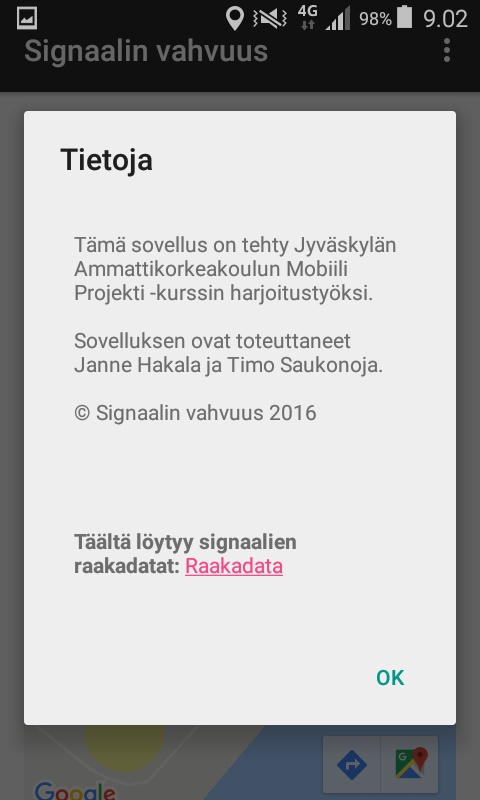
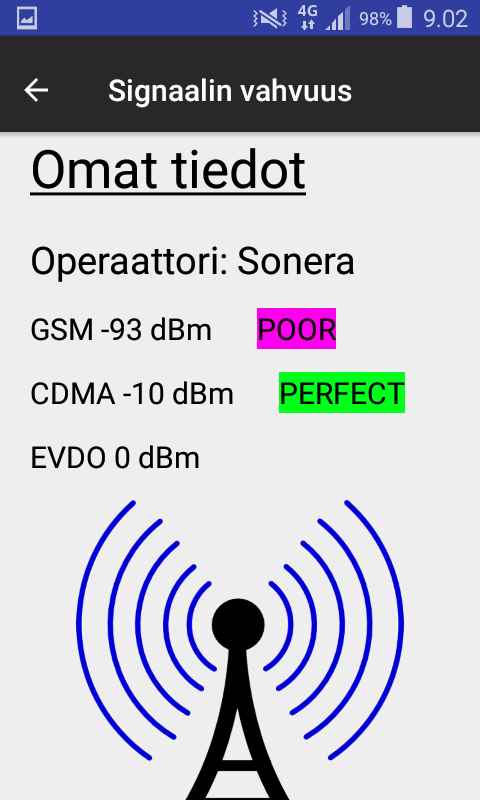
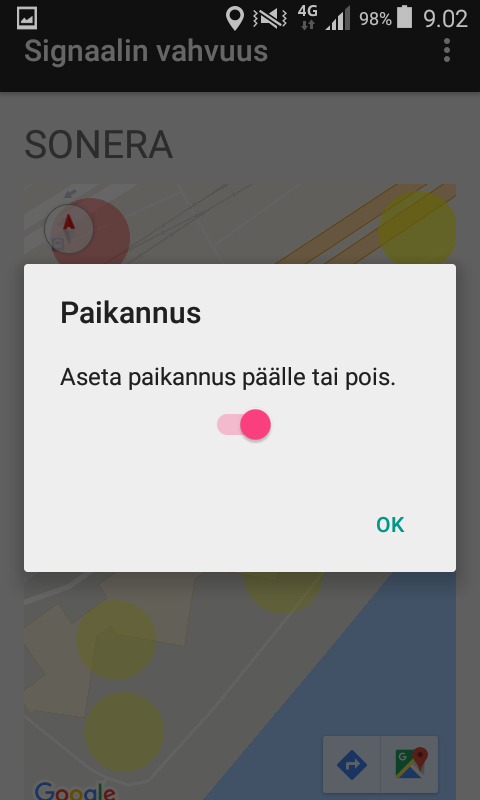
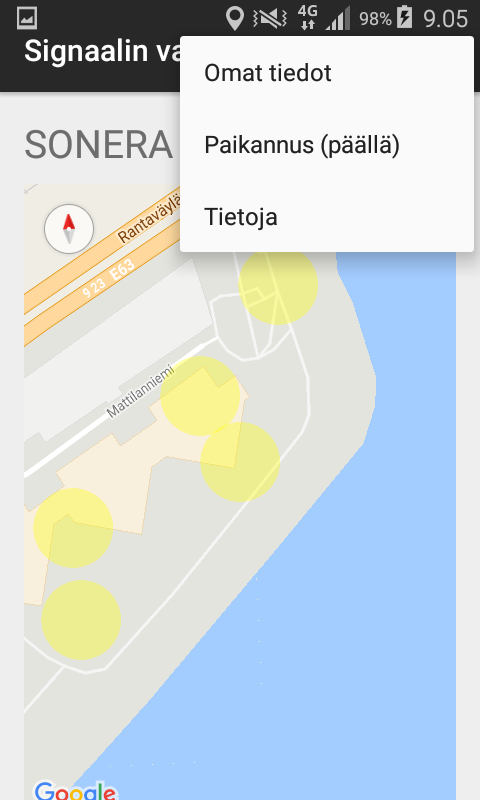
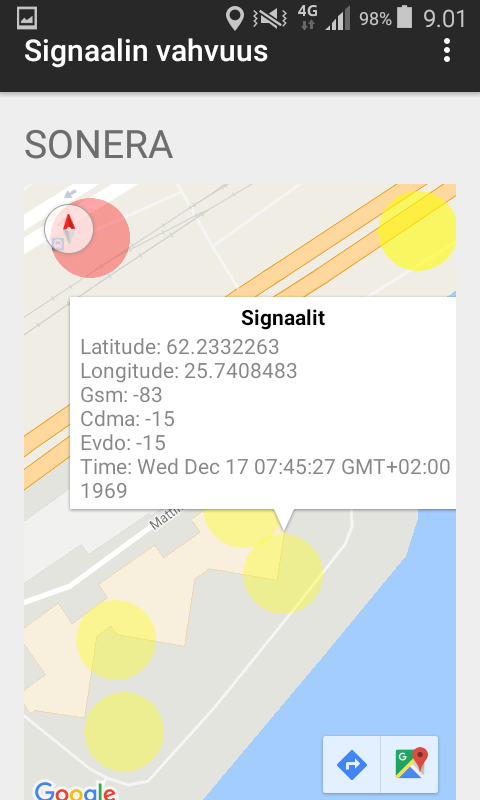
REST-rajapinta toteutettiin Spring Frameworkilla, joka on Javalla toteutettu open source kirjasto. Valitsimme Springin koska se tuli ensimmäisenä Googlesta vastaan ja sille oli olemassa hyviä tutoriaaleja.

