Henrik Tyrväinen

MOBIILISOVELLUKSEN KEHITTÄMINEN
Case: Limab Oy

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma

Marraskuu 2015

MAMK
University of Applied Sciences

KUVAILULEHTI

MAMK University of Applied Sciences	**Opinnäytetyön päivämäärä** **27.11.2015**
Tekijä(t) Henrik Tyrväinen	**Koulutusohjelma ja suuntautuminen** Tietojenkäsittelyn koulutusohjelma
Nimeke Mobiilisovelluksen kehittäminen - Case: Limab Oy	

Tiivistelmä

Tämän opinnäytetyön tarkoituksena on tehdä toimeksiantajalleni Limab Oy:lle mobiilisovellus. Mobiilisovelluksen tekemisellä halutaan tietää, voidaanko se toteuttaa heidän tarpeisiinsa, joita ovat mittarilta saadun tiedon lukeminen ja siihen reagoiminen mobiililaitteessa.

Sovelluksen kehittäminen onnistui, sen avulla voidaan saada mittarilta tieto, jos siinä ilmenee jokin vika tai huoltoväli. Mobiilisovellus koostuu kahdesta osasta. Ensimmäinen oli tietokannan ja web-palvelun rakentaminen. Web-palvelu lukee tietokantaa ja kirjoittaa sinne, kun taas mobiilisovellus on käyttöliittymä ja keskustelee web-palveluiden kautta tietokannalle. Web-palvelu toimii verkossa toimivalla palvelimella. Käytimme testauksessa Microsoftin Internet information services -palvelua, jolla saatiin tehtyä web-palvelin, johon voidaan selaimen URL:n kautta ottaa yhteyttä. Tämä mahdollistaa mobiililaitteiden keskustelun, web-palveluille, joka hoitaa tietokantaan tehdyt SQL-kielen kyselyt. Yksi palveluista, joka toimii notification-viestin lähettäjänä, pyörii web-palvelimella aina. Sen on tarkoitus ilmoittaa käyttäjää, jos tietokantaan tulee uusi tietue, eli mittari on lähettänyt viestin tietokantaan. Tämä mahdollistaa tiedon saamisen aina, vaikka sovellus mobiililaitteessa olisi kiinni.

Työssä saatiin siis toimiva prototyyppi sovelluksesta, jota toimeksiantaja pääsee testaamaan oikeassa ympäristössä ja sen kautta kehitetään sovellusta eteenpäin. Työstä kävi ilmi, että on mahdollista kehittää mobiilisovellusta tehtaissa toimiviin mittareihin, vaikka näillä kahdella voi olla aivan huimasti eroavaisuuksia tekniikan ja toimivuuden suhteen.

Asiasanat (avainsanat)

mobiilisovellukset, sovellukset, ohjelmointi, tietokannat, JavaScript, jQuery mobile, Cordova, SQL

Sivumäärä 28	**Kieli** Suomi

Huomautus (huomautukset liitteistä)

Ohjaavan opettajan nimi Janne Turunen	**Opinnäytetyön toimeksiantaja** Limab Oy

DESCRIPTION

MAMK University of Applied Sciences	**Date of the bachelor's thesis** 27 November 2015
Author(s) Henrik Tyrväinen	**Degree programme and option** Business Information Technology

Name of the bachelor's thesis

Development of a mobile application – Case: Limab Oy

Abstract

The purpose of this thesis was to make a mobile application for my employee Limab Oy. By making the mobile application we want to know it could meet their needs of reading the data produced by a measuring device and showing it on a mobile device.

Developing an application was successful it could be used to get the meter data in case of a fault or a maintenance interval. The mobile application consisted of two parts. The first was the construction of a database and a web service. The web service reads the database and writes there, while the mobile application is the user interface and communicates with the data via a web services repository. The web service runs on network by using online web server. In testing we used Microsoft's Internet Information Services which we used to make a web server which can be connected via a browser using URL. This allows mobile devices to talk with the web services which makes the queries for database in SQL-language. Working on a sender of notification messages is running on the server all the time. It is meant to inform the user if the database gets a new record that is, the measuring device has sent message to database. This makes it possible to access the information at all times even if an application on a mobile device would be closed.

In sum, the project rounded in a working prototype application for the client to test on a real environment, and through, that test they can develop the application forward. The Work revealed that it was possible to develop a mobile application what is operating on factories measuring devices, although the two could be quite dramatically different in terms of technology and functionality.

Subject headings, (keywords)

mobile applications, applications, programming, databases, JavaScript, jQuery mobile, Cordova, SQL

Pages 28	**Language** **Finnish**

Remarks, notes on appendices

Tutor Janne Turunen	**Bachelor's thesis assigned by** Limab Oy

SISÄLTÖ

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena on tehdä Limab Oy:lle mobiilisovellus, jolla voidaan tarkkailla puun mittauslaitteiden toimintaa mobiilisti. Limab Oy on yritys, joka toimittaa mittausjärjestelmiä saha-, levy- ja terästeollisuuteen. Mittausjärjestelmien toimittamisen ohella Limab Oy huolehtii niiden käyttöönotosta ja ylläpidosta. Joskus Mittausjärjestelmät voivat tarvita huoltoa tai niissä ilmenee jokin vika, silloin tarvitaan ilmoitus mittarilta jonka pitäisi tulla nopeasti huomatuksi, joten sovellus tehdään mobiilille. Tällöin ei tarvita jatkuvaa tarkkailua vaan mittari ikään kuin ilmoittaa itse, kun se tarvitsee huoltoa. Mittarista tulevan viestin on tarkoitus sisältää heidän puun mittauslaitteista tulevaa dataa. Data on numeraalista, ja arvot muuttuvat. Tämän datan saattaminen tiedoksi puhelimeen antaa heille mahdollisuuden tarkkailla ja pysyä ajan tasalla tuotantolinjan toiminnasta. Tarkoitus on myös siis parantaa tuotantolinjan luotettavuutta ja tehokkuutta.

Mobiilisovelluksen toteutuksessa käytän koulussa oppimiani taitoja ja opettelen myös paljon uusia asioita. Työnä aihe on sopivan haastava ja realistinen. Halusin päästä tekemään mobiiliohjelmointia opinnäytetyössä, ja tämä sopii tarkoitukseen hyvin. Työssä teen Limabin toimistolla sovellustani noin 6–8 tuntia päivässä arkisin parin kuukauden ajan. Työssä saan jonkun verran apua henkilöstöltä, mutta suurin osa työstä pitää tehdä itse.

Aikaisempi kokemus mobiilisovellusten tekemisestä jää yhteen koulussa käytyyn opintojaksoon, joten minulla oli vielä paljon opittavaa sovellusten kehittämisestä ja toteuttamisesta. Etenkin sovelluksen saaminen oikeaan käyttöön asettaa paljon huomioitavia kohtia, kuten käyttöönotto ja ylläpitäminen. Näitä ei koulussa välttämättä opi, koska siellä tehdään harjoitustehtäviä, jotka toimivat niin kuin on pyydetty ja ei sen enempää.

Luvussa 2 aloitan hahmottelemalla, minkälainen on tilanne maailmalla sovellus ja käyttöjärjestelmämarkkinoilla, ja mitkä ovat minkäkin alustan heikkoudet ja vahvuudet. Tällä voin perustella, miksi juuri valitsin kehitysvälineeksi alustan, jolla kehitän ja toteutan sovelluksen.

Luvussa 3 on tarkoitus selvittää eri sovelluskehittämisen tekniikoita. Nämä tekniikat helpottavat työn tekemistä ja ovat laajalti käytettyjä sovelluskehityksessä. Kerron sy-

vemmin, mitä sovelluskehykset tekevät ”pellin alla” eli, miten kehys toimii juuri sillä tavalla, joista normaalin käyttäjän ei välttämättä tarvitse tietää mitään. Myös tietokannat ovat keskeinen osa työssäni ja muutenkin sovelluksissa, joten on hyvä käydä läpi sen toimintaa.

Luvussa 4 kerron, kuinka toteutin sovellukseni ja mitä haasteita siihen liittyi. Uusia ratkaisuja täytyi keksiä paljon, koska en ollut ennen tehnyt, työssä törmäämiini asioihin. Monet ratkaisut selvisivät huolellisella tutkimisella ja päättelyllä. Toteutuksen on tarkoitus selvittää ajatuksenkulkuani sovellusta kehittäessäni ja esittää vastaukset haasteisiin, joita on aina tällaista työtä tehdessä.

2 MOBIILIOHJELMOINTI

Perinteisesti mobiiliohjelmointi alkaa alustan valitsemisesta, mutta nykyään on mahdollista tehdä kaikilla alustoilla toimiva sovellus samalla koodilla. Tämä helpottaa sovelluksen levittämistä ja ylläpitoa. Muutamat käyttöjärjestelmät eroavat jonkun verran toisistaan toimintojensa ja eri selainten käytön vuoksi. Tämä näkyy sovelluskehityksessä, kun ilmaantuu odottamattomia virheitä.

2.1 Käyttöjärjestelmät

Mobiilisovellusta kehitetään yleensä alustakohtaisesti, ja alustojen käyttäjämäärät vaihtelevat. Mobiilisovellus halutaan tietenkin mahdollisimman monelle käyttäjälle, yrityksen intressejä huomioiden. Näin saadaan maksimi hyöty kehitetylle sovellukselle, oli sen tarkoitus mikä tahansa.

Android-käyttöjärjestelmä on Googlen kehittämä, ja se toimii monissa eri mobiililaitteissa, kuten puhelimissa ja tableteissa. Android perustuu Linux-käyttöjärjestelmään ja se on suunniteltu käytettäväksi kosketusnäytöllä. Siihen voi ladata sovelluksia (apps) Google Play Storesta.

Android on ilmainen ja vapaa kaikille. Jokainen voi ottaa Androidin lähdekoodin ja alkaa kehittää käyttöjärjestelmää omalle polulleen. Android toimii erittäin monella lait-

teella, ja sen päätarkoituksiin kuuluu luoda avoin sovellusalusta kaikenlaisille elektroniikoille, jotta kehittäjät voisivat luoda lennokkaita ideoitaan tuotteiksi ja parantaa näin mobiilin käyttökokemusta. (Android 2015.)

Android tarjoaa rikkaan sovelluskehyksen sovellusten kehittämiseen. Androidille ohjelmoidaan Java-ohjelmointikielellä. Androidin sovelluksen rakennukseen on monta eri ohjelmointirajapintaa (API), joita voi käyttää vapaasti haluamaansa tarkoitukseen (Android Developer 2015). Esimerkiksi yksittäinen käyttötapaus voi olla yhdellä ruudulla toimiva käyttöliittymä ja palvelut suoritetaan itsenäisesti takana.

Androidin SDK (Software Development Kit) kääntää Java-koodin APK:ksi (Android Package). Kaikki data sovelluksesta ja sen resurssit menevät arkistotiedostoksi, joka saa päätteen .apk. APK sisältää kaikki tarvittavat osat Androidissa toimimiseen ja se on samalla asennus paketti. (Android Developer 2015.)

iOS on Applen kehittämä käyttöjärjestelmä mobiililaitteille. Sillä on pienempi markkinaosuus kuin Androidilla, mutta se on hyvin tunnettu maailmalla. Applen iPhone ja iPad käyttävät iOS-käyttöjärjestelmää ja niille voi ladata sovelluksia App Storesta.

