Maxim Soila

PHONEGAP PUHELINSOVELLUS RIISTATIETOJEN KERÄÄMISEEN

Opinnäytetyö
Tietotekniikan koulutusohjelma

Toukokuu 2014

KUVAILULEHTI

MAMK University of Applied Sciences	**Opinnäytetyön päivämäärä** 31.5.2014
Tekijä(t) Maxim Soila	**Koulutusohjelma ja suuntautuminen** Tietotekniikan koulutusohjelma

Nimeke

Phonegap puhelinsovellus riistatietojen keräämiseen

Tiivistelmä

Opinnäytetyön tavoitteena oli luoda toimiva älypuhelinsovellus riistatietojen keräämiseen. Sovelluksen tuli kerätä käyttäjän valitsemat tiedot riistatapahtumasta ja lähettää ne palvelimelle. Opinnäytetyössä tutkittiin alustariippumatonta sovelluskehitystä älypuhelin alustoille PhoneGapin avulla. Työstä tehtiin toimiva esittely versio riistatietojen keruusta, Oravannahka Oy:lle.

Tavoitteena oli tuottaa yksi sovellus, joka toimisi kolmella eri älypuhelinkäyttöjärjestelmällä: Windows Phonella, Androidilla ja iOS:llä. Työ tehtiin PhoneGap sovelluskehyksen ympärille. Puhelinohjelman toiminnallisuus toteutettiin käyttäen Javascriptiä ja käyttöliittymä toteutettiin käyttäen HTML:iä ja CSS:iä.

Sovelluksen tarkoitus on ilmoittaa riistatietoja koskevasta tapahtumasta Suomen Riistakeskukselle. Tapahtuman yhteydessä valitaan tapahtumatyyppi ja eläin jota tapahtuma koskee. Sovellus paikantaa käyttäjän käyttäen GPS-paikannusjärjestelmää. Paikannustiedot ja käyttäjän syöttämät tiedot lähetetään palvelimelle, joka tallentaa tiedot muistiin. Käyttöliittymä tehtiin mahdollisimman intuitiiviseksi ja helposti ymmärrettäväksi. Sovelluksen toimivuus testattiin kaikilla vaadituilla älypuhelinkäyttöjärjestelmillä. Lopputulos täytti odotukset, toimiva sovellus saatiin aikaiseksi.

Asiasanat (avainsanat)

Tietotekniikka, IT, ohjelmointi, Javascript, Phonegap, älypuhelin, sovellus, alustariippumaton

Sivumäärä	**Kieli**	**URN**
30	Suomi	

Huomautus (huomautukset liitteistä)

Ohjaavan opettajan nimi	**Opinnäytetyön toimeksiantaja**
Jukka Selin	Janne-Pekka Surakka, Oravannahka Oy

DESCRIPTION

MAMK University of Applied Sciences	**Date of the bachelor's thesis** 31.5.2014
Author(s) Maxim Soila	**Degree programme and option** Information Technology

Name of the bachelor's thesis

PhoneGap smartphone application for collecting wildlife information

Abstract

The aim of the thesis was to create a working smartphone application. Application collects wildlife data and sends it to the Finnish Wildlife Agency's server. The thesis examined PhoneGap's usage for smartphone cross-platform application development. The work was done for Oravannahka Oy, and the main goal was to make a working demo for the company.

The aim was to make the application work for all the major smartphone operating system. Specifically the three most prelevant, which are Windows Phone, Android and iOS. Application was built around PhoneGap's framework. The functionality for the application was built with Javascript and the user interface was built with HTML and CSS.

The main idea for the application is to report wildlife occurrences to the Finnish Wildlife Agency. User inputs data for the type occurrence that is happening and the animal that it concerns. Application uses GPS-tracking system. Tracking information and the inputted data are sent to a server, which saves them. The interface was made to be intuitive and easy to understand. The application was tested for all of the three main operating systems. The end result fulfilled expectations and a working application was delivered to the client.

Subject headings (keywords)

IT, cross-platform, PhoneGap, app, application, Javascript, programming, smartphone

Pages	**Language**	**URN**
30	Finnish	

Remarks, notes on appendices

Tutor	**Bachelor's thesis assigned by**
Jukka Selin	Janne-Pekka Surakka, Oravannahka Oy

SISÄLTÖ

1 JOHDANTO 1

2 ALUSTARIIPPUMATON SOVELLUSKEHITYS 2

2.1 Verkkoselaimen hyödyntäminen 3

2.2 Palvelintekniikoiden hyödyntäminen 5

3 PHONEGAP 6

3.1 Arkkitehtuuri 6

3.2 Ohjelmointirajapinnat 9

4 RIISTATIEDON KERUUSOVELLUS 17

4.1 Sovelluksen vaatimukset 17

4.2 Arkkitehtuuri 18

4.3 Teknisen toteutustyylin ja kehitysympäristön valinta 22

4.4 Kehitys 23

5 PÄÄTELMÄT 28

LIITE/LIITTEET

1 Yksisivuinen liite
2 Monisivuinen liite

LYHENTEET

OS: Operating System, Käyttöjärjestelmä

HTML: Hyper Text Markup Language, Ohjelmointi kieli verkkosivuille.

CSS: Cascading Style Sheets, Muotoilukieli verkkosivuille.

PHP: Hypertext Preprocessor, Ohjelmointikieli verkkosivujen toiminnallisuuksien tekoon.

RSS: Really Simple Syndication, Verkkosyötemuoto päivittyvälle digitaaliselle sisällölle.

UDID: Unique Device Identifier, Avoin tunnistestandardi.

API: Application Programming Interface, Ohjelmointirajapinta.

SDK: Software Development Kit, Tyypillisesti paketti sovelluksen kehittäjälle, joka koostuu erilaisista työkaluista.

SSL: Secure Sockets Layer, Salausprotokolla, jolla suojataan Internet-sovellusten tietoliikennettä.

1 JOHDANTO

Ohjelmien kehitys on muuttunut radikaalisti viimeisen kymmenen vuoden aikana. Varsinkin mobiilialustojen suosio on tuonut mahdollisuuden pienemmille, muutaman henkilön yrityksille tehdä nimeä itselleen koodaten yksinkertaisia sovelluksia. Enää ei ole pakko kuulua suuriin sovellustaloihin, jotka tuottavat massiivisia projekteja. Voi haalia ympärilleen osaavia tekijöitä ja perustaa esimerkiksi koulun ohella kavereiden kanssa projektin, josta voi muovautua myöhemmin oma tuleva työpaikka. Toisaalta tämä mobiilialustojen suosio on tuonut mukanaan uudenlaisen ongelman. Mihin alustaan paneutua? Kaikkien tulee tuntea alusta hyvin, ennen kuin mitään kunnollista voi alkaa tekemään. Entä jos alusta epäonnistuu? Entä jos sovellus olisikin myynyt paremmin toisella alustalla? Mitä jos alustan omistava yhtiö määrää uuden politiikan sovellusten kehitykseen, ja yhtäkkiä sovellus ei sovikaan enää kyseiselle alustalle? Alustariippumaton ohjelmointikehitys on kehitetty ratkaisemaan edellä mainittuja ongelmia ja tarjoamaan yhteistä ohjelmointikieltä kaikille alustoille.

Suomen Riistakeskus hoitaa ja edistää kestävää riistataloutta Suomessa. Heidän tehtäviinsä kuuluu riistaeläinkantojen tilan, kehityksen, kestävyyden ja elinvoimaisuuden seuraaminen. Riistakeskus on alkanut tarjoamaan palvelujaan sähköisesti internetin välityksellä. Kokeilut ovat olleet onnistuneita, kuten metsäkauris- ja ilvessaaliin ilmoittaminen sähköisesti riistakeskuksen verkkopalvelussa (Suomen Riistakeskus 2014). Suurempia järjestelmiä on alettu siirtämään verkkoon ja tulevaisuudessa toivotaan, että kaikki asiat voitaisiin hoitaa verkkopalvelujen ja puhelinsovellusten kautta. Riistatiedon kerääminen on metsästysseurojen vastuulla. Se tehdään normaalisti käsin täyttämällä lomake ja postittamalla se Riistakeskukselle. Yksinkertainen sovellus riistatietojen keräämiseksi vähentäisi työmäärää metsästysseuroissa ja Riistakeskuksessa.

Sijoitusyhtiö Oravannahka sijoittaa päätoimisesti pääomaa ja henkilöstöapua yrityksille, mutta sillä on myös vahvat juuret tietotekniikassa, sillä se toimittaa erilaisia ohjelmistoprojekteja ja IT-asiantuntijuutta yrityksille. Oravannahka lupasi auttaa kehityksessä visioimalla mahdollisen tulevan mobiilisovelluksen Suomen Riistakeskukselle. Kehitysversion tulisi toimia kolmella suurimmalla älypuhelinalustalla, ja sen pitäisi integroitua riistakeskuksen tulevaan verkkopalvelujärjestelmään. Sovellukseen kirjauduttaisiin käyttäjätunnuksella. Kirjautumisen jälkeen käyttäjien tulisi voida ilmoittaa saalis-, kaato- tai havainto-tapahtumasta mahdollisimman yksinkertaisesti ja lähettää

tapahtuman tiedot paikannustietojen mukana Riistakeskuksen järjestelmään. Sen jälkeen niitä pystyisi vielä kotona muokkaamaan.

2 ALUSTARIIPPUMATON SOVELLUSKEHITYS

Yleisesti markkinoille on vakiintunut kolme suurta mobiilialustaa: Windows Phone, iOS ja Android. Muita pienempiä tai pikku hiljaa häviäviä alustoja ovat mm. Symbian OS, Bada ja Blackberry OS (Gartner 2014). Mobiilialustalla tarkoitan puhelimen käyttöjärjestelmäympäristöä, johon kuuluu kaikki sen tukemat käyttöjärjestelmän versiot. Kolme suurinta alustaa ovat käytännössä kaapanneet koko markkinan itselleen, ja häviäville tai pienimmille alustoille sovelluksen kehitystä ei ole järkeä aloittaa. Näille kolmelle alustalle sovelluksen kehittäminen vaatii eri asioita. iOS:n kehitykseen tarvitaan Mac-tietokone, Windows Phonelle Microsoft Windows-lisenssillä varustettu tietokone ja Androidia voi kehittää joko Windowsilla, Macillä tai Linuxilla. Muita eroavaisuuksia ovat kehitysympäristöt. Kehitysympäristöllä tarkoitan niitä ohjelmia tai ohjelmajoukkoja, joiden avulla ohjelmoija suunnittelee ja toteuttaa sovelluksia. Androidin kehitystä tuetaan Eclipse-sovelluksen avulla, Windows Phonen kehitystä Visual Studiolla ja iOS:n Xcodella. Kaiken tämän lisäksi jokaista ympäristöä hallitaan eri koodikielillä, kuten taulukosta 1 näkyy. Kaikki tämä hajottaa älypuhelinmarkkinoiden kehittäjät painottumaan eri alustoille, ja tämän hajanaisuuden yhdistämiseen alustariippumaton sovelluskehitys pyrkii.