REST rajapinta toteuttaa kahta toimintoa: asettaa puhelimesta lähetettyä dataa Cassandra kantaan ja hakee dataa kannasta puhelimessa näytettäväksi. REST käyttää datan liikutteluun DataLocation luokkaa jonka olio sisältää yhden kannan rivin tarvitsemat tiedot kun kuljetaan sekä sisään että ulospäin.

Kun puhelin haluaa näyttää dataa se lähettää get-pyynnön REST:lle. Rest hakee puhelimen haluaman taulun sisällön, kokoaa sen DataLocationList olioksi, joka on lista DataLocation olioista ja parsii sen listasta JSON muotoon Jackson kirjaston avulla ja tulostaa sen puhelimen saataville.

Kun puhelin haluaa lähettää dataa kantaan se lähettää post-pyynnön REST:lle. Puhelimessa oleva DataLocation olio parsitaan JSON muotoon ja lähetetään post-pyynnössä REST:lle. REST parsii JSON:sta DataLocation olion, josta se asettaa datan kantaan.

# Screenshotit



# UML

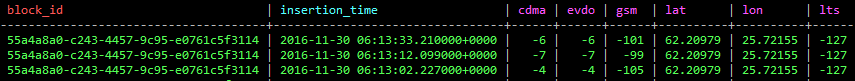
UML-kaavio löytyy tästä linkistä

<http://student.labranet.jamk.fi/~H3694/JSON/SignalUML.class.violet.html>

# Tietokanta

Tietokantana käytimme Cassandraa, joka on NoSQL(Big data)-kanta. Kannasta löytyy yksi Keyspace(tietokanta), joka sisältää oman taulun jokaiselle operaattorille. Työn aikana käytimme vain yhtä taulua, koska käytössämme oli vain yksi liittymä.

Alla kentät ja hieman dataa, jotka löytyvät taulusta.



# Aikataulutus vs toteuma

Aikataulutus:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Vastuu** | **Tuntiarvio** | **Valmis (viimeistään)** |
| MainActivity | Janne | 3H | 10.11.2016 |
| KaaActivity | Timo | 1H | 10.11.2016 |
| MapsActivity | Janne | 7H | 13.11.2016 |
| DataParser | Timo | 8H | 13.11.2016 |
| CassandraConnector | Janne | 4H | 19.11.2016 |
| Entry | Timo | 1H | 19.11.2016 |
| MyPhoneStateListener | Janne | 1H | 23.11.2016 |
| Viimeistely | Janne & Timo | 10H | 1.12.2016 |

Toteuma:



Iso osa koodista saatiin aiemmista mobiili-moduulin projekteista. Tämä nostaa molempien tuntimäärää n. 40-50 tunnilla ylöspäin.

Toteumaa ja suunnitelmaa verratessa huomaa, että suunnitteluvaiheessa emme ajatelleet tarvitsevamme erillistä REST –rajapintaa Cassandra –kannan käyttöön. Kuitenkaan Cassandralle ei ole olemassa Androidin kanssa yhteensopivaa kirjastoa, jolla siihen voisi ottaa yhteyden suoraan puhelimesta, joten päädyimme toteuttamaan oman REST –rajapinnan, johon Timolta kului käytännössä kaikki aika ja UI jäi siis käytännössä kokonaan Jannen tehtäväksi.