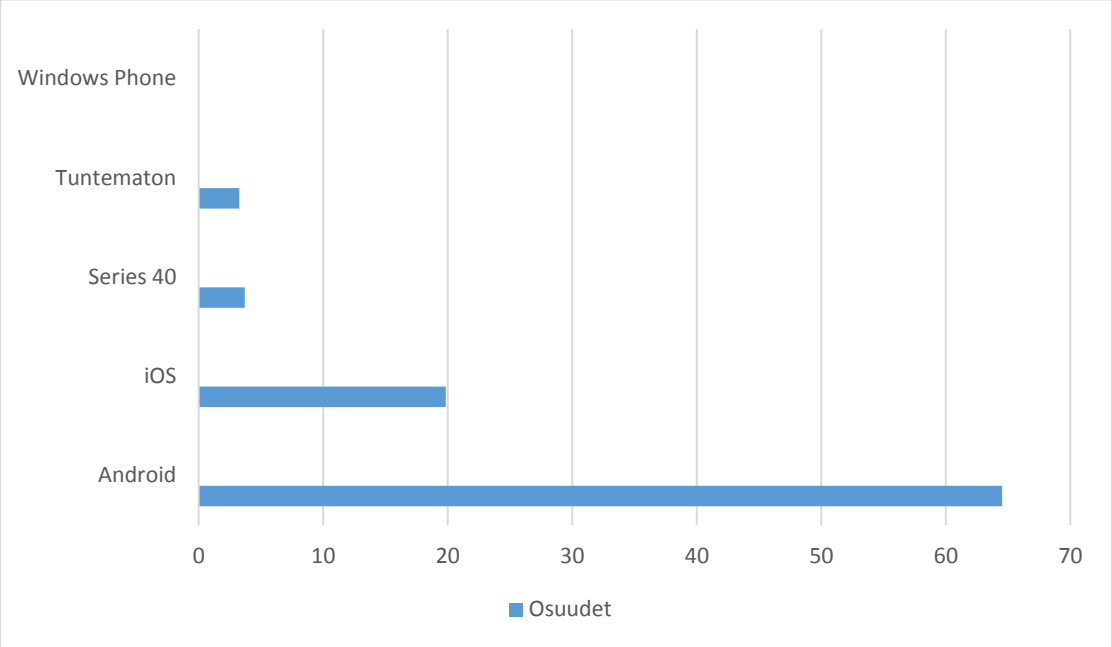
iOS 9 on tällä hetkellä uusin käyttöjärjestelmä, joka toimii Applen monilla eri tuotteilla ja sen ominaisuuksia on lisätty jokaisella versiolla. iOS ei ole avoin kehitykselle vaan se on Applen itse kehittämä käyttöjärjestelmä. iOS sai alkunsa 2007, kun se julkaistiin ensimmäiseksi iPhonelle ja on sen jälkeen levinnyt muihinkin tuotteisiin. iOSin suurimpia valtteja on sen helppo käytettävyys ja valtava sovellusvalikoima App Storessa, jossa on jo yli 1,5 miljoonaa erilaista sovellusta. (Apple 2015.)

iOS 9 SDK tarjoaa taas edellistä paremmat palvelut ja uusia rajapintoja sovellusten kehittäjille. Sillä voi tehdä monta tehtävää samanaikaisesti (Multitasking). Uusia ominaisuuksia on lisätty, kuten tuki CloudKit, HomeKit, HealthKit ja MapKit-palveluille laajentavat kehitysmahdollisuuksia paljon (Apple Developer 2015). SDK tarjoaa sovelluskehitykselle aina uusia ohjelmointirajapintoja, joilla kehittäjät pääsevät leikkimään ja näin Apple saa laajennettua App Store -valikoimaansa.

Sovellusten kehittäminen on maksullista ja siihen tarvitaan Applen kehittäjäjäsenyys, joka maksaa vuodessa 99 dollaria. Ilman jäsenyyttä käyttäjä ei voi testata sovellustaan

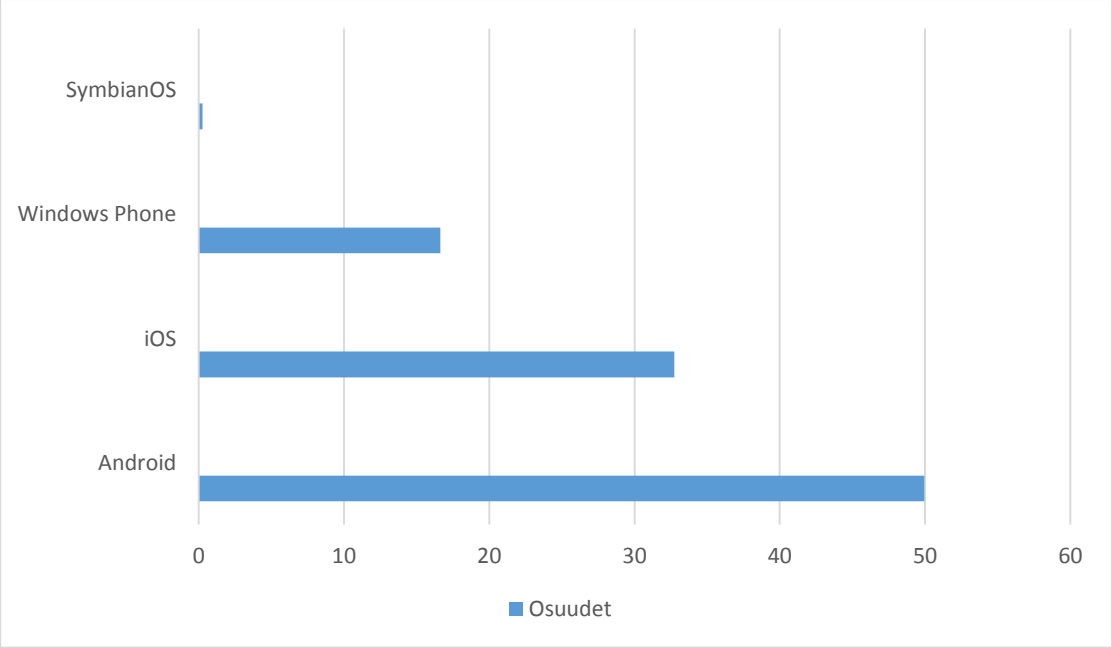
oikealla laitteella eikä julkaista sovellusta App Storessa. iOS SDK:ta kehitetään Xcode-kehitysympäristöllä. Xcode toimii Objective-C-ohjelmointikielellä, ja sillä voidaan tehdä graafisia käyttöliittymiä ja tietokantaohjelmointia. (Apple Developer 2015.)

Tällä hetkellä maailmanlaajuinen mobiilikäyttöjärjestelmien kilpailu suosii selvästi kahta tekijää. Näillä tarkoitetaan tietenkin iOS- ja Android-käyttöjärjestelmiä. Niiden osuus maailmanmarkkinoilla on selvästi suurin (Ristola 2014). Kuten kuvasta 1 näkyy, Android on suvereeni hallitsija käyttöjärjestelmien markkinaosuudessa, mutta ei iOS kaukana ole. Näiden statistiikkojen perusteella sovelluskehittäjät yleensä valitsevat Androidin tai iOS:n. Näin saavutat suurimman mahdollisen käyttäjäkunnan.



KUVA 1. Mobiilikäyttöjärjestelmät maailmalla kesäkuu 2015–syyskuu 2015 (StatCounter 2015, mukaillen)

Suomen markkinoilla tilanne taas on eri (kuva 2). Suomessa menestyy selvästi kolme käyttöjärjestelmää. Yllättäjänä on Windows Phone, joka selittyy tilaston löytymisestä tietenkin Nokian takia. Suomessa myös Windows Phone on suosittu työsuhdepuhelin, joka vaikuttaa myös (Ristola 2014).



KUVA 2. Mobiilikäyttöjärjestelmät Suomessa kesäkuu 2015-syyskuu 2015 (StatCounter 2015, mukaillen)

Tämän takia on Suomessa hyödyksi käyttää alustariippumatonta sovelluskehystä, kuten esimerkiksi Cordova. Tämä mahdollistaa sovelluksen julkaisun monille alustoille samalla lähdekoodilla. Näin voidaan saavuttaa suurin mahdollinen käyttäjäkunta Suomessa.

3 KEHITYSTEKNIIKAT

Mobiilille kehittäessä käytetään monenlaisia sovelluskehyksiä, joiden avulla ohjelmoija välttyy pyörän keksimiseltä uudelleen. Sovelluskehykset ovat suosittuja, koska ne vähentävät koodin pituutta, mikä taas vähentää ohjelmointiin tehtävää työtä. Tarkoitus on myös helpottaa koodin kirjoittamista, koska vaikeimmat tehtävät ja rakenteet voivat olla sovelluskehyksessä rakennettu paljon helpommin käytettäväksi, jolloin sovellusten kehittäjän ei tarvitse kuin osata käyttää sitä.

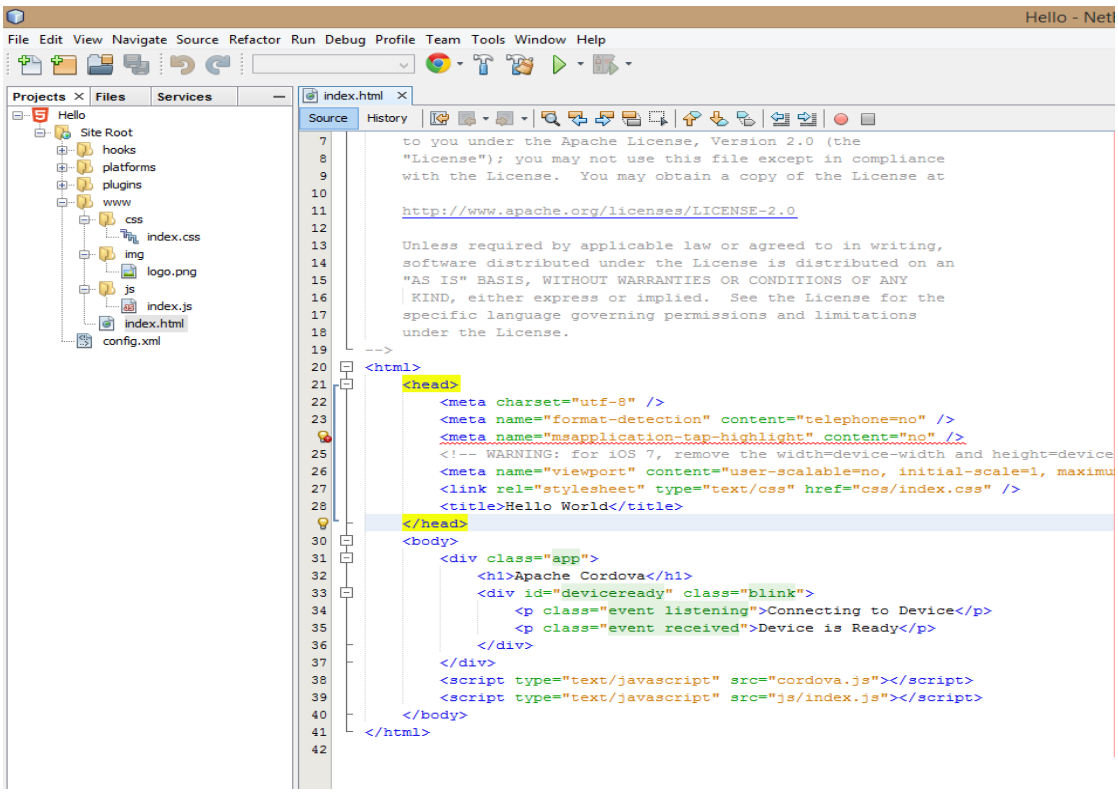
3.1 Apache Cordova

Cordova on avoimeen lähdekoodiin perustuva mobiilisovellusten kehitykseen tarkoitettu sovelluskehys. Se mahdollistaa normaaleilla web-tekniikoilla, kuten HTML, CSS

ja JavaScriptillä kehittämisen alustariippumattomasti. Näin vältetään eri mobiilialustojen natiivikielet (alustan käyttämä ohjelmointikieli) ja voidaan helposti päästä käyttämään rajapinnan kautta laitteiden ominaisuuksia, kuten kameraa ja mikrofonia (Apache Cordova 2015). Sama ohjelmakoodi toimii siis kaikissa mobiililaitteissa, mikä tekee mobiilisovelluksen kehittämisestä nopeaa ja kannattavaa.