Mobiilikäyttöjärjestelmä	Tietokoneen käyttöjärjestelmä	Kehitysympäristö	Ohjelmointikieli
Android	Windows/Mac/Linux	Eclipse + Java + Android SDK	Java
iOS	Mac	Xcode	Objective-C
Blackberry OS	Windows	Eclipse + JDE + Java	Java
Windows Phone	Windows	Visual Studio 2010/2012	C#/.NET/Silverlight/WPF

TAULUKKO 1. Mobiilialustojen riippuvuudet

Alustariippumattomuudella tarkoitan ohjelmointikieltä tai sovellusta, joka ei ole sidoksissa mihinkään tiettyyn laitteistoalustaan tai käyttöjärjestelmään. Alustariippumaton sovelluskehitys ei ole uusi asia tietotekniikan maailmassa. Esimerkiksi Java on tehnyt sitä jo pitkään. Tarjoamalla asennettavan Java-ympäristön mahdollisimman monelle eri alustalle, jolla Javalla tuotetut sovellukset toimivat alustasta riippumatta. Myös Web-ohjelmointi on alustariippumatonta. Web-ohjelmoinnissa verkkoselain toimii käyttöympäristönä, joka lukee ja prosessoi tietoa. Toisin kuin Javassa, verkkoselaimia on monia erilaisia monilta eri yrityksiltä. Useimmat nykyaikaisista verkkoselaimista näyttävät verkkosivut identtisinä toisiinsa verrattuina, joten selaimen valinta on lähinnä kosmeettinen tai käyttötottumuksiin liittyvä (Sascha P. Corti 2011). Verkkosivuja ja toimintoja sivuille voi tehdä melkein millä tahansa tekstinkäsittelyohjelmalla, mikä vapauttaa kehittäjän tekemään sisältöä millä tahansa käyttöjärjestelmällä, jossa on tekstimuokkain. Näin ollen myös kehitysympäristö on vapaasti valittava.

Alustariippumattomassa ohjelmoinnissa on myös ongelmia ja haasteita. Jotkut alustat saattavat käyttäytyä hieman eri tavoin kuin toiset, ja näin tuoda mukanaan toiminnallisia eroja tai semanttisia ohjelmointivirheitä ja alustariippumattomuuskin on hieman harhaan johtava termi, sillä ohjelma on usein jollain tavoin alustariippuvainen. Yleensä tämä tarkoittaa sitä, että ohjelmalla on tietyt vähimmäisvaatimukset, jotka rajoittavat alustamahdollisuuksia. Alustariippumattomasti ohjelmoidut sovellukset vaativat yleisesti enemmän tehoa alustalta kuin natiivisti ohjelmoidut sovellukset. Natiivisovelluksella tarkoitetaan ohjelmaa, joka on ohjelmoitu käyttäen mobiilialustan kehittäjän tukemia työkaluja ja ohjelmointikieliä. Esimerkiksi Windows Phone-natiivisovellus kehitetään Visual Studio-ohjelmalla ja C#-ohjelmointikielellä, ja siinä noudatetaan tarjottuja suunnittelu- ja tyyliohjeita. Tietoturvan takaaminen jokaiselle eri alustalle on myös vaikeampaa kuin natiivisti tehdyissä sovelluksissa. (Miika Raivio 2013)

2.1 Verkkoselaimen hyödyntäminen

Verkkoselain on nykyään melkein joka laitteessa, vähintäänkin jokaisessa viimeisen vuosikymmenen aikana ostetussa tietokoneessa ja monissa nykyajan puhelimista. Verkkoselaimen laaja levinneisyys tekee siitä ihanteellisen kohteen alustariippumattomalle sovelluskehitykselle. Verkkoselainympäristöä tuetaan eri standardein ja kehittäjiä ohjeistetaan toimimaan niiden mukaan. Ohjeistuksella pyritään opettamaan uusia

ja vanhoja kehittäjiä toimimaan muiden kehittäjien hyväksymien normien puitteissa. Esimerkkinä tämäntyyliseen ohjeistukseen on tietyn tyylinen semantiikka ohjelmoidessa, jossa noudatetaan koodin oikeinkirjoitusta.

Kaikki esimerkit tuottavat saman toiminnan, mutta standardoinnilla pyritään saamaan kehittäjät käyttämään samaa mallia ohjelmoitaessa. Helpottaen koodin lukua kehittäjien välillä tms.

`<br></br>` ei — Verkkoselain näyttää useimmiten koodin oikein, mutta se noudata HTML5 tai XHTML standardeja.

`<br /> ja <br/>` — Noudattaa XHTML-standardia verkkosivujen kirjoitukseen.