# Analyysi

## Janne

Timon tapellessa RESTIN ja Cassandran kanssa, niin minun kontolleni tuli Androidin puoli ja se, miltä UI –leiska tulisi puhelimessa näyttämään. Aluksi lähdin toteuttamaan suunnitelman pohjalta ollutta Fragment –switcheriä manuaalisesti, kunnes huomasin valmiin activityn ”Tab Activity”, jonka avulla pystyi luomaan valmiiksi kolmen sivun vaihto Fragmentit. Tämän jälkeen ongelmaksi muodostui GoogleMapin implementointi kyseiseen fragmenttiin, koska GoogleMap oli itsessään Fragment, joka olisi tullut olemassa olevan Fragmentin sisälle. Ainut vaihtoehto,jolla tällaisen pystyi toteuttamaan, oli MapView -elementti, minkä jälkeen GoogleMapin käyttö toimii samaan tapaan kuin ennenkin.

Muita ongelmia, mitä en saanut korjattua oli SoneraFragmentin sisällä oleva Asynctask –luokka, jonka olisi voinut eristää erilliseen tiedostoon, kuitenkaan jäälleen kerran ongelmia tuotti Fragment, enkä saanut datan välittyvän toisesta luokasta toiseen. Tällä hetkellä siis SoneraFragment sisältää Asynctask -luokan samassa tiedostossa. Muuta ongelmaa tässä ei ole kuin se, että koodin toistamista tulisi paljon, koska muut Fragmentit käyttäisivät samaa luokkaa.

Muita harmillisia devausongelmia tuotti se, että en omista Android –laitetta, joten jouduin tyytymään Android Studion emulaattoriin. Omalla pöytäkoneella emulaattori ei toiminut yhteen prosessorin kanssa, joten jouduin tyytymään kannettavalla koodaamiseen. Pari vuotta vanhalla kannettavalla emulaattori kaatui säännöllisin väliajoin ja uuden Gradle Buildin ajaminen otti aina 5-10 min. Tämä hidasti merkittävästi devaamista sekä aiheutti pieniä harmistuksen tunteita. Kuitenkin sain oman osuuteni projektista vietyä kunnialla loppuun saakka osaksi koulun koneiden avulla. Kaiken kaikkiaan tämä projekti oli erittäin hyvä ja opettava kokemus, josta tulee luultavasti olemaan hyötyä vielä tulevaisuudessa.

## Timo

Projektia ideoidessa tarkoituksena oli käyttää aiempaa kokemustani Cassandrasta ja sen Java kirjastosta. Seinä nousi vastaan heti alkumetreillä kun yritin käyttää tuttua Datastaxin Cassandra kirjastoa suoraan puhelimesta. Androidista puuttui kuitenkin tuki tälle kirjastolle ja useiden tuntien, yritysten ja runsaan Googlailun jälkeen suorasta tavasta oli luovuttava. Lueskelin aiheesta ja muut samankaltaista sovellusta kehittäneet olivat päätyneet käyttämään REST-rajapintaa apuna NoSQL:än ja mobiilisovellusten kanssa. REST oli käsitteenä tuttu, mutta kokemusta uupui. Päätimme kuitenkin jatkaa saman aiheen parissa, joten googlasin simppeliä REST ohjetta ja törmäsin Spring Frameworkkiin. Sain noin tunnissa tutorialin avulla tehtyä pienen Hello World REST:in ja homma vaikutti lupaavalta. Seuraava askel oli saada REST palauttamaan dataa kannasta ja JSON muodossa, josta se olisi helppo parsia puhelimen päässä. Tämä onnistui vielä suhteellisen helposti. Käytin Jackson kirjastoa JSON:in luomiseen. REST:in kanssa suurimmaksi haasteeksi muodostui usean kannasta haetun rivin parsiminen JSON muotoon. Tämä onnistui jälkeenpäin ajateltuna aika helpolla Jacksonin avulla, mutta söi työtunteja kuitenkin kiitettävästi. Jacksonilla pystyi luomaan olioista koostetusta listasta tulostettavan JSON:in josta puhelin pystyi parsimaan datan kartalle. Androidi puolelle ohjelmoin melko vähän ainoastaan samanlaisen DataLocation luokan kuin RESTillekin, jonka avulla Jackson pystyi helposti parsimaan molemmissa päissä ja HttpRequestTaskin, joka käytti Jacksonia DataLocation luokan lähetykseen REST:lle. Toisaalta puhelimen signaaleihin liittyvä koodi oli valmista aiemmista moduulin projekteista.

## Lopputulema

Lopputulemana olemme erittäin tyytyväisiä tuotokseen. Projektin parissa oli erittäin mielenkiintoista työskennellä ja sen kanssa käytimme useita uusia ja ”kuumia” teknologioita. Jos aikaa olisi ollut enemmän olisimme halunneet implementoida vielä ainakin: dataa kerätään myös näyttö sammuksissa, datan lähetyksen intervalli laskettaisiin käyttäjän nopeudesta, ympyrät eivät piirry päällekkäin ja haettua dataa rajattaisiin käyttäjän kaupungin mukaan. Saatamme työskennellä projektin parissa vielä jatkossa.