PhoneGap eli Cordovan ”isä” luotiin noin 2009 aloittavan yrityksen toimesta nimeltä Nitobi. Sen tarkoitus oli siirtää mobiilisovellusten kehitys kokonaan web-teknologioille, mutta silti pitää reitti avoinna natiivin koodin kutsumiseen. 2011 Adobe osti yrityksen ja siirsi tärkeimmät osat avoimesta lähdekoodista Apache Software Foundationille nimellä Cordova (Ionic Blog 2014; PhoneGap Blog 2012). Kaikki pystyvät rakentamaan Cordovan päivityksiä ja parannuksia, sekä lisäämään osia siihen. Näin Adobe saa myös laittaa niitä omaan sovelluskehykseensä eli PhoneGap:lle. Tämän takia nämä kaksi sovelluskehystä muistuttavat hyvin paljon toisiaan.

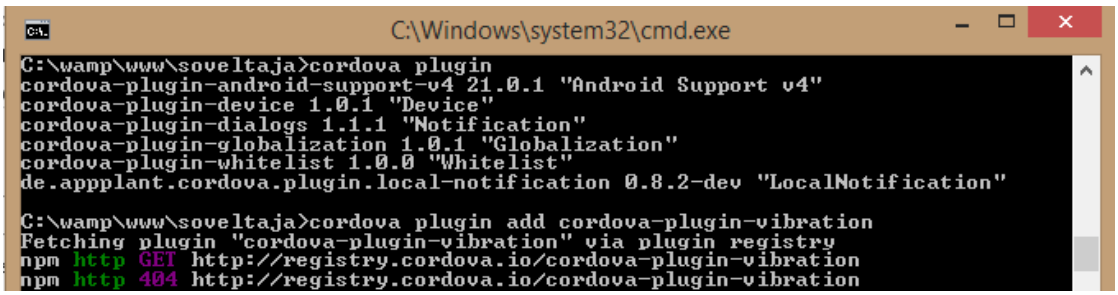
Apache Cordova toimii config.xml-tiedoston avulla, josta se saa tietoja sovelluksesta ja määrää, kuinka sen kuuluu toimia. Esimerkiksi reagoiko puhelin liikkeisiin. Sovelluksen teko näkyy saman lailla, kuin verkkosivut. Sovellus saa index.html-tiedoston, jossa on viittaukset kaikkeen verkkosivujen kehittämiseen tarvittavaan, kuten CSS, JavaScript, kuvat ja media-tiedostot, sekä muita tarpeellisia hyödykkeitä, joita sovellus tarvitsee toimiakseen. Sovellus suoritetaan WebView-näkymänä laitteen natiivissa rajapinnassa (Apache Cordova 2015). Cordova-projektin luonti antaa hyvän pohjan mobiilisovelluksen kehittämisen aloittamiseen (kuva 3).



KUVA 3. Cordova-projektipuu ja index.html

Cordovan oleellisimpia asioita eli liitännäisten (plugin) käyttö toimii natiivien komponenttien ja Cordovan kommunikoinnilla. Liitännäisiä kutsutaan JavaScript-kielellä natiivikoodista ja parhaimmillaan JavaScriptiä käyttävien rajapintojen avulla se toimii useilla eri laitteilla (Apache Cordova 2015). Suurimmat kehitykset saavutetaan kuitenkin kolmannen osapuolen liitännäisillä. Niitä löytyy huomattava määrä, ja ne ovat välttämättömyys joihinkin natiiveihin toimintoihin.

Cordovaa käytetään yleensä komentorivikehotteelta, joka on yleisin käyttötapa. Komentoriviltä käyttäjän ei tarvitse tietää, mille alustalle lopullinen sovellus tehdään vaan sen voi rakentaa jokaiselle alustalle suunnatusti. Alustapohjainen kehittäminen on taas vastakohta, jossa sovellus tähdätään yhdelle alustalle. Komentoriviltä tehdessä ei pääse käsiksi kaikkiin toimintoihin, kun taas alustalle rakennettaessa voidaan itse määrätä uusia toimintoja (Apache Cordova 2015). Komentoriviltä tehdessä on helpompi aloittaa mobiilisovelluksen kehittäminen ja liitännäisten lisääminen käy komentoriville oikean syötteen kirjoittamalla. Kuvassa lisätään liitännäinen, joka mahdollistaa puhelimen tärinä toiminnon käyttämisen (kuva 4).



KUVA 4. Cordova-projektiin liitännäisen lisääminen komentoriviltä.

Suurin osa Cordovan toiminnoista tulee config.xml-tiedostosta, joka on eräänlainen globaali määritystiedosto alustalle. Alustat kuten iOS, Android ja Blackberry 10 saavat oman config.xml-tiedoston, kun aloitetaan projektin luonti Cordovan CLI(Command line interface, komentorivi) toiminnolla. Config.xml-tiedosto sisältää alustapohjaisia ohjelmointirajapintoja, jotka ohjaavat mobiilin natiiveja toimintoja, kuten kameraa ja mikrofonia (Apache Cordova 2015).

```
<?xml version='1.0' encoding='utf-8'?>
<widget id="fi.testi.sovellus" version="0.0.1" xmlns="http://www.w3.org/ns/widgets" xmlns:cdv="http://cordova.apache.org/ns/1.0">
    <name>Heippa Maailma</name>
    <description>
        A sample Apache Cordova application that responds to the deviceready event.
    </description>
    <author email="dev@cordova.apache.org" href="http://cordova.io">
        Apache Cordova Team
    </author>
    <content src="index.html" />
    <plugin name="cordova-plugin-whitelist" version="1" />
    <access origin="*" />
    <allow-intent href="http://*/*" />
    <allow-intent href="https://*/*" />
    <allow-intent href="tel:*" />
    <allow-intent href="sms:*" />
    <allow-intent href="mailto:*" />
    <allow-intent href="geo:*" />
    <platform name="android">
        <allow-intent href="market:*" />
    </platform>
    <platform name="ios">
        <allow-intent href="itms:*" />
        <allow-intent href="itms-apps:*" />
    </platform>
</widget>
```

KUVA 5. Config.xml-tiedosto sisältö, jonka Cordova luo kun tehdään projekti komentoriviltä.

Kuvassa 5 ylimmät asetukset pätevät kaikkiin alustoihin, kuten id, versio ja nimi. Kun taas alempana voidaan määritellä alustakohtaisia asetuksia, kuten avausruutu, äänet ja kuvat. Alustoille määritetään yleensä asetuksia preference (suositus) merkinnällä. Näillä preference-merkinnällä voidaan vaikuttaa mobiilin natiivitoimintoihin, kuten esimerkiksi kuvassa 6 asetetaan mobiili kokoruututilaan, jolloin ei näy edes ylälaidan tilarivi (Statusbar).

```
<preference name="Fullscreen" value="true" />
```

KUVA 6. Config.xml preference esimerkki.

Config.xml:n ilmestyy määrittelyt myös lisäosille, jotka ovat asennettuna alustalle. Ne tulevat merkinällä feature (ominaisuus). Eri alustoille voi määrätä omat suosituksensa ja ominaisuutensa, monet alustat ja laitteet eroavat paljon, joten toimintojen erottelulla saadaan sovellus toimivaksi kaikilla alustoilla (Apache Cordova 2015). Kuvassa 4 asennettu *vibration*-lisäosa näkyy config.xml-tiedostossa (kuva 7).

```
<feature name="Vibration">
    <param name="android-package" value="org.apache.cordova.vibration.Vibration"/>
</feature>
```

KUVA 7. Config.xml feature esimerkki.

Config.xml sisältää perustan alustan toiminnalle ja sen käyttäytymiselle. Se on myös tärkeä tiedonlähde sovelluksen tekijälle, koska sitä voidaan käyttää versioinnissa ja asennettujen ohjelmien muistiona.

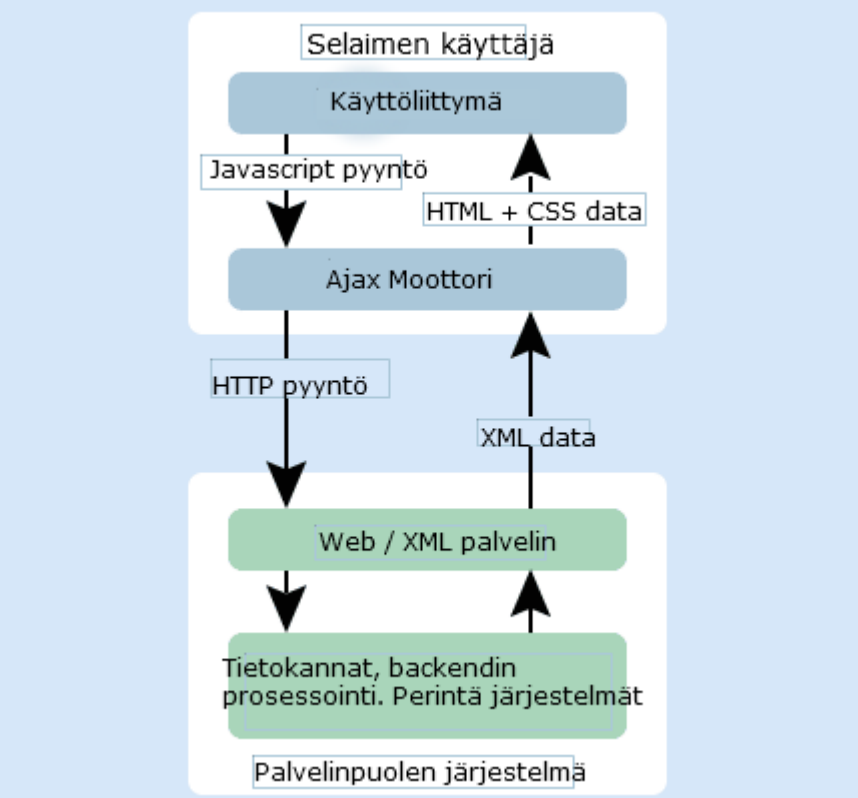
3.2 JQuery Mobile

jQuery Mobile on jQuerysta kasvanut mobiilisovelluksiin tarkoitettu sovelluskehys. jQuery on tietokoneiden selainten verkkosivujen tekoon käytetty sovelluskehys, joka on noussut suosioon maailmalla, helpon käytettävyytensä ja erinomaisten toimintojensa avulla. jQuery Mobile laajentaa jQuerya tuomalla samoja toimintoja mobiiliin kehitykseen ja tarjoamalla myös uuden tavan kehittää mobiilisovelluksia.