`<br>` — Noudattaa HTML5-standardia verkkosivujen kirjoitukseen.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//FI" "http://www.mamk.fi/">
```

On esimerkki XHTML:n vaatimasta standardista aloituksesta.

Yleisesti verkkosivujen koodaamiseen tarkoitetulla HTML-kielellä ei voi tehdä monimutkaisia toimintoja sivuille. Esimerkiksi laskimen tai pelien teko ei onnistu HTML- kielellä (Miika Raivio 2013). Monimutkaisten asioiden tekoon suositellaan muita ohjelmointikieliä, kuten Javascriptiä, Hypertext Preprocessoria (PHP) tai Rubyä tilanteesta riippuen. Verkkoselaimella on kehitetty sisällön ja toimintojen tuottoon muitakin ohjelmointikieliä sekä erilaisia kehitysympäristöjä, sovelluskehitystekniikoita ja tyyliohjeita, kuten Cascading Style Sheets (CSS), Ajax, Flash, Really Simple Syndication (RSS), Perl, Python ja Java. Näistä kaikki eivät toimi suoraan jokaisella verkkoselaimella. Modernit tietokoneiden ja älypuhelimien selaimet tukevat suoraan Javascriptiä sekä CSS- ja HTML-kieliä. Lisätekniikat (plugin) vaativat yleensä jonkin liitännäisen asentamisen. Tässä tapauksessa liitännäinen toimii vuorovaikutuksessa verkkoselaimen kanssa tarjotakseen tietyn toiminnon. Esimerkiksi joihinkin selaimiin pitää ladata ja asentaa Flash ympäristö, jotta sillä tuotettu sisältö näkyy käyttäjälle. Verkkoselaimen eri tekniikoiden monipuolisuuden ja laajuuden kirjon avulla voidaan luoda monimutkaisiakin ohjelmia ja pelejä. Näitä teknologioita aletaan pikku hiljaa toteuttamaan eri laitteisiin, joissa toimii verkkoselain. Puhelimissa verkkoselainta on alettu käyttämään sovellusten kehityksessä. Tällaista kehitystä tukee esimerkiksi PhoneGap, jolla voi luoda puhelimelle sovelluksia käyttäen verkkoselaimelle tarkoitettuja tekniikoita.

Mobiilikäyttöjärjestelmä	Ohjelmointikieli
Android	HTML, CSS, Javascript
iOS	
Blackberry OS	
Windows Phone	

TAULUKKO 2. Mobiilialustojen riippuvuudet, jos kehityksessä käytetään verkkoselainta hyödyksi.

Sovelluskehityksen painotus verkkoselain pohjaiseksi vapauttaa kehittäjän aikaisemmin mainituista alustariippuvuuksista (taulukko 1). Selainpohjainen sovelluskehitys ei ole rajattuna mihinkään tiettyyn käyttöjärjestelmään tai kehitysympäristöön. Jolloin ainoaksi riippuvuudeksi jää selainpohjaiset kehitystekniikat ja ohjelmointikielet, kuten taulukosta 2 näkyy.

2.2 Palvelintekniikoiden hyödyntäminen

Verkkoselaimelle sovelluksia tehtäessä tulee muistaa palvelimien monipuolisuus. Palvelin voi toimia esimerkiksi tiedonsäilöntäpalveluna sovellukselle, tietokantana, videon ja äänen suoratoistopalveluna, chattipalveluna tai sitten sovellus voi toimia vain yhdistyssiltana, jolloin kaikki toiminnallisuus suoritetaan palvelimen avulla. Palvelimella tarkoitan erillistä palvelintietokonetta, jossa on eri toiminnallisuuksia riippuen sen käyttötarkoituksesta ja johon otetaan yhteys verkon välityksellä.

Palvelinta hyödynnettäessä verkkosivuun upotetaan skripti eli yksinkertainen ohjelma, jonka palvelin suorittaa kun skriptiä kutsutaan. Skripteillä voidaan siis käyttää palvelinjärjestelmää ja hyödyntää sen tuomia etuja, ja sillä voidaan myös rajoittaa käyttäjien pääsyä tiettyihin asioihin sekä parantaa tietoturvaa. Usein raskaan prosessoinnin siirtäminen keskitetylle palvelimelle on suosittua verkko-ohjelmoinnissa. Näin voidaan luoda laajempia sovelluksia, niin että ne toimivat laskentakyvyltään tehottomimmilla laitteilla. Näin sovellus voidaan suorittaa millä tahansa laitteella, jossa on toimiva verkkoselain tehosta riippumatta. Esimerkkejä tämätyylisistä palvelinpuolen

ohjelmointikielistä ovat PHP, Active Server Page.NET (ASP.NET), C, Java, Ruby, Python, joilla luodaan dynaamisia verkkosivuja ja sisältöä niille. (Jukka Korpela)

3 PHONEGAP

Kanadalainen yritys Nitobi laittoi ensimmäisen PhoneGapin koodin aluilleen San Franciscon iPhoneDevCampissä elokuussa 2008. Päällimmäinen syy sen kehittämiseen oli, se että aloitteleva iPhonen kehittäjä ei ymmärtänyt Objective-C-ohjelmointikieltä, ja tasokuilu webpohjaisesta ohjelmoinnista olio-ohjelmointikieleen, kuten Objective-C, oli suuri (Thomas Myer). Tarvittiin jotain millä hyödyntää kaikkia webtaustaisia ohjelmoijia ja älypuhelimien suurta suosiota. Siihen väliin PhoneGap pyrki tuomalla normaalit puhelimen ominaisuudet ja webpohjaisen ohjelmoinnin helppouden yhteen.

Vuoden sisällä perustamisesta PhoneGap voitti People's Choice -palkinnon Web 2.0 Expo -tapahtumassa, ja sai laajempaa suosiota muiltakin kuin iPhonen kehittäjiltä. Suuret mobiilimaailman yritykset, kuten IBM, SonyEricsson, Symbian ja Palm alkoivat myös tukea PhoneGapin kehitystä (Gowell & McWherter 2012, 309.). PhoneGapin tuki laajennettiin Androidillekin (Thomas Myer). Vuonna 2011 Adobe osti Nitoben. Kaupan jälkeen Nitobi lahjoitti PhoneGapin lähdekoodin Apache Software Foundationille (ASF). Projektin nimeksi valittiin Cordova. (Giorgio Natilli)

Yksi suurimmista eduista koodin siirtämisestä Apache Software Foundationille on se, että suuret yritykset ja kuka tahansa projektista kiinnostunut voi helposti edistää sen etenemistä. Lisäksi projektia hallitsee avoin ja läpinäkyvä johto: projektin yhteisö. Tällöin koodin kehityssuunta pysyy yleensä tarkoituksenmukaisena, koska myynti ja tuotto eivät ole projektin tavoitteena.

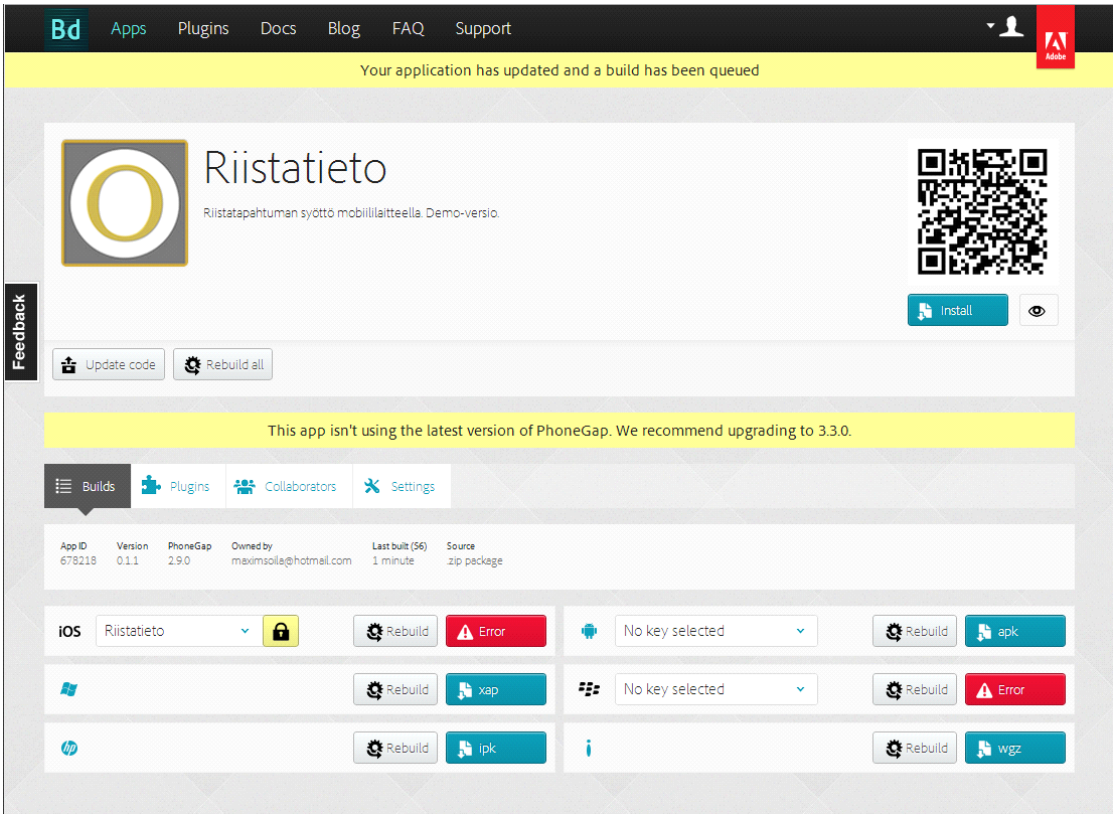
3.1 Arkkitehtuuri

PhoneGap on mobiilisovelluksille tarkoitettu kehitysympäristö. Lyhyesti määriteltynä PhoneGap käyttää älypuhelimen natiivia verkkoselainta, naamioi verkkosivun sovelluksen näköiseksi siitä luo tällä tavoin vaikutelman ja että sovellus pyörisi samoin kuin mikä tahansa muukin sovellus. Käyttöjärjestelmälle tyypillistä koodikieltä ei tar-

vita vaan koko sovellus luodaan käyttämällä HTML5:ttä, CSS3:a, Javascriptiä ja PHP:ia.

Sovelluksen luonti tapahtuu helpoiten käyttäen PhoneGapin tukemia kehitysympäristöjä, joita ovat Eclipse, NetBeans, Microsoft Visual Studio ja Xcode. Kehitysympäristöön ladataan PhoneGapin vaatimat tiedostot. Tarvittavat tiedostot sekä ohjeet niiden asentamiseen ovat PhoneGapin omilla sivuilla. Asennuksen jälkeen aloitetaan uusi sovellusprojekti käyttäen näitä tiedostoja. Ohjelmoitaessa sovellusta PhoneGapin avulla yleisesti hyvä käytäntö on tehdä sovelluksen visuaalinen puoli käyttäen CSS:ää ja HTML:ää, ja käyttää apuna jQueryn mobiilisivuille tarkoitettuja muotoiluja. Toiminnallisuuden ohjelmointiin suositellaan Javascriptiä sekä PHP:tä.

Jos projekti halutaan jakaa usealle laitteelle tai se halutaan laittaa eri valmistajien sovelluskauppaan, kehittäjän tulee joko kääntää sovellus käsin tai luoda käyttäjätunnus PhoneGapin sivuille ja luoda tätä kautta sovellukselle projektin hallintasivut. Hallintasivujen kautta sovelluksen voi muuntaa eri puhelinvalmistajille sopivaksi ja päivittää projektiin liittyviä tietoja. Sovelluksen muunto sivujen kautta vaatii, että valmiin sovelluksen projektitiedot ja tiedostot pakataan yhdeksi zip-tiedostoksi. Zip-tiedosto lähetetään projektin hallintasivun kautta PhoneGapin palvelimelle, jossa automaattinen kääntäjä kääntää sovelluksen sopimaan eri alustoille. Kääntäjä ilmoittaa alustakohtaisista virheistä käännön yhteydessä. Kääntämisen jälkeen sovelluksen eri puhelinalustojen versiot voi ladata sivujen kautta.



KUVA 1. PhoneGap sovelluksen kääntäminen hallintasivun kautta

Itse käännettäessä kehittäjän tulee noudattaa PhoneGapin uusimpia sovelluksen kääntämisohjeita. Sovelluksen kääntäminen käsin vaatii komentoriviltä annettuja käskyjä ja tiettyjen apuohjelmien latausta. Pääpiirteittäin kääntäminen tapahtuu seuraavalla tavalla. Komentorivin kautta luodaan projekti sovellukselle. Siihen lisätään halutut alustat, kuten Windows Phone, iOS ja Android. Sitten projekti käännetään eri tiedostomuotoihin. Tiedostomuodoilla tässä tapauksessa tarkoitan niitä tiedostoja, joita voidaan asentaa puhelimeen, minkä jälkeen sovellusta voi käyttää puhelimesta käsin. Androidin tiedostomuoto on apk, Windows Phonen tiedostomuoto on xap ja iOS tiedostomuoto on ipa. Viimeisenä testataan käännetyn sovelluksen toimivuus puhelimessa.

Käytettäessä puhelimen ominaisuuksia kuten GPS:ää, tiedoston selausta, kameraa, yhteystietoja jne, tarvitaan yhteyssilta laitteen ja sovelluksen välille. Normaalisti näihin pääsee käsiksi puhelimen käyttöjärjestelmän omalla natiivilla koodikielellä. PhoneGap tarjoaa ratkaisun tekemällä liitännät itse alustasta riippuen. PhoneGap toimii yhteyssiltana kehittäjän tekemien kutsujen ja puhelimen ominaisuuksien välillä. Käytännössä siis tehty sovellus kutsuu PhoneGapin toiminnallisuutta esimerkiksi kameral-

le. PhoneGap huomaa kamerakutsun ja välittää sen puhelimen järjestelmälle. Järjestelmä toimii annettujen parametrien mukaan ja antaa oikeuden PhoneGapille käyttää kyseistä toimintaa, jolloin toiminta aukeaa sovellukseen. Esimerkkitapauksessa sovellus avaa puhelimen kameran, jota se käyttää annettujen parametrien mukaan. Parametrit vaihtelevat eri toimintojen mukaan, mutta niihin kuuluvat yleisesti mitä osaa toiminnosta halutaan käyttää ja millä tavoin sekä mitä tapahtuu kun toiminto on suoritettu. Esimerkiksi tiedoston selauksessa voidaan antaa käyttöön luku-, kirjoitus-, tai suoritusoikeus eri parametreillä, jotka käyttävät toiminnallisuutta eri tavoin.

Yhteyssillan lisäksi tarvitaan järjestelmä, jolla sovellus voidaan siirtää alustalta toiselle. PhoneGap tarjoaa verkossa ilmaiseksi yhdelle sovellukselle tämän mahdollisuuden. Käyttäjä lähettää PhoneGapin ympäristössä luodun sovelluksen heidän palvelimilleen, ja jos koodissa ei ole virheitä, käännös tapahtuu automaattisesti eri alustoille. Jotkin alustat vaativat käyttäjiltä kehittäjänlisenssin, jotta käännös voidaan viedä loppuun. Kehittäjänlisenssi on yleensä maksullinen lisenssi, jonka avulla kehittäjä saa oikeuden myydä valmistamiaan sovelluksia sovelluskaupassa sekä oikeuden myös testata kehittämiään sovelluksia puhelimella. Käyttäjä lataa valmiiksi käännetyn paketin ja käyttää sitä niin kuin haluaa. Tämä paketointiprosessi mahdollistaa sovelluksen lähettämisen eri mobiilialustojen sovelluskauppaan ja myös sovelluksen testauksen eri valmistajien puhelimilla.

Koska sovellus on kirjoitettu käyttäen web-pohjaisia kieliä ja se toimii käytännössä puhelimen verkkoselaimella, se täytyy naamioida jotenkin näyttämään sovellukselta, etteivät käyttäjät hämäänny sen käyttötarkoituksesta. Käyttöliittymäkerrokseen on toteutettu WebView-komponentti, johon sovelluksen aloitussivu latautuu. WebView vie koko tilan näytöltä, joten esimerkiksi osoitepalkkia ei näy sitä käytettäessä. Verkkosivua vaihdettaessa WebView pitää näkymänsä samanlaisena, joten sivujen latautuessa ei pitäisi tapahtua mitään ihmeellistä. (Wargo 2012)

3.2 Ohjelmointirajapinnat

Puhelimen ominaisuuksiin ja toiminnallisuuksiin pääsee käsiksi PhoneGapin Javascript-rajapinnan kautta. Rajapinnalla tarkoitan ohjelmointirajapintaa, jonka avulla

ohjelmat, tai ohjelma ja käyttöjärjestelmä keskustelevat keskenään. Rajapinnassa on yleensä abstraktialue ns. "musta laatikko tai black box". Abstraktikerroksen avulla ohjelmoijan ei tarvitse tietää rajapinnan alaisista menetelmistä. Esimerkiksi PhoneGapin Javascript-rajapinnalla tarjotaan käyttömahdollisuus puhelimen toiminnoille. Yleisesti toimintoihin pääsee käsiksi vain käyttöjärjestelmän natiivikielen kautta. Joten PhoneGap toimii ikään kuin välikätenä Javascript-koodin ja puhelimen natiivikoodin välillä kääntämällä annetut käskyt puhelimelle sopivalle kielelle. Tällaista käännösoperaatiota tarvitaan, jos halutaan ottaa hyöty irti puhelimen tarjoamista ominaisuuksista, kuten kamerasta, sijaintista, kompassista ja kiihtyvyysmittarista.

PhoneGap version 3.4.0 ohjelmointirajapintoihin eli Application Programming Interfaceihin (API) kuuluvat: akun tila, kamera, yhteystiedot/kontaktit, laitteen tiedot, kiihtyvyysmittari, kompassi, tiedostojen selaus tai tiedostopuu, sijainti, äänentoisto, verkon tila, värinätoiminto, latausruutu, videon-, äänen- ja kuvien kaappaus, lokalisointi, tiedoston lähetys, verkkoselaimen käyttö ja laiteilmoitukset. Eri käyttöjärjestelmin tukemat API:t näkyy taulukosta 3.

	iPhone / iPhone 3G	iPhone 3GS ja uudemmat	Android	Blackberry OS 6.0+	Blackberry 10	Windows Phone 8	Ubuntu	Firefox OS
Kiihtyvyysmittari	✓	✓	✓	✓	✓	✓	✓	✓
Kamera	✓	✓	✓	✓	✓	✓	✓	✓
Kompassi	x	✓	✓	x	✓	✓	✓	✓
Yhteystiedot	✓	✓	✓	✓	✓	✓	✓	✓
Tiedostot	✓	✓	✓	✓	✓	✓	✓	x
Paikkatieto	✓	✓	✓	✓	✓	✓	✓	✓
Media	✓	✓	✓	x	✓	✓	✓	x
Verkko	✓	✓	✓	✓	✓	✓	✓	✓
Ilmoitukset (Viesti)	✓	✓	✓	✓	✓	✓	✓	✓
Ilmoitukset (Ääni)	✓	✓	✓	✓	✓	✓	✓	✓
Ilmoitukset (Värinä)	✓	✓	✓	✓	✓	✓	✓	✓
Muisti	✓	✓	✓	✓	✓	✓	✓	✓
Verkkoselain	✓	✓	✓	✓	✓	✓	✓	x
Lokalisointi	✓	✓	✓	x	x	✓	✓	x
Laitetiedot	✓	✓	✓	✓	✓	✓	✓	✓
Latausruutu	✓	✓	✓	✓	✓	✓	✓	x
Median kaappaus	✓	✓	✓	✓	✓	✓	✓	x
Verkon tila	✓	✓	✓	✓	✓	✓	✓	x

TAULUKKO 3. Käyttöjärjestelmät ja ohjelmointirajapinnat joita ne tukevat PhoneGapin kehyksessä.

Akun tila -API tarkkailee muutoksia akun varauksessa. API käynnistyy, kun se havaitsee vähintään 1 % muutoksen akun tilassa, tai kun akkulaturi kytketään puhelimeen tai kun se poistetaan. API:n päällimmäinen tarkoitus on suorittaa jokin komento muutoksen tapahtuessa. Esimerkkinä komento ilmoittaa akun latauksen olevan alhainen.