JQuery Mobile auttaa responsiivisessä suunnittelussa ja kaikille alustoille sopivia mobiilisovellusten ja sivustojen rakentamisessa. Se perustuu jQuery ja jQuery UI -säätiöön ja tarjoaa Ajaxin avulla navigaatiota sivujen vaihdossa, kosketustapahtumissa ja eri *widgettejä* (pienoisohjelmia). Sitä on helppo käyttää ja se on tehty kevyeksi ja joustavaksi (jQuery Mobile About). jQuery Mobile on aloittelijaystävällinen ja hyvä sovelluskehys mobiilisovellusten tekemiseen. Sovelluksen responsiiviseen suunnitteluun ei tarvitse käyttää aikaa, koska kehys skaalaa sovelluksen jokaiselle näyttökoolle. Tämä nopeuttaa huomattavasti sovelluksen kehittämistä. Se helpottaa monia käytettyjä toimintoja, kuten DOM-elementtien hallintaa ja Ajax-tiedonsiirtotavan toteutusta.

Perinteisesti nettisivut päivittivät sisältönsä vasta uudelleen ladattaessa. Päivittäminen saattoi olla erittäin raskas prosessi, esimerkiksi sähköpostia tarkastaessa piti käyttäjän

manuaalisesti painaa päivitystä ja koko sivu ladattiin uudelleen HTML, CSS ja JavaScriptia myöten, sekä myös käyttäjän sähköposti. Tämä ongelma ratkaistiin kuitenkin 2003, kun suurimmat selaimet ottivat käyttöönsä XMLHttpRequest (XHR) olion, joka mahdollisti kommunikoinnin palvelimen kanssa, ilman että sivua tarvitsi ladata uudelleen (jQuery Ajax 2015).



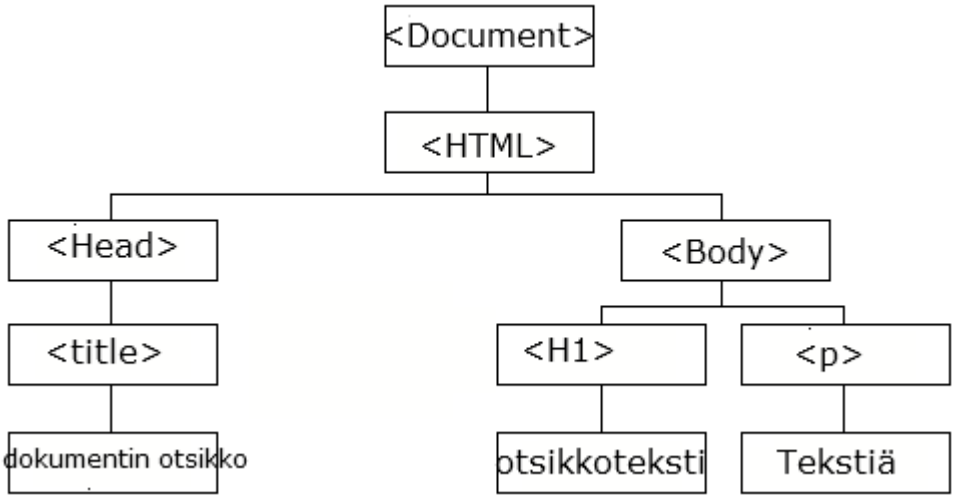
KUVA 8. Ajax-toiminta (Ajax programming 2015, mukaillen)

XHR-olio on osa Ajax (Asynchronous JavaScript and XML) teknologiaa. Ajaxia käyttämällä voidaan välittää viestejä palvelimen ja selaimen kesken ilman, että tarvitaan koko sivun uudelleen päivittämistä. Tässä hyödynnetään XHR:n suomaa rajapintaa, joka muuntaa datan XML-muotoiseksi ja lähettää sen Http-protokollan kautta selaimelle (kuva 8). Myös asynkroninen tiedonsiirtotapa ei häiritse koodin suorittamista, joten ohjelma voi jatkaa toimintaansa sillä välin, kun kutsua käsitellään (jQuery Ajax 2015). Kutsun jälkeen pitää vain muistaa ottaa vastaan tiedot (callback), kun se on valmis.

DOM (Document Object Model) on puumainen tiedostorakenne, joka on web-tekniikoiden yksi yleisimmistä tietorakenteen malleista. DOM-puun avulla voidaan hallin-

noida sivuilla näkyvää tietoa kätevästi JavaScriptilla. Kaikki käyttäjän aktiviteetti sivuilla, kuten hiiren klikkaukset ja näppäimistön painallukset, voidaan ottaa vastaan DOM:n kautta.

Yksi keskeisimmistä toiminnoista jQueryssa on DOM-puun manipulointi. DOM:lla tarkoitetaan HTML, XHTML ja XML -tyyppisten dokumenttien esitys- ja käyttäytymistapaa. DOM:lla on tarkoitus auttaa sivun elementtien navigoinnissa, kun koodataan toimintoja eri elementeille, jolloin yleensä koodauskielenä selaimissa puhuttaessa on JavaScript. JavaScriptillä koodatessa voi DOM:n navigointi olla raskasta ja hankalaa, jQuery-sovelluskehyksessä tulee onneksi mukana monia helpottavia metodeja DOM:n manipuloimiseksi (jQuery tutorial 2015).



KUVA 9. Document Object Model -esimerkki

DOM-puu rakentuu loogisesta järjestyksestä ja näyttää tämän vuoksi puumaiselta rakenteelta (kuva 9). jQueryn dokumentaatiosta löytyy monta eri tapaa manipuloida DOM-puuta. jQuerylla voi vaikuttaa elementteihin lisäämällä, muokkaamalla, poistamalla tai monilla muilla tavoilla. jQuerylla voidaan lisätä elementtiin tekstiä tai lukea sitä. Näitä metodeja kutsutaan settereiksi ja gettereiksi (Category: Manipulation 2015). Esimerkiksi *.append()* ja *.remove()* ovat settereitä, koska ne suorittavat sivun elementeille toimintoja DOM:n kautta, kun taas *.html()* ja *.val()* ovat gettereitä, jotka hakevat tietoa sivulta.

```
<script>
// Hakee ensimmäinen esine listasta
var listaEsine = $( 'li' ).first(); // Viimeisen haku: .last()

// Hakee sisarukset listan esineelle
var sisarukset = listaEsine.siblings();

// Hakee seuraava sisar listassa
var seuraavaSisar = listaEsine.next(); // Aikaisempi sisar: .prev()

// Hakee listan vanhemman
var lista = listaEsine.parent();

// Hakee lähimmät listan lapset
var listaEsineet = lista.children();

// Hakee kaikki listan esineet
var kaikkiListanEsineet = lista.find( 'li' );

// Hakee kaikki listan esi-isät, joiden luokkana on "module"
var moduulit = listaEsine.parents( '.module' );

// find the closest ancestor of the list item that has a class of "module"
// Hakee lähimmän listan esi-isän, jolla on luokka "module"
var moduuli = listaEsine.closest( '.module' );
</script>
```

KUVA 10. jQuery esimerkki DOM:n käytöstä

DOM-rakennetta voi kuka tahansa alkaa käyttää, se ei tarvitse erillistä asennusta. Monet eri selaimet ovat sisällyttäneet eri variaatioita DOM-rakenteesta, mutta perusperiaate on kaikissa sama (MDN 2015). Kuvassa 10 haetaan html-tiedoston listan elementeistä tietoja. jQuery tarjoaa paljon nopeamman syntaksin DOM:n käytölle kuin JavaScript-lauseet. jQueryn ohjelmointisyntaksi on tiivistetty hyvin lyhyeksi, jotta koodi on nopeampi ja selkeämpi kirjoittaa.

3.3 Tietokannat

Tietokantojen käyttö helpottaa datan hallitsemista ja muistamista. Sovelluksesta saa paljon älykkäämmän, kun käytetään tietokantoja, koska tietoa voidaan tallettaa ja lukea. Tietokantaan saa esimerkiksi käyttäjätiedot, joiden avulla voidaan yksilöidä viestitys. Tietokannoista saadaan dataa, jota käyttäjä tarvitsee sovelluksessa.

Tietokanta koostuu samankaltaisesta tiedosta. Tieto voi olla tosiasioita, joita voidaan kirjata ja niillä on merkitys. Esimerkiksi videovuokraamon elokuvakokoelma tai varas-

ton tuotelista voivat olla tietokantoja (Lahtonen 2002, 2). Tietokantaan tulee projektissani tietoa mittarilta ja sovellukselta. Tietokantaa käytetään tiedon tallentamiseen ja lukemiseen.

Vuonna 1970 E.F. Codd esitteli relaatiomallin. Se mullisti tietokantamallin vaatimukset, koska se oli yksinkertaisin ja joustavin. Ainoana miinuksena voidaan sanoa suuri koneresurssien tarve. Relaatiotietokannassa tiedot ovat tauluina (table). Yhtä riviä kutsutaan tietueeksi (record). Jokaisella taulun rivillä täytyy olla yhtä monta kenttää (field) ja jokaisella kentällä perusavain. Yksittäistä tietoa voidaan hakea relaatiotietokannoissa äsken mainitsemilla käsitteillä, esimerkiksi taulun nimellä, sarakenimellä tai avaimen arvolla (Lahtonen 2002, 4). Projektissani käytän Microsoftin SQL Server 2008 tietokantaa, joka on relaatiotietokanta. Relaatiotietokanta on hyvin yleinen tietokantamalli ja tämän takia osaamalla yhden tietokannan käytön, kuten MySQL osaat muidenkin käytön, joten uutta on vain käyttöliittymä.

1990-luvulla alkoi oliopohjainen ajattelu ja sitä alettiin soveltamaan myös tietokantoihin. Aluksi tehtiin puhtaasti oliopohjaisia tietokantoja, mutta ne eivät saavuttaneet samanlaista suosiota, kuin relaatiotietokannat, joihin on lisätty olio-ominaisuuksia. Nämä ns. oliorelaatiotietokannat helpottavat tiedon käsittelyä tietokannassa, etenkin multimediatiedon käsittely loi uusia haasteita tietokannoille, mutta oliorelaatiotietokannat pystyvät erikoistumaan niiden tallentamiseen. (Lahtonen 2002, 5.) Oliot noudattavat samanlaista jäsenten nimeämistä kuin tietokannan kentät, tämä helpottaa ohjelman ylläpidettävyyttä ja ymmärtämistä. Esimerkiksi tietokannan taulun nimi voi olla henkilö ja sen kenttänä voi olla nimi. Olio-ohjelmointi ajattelussa olioksi tulisi siis henkilö, jolla on sisäinen arvo nimi.

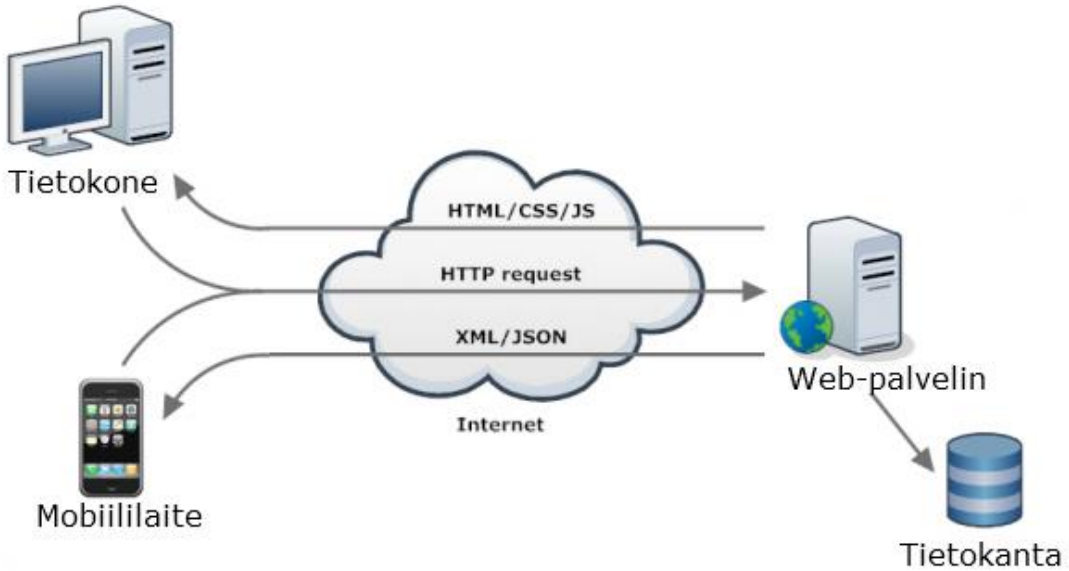
Suurin osa maailman tietokoneiden informaatiosta on tallennettuna tietokantoihin. Tiedon saaminen ja vieminen vaatii dynaamisuutta nettisivuilta. Kaikki haluavat saada tietoa jatkuvasti, joten sivun täytyy muokkaantua tiedon vaihtuessa. Tähän tarkoitukseen on hyvä käyttää palvelua, joka toimii web-palvelimella ja se hoitaa tiedonvälityksen selaimen ja tietokannan välillä. Miinuksena voidaan sanoa, että kuormitusta syntyy enemmän palvelimelle. Monesti luullaan että verkkoyhteydessä on jotain vialla, vaikka ongelma onkin palvelimessa, joka yrittää tuoda tietoa sivulle tietokannasta. (Lahtonen 2002, 164.) Samanlaista toiminallisuutta hyödynnän sovelluksen tiedonvälityksessä ja

jopa eräät toiminnot, kuten push-viestien lähetys tarvitsee tietokannan, josta se saa tarvittavat parametrit viestin lähettämiseen.

3.5 Web-palvelut (Web services)

Nykyään on yleistä tehdä toimintoja muualla kuin laitteessa. Tämä vähentää etenkin laitteelle aiheutuvaa rasitusta sovelluksen toimiessa, mikä tarkoittaa mobiilin akun keston pidentymistä ja vähentää prosessorin käyttöä. Web-palvelut mahdollistavat tiedonkäsittelyn selainpohjalla, jolloin mobiililaitteet voivat hyödyntää näitä ohjelmia kutsumalla niitä. Tämä mahdollistaa esimerkiksi notification (Push-viesti, ilmoitus) toiminnon puhelimessa. Ilmoitukset tulevat web-palvelulta, joka tekee kyselyjä kantaan ja lähettää viestit puhelimeen. Viestit voi lähettää tai vastaanottaa notificationina, jolloin käyttäjä huomaa varmasti uuden tiedon tulleen sovellukseen (kuva 11).

Web-palvelut ovat itsenäisiä modulaarisia, hajautettuja ja dynaamisia sovelluksia, jotka voidaan kuvata, julkaista, sijaita tai kutsua netin välityksellä. Näin saadaan tuotteita, prosesseja ja toimitusketjuja. Web-palvelu voi sijaita paikallisesti, jaetusti tai web-pohjaisesti. Web-palvelut rakennetaan avoimilla standardeilla, kuten TCP/IP, HTTP, Java, HTML ja XML. (Tutorialspoint 2015.)



KUVA 11. Web-palvelimen sijainti verkossa ja sen toiminta (Pietrino Atzeni 2011, mukaillen).

Web-palvelu on ohjelma tai osa sitä, jota on mahdollista käyttää netissä hyödyntämällä standardisoitua XML-kieltä viestittelyyn. XML:llä koodataan kaikki keskustelu web-palvelun kanssa. Tämä mahdollistaa alustariippumattoman käytön, jolloin esimerkiksi Windows-tietokone voi hakea tietoa Unix-koneesta, jossa tietokanta on (Tutorialspoint 2015). Tämä on erittäin käytännöllistä myös mobiiliin, koska silläkin alueella riittää eri alustoja ja selaimia joiden keskinäinen kommunikointi voi olla mahdotonta muuten, kuin web-palvelun kautta.

4 TOTEUTUS

Limab Oy on yritys, joka toimittaa mittausjärjestelmiä saha-, levy- ja terästeollisuuteen. Mittausjärjestelmien toimittamisen ohella Limab Oy huolehtii niiden käyttöönotosta ja ylläpidosta. Limab Oy tarjoaa myös huoltosopimuksia, joiden avulla turvataan asiakkaiden järjestelmien jatkuva optimaalinen sekä häiriötön toiminta tuotantokäytössä (Limab etusivu 2015). Yrityksellä ei siis ollut entuudestaan valmiuksia mobiiliohjelmointiin, vaan he toimivat enemmän tekniikan ja tuotteiden avulla tehtaissa. Heidän mittausjärjestelmänsä ovat herkkiä laitteita, jotka vaativat huoltoa. Niissä voi myös esiintyä yllättäviä ongelmia, kuten ne saattavat mennä lukkoon. Näistä ongelmista heille tulee viesti, jonka avulla he saavat tiedon missä meni mikä pieleen. Viestin saaminen vain on hankala prosessi, jossa henkilö joutuu tarkastamaan häiriöt tietokannasta tai jollain vanhalla ohjelmalla.

Mobiilisovellus on tarkoitettu Limab Oy:n yritysasiakkaille parannukseksi heidän mittausjärjestelmien ylläpitoon ja huoltoon. Sovelluksen tarkoitus on helpottaa järjestelmien häiriön ilmaantumista. Mittari pystyy lähettämään häiriösignaalin, kun siinä ilmenee häiriö. Tämä signaali rekisteröidään tietokantaan. Tähän päättyykin häiriöviestin kulku, huoltohenkilöstön pitää siis tarkistaa häiriö tietokannasta ja vielä selvittää mistä häiriöstä on kyse ja missä. Sovellus helpottaisi huomattavasti häiriön tietoon tulemista, koska jokaisella on nykyään puhelin taskussa. Sovelluksen tarve on juuri häiriön ilmoittaminen selkeästi huoltohenkilölle, jolloin huoltoprosessi nopeutuu ja tarkentuu.

4.1 Suunnitelma

Suunnittelussa aloitimme hahmottelemaan tietokantayhteyden kommunikaatiota ja sovelluksen päätoimintoja. Sovimme, että kehitän sovellusta yksin heidän yrityksessään ja saan tarvittavat kehitysvälineet sekä pääsen käyttämään heidän tietokantaansa, testausympäristön luomisen syissä. Suunnittelu piti tehdä kevyeksi, koska heillä ei ollut tarkkaa määritelmää, mitä he haluavat sovellukselta. Tarve sovellukselle kuitenkin oli ja aikataulu piti sopia yhteen valmistumisen kannalta, joten päätimme aloittaa sovelluksen teolla ja päättää lisää toimintoja sovellukseen sitä mukaa, kuin edelliset valmistuivat.

Tärkeimmiksi ongelmanratkaisu kohdiksi tulivat mobiilin yhdistäminen asiakkaan tietokantaan, johon tulee dataa mittareista. Näitä mittareita haluttaan seurata etenkin, jos niihin ilmenee jotain ilmoitusluontoista. Ongelmat suodatetaan tulevaan tietokantaan erillisellä ohjelmalla ja tästä suodatetusta tiedosta, pitäisi saada viestejä puhelimeen ja niille. Käyttäjän pitää myös pystyä kuittaamaan ilmoituksia, jotta pysytään perillä, kuka hoitaa asiaa. Tästä kävi myös ilmi, että tarvitaan pienimuotoinen käyttäjän hallinta, jotta saadaan eroteltua sovelluksen käyttäjät, koska samaa ilmoitusta voi hoitaa moni henkilö.

Lopullisesta tuotteesta ei oltu varmoja vaan yritys haluaa kartoittaa mahdollisuuksia sovelluksella. Sovelluksen testaus ja myynti tapahtuvat vasta, kun on todettu konsepti mahdolliseksi ja toimivaksi. Tästä on tarkoitukseni tehdä tutkimusta, joka kääntyy toteutukseksi lopuksi. Rajaksi sovittiin, että saisin ohjelman toimimaan käyttöliittymästä vielä sen suurempia välittämättä ja että sovellus oli hyvin dokumentoitu heidän tulevaa testausta varten, koska en itse ole todennäköisesti myynti tai testaus tilanteessa toisella puolen maailmaa. Sovellus on siis samalla tuote, jonka pitää olla mahdollista yrityksen henkilöstön pystyttää ja asentaa.

4.2 Tietokantayhteydet ja Web-palvelin (Web services)

Tietokantayhteyden luominen aloitettiin Limab Oy:n tietokantapalvelimelle, joka on Microsoft SQL Server 2008 R2. Yhteyden luominen tapahtuu PHP-ohjelmointikielellä, johon se soveltuu erinomaisesti nopean ja kevyen suorituksensa ansiosta (kuva 12).

```php
<?php
$server = [redacted];
$user = [redacted]
$pass = [redacted]
$database = "TEST";
try{
$yhteys = new PDO( "sqlsrv:server=$server ; Database = $database", "$user", "$pass");
} catch (Exception $e) {
exit("Yhteyden muodostus ei onnistu!");
}

$json=array();

$yhteys->exec("set names Finnish_Swedish_CI_AS");
$sql="SELECT * FROM tblTesti ORDER BY nimi ASC";
$kysely=$yhteys->prepare($sql);
$kysely->execute();

$tulokset=$kysely->fetchAll(PDO::FETCH_ASSOC);

print json_encode($tulokset);
?>
```

KUVA 12. PHP-ohjelma, joka luo yhteyden ja kyselyn kantaan

Tämä PHP-ohjelma toimii rajapintana käyttöliittymälle ja tietokannalle. Ohjelma toimii web-palvelimella, jotta siihen voidaan ottaa yhteyttä suoraan käyttäjältä. Tällainen kommunikointi tietokantapalvelimen kanssa perustuu palvelukeskeiseen arkkitehtuuriin, jossa juuri avoimella rajapinnalla tai tekniikalla luodaan yhteyttä vanhoihin tietokantajärjestelmiin.