```
window.addEventListener("batterylow", onBatteryLow, false);
function onBatteryLow(info) {
        alert("Akun tila on alhainen" + info.level + "%");
}
```

Kamera API:lla voi ottaa kuvia puhelimen kameralla tai kameroilla. Sillä voi myös valita aikaisemmin otettuja kuvia kameran tiedostoista. Esimerkkitoiminto kamera-API:lle on ottaa kuva ja koodata se sopivaan tiedostomuotoon.

```
navigator.camera.getPicture(onSuccess, onFail, { quality: 50,
            destinationType: Camera.DestinationType.DATA_URL
});
function onSuccess(imageData) {
            var image = document.getElementById('myImage');
            image.src = "" + imageData;
}
function onFail(message) {
            alert('Jokin meni vikaan: ' + message);
}
```

Kiihtyvyysmittari-API tarkkailee laitteen kiihtyvyysmittaria ja toimii, kun se havaitsee muutoksen (delta) liikkeessä suhteessa nykyiseen laitteen suuntaan. Kiihtyvyysmittari tarkkailee liikettä kolmiulotteisesti akseleilla x,y ja z. Esimerkki koodi tarkkailee kiihtyvyysmittarin antureita ja syöttää saadut tulokset käyttäjän ruudulle.

```
navigator.accelerometer.getCurrentAcceleration(onSuccess, onError);
function onSuccess(acceleration) {
            alert('Akseli X: ' + acceleration.x + '\n' +
            'Akseli Y: ' + acceleration.y + '\n' +
            'Akseli Z: ' + acceleration.z + '\n' +
            'Aikaleima: ' + acceleration.timestamp + '\n');
};
function onError() {
            alert('Virhe viesti!');
};
```

Kompassi-API tarkkailee laitteen anturia, joka tunnistaa mihin suuntaan laite osoittaa. Sensori sijaitsee yleensä laitteen yläosassa. API mittaa suuntaa asteina 0-359,99, missä 0 on pohjoiseen. Esimerkkikoodi tarkkailee kompassin anturia, ja antaa saadut tulokset käyttäjän ruudulle.

```
navigator.compass.getCurrentHeading(onSuccess, onError);
function onSuccess(heading) {
            alert('Suunta: ' + heading.magneticHeading);
};
```

```
function onError(error) {
            alert('KompassiVirhe: ' +error.code);
};
```

Laiteviestit näyttävät ruudulla viestin käyttäjälle. Viesti on yleensä ohjeistus tai jokin muu vastaava. Viestiin voi laittaa valintapainikkeita käyttäjille, jotka tekevät eri asioita niitä painettaessa. Esimerkkikoodi ilmoittaa viestin käyttäjälle ja antaa painikkeen sen hyväksymiseen.

```
function alertDismissed() {
            // tee jotain, kun viesti on hyväksytty
}
navigator.notification.alert(
            'Olet voittaja!',      // viesti
            alertDismissed,        // takaisinkutsu
            'Peli ohi',            // otsikko
            'Ok'                   // painonapin teksti
);}
```

Laitteen tiedot API:lla voi hakea tietoja asennetun laitteen ohjelmistosta. Tietojen avulla voi estää tai sallia tiettyjä toimintoja ohjelmassa, riippuen laitteesta tai sen ohjelmistosta. Esimerkkikoodi tulostaa käyttäjälle tiedot laitteen mallista, alustasta, alustan versiosta, PhoneGapin versiosta ja UUID-tunnisteen.

```
document.addEventListener("deviceready", onDeviceReady, false);
function onDeviceReady() {
            var element = document.getElementById('deviceProperties');
            element.innerHTML = 'Laitteen malli: ' + device.model + '<br
/>' +
            'Laitteen alusta: ' + device.platform + '<br />' +
            'Alustan versio: ' + device.version + '<br />' +
            'UUID tunniste: ' + device.uuid + '<br />' +
            'PhoneGapin versio: ' + device.cordova + '<br />';
}
```

Latausruutu näkyy sovellusta käynnistettäessä, ja sen voi kytkeä päälle tai pois päältä.

```
navigator.splashscreen.hide();
navigatos.splashscreen.show();
```

Lokalisoinnilla voi tehdä muutoksia ohjelmaan käyttäjän tapahtumapaikan tai aikavyöhykkeen perusteella. Esimerkki koodi valitsee sopivan kielen käyttäjän paikan perusteella.