Tietokantaan tehdyssä kyselyssä saadaan vastauksesi kaikki taulussa olevat arvot ja ne muunnetaan JSON-muotoisiksi dataksi, jotta niitä pystytään kutsumaan myöhemmin. (kuva 13) tiedostossa index.js on Ajax kutsu, jossa JSON-muotoinen datamme kutsutaan.

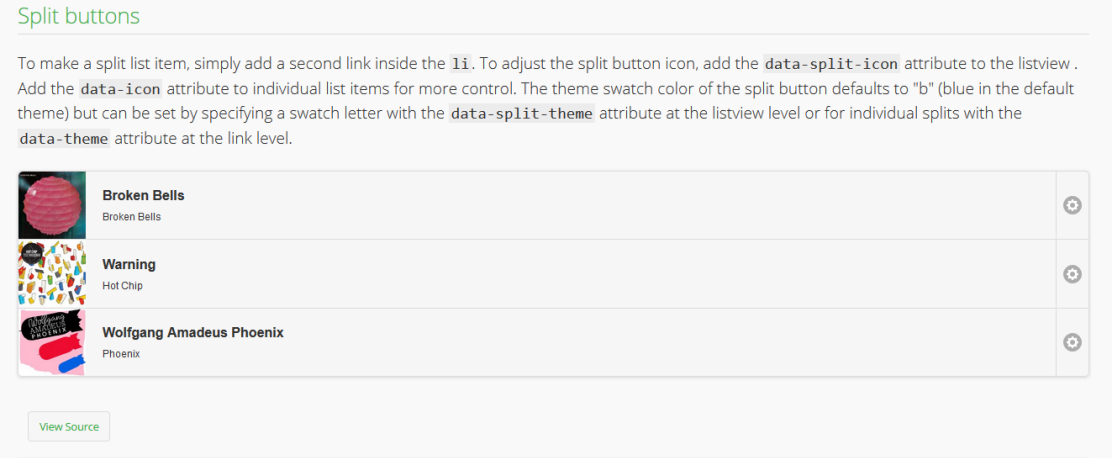
```javascript
    var hal_hinta;
    var hal_tuote;
    var hinta2;
    var tuote2;
    var muokattavan_id;

    function myFunction() {
    $.ajax({
            url: "http://localhost/ws/tuotteet.php",
            dataType: "json",
            success: function(palautettuData) {

                tuotteet = palautettuData;

                var tuote = "";

                for (var i = 0;i < tuotteet.length;i++) {

tuote += "<li id="+ i +"><a href='#'><h3>"+ tuotteet[i].nimi + "</h3><p>" + tuotteet[i].hinta +"</p></li>";

                }

                $("#lista01").html(tuote);
                $("#lista01").listview("refresh");

            },
            error: function () {
                alert("AJAX-kutsu epäonnistui");
            }
        });
    };
```

KUVA 13. Ajax-kutsu, joka hakee kannasta dataa ja tulostaa ne käyttöliittymälle

Kutsulle annetaan osoite (URL) ja datatyyppi. Kun kutsu tuottaa onnistuneen (success) haun, voidaan data ottaa haltuun funktion parametrina. Tiedot ovat json-tyyppisessä taulukossa, joka voidaan käydä läpi *for*-lauseella (looppi). Näin data saadaan eriteltyä ja tulostettua haluamallaan tavalla.

4.3 HTML ja jQuery Mobile

Sivun ulkoasu luodaan jQuery Mobile -sovelluskehystä käyttämällä. jQuery Mobile tarjoaa laajan ja helposti käytettävän dokumentaation, mobiilisovelluksen ulkoasun tekemiseen. Projekti tarvitsi selkeän tavan ilmaista tietokannasta tulevaa tietoa ja tietoa pitäisi myös päästä muokkaamaan, joten päädyin *split buttons* -elementtiin (kuva 14).



KUVA 14. jQuery Mobile-demo dokumentaatio

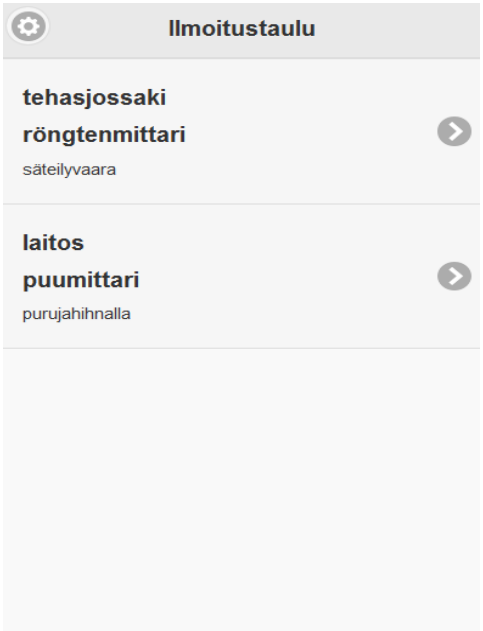
Sivuilta löytyy demo-osio, josta saamme valmiita käyttöliittymäelementtejä, joita voimme hyödyntää sovelluksen ulkoasun luomiseen. Elementtejä voidaan sitten muokata omaan tarkoitukseen haluamallamme tavalla. Näin saadaan nopeammin runko käyttöliittymälle.

Elementti otetaan käyttöön antamalla sille ominaisuuksia, kuten *data-role*, *data-split-icon* ja *data-split theme* (kuva 15). Näille ominaisuuksille annettu arvo muokkaa listanäkymää (ListView), kuten tässä tapauksessa olen jättänyt kokonaan kuvat pois, koska halusimme vain päällekkäin menevät tekstit ja ikonin tekstien vierelle jotta käyttäjä tietää, että elementtiä painamalla (tap) pääsee siitä tekemään toimintoja.

```
<!DOCTYPE html>
<html>
    <head>
        <title>Limab sovellus</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">

        <link rel="stylesheet" href="css/jquery.mobile-1.4.5.min.css">
        <link rel="stylesheet" href="css/index.css">

        <script src="cordova.js"></script>
        <script src="js/jquery.js"></script>
        <script src="js/jquery.mobile-1.4.5.min.js"></script>
        <script src="js/index.js"></script>
    </head>
    <body>
            <!-- Sivu 1 -->

        <div data-role="page" id="sivu1">
            <div data-role="header" data-position="fixed">
                <h1>Ilmoitustaulu</h1>
            </div>
            <div data-role="main" class="ui-content">

                <ul  data-role="listview" id="lista01" data-split-icon="gear" data-split-theme="d">
                </ul>

                <div id="ilmoitus"></div>

            </div>
            <div data-role="footer" data-position="fixed">
            </div>
        </div>

```

KUVA 15. Listanäkymän käyttö index.html-tiedostossa

Sovelluksen käyttöliittymän rakentaminen tapahtuu nopeasti ja selkeästi jQuery Mobilella. Sen avulla toteutamme sovelluksen käyttöliittymän ja se helpottaa myös joitakin Ajax-kutsujen tekemistä (kuva 13). Kuvassa 16 on mobiilissa näkyvä lista, johon tiedot tulevat.



KUVA 16. Esimerkkikuva ilmoitustaulusta mobiilissa.

Näitä voi listassa valita ja kuitata. Listaan tulevat tiedot päivitetään käyttäjälle aina kuittauksen jälkeen. Listaa päivitetään myös ohjelmasta 10 sekunnin välein. Joten tiedot muuttuvat, jos joku muu on kuitannut ilmoituksen.

4.4 Sovelluksen toiminnot

Sovelluksen päätoiminnot ovat käyttäjän rekisteröiminen, ilmoitukset ja kuittaus. Nämä kolme toimintoa tarvitaan että sovellus täyttää tarpeensa. Kuittaus sitä varten, että mittarin viasta joku on saanut tiedon ja hoitaa asian. Ilmoitus eli *notification* antavat käyttäjälle tiedon vaikka sovellus olisi pois päältä. Mittarista lähetettävän viestin on tultava käyttäjälleen tiedoksi mahdollisimman selkeästi ja huomattavasti. Käyttäjän rekisteröiminen mahdollistaa kuittauksen ja ilmoituksen yksilöinnin, koska mittareita ja henkilöitä jotka käyttävät tai huoltavat niitä voi olla monia. Myös asiakkaat voivat olla eri firmoista, joten tarvitaan erittelyä kenelle lähetetään miltäkin laitokselta ja mikä viesti. Kuvassa 17 näkyy käyttäjä, joka on oman yhtiönsä taulussa, jotta viestit saadaan yksilöityä yritysten puolesta. *DeviceRegID* on myös yksilöllinen, joten voidaan lähettää viestejä vain halutuille käyttäjille.

	DeviceID	Enabled	User	DeviceName	DeviceRegID
	55	True	Henrik Tyrväinen	GT-I8190	APA91bElMkOcBfZ3XgzEC--_nqKNPsaqaJ-W9oujfIB-BfVCbCZzb1anGzAlF47xDkmD1ci-9CkSAdtpR...
▸*	NULL	NULL	NULL	NULL	NULL

KUVA 17. Esimerkki käyttäjästä tietokannassa.

Kun käyttäjä ensimmäistä kertaa käynnistää mobiililaitteessaan sovelluksen, kysytään hänen nimensä, kuten kuvassa 18. Nimi tallennetaan kantaan ja samalla otetaan talteen myös *register identification* (*DeviceRegID*, tunnisteavain). Tämän avulla voidaan lähettää yksilöityjä viestejä käyttäjille.


```
$("#kirjaudu").on("tap",function(){
    etunimi = $("#etunimi").val();
    sukunimi = $("#sukunimi").val();
    pushReg();
    window.location.href = "#sivu1";
    Listaus();
    //alert(etunimi);
});

// pushReg funktio lahettaa testicon.phpeelle, joka on  web-palvelimella, puhelimen rekisteri ID
// testicon tallentaa sen kantaan, jos siellä ei ole sitä. Tämä mahdollistaa
// pushviestit
function pushReg() {
    devicemodel = device.model;
    kayttaja = etunimi+" "+sukunimi;
$.ajax({
        url: url+"/testicon.php?\n\
        regID="+regID+
        "&devicemodel="+devicemodel+
        "&kayttaja="+kayttaja,
        dataType: "POST",
        success: function() {
            //alert("Push onnistui");
        },
        error: function (e) {
            //alert("Jokin meni pieleen Pushviestin hakemisessa"+ JSON.stringify(e));
        }

    });
}
```

KUVA 18. Kirjautumis-näkymä sovellukseen mobiilissa ja sen koodi.

Sovelluksen lista-näkymälle rakennetaan suurin osa toiminnoista. Listaan tulevia tietoja on käyttäjän tarkoitus kuitata. Listaan tulevat tiedot haetaan kannasta, kuten kuvassa 19 näkyy. Funktio kutsuu itseänsä 10 sekunnin välein, jolloin sisällä oleva Ajax-kutsu hakee web-palvelun avulla dataa kannasta ja näin saadaan aina päivitettyä lista. Dataa tutkitaan *if*-lauseella, jos siellä esiintyy haluamamme tiedonpätkä, niin päivitämme listan. Näin listalla saadaan näkymään aina uusin tieto kannasta.

MessageID	PlantText	MeasuringD...	Message	SendMessage	DeviceIDTo...	NoticeTime	NoticeUser
1	tehasjossaki	röngtenmittari	säteilyvaara	True	0	2015-10-15 10:32:21.000	Henrik Tyrväinen
2	laitos	puumittari	purujahihnalla	True	0	2015-10-15 10:33:44.000	Henrik Tyrväinen
3	sahala	sirkkelimittari	terä löysänä	True	0	2015-10-15 10:33:41.000	Henrik Tyrväinen
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

KUVA 19. Esimerkkikuva tietokannan viestitaulusta.

Kun sovellus käynnistetään ensimmäisen kerran, otetaan talteen käyttäjän nimi, laitetietoja ja rekisterinumero. Etenkin rekisterinumero on tärkeä, koska sitä tarvitaan *push*(ilmoitus, *notification*) viestien lähettämiseen. Puhelimen rekisterinumero otetaan talteen joka kerta, kun sovellus käynnistetään (kuva 20). Numeroa ei kuitenkaan tallenneta joka kerta, vaan vertaillaan kannassa oleviin. Jos sellainen on jo kannassa, ei sitä tallenneta. Näin tehdään siksi, koska sovellusversion muutokset tai Googlen palvelu saattaa välillä vaihtaa rekisterinumeroa.

```
52      // PushNotification lähettää Googlelle viestin, jossa on senderID
53      // senderID tulee gcm(Google Cloud Messaging) palvelusta, joka saadaan,
54      // kun luodaan projekti, heidän palveluunsa ja aktivoidaan gcm.
55      var push = PushNotification.init({ "android": {"senderID": "842297115580"},
56          "ios": {"alert": "true", "badge": "true", "sound": "true"}, "windows": {} } );
57
58      // push.on palauttaa meille funktion parametrinä data, registrationId, jota
59      // tarvitsee, kun lähetämme puhelimeen push viestejä.
60      push.on('registration', function(data) {
61          regID = data.registrationId;
62          onkoUusiKayttaja();
63          //alert(regID);
64          //pushReg();
65          //console.log(regID);
66      });
67      // käsittelee push viestit
68      push.on('notification', function(data) {
69          data.message,
70          data.title,
71          data.count,
72          data.sound = true,
73          data.image;
74      });
75
76      push.on('error', function(e) {
77          console.log("VIRHE "+e.message);
78      });
```

KUVA 20. Googlen cloud messaging palveluun liittyminen koodissa.

Ilmoituksen lähetys tapahtuu web-palvelussa. Palvelu hakee käyttäjät tietokannasta, jonka avulla se tietää kenelle ilmoitukset lähetetään. Palvelu käynnistetään kerran, jolloin sen sisällä oleva *while*-looppi pyörii ikuisesti, koska sen arvo on aina true. Tämä mahdollistaa tietokannan jatkuvan kyselyn ja siitä voidaan suodattaa aina uusin viesti.

Kun kantaan tulee uusi tieto, tehdään siitä GCM (Google Cloud Messaging) parametrien mukainen viesti, joka lähetetään heidän palveluunsa (kuva 21). Palvelu lähettää viestin puhelimeen, jolloin ei ole väliä onko sovellus päällä vai ei. Kantaan on tallentunut *DeviceRegID*, jota Googlen GCM-palvelu käyttää, ikään kuin puhelinnumerona lähettäessään viestin puhelimeen. *DeviceRegID* on Googlen palvelusta saatava rekisterinumero laitteelle ja se on yksilöllinen.

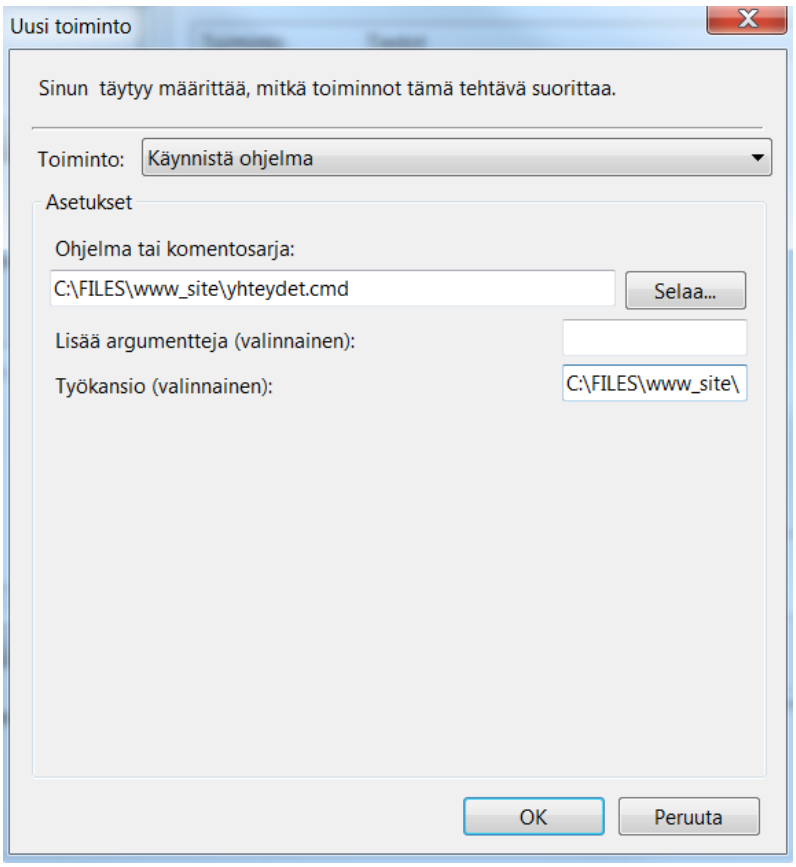
Ohjelma hakee kannasta kaikki viestit 10 sekunnin välein ja tekee kuvassa 21 erottelun viesteille. Erottelussa eli *if*-lauseissa saadaan kannasta aina uusin viesti eroteltua, joka sitten lähetetään Googlen palvelulle PHP:n *curl*-lähetys protokollan avulla. Parametrina annetaan viestin sisältö, otsikko ja rekisterinumero.

```
37      if ($kaikkiviestit == $vanhatviestit) {
38      //Ei ole tullut uusia viestejä
39      $uudetviestit = $vanhatviestit;
40      }
41  else if($uudetviestit != null && $kaikkiviestit > $uudetviestit) {
42      //asetetaan uudet viestit vanhoiksi
43      $vanhatviestit = $kaikkiviestit;
44      //Lähetetään push viesti
45      $viesti = array
46      (
47          'message'   => 'Uusia ilmoituksia on tullut',
48          'title'     => 'Ilmoitus',
49
50      );
51      $kentat = array
52      (
53          'registration_ids'  => $registrationIds,
54          'data'          => $viesti
55      );
56
57      $otsikot = array
58      (
59          'Authorization: key=' . API_ACCESS_KEY,
60          'Content-Type: application/json'
61      );
62      $lahetys = curl_init();
63      curl_setopt( $lahetys,CURLOPT_URL, 'https://android.googleapis.com/gcm/send' );
64      curl_setopt( $lahetys,CURLOPT_POST, true );
65      curl_setopt( $lahetys,CURLOPT_HTTPHEADER, $otsikot );
66      curl_setopt( $lahetys,CURLOPT_RETURNTRANSFER, true );
67      curl_setopt( $lahetys,CURLOPT_SSL_VERIFYPEER, false );
68      curl_setopt( $lahetys,CURLOPT_POSTFIELDS, json_encode( $kentat ) );
69      $tarkastus = curl_exec($lahetys );
70      curl_close( $lahetys );
71      }
72  else {
73      // uusien viestien asetus
74      $vanhatviestit = $kaikkiviestit;
75      }
```

KUVA 21. Web-palvelu, joka hoitaa mobiilin *notification* viestit.

Ongelma ohjelmassa on se, että sen tulisi aina olla päällä. Tuotantokäytössä sovelluksen täytyy ilmoittaa käyttäjälleen mittarin tulleesta viasta 24/7. Ongelmasta kiperän teki se, että palvelun lopullista sijoituspaikkaa ei ollut tiedossa, mutta löysimme ratkaisun matkalla joihinkin web-palvelimiin, esimerkiksi Microsoftin Iis(Internet information services) web-palvelin tekniikkaan.

Ohjelma joka pyörittää toista ohjelmaa säännöllisin väliajoin, kutsutaan cronjobeiksi. Cronjob tulee Unix-käyttöliittymästä, koska yleensä palvelinpuolta hallinnoidaan sillä. Cronjob-ohjelmalle annetaan tiedot mitä ohjelmaa pitää käynnistää, millä asetuksilla ja kuinka usein. Windowsilla tämä toiminto hoidetaan *task schedulerilla* (Ajoitetut tehtävät) (kuva 22).



KUVA 22. Windowsin *task scheduler*-toiminnon käyttäminen.

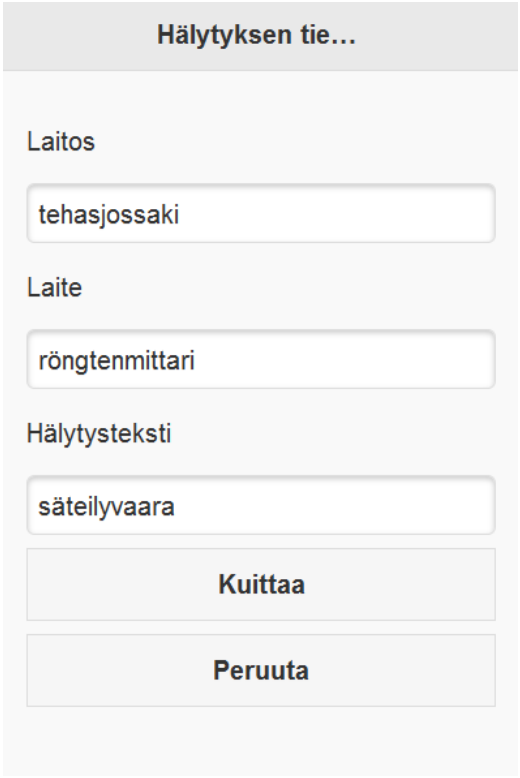
Kuvassa 22 näkyy toiminnon asentaminen, joka vaaditaan aina. Tässä annetaan tiedot toiminnolle. Parametrina annettu tiedoston sijainti ja sen kansio vaaditaan. Muut asetukset kuten käynnistin, asetetaan viiden minuutin välein, koska ohjelmamme pyörii 300 sekuntia, ennen kun se pysähtyy. Yhteydet.cmd-tiedosto näkyy kuvassa 23, tässä käynnistetään PHP-ohjelma web-palvelimella. Tämän komentorivikehotteen avulla Windows-käyttöjärjestelmä osaa avata oikean tiedoston, kun PHP on lisätty ympäristömuuttujiin PATH-muuttujaksi, voidaan ohjelma käynnistää komentoriviltä. Näin ohjelmalle ei tule katkoja viestien lähetyksessä vaan saadaan ylläpidettyä 10 sekunnin uusien viestien tarkastus tahtia tietokannasta.

```
1 start C:\FILES\Downloads\Mobiili\PHP\php-5.5.30-nts-Win32-VC11-x86\php.exe -f C:\FILES\www_site\Kuhmo\longpolling.php
2 start C:\FILES\Downloads\Mobiili\PHP\php-5.5.30-nts-Win32-VC11-x86\php.exe -f C:\FILES\www_site\Verso\longpolling.php
```

KUVA 23. Yhteydet.cmd-tiedostossa näkyvät komentorivi käskyt.