```
navigator.globalization.getPreferredLanguage(
            function (language) {alert('Kieli: ' + language.value+'\n');
},
            function () {alert('Virhe kieltä haettaessa\n');}
);
```

Sijainti-API paikantaa puhelimen nykyisen sijainnin ja antaa siitä tarkennettuja tietoja. Paikannukseen puhelin käyttää eri lähteitä, kuten GPS:iä , IP-osoitetta, langatonta verkkoa, Bluetoothia, MAC-osoitetta, Radio Frequency Identificationia (RFID) tai GSM/CDMA puhelintunnisteita. Esimerkkikoodi näyttää käyttäjälle nykyisen sijainnin koordinaatit ja antaa siitä lisätietoja.

```
navigator.geolocation.getCurrentPosition(onSuccess, onError);
var onSuccess = function(position) {
            alert('Leveysaste: '   +position.coords.latitude
            +'\n'+
            'Pituusaste: '         +position.coords.longitude
            +'\n'+
            'Korkeus: '            +position.coords.altitude
            +'\n'+
            'Tarkkuus: '           +position.coords.accuracy
            +'\n'+
            'Korkeuden tarkkuus: ' +position.coords.altitudeAccuracy
            +'\n'+
            'Kulkusuunta: '        +position.coords.heading
            +'\n'+
            'Nopeus: '             +position.coords.speed
            +'\n'+
            'Aikaleima: '          +position.timestamp
            +'\n');
};
function onError(error) {
            alert('code: ' + error.code + '\n' +
            'message: ' + error.message + '\n');
}
```

Tiedostonlähetys-API:lla voi lähettää ja ladata tiedostoja. Tiedostoja voi lähettää palvelimelle tai käyttäjien laitteiden välillä. Esimerkkikoodi lähettää valitun tiedoston palvelimelle.

```
var win = function (r) {
            console.log("Koodi = " + r.responseCode);
            console.log("Vastaus = " + r.response);
            console.log("Lähetetty = " + r.bytesSent);
}
var fail = function (error) {
            alert("Virhe tapahtui: Koodi = " + error.code);
            console.log("Lähetysvirhe lähde " + error.source);
            console.log("Lähetysvirhe kohde " + error.target);
}
var options = new FileUploadOptions();
options.fileKey = "file";
options.fileName = fileURL.substr(fileURL.lastIndexOf('/') + 1);
options.mimeType = "text/plain";
var params = {};
params.value1 = "test";
params.value2 = "param";
options.params = params;
var ft = new FileTransfer();
ft.upload(fileURL, encodeURI("http://oma.serveri.com/upload.php"), win,
fail, options);
```

Verkkoselain-API:lla voi avata laitteen verkkoselaimen. Verkkoselain avautuu normaalisti kaikkien sille normaalisti ominaisten toimintojen kanssa. Vaikka PhoneGap käyttääkin jo verkkoselainta apunaan, niin tällä tavoin avattu selain ei pysty käyttämään PhoneGapin tarjoamia toimintoja. Esimerkki koodi avaa osoitteen antaman verkkosivun uuteen selain ikkunaan.

```
var ref = window.open('http://mamk.fi', '_blank', 'location=yes');
```

Verkontila antaa tietoja mitä verkon yhdistämis mahdollisuuksia laitteessa on ja mitkä niistä ovat päällä. Sen avulla voi myös poistaa ja lisätä toimintoja riippuen siitä mitkä yhteydet ovat saatavilla. Esimerkki koodi tarkastelee eri yhteyksiä ja kertoo mitkä niistä ovat päällä.

```
checkConnection();
function checkConnection() {
```

```
            var networkState = navigator.connection.type;
            var states = {};
            states[Connection.UNKNOWN]        = 'Ei tunnettu yhteys';
            states[Connection.ETHERNET]       = 'Ethernet yhteys';
            states[Connection.WIFI]           = 'WiFi yhteys';
            states[Connection.CELL_2G]        = '2G yhteys';
            states[Connection.CELL_3G]        = '3G yhteys';
            states[Connection.CELL_4G]        = '4G yhteys';
            states[Connection.CELL]           =    'Geneerinen    puhelin
yhteys';
            states[Connection.NONE]           = 'Ei verkko yhteyttä';
            alert('Verkkoyhteyden tyyppi: ' + states[networkState]);
}
```

Median kaappaus API:lla saadaan yhteys puhelimen eri media toimintoihin kuten ääni, kuva ja video. API:lla voidaan siis tallentaa ääntä, kuvaa ja videota. API poikkeaa hieman kamera API:sta ja äänen nauhoitus API:sta. Kaappausta voidaan tehdä sovelluksen taustalla, ilman että kameran tai ääninauhurin käyttöliittymät tulevat näkyviin. Esimerkki koodi nauhoittaa videota.

```
navigator.device.capture.captureVideo(captureSuccess, captureError, {li-
mit:2});
var captureSuccess = function(mediaFiles) {
            var i, path, len;
            for (i = 0, len = mediaFiles.length; i < len; i += 1) {
                        path = mediaFiles[i].fullPath;
                        // tee jotain tiedostolle
            }
};
var captureError = function(error) {
            navigator.notification.alert('Virhekoodi:  '  +  error.code,
null, 'Kaappaus virhe');
};
```

Värinähälytys API:lla voi kytkeä laitteen värinähälytyksen ja yhdistää sitä muiden toimintojen mukaan. Esimerkki koodi aktivoi laitteen värinähälytyksen 2,5 sekunnin päästä.

```
navigator.notification.vibrate(2500);
```

Äänentoisto ja nauhoitus API:lla voi toistaa ääni-tiedostoja tai käyttää laitteen mikrofonia nauhoitukseen. Esimerkki koodi hakee osoitteesta äänitiedoston ja soittaa sen.

```
function onDeviceReady() {
            playAu-
dio("http://audio.tiedosto.com/sisalto/bluesguitar.mp3");
}
function playAudio(url) {
            // Soittaa osoitteen osoittaman audio-tiedoston
            var my_media = new Media(url,
            function() {
                        console.log("playAudio():Audio Ok");
            },
            funciont(err) {
                        console.log("playAudio():Audio Virhe:" + err);
            };
            my_media.play();
}
```

4 RIISTATIEDON KERUUSOVELLUS

4.1 Sovelluksen vaatimukset

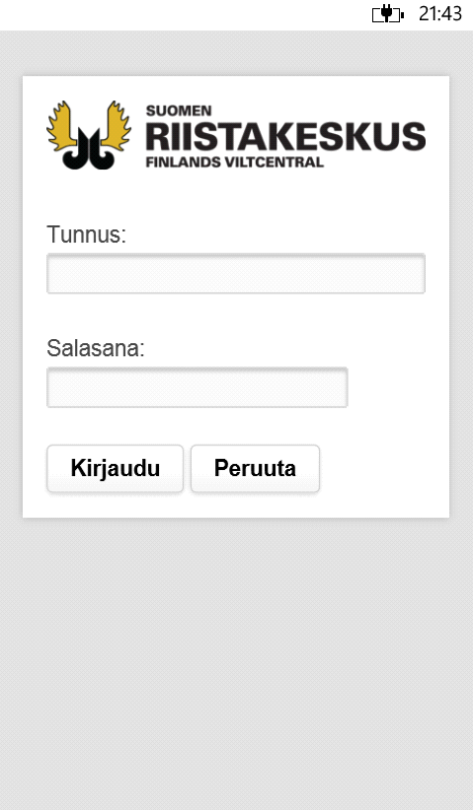
Opinnäytetyön tarkoituksena on luoda mobiilisovellus riistatietojen keräämiseen. Sovelluksen tulisi toimia Windows Phone-, iOS- ja Android -alustoilla. Näiden eri versioilla ei saisi olla toiminnallisia eroja eikä suuria kosmeettisia eroja, jotta käyttökokemus ei vaihtelisi eri käyttäjäkunnittain. Sovellukseen syötettäisiin riistaeläintä koskevasta tapahtumasta tarvittavat tiedot, jotka lähetään eteenpäin riistakeskuksen verkkopalvelimelle. Tiedot tulisi toimittaa ennalta sovitussa muodossa ja sovellukseen kirjauduttaisiin riistakeskuksen järjestelmässä valmiina olevilla tunnuksilla.

Normaalisti havainto-, saalis-, tai kaatotapahtumaa tehtäessä merkitään ylös tapahtumapaikan koordinaatit, kellonaika ja eläin, joten yksityiskohtien tulisi näkyä myös puhelinsovelluksessa. Ensimmäisessä kehitysversiossa nämä asiat ovat pakollisia, ja sen jälkeen sovellusta voitaisiin kehittää eteenpäin, mikäli se koetaan tarpeelliseksi. Visioita kehityksen viemiseksi ovat mm. valokuvan ottaminen tapahtumasta ja sen lähetys Riistakeskuksen palvelimelle muiden tietojen mukana, laajennus eri tapahtumiin ja eläimiin sekä Riistakeskuksen tiedotusten näkyminen ja tärkeimpien tapahtumien uutisointi käyttäjille.

Tietoturvaan ja käyttöliittymään ei alussa panosteta. Lähtökohtana on saada yhteys toimimaan paikallisella virtuaalisella palvelimella, joihin sovellus tekee tietojen lähetyksen ilman salausmenetelmiä. Salausmenetelmät lisätään jälkeenpäin, kun kaikki muu toiminnallisuus on kunnossa. Tiedoston tallennusmenetelmän tulee olla helposti avattavissa. Käyttöliittymän tulee olla selkeä ja intuitiivinen käyttäjälle, niin että käyttäjä tietää heti miten sitä käyttää ilman suurempaa ohjausta. Graafisesti käyttöliittymän tulee antaa kuva, siitä että käyttäjä on käyttämässä Riistakeskuksen virallista sovellusta.

4.2 Arkkitehtuuri

Riistatiedon keruu sovellus koostuu neljästä eri tapahtumakohdasta: kirjautumisesta, päävalikosta, tapahtumankäsittelijästä ja tiedon lähetyksestä. Näistä ensimmäinen on sisäänkirjautuminen (kuva 2). Sisäänkirjautumisessa puhelin lähettää pyynnön riistakeskuksen palvelimelle, joka etsii tietokannasta täsmäävän käyttäjätunnuksen ja tarkistaa, täsmääkö mukana lähetetty salasana käyttäjätunnukselle määritettyyn salasanaan. Tämän jälkeen palvelin lähettää puhelimelle takaisin vastauksen. Jos kaikki on kunnossa, vastaukseen sisältyvät tarvittavat käyttäjätiedot ja hyväksyntä kirjautumiselle. Jos jokin epäonnistuu, palvelin lähettää paluuviestinä virheilmoituksen, jonka sisältö riippuu siitä, mikä epäonnistui. Sovellus näyttää virheilmoituksen käyttäjälle. Jos esimerkiksi salasana on väärä, sovellus kertoo tästä ja ohjeistaa käyttäjää tarkistamaan salasanan oikeinkirjoituksen. Kun sovellus on hyväksynyt kirjautumisen, se saa käyttäjän tiedot joita se ylläpitää istunnon ajan ja käyttää niitä tietoja lähetettäessä.

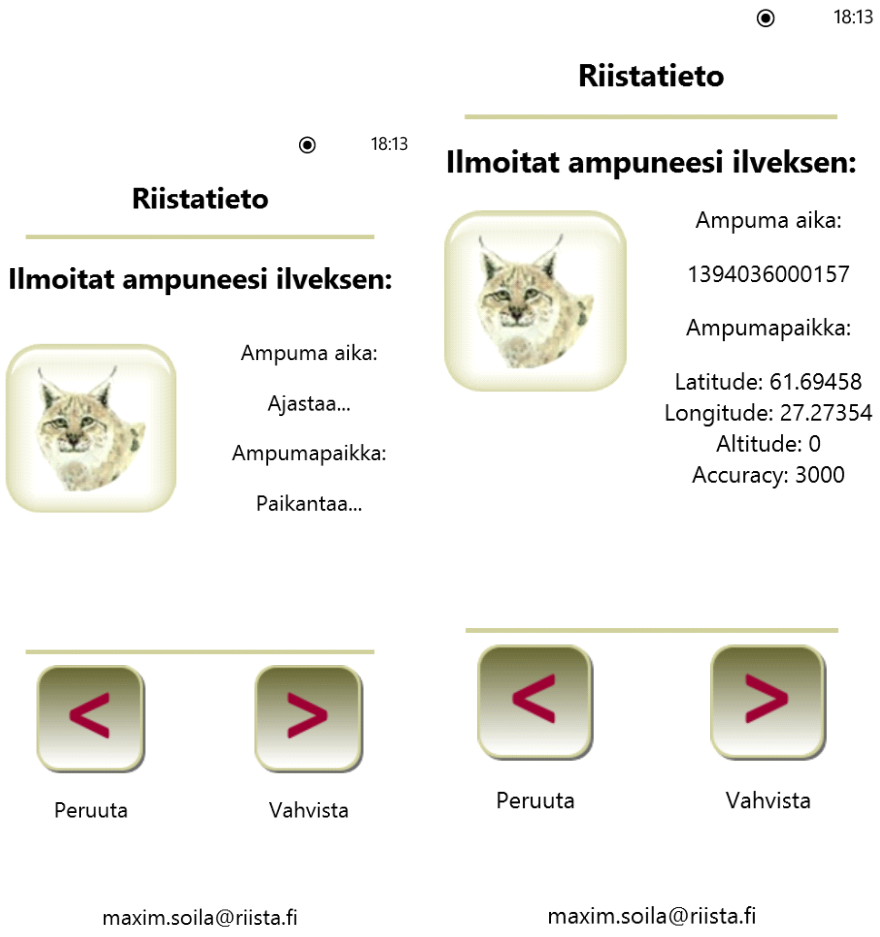


KUVA 2. Sovelluksen kirjautumisruutu

Kirjautumisen jälkeen avautuu päävalikko (kuva 3), jonka tarkoituksena on ohjata käyttäjä lähettämään oikean riistatapahtuman tiedot. Valikossa on tapahtumakäsittelijät kolmelle eri eläimelle, käsittelijä muulle tapahtumalle ja uloskirjautumisen mahdollisuus. Tapahtumakäsittelijällä tarkoitan sitä osaa koodissa, joka suorittaa halutun toiminnon. Yleisesti koodissa on tapahtuma ja tapahtumankäsittelijä. Tapahtumat ovat toimintoja, jotka yleensä ilmoittavat ohjelman muuttuneesta tilasta. Tapahtumankäsittelijät taas reagoivat tähän muuttuneeseen tilaan ja suorittavat toimintonsa. Esimerkiksi painikkeen painaminen laukaisee tapahtuman, jonka aiheuttama muutos käynnistää tapahtumankäsittelijän. Kun käyttäjä painaa eläimen painiketta, puhelin avaa omasta kirjastostaan sisäisen linkin, joka tuo valitun kohteen tiedot näkyviin. Päävalikossa on myös käyttäjälle pieni ohjeistus, joka kehottaa valitsemaan kaadetun eläimen painikkeen valikosta. Sovelluksen alalaidassa näkyy myös kirjautuneen käyttäjän käyttäjätunnus.

KUVA 3. Sovelluksen päävalikko

Tiedon- ja tapahtuman käsittelijä avautuu eläintä tai muu tapahtuma painiketta painettaessa. Käsittelijä lataa kyseisen sivun auki (kuva 4). Sen jälkeen sovellus kutsuu puhelimen GPS-navigointiominaisuutta. Jos GPS ominaisuutta ei saada auki, siitä ilmoitetaan käyttäjälle virheilmoituksella. Jos taas ominaisuus saadaan ilman ongelmia auki, sovellus pyytää seuraavaksi GPS-navigaattoria hakemaan senhetkiset paikannustiedot. Paikannustiedot tuodaan näkyviin ruudulle luettavassa muodossa (kuva 4). Tiedoissa näkyvät koordinaatit, GPS:n tarkkuus, korkeus merenpinnasta ja aikaleima. Tietojen tullessa näkyviin ruudulle avautuu painike niiden lähettämiselle. Vieressä on myös painike tapahtuman perumiselle.



KUVA 4. Paikannustietojen haku ja ajan mittaaminen

Viimeinen osa tapahtumaketjua on itse tietojen lähettäminen. Siinä puhelin ottaa jälleen yhteyden Riistakeskuksen verkkopalvelimeen ja pyytää saada luvan lähettää tapahtuman tiedot kirjautuneen käyttäjän nimissä. Jos palvelin vastaa hyväksyvästi, puhelin aloittaa tietojen lähetyksen palvelimelle. Kun kaikki tiedot on lähetetty, palvelin lähettää paluuviestin, jossa todetaan että kaikki on mennyt hyvin ja että yhteys voidaan sulkea. Sen jälkeen puhelin sulkee yhteyden, ja ilmoittaa käyttäjälle tapahtuman onnistumisesta (kuva 5). Jos tapahtuu jokin virhe, verkkopalvelin lähettää paluuviestinä virheilmoituksen, jonka sisältö riipuu virheen laadusta. Käyttäjää ohjeistetaan virheen mukaan. Jos internetyhteyttä ei voida luoda laisinkaan, puhelin ilmoittaa että yhteyttä ei saatu luotua.

KUVA 5. Tietojen lähetyksestä saatu vastaus

Käyttäjälle annetaan loppuruudussa painike päävalikkoon palaamiselle. Päävalikosta tapahtumaketju voidaan joko aloittaa uudestaan tai poistua sovelluksesta. Onnistuneen tiedon lähetyksen ohessa käyttäjää ohjeistetaan lisäämään puuttuvat tiedot Riistakeskuksen sivuilta kotoa käsin. Tiedon lisäys eri kautta on tehty siksi, että ilmoittamisen tuli olla nopeaa ja helppoa, jottei aikaa mene metsästysalueella siihen. Ja ylimääräisten arvojen ja tietojen lisääminen sovelluksella tekee käyttöliittymästä raskaan ja kankean.

4.3 Teknisen toteutustyylin ja kehitysympäristön valinta

Sain melko vapaat kädet siihen miten ohjelmiston tekisin. Olin käynyt keskusteluja aikaisemmin opettajani kanssa uudesta tyylistä tehdä puhelinsovelluksia käyttäen webkit-pohjaista koodausta. Kun olin tutkinut asiaa tarkemmin, löysin sopivan ohjelmointikehyksen, jolla voisi kääntää ohjelmistokoodin eri alustoille ilmaiseksi. Tämä PhoneGap sopi tarkoitusperiini. Sovelluksen vaatimuksena oli datan lähetys ja vastaanotto, helppo käytettävyys ja käyttöliittymä, sovelluksen tuli toimia kolmella suu-

rimmalla älypuhelin alustalla (Windows Phone, iOS, Android), sekä GPS paikannustietojen talteen kerääminen. Kaikkia näitä pystyttiin hallitsemaan PhoneGapin kautta.

Kehitysympäristön valintaan on käytännössä neljä järkevää vaihtoehtoa: Xcode, Eclipse, NetBeans ja Microsoft Visual Studio 2012. Nämä ovat parhaiten tuetuimmat ympäristöt PhoneGapille. Xcode oli huono valinta, sillä se vaatii Mac-tietokoneen toimiakseen. Eclipse ja NetBeans kilpailevat toisiaan vastaan samantyylisinä kehitysympäristöinä. Eclipse on virallisesti tuettu Android-kehitysympäristö, joten sille löytyisi varmasti helpommin ohjeita tai apua ongelma tilanteiden varalle. Päädyin valitsemaan Microsoft Visual Studio 2012:n, koska sillä sai kehittää kouluun liittyviä projekteja. Valitsin Visual Studion myöskin sen takia, että omistin Windows Phonen, jolle kehitysympäristö on muun muassa tarkoitettu. Toiseksi ympäristöksi valitsin Eclipsen, lähinnä sen tukeman Androidin takia. Projekti tulisi testata mahdollisimman monella eri alustalla jo kehitysvaiheessa, ettei jälkeenpäin tule ongelmia sen siirrettävyydestä eri alustoille.

Varsinainen lopullinen testaus suoritettiin laitteilla sekä käyttäen apuna puhelinemulaattoreita. Windows Phonen testauksen suoritin omalla puhelimellani. iOS-testauksessa apuna toimi Janne-Pekka Surakka ja hänen iPhonensa. Android-testauksessa käytin Eclipse-kehitysympäristöstä löytyvää Android-puhelinemulaattoria. Emulaattoritestauksen huono puoli on, etteivät kaikki toiminnot toimi siinä samoin kuin puhelimissa, esimerkiksi GPS-paikannus ei onnistu emulaattorilla, koska virtuaalinen emulaattori ei sijaitse fyysisesti missään. Hyvä puoli emulaattorilla testauksessa on sen nopeus ja tietenkin se, ettei laitetta tarvitse omistaa testatakseen sovellusta. Siksi käytin emulaattoria lähinnä muun toiminnallisuuden sommittelussa, suunnittelussa sekä testauksessa.

4.4 Kehitys

Sovelluksen kehityksen aloitin puhtaalta pöydältä, vaikka olin jo aikaisemmin tehnyt muutaman kännykälle sopivan verkkosivun. Näitä sivuja ei kuitenkaan voinut käytännössä hyödyntää tässä projektissa, sillä asettelu tulisi olemaan aivan erilainen kuin aiemmissa töissäni. Ensimmäiseksi asensin kehitysympäristöt Eclipsen sekä Visual Studion, joihin liitin PhoneGapiin tarvittavat lisäosat. Seuraavaksi testasin toimiko itse ympäristö. Toteutin testauksen kehittämällä mahdollisimman yksinkertaisen sivun,

joka näytti ruudulla vain tekstin pätkän. Testauksen tarkoituksena oli varmistaa ettei ympäristössä ole ongelmia ennen kehityksen aloittamista. Tämä on mielestäni yleisesti hyvä käytäntö uusille alustoille tai uudella ympäristöllä aloitettaessa. Toimivuuden todettua pystyi itse kehityksen aloittamaan. Testasin alkuun paria eri asetteluvaihtoehtoa, jotka lähetin tilaajalle arvioitavaksi. Keskustelujen kautta päädyimme yksinkertaiseen vaihtoehtoon, jossa sivun jaettiin kolmeen eri osaan: ylätunnisteeseen, sisältöosuuteen ja alatunnisteeseen.

Ensimmäisenä sivun tilan jaottelussa sommittelin tilan logolle ylätunnisteeseen. Ylätunniste eli header pysyy samanlaisena sivulta toiselle siirryttäessä säilyttäen koheesion sovelluksen sisällä. Ylätunnisteeseen kuuluu myös tilanjako viiva, joka toimi tilan erottajana sivulla ja visuaalisena loppuna ylätunnisteelle. Sivun keskelle jätin tyhjää tilaa sisällölle. Loin myös sivun loppuun alatunnisteen, jossa oli aluksi vain tilanjakoviiva tunnisteen alussa, samalla periaatteella erottaen sen sisällöstä visuaalisesti kuin ylätunnisteessakin.

Tunnisteiden valmistuttua sovelluksen perussivu alkoi näyttää varteenotettavalta. Testasin kuinka sisällölle luotu tila käyttäytyi eri asioiden kuten kuvien ja tekstin kanssa. Kun olin tyytyväinen siihen miten eri sisältö asettui sivulle, aloitin kehittämään GPS-toiminnallisuutta sovellukseen. Tuskastelin turhankin pitkää GPS-toiminnallisuuden kanssa, sillä aluksi en saanut sitä millään toimimaan. Myöhemmin tajusin ettei virtuaalinen puhelinemulaattori fyysisesti sijaitse missään paikannettavassa tilassa, jolloin GPS-toiminnallisuus ilmoitti virheestä tai ei vain saanut paikannustietoja haettua. Vaihdoin testauksen fyysiselle puhelimelle ja GPS alkoi toimimaan muutaman pienen viilauksen jälkeen. Poistin turhaksi kokemiani tietoja, kuten hetkellinen nopeus, merenpinnan korkeus, merenpinnan korkeuden mittauksen tarkkuus yms. Kyseiset tiedot ei ole tarpeellisia riistatapahtuman ilmoituksessa.

Värit sovellukseen päätettiin opinnäytetyön tilaajan kanssa, keskustelujen kautta. Niihin vaikuttivat eniten Riistakeskuksen omilla sivuilla käytetyt värit, joihin lisäsin itse yhden vastavärin tuomaan huomioita sovelluksen tärkeille asioille. Tässä tapauksessa vihreän vastavärinä toimi punainen, joka mielestäni toimii hyvin huomiota herättävänä värinä. Värimaailmaan oltiin tyytyväisiä, joten implementoin ne sovelluksen eri kohtiin, kuten painikkeisiin ja tilanjakoviivoihin. Taustan pidin valkoisena, jotta sovelluksen teksti olisi helposti luettavaa. Osan projektin grafiikoista, kuten eläinten kuvat

painonapeissa, Riistakeskuksen logon ja Oravannahka Oy:n logon sain käyttööni projektin tekemiseen kyseisten tahojen verkkosivujen kautta.

Väriteeman asetuttua pystyin aloittamaan muiden visuaalisten elementtien teon. Aloitin luomalla pari vaihtoehtoa painikkeisiin. Sovelluksessa tarvittiin kahta eri tyylistä painonappia, toinen eläimille ja toinen muulle toiminnallisuudelle. Kun viimeisiin muotoihin napeista oli päädytty, valmistin loput tarvittavat napit samalla tyylillä. Lisäsin myöhemmin kaikkiin painikkeisiin visuaalisen efektin, kun nappia painaa. Eläinpainikkeissa taustaväri muuttuu ilmoittaen käyttäjälle, että painikkeen painaminen on rekisteröity. Muissa painikkeissa tein niille varjostuksen, joka loi 3D-maisen efektin. Kun painiketta painaa, painuu painikkeita alaspäin visuaalisesti. Efektin loin tekemällä toisen kuvan painikkeesta, jossa varjostus on poistettu ja kuvaa on siirretty hiukan aikaisemman kuvan varjon suuntaan. Painettaessa painiketta, oikeasti vain kuva vaihtuu sovelluksen sisällä toiseen jossa varjoa ei ole, mutta efekti näyttää silmälle luonnolliselta. Painike nousee takaisin ylös painon poistuttua sen kohdalta.

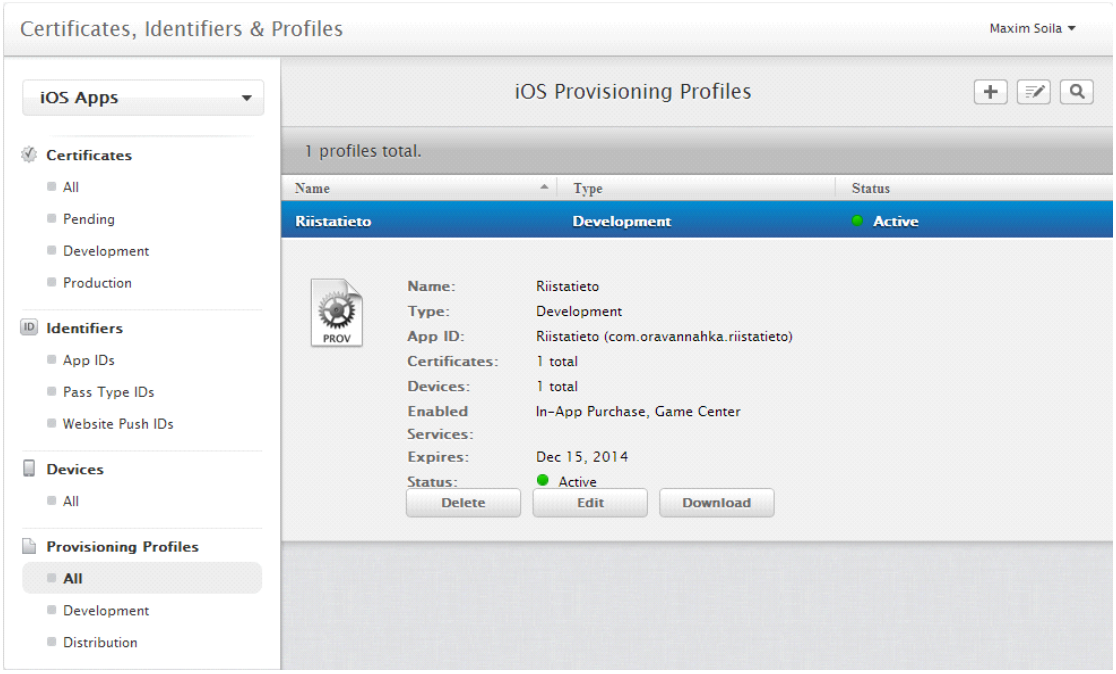
Viimeisenä sovelluksen visuaalisen ilmeen rakentamisessa oli kaiken kokoaminen sovelluksen sisälle ja erinäisten lataus- ja sovelluskuvien työstäminen eri puhelinten käyttöjärjestelmille. Työvaiheena tämä oli yksinkertainen, koska sivujen asettelu oli muotoiltu vastaanottamaan tekemäni sisältö. Sijoitin tehdyt painikkeet, kuvat ja tekstit niille tarkoitetuille paikoille ja yhdistin toiminnallisuuden niihin. Lisäsin sovellukselle käynnistyskuvakkeet puhelimen sovellusvalikkoon, sekä tein eri kokoiset versiot sovelluksen latautumisruudulle. Erikokoiset latausruudut tarvitaan sen takia, että monet valmistajat käyttävät eri näyttökokoja puhelinten valmistuksessa. Tein jokaisen valmistajan asettamilla standardeilla eri näyttökooille sopivat latausruudut. Yksinkertaisinta oli tehdä ensin kaikista suurimmalle näyttökoolle latausruutu, jonka jälkeen pienensin kuvaa kuvanmuokkausohjelmalla sopimaan eri näytöille. Sovelluksen käynnistyskuvakkeissa piti tehdä pieniä muokkauksia, sillä pelkkä suurentaminen tai pienentäminen ei riittänyt. Asettelu piti tehdä yleensä eri tavalla sillä valmistajien standardikoot poikkesivat paljon toisistaan.

Visuaalisen ilmeen valmistuttua, loin sovellukselle kirjautumisruudun, johon käyttäjät voisivat syöttää Riistakeskukselta saamansa käyttäjätunnuksen ja salasanan. Ohjelmoin kirjautumiseen toiminnallisuuden, jonka erotin muusta sovelluksen koodista.

Kaikki koodi yhdessä saattaisi muodostaa tietoturvariskin ja koodin hajauttaminen eri tiedostoihin selkeytti koodin jäsentelyä.

Sovelluksen testauksessa käytin omaa puhelintani, joka oli avattu kehitystarkoitukseen. Näin siksi, että sillä pystyi ajamaan sovelluksia jotka eivät vielä olleet sovelluskaupassa. Windows Phonen avaamiseen asennetaan Windows Phone SDK-paketti tietokoneella, jonka avulla puhelin avataan. Avaus tarvitsee myös Microsoft käyttäjätilin, jonka voi luoda Microsoftin omilta sivuilta. Puhelin rekisteröidään kyseiselle Microsoft-tilille.

IPhonen testaukseen hankimme kehittäjälisenssin. Sen aktivoiminen ja puhelimen rekisteröinti kehittäjälaitteeksi ei ollut helpoimmasta päästä, koska Mac-pohjaista konetta ei ollut käytettävissä. Rekisteröitäessä puhelin kehitystarkoitukseen Macillä lisenssinhaltijan tulee luoda Keychain-ohjelman kautta CertificateSigningRequest-tiedosto. Tiedoston luomisessa syötetään email osoite ja nimi, jolle kehittäjälisenssi on rekisteröity. Ohjelma tuottaa tiedoston, joka lähetetään Applen iOS-kehittäjäsivustolle hyväksytettäväksi. Prosessin onnistuttua avautuu valikko hyväksytyn tiedoston lataukseen. Hyväksytty tiedosto lisätään Keychain-ohjelmaan. Kehittäjäsivustolla lisätään myös rekisteröitävän puhelimen Unique Device Identifier -tunniste (UDID). Tunniste löytyy esimerkiksi iTunes ohjelman kautta. Hallintasivuilla luodaan kehitettävälle sovellukselle myös oma tunnisteensa. Kaikki nämä tiedot yhdistetään Provisioning Profile -tunnisteeksi (kuva 6). Tämä tunniste ladataan ja muutetaan Keychain-ohjelman kautta siirrettävään muotoon. Tiedosto lisätään PhoneGapin kääntäjään, jolloin itse tehtyjä sovelluksia voi testata iPhonella.



KUVA 6. Applen kehittäjäsivuston sertifikaattien, tunnisteiden ja profiilien hallinta-paneeli

Puhelimen rekisteröinti ja lisenssin aktivoiminen Windowsin kautta vaatii Open Secure Sockets Layer -ohjelman (OpenSSL), jonka avulla luodaan tarvittavat tiedostot. Ensimmäisenä tulee käyttäjän luoda avain. Avaimen luomiseksi syötetään komentoriviltä seuraava koodi:

```
openssl genrsa -des3 -out ios.key 2048
```

Komento luo avaimen ios.key-tiedostoon, siihen kansioon, johon komentorivin tiedostoselain osoittaa sillä hetkellä. Komento pyytää myös antamaan salasanan. Salasanaa tarvitaan vielä myöhemmässä vaiheessa.

Seuraavaksi luodaan Certificate Signing Request -tiedosto (CSR). Tiedosto luodaan seuraavalla komennolla:

```
openssl req -new -key ios.key -out ios.csr -subj "/emailAddress=SÄHKÖPOSTI-OSOITE, CN=YRITYKSEN-NIMI, C=MAAKOODI"
```

Sähköposti-osoite, yrityksen nimi ja maakoodi tulee korvata oikeilla tiedoilla, esimerkiksi:

```
openssl req -new -key ios.key -out ios.csr -subj "/emailAddress=maxim.soila@oravannahka.fi, CN=Oravannahka, C=FI"
```

Komento tuottaa sertifikaatin allekirjoituspyynnön ja tallentaa sen ios.csr-tiedostoon.

Tiedosto hyväksytetään Applen kehittäjäsivujen kautta, joka vaatii kehittäjätunnuksen, joka taas luodaan kehittäjälisenssin oston yhteydessä. Hallintapaneelista löytyvän Certificate-toiminnon kautta tiedosto lähetetään sivuston palvelimelle. Palvelimen hyväksyttyä tiedoston valikko avautuu hyväksytyn tiedoston lataukseen. Tämä hyväksytty tiedosto tulee muuntaa Privacy Enhanced Mail Security Certificate-tiedostoksi (PEM). Muunto tapahtuu komennolla:

```
openssl x509 -in ios_development.cer -inform DER -out ios_development.pem -outform PEM
```

Jossa ios_development.cer on tiedosto, joka hyväksytettiin kehittäjäsivujen kautta. Ja ios_development.pem on tiedosto, joka syntyy muunnosta.

Seuraavana muodostetaan p12-tiedosto. Tiedoston tekoon tarvitaan avain tiedosto (ios.key) ja kehitys sertifikaatti (ios_developtment.pem).

```
openssl pkcs12 -export -inkey ios.key -in ios_development.pem -out ios_development.p12
```

Tiedoston luonnin yhteydessä kysytään salasanaa, joka annettiin aikaisemmin ios.key-luonnin yhteydessä.

Viimeinen tiedosto luodaan Applen kehittäjäsivujen kautta, jossa luodaan Provision Profile-tiedosto. Tiedosto luodaan sertifikaattihallintasivujen kautta, jossa täytetään mm. käyttäjätietoja, tietoja ohjelmasta ja puhelimen UDID. Lopuksi avautuu valikko, josta tiedoston voi ladata. Tiedosto tulee asentaa kehittäjän puhelimeen, jolla sovelluksia testataan. Tiedoston asennus onnistuu iTunesista raahaamalla tiedosto iTunessin päälle. iTunessin tulisi tunnistaa tiedosto ja lisätä se sen kirjastoon sovelluksen. Sovellus asennetaan puhelimeen raahaamalla sovellus asennettavan puhelimen kuvakkeeseen.

5 PÄÄTELMÄT

Laitevalikoiman laajentuessa ja uusien järjestelmien syntyessä moni aloitteleva ohjelmoija havahtuu yllättävään ongelmaan: ohjelmointikielien ja järjestelmien suureen määrään. Syventyminen johonkin näistä on suuri ajallinen sitoumus. Entä jos tuki lakkautetaan tai parempia järjestelmiä eri teknologioilla tulee? PhoneGap seuraa nykyajan trendiä tuomalla laitteita lähemmäs toisiaan yhteisen tekijän verkkoselaimen kautta. Nykyään verkkoselaimia on jääkaapeista televisioihin ja autoista aina tietoko-

neisiin asti. Tälle alueelle ollaan kovaa vauhtia tekemässä teknologisia standardeja ja ohjeistuksia, jotka helpottavat tulevaisuudessa ohjelmoijan elämää. Standardien sekä ohjeistuksien myötä, tulevaisuudessa voidaan laajentaa verkkoselaimen käyttömahdollisuuksia, jotka ovat nykyäänkin jo mittavat.

Uusien verkkosovelluskehitystekniikoiden yleistyessä, laajenee samalla verkkoselaimen käyttötarkoitus. Uskon, että jatkossa painotetaan yhä enemmän verkkoselaimella toimiviin sovelluksiin. Sillä nykyäänkin voi jo hoitaa esimerkiksi verot, pankkiasiat, pitsan tilaukset, uutisten lukemisen, pelaamisen, tv:n katsomisen ja paljon muita asioita verkkoselaimen kautta. PhoneGap tuo yhteen puhelinsovelluskehityksen ja verkkoselainteknologiat, joka avaa verkkoselaimiin painottuvalle kehittäjälle tien puhelinsovellusten pariin.

Kuten jo alussa mainittiin, alustariippumaton sovelluskehitys ei ole ilman puutteita ja ongelmia. PhoneGap kohtaa samoja ongelmia, kuin muutkin alustariippumattomat tekniikat. Sovellukset on yleisesti hieman raskaampia ja kaikkia puhelimen ominaisuuksia ei saa helposti käyttöön. Silti sillä oma käyttötarkoituksensa. PhoneGapillä on helppo luoda yksinkertaisia sovelluksia, jotka toimivat kaikilla suurimmilla älypuhelin alustoilla. Sovellukset on myös yksinkertaista siirtää toimimaan tietokoneen verkkoselaimellekin. PhoneGap soveltuu erinomaisesti pienille, muutaman henkilön kehitysryhmille joilla ei ole aikaa tai resursseja kehittää sovellusta erikseen jokaiselle alustalle.

Kaiken kaikkiaan hanke oli kiinnostava ja tutkimustyö herätti innostuksen alustariippumattomaan sovelluskehitykseen. Aluksi kehitystyö oli hiukan kankeaa verrattuna natiiveihin työkaluihin. Esimerkiksi Microsoftin Visual Studiolla natiivien ohjelmien kehitys on paljon mutkattomampaa hyvien ohjeiden ja sen antaman valmiin pohjan ansiosta. Kehitystyön aikana ei tapahtunut suuria muutoksia ensimmäisen tapaamisen aikana tehtyyn vedokseen, joka helpotti projektin eteenpäin viemistä. Väriskaalan valinta ja painikkeiden teko toi jotain uutta ohjelmoijan arkeen. Yleensä vastaavat tehtävät annetaan graafikon hoidettavaksi

Riistatieto-sovellus valmistui demo-versioksi. Aika näyttää tuleeko se suurempaan jakoon viralliseksi Riistakeskuksen sovellukseksi, ja jatkuuko kehitystyö sen saralla. Jatkokehitys mahdollisuutena sovellukseen voisi lisätä esimerkiksi tiedote-palvelun.

Palvelu voisi tiedottaa Suomen Riistakeskusta koskevista tärkeistä asioista, kuten metsäpalovaarasta, tietyn eläin kannan noususta, karhu havainnoista jne. Muita mielenkiintoisia kehitysideoita on kuvien lisääminen tapahtumien ilmoitukseen. Karttapalvelun lisääminen, jossa käyttäjä saa näkyviinsä maastokartan alueesta. Lisälaitteiden integroiminen, johon voisi esimerkiksi lisätä bluetooth-yhteydellä ammattitasoisen GPS-paikantimen. Sovellukseen voisi myös tehdä varatoiminnon, jos se ei saa Internet-yhteyttä luotua metsän keskellä. Jos yhteyttä ei saada luotua, tallentaisi puhelin tiedot muistiin ja lähettäisi se ne, kun Internet-yhteys