Kuittaus suoritetaan listassa klikkaamalla ilmoitusta jolloin päästään katsomaan tarkempia tietoja ilmoituksesta (kuva 24). Kuittauksessa tallennetaan kantaan viestin viereen aika sekä nimi, kuten kuvassa 19 näkyi. Nimi on se, jonka käyttäjän ensimmäisellä käynnistyksellään on laittanut, jolloin se tallennettiin tietokantaan web-palvelun avulla.

Kuittaus palauttaa käyttäjän takaisin ilmoitustaulu-näkymään, ja päivittää ruudun samalla.



KUVA 24. Kun ilmoitusta on klikattu päästään ikkunaan, jossa on tarkemmat tiedot ilmoituksesta sekä kuittaus.

Kuittauksesta haluttiin yksinkertaista ja nopeaa. Tässä onnistuttiin Ajaxin asynkronisella lähetystavalla, joka mahdollistaa koodin ajamisen taustalla. Web-palvelussa hoidetaan kuittausajan lisääminen, jotta sovelluksen ei tarvitse tehdä sitä itse. Mobiilit laitteet ovat huomattavasti hitaampia laskemaan, kuin pöytäkoneet, joten kaikki laskeminen kannattaa ulkoistaa web-palvelimelle, jos vain mahdollista.

5 PÄÄTÄNTÖ

Mobiilisovelluksen kehittäminen oli erittäin mielenkiintoista ja antoisaa. Projektissa kohdattiin aivan uusia haasteita ja tehtäviä koulun tehtäviin verrattuna. Nautin kuitenkin tehdä työtä toimeksiantajalleni ja etenkin työympäristöstä olen hyvin kiitollinen. Sovelluksesta saatiin tehtyä valmis prototyyppi, jolle voidaan saada käyttäjiä, joilta voidaan

laskuttaa sovelluksen käytöstä. Sovellus skaalautuu käyttäjiin nähden. Jakelu suoritetaan antamalla asiakkaalle ohjelman asennusversio, jonka Cordova kääntää esimerkiksi Androidilla *apk.*

Sovellusta testattiin toimeksiantajan tarjoamassa suljetussa verkkoympäristössä, jossa simuloitiin oikeaa ympäristöä. Joten lähetimme tietokantaan viestejä, jotka web-palvelu luki tietokannasta ja lähetti viestin mobiilisovellukselle joka otti ne vastaan. Näitä viestejä pystyttiin kuittaamaan laitteella, jolloin tietokantaan ilmestyi kuittaajan nimi ja aika. Sovellus suorittaa sille asetetut vaatimukset, johon olimme tyytyväisiä toimeksiantajan kanssa.

Eniten haasteita tuotti uusien ympäristöjen oppiminen, etenkin oman web-palvelun tekeminen mobiilisovellukselle oli suuritöinen ja myös kaikkein kriittisin. Jotkin web-palvelut toimivat Apachen web-palvelinympäristössä, mutta eivät Microsoftin Iis web-palvelinympäristössä. Nämä kaikki ongelmat selvisivät kuitenkin syventymällä tutkimiseen ja loogiseen päättelyyn.

Sovelluksen tulevaisuus riippuu sen saamasta palautteesta. Sovellusta ei päästy minun työssäoloaikana testaamaan oikeassa ympäristössä. Toimeksiantaja aikoo testata sitä asiakkaalla, jos tuotteesta eli sovelluksesta kiinnostutaan, jolloin sitä saadaan myytyä. Sovellukseen voidaan kehittää lisää toimintoja perustuen mittarilta tulevaan dataan. Suunnitelmissa oli tehdä seurantasovellus, joka näyttäisi reaaliaikaisesti mittarilta tulevaa dataa esimerkiksi kaaviona. Tästä käyttäjä voisi itse päätellä, onko kaavion näyttämät arvot oikein ja nähdä pitkäaikaisesta seurannasta, että onko esimerkiksi tahti hidastunut jollain hihnalla tai tukin laatu huonontunut. Näistä uusista lisäosista saadaan kehittämiseen liittyvät kustannukset pois, jos voidaan myydä tekemäämme ensimmäistä sovellusta, niin että se vakuuttaa asiakkaan tällaisen sovelluksen tarpeellisuudesta. Toisin sanoen sovellus voisi olla osa uutta tuoteperhettä, johon kuuluu sovelluksia ja ohjelmia, joilla on jokin käyttötarkoitus tehdastuotannon mobiilisoimisessa ja sen globaalissa hallitsemisessa.

Ensimmäisenä päivänä kun saavuin Limab Oy:lle, en tiennyt ehdotetusta opinnäytetyöaiheesta muuta, kuin että sillä oli jotain tekemistä mobiililaitteiden kanssa. Päätin tarttua tilaisuuteen, koska syksyni olisi muuten ollut tyhjää täynnä ja aihe oli mielenkiintoinen.

Olin tykännyt koulun mobiiliohjelmointikurssista ja olin pärjännyt siinä opettajan mielestä kolmosen arvoisesti. Ensivaikutelma saattoi jäädä huonoksi, koska en hallinnut teoriaa ja käsitteitä. Ymmärsin kyllä, mitä pitää tutkia ja tehdä, mutta en saanut sanotuksi asioita, joita olisin halunnut sanoa. Joka tapauksessa työn aloittaminen oli pelottavaa, koska ensimmäiset päivät piti olla muiden koneilla, joihin ei saanut asentaa tarvittavia ohjelmia, joten päädyin yksin nurkkaan tutkimaan laitteita ja tekniikoita. Yllättävää oli miten nopeasti pääsin kärryille asiasta ja työ alkoi rakentumaan jo ensimmäisestä viikosta lähtien. Toinen viikko alkoi samoin kuin seuraavat 10 viikkoa sen jälkeen. Puurtamista ja ongelmanratkaisua tehtiin 8 tuntia päivässä arkisin ja välillä enemmänkin, mutta se ei ollut raskasta vaan jostain syystä mukavaa ja helppoa.

Pelkäsin opinnäytetyön tekemistä mutta nyt työn tehneenä haluaisin sanoa samaa poteville, että sen aloittaminen ei ole pahin kohta vaan se keskiväli. Haluan vielä kiittää Janne Turusta, opinnäytetyön ohjaajaani, hyvistä neuvoista ja vinkeistä sekä motivoinnista. Myös Jouni Turusta, joka toimi työkaverina ja auttoi tutkimaan asioita sekä opetti lisää alasta, myös toimeksiantajaa aiheesta ja mahdollisuudesta päästä heille tekemään työtä heidän välineillään.

LÄHTEET

Apache Cordova. 2015. Apache Cordova. WWW-dokumentti. http:// apache.cordova.com/. Ei päivitystietoja. Luettu 2.9.2015.

Etusivu. 2015. Limab Oy. WWW-dokumentti. http://www.limab.fi/fi/etusivu. Ei päivitystietoja. Luettu 16.9.2015.

FAQ PhoneGap. 2015. PhoneGap. WWW-dokumentti. http://phonegap.com/about/faq/. Ei päivitystietoja. Luettu 16.9.2015.

PhoneGap, Cordova, and what's in a name?. 2012. PhoneGap Blog. Blogi. http://phonegap.com/2012/03/19/phonegap-cordova-and-what%E2%80%99s-in-a-name/. Päivitetty 19.3.2012. Luettu 16.9.2015.

The Last Word on Cordova and PhoneGap. 2014. Ionic Blog. Blogi http://blog.ionic.io/what-is-cordova-phonegap/. Päivitetty 6. 3.2014. Luettu 16.9.2015.

JQuery Mobile About. 2015. jQuery Mobile. WWW-dokumentti. http://jquerymobile.com/about/. Ei päivitystietoja. Luettu 22.9.2015.

Ajax. 2015. jQuery. WWW-dokumentti. http://learn.jquery.com/ajax/. 11.3.2015. Luettu 22.9.2015.

Ajax programming. 2015. evolve inc. WWW-dokumentti. http://www.evolve-incorporated.com/services/ajax-programming/. Ei päivitystietoja. Luettu 28.9.2015.

Introduction to DOM manipulation. 2015. jQuery tutorial. WWW-dokumentti. http://www.jquery-tutorial.net/dom-manipulation/introduction/. Ei päivitystietoja. Luettu 28.9.2015.

Category: Manipulation. 2015. jQuery. WWW-dokumentti. https://api.jquery.com/category/manipulation/. Ei päivitystietoja. Luettu 28.9.2015.

Philippe Le Hégaret. 2002. The W3C Document Object Model (DOM). WWW-dokumentti. http://www.w3.org/2002/07/26-dom-article.html. 26.07.2002. Luettu 29.9.2015.

MDN. 2015. Introduction to the DOM. WWW-dokumentti. https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction. Ei päivitystietoja. Luettu 29.9.2015.

Tommi Ristola. 2014. Markkinakatsaus Q1/2014 – Mobiilikäyttöjärjestelmät. Blogi. https://ristola.wordpress.com/2014/03/27/markkinakatsaus-q12014-mobiilikayttojarjestelmat/. 27.3.2014. Luettu 27.3.2014.

Introduction to Android. 2015. Android Developers. WWW-dokumentti. http://developer.android.com/guide/index.html. Ei päivitystietoja. Luettu 29.9.2015.

The Android Source Code. 2015. Android. WWW-dokumentti. https://source.android.com/source/index.html#governance-philosophy. Ei päivitystietoja. Luettu 13.10.2015.

iOS 9. 2015. Apple. WWW-dokumentti. http://www.apple.com/ios/what-is/. Ei päivitystietoja. Luettu 13.10.2015.

Developing for iOS 9. 2015. Apple Developer. WWW-dokumentti. https://developer.apple.com/ios/. Ei päivitystietoja. Luettu 13.10.2015.

Lahtonen, Tommi 2002. SQL. Jyväskylä: Docendo.

REST webservice with symfony. 2011. Pietrino Atzeni. Blogi. http://di-side.com/di-side/services/web-solutions/rest-webservice-symfony/. 2.7.2011. Luettu 21.10.2015.

Top 8 Mobile Operating Systems from July to Sept 2015. 2015 StatCounter Global Stats. WWW-dokumentti. http://gs.statcounter.com/#mobile_os-ww-monthly-201507-201509-bar. Ei päivitystietoja. Luettu 21.10.2015.

Top 8 Mobile Operating Systems in Finland from July to Sept 2015. 2015 StatCounter Global Stats. WWW- dokumentti. http://gs.statcounter.com/#mobile_os-FI-monthly-201507-201509-bar. Ei päivitystietoja. Luettu 21.10.2015.

What are Web Services. 2015. Tutorialspoint. WWW-dokumentti. http://www.tutorialspoint.com/webservices/what_are_web_services.htm. Ei päivitystietoja. Luettu 21.10.2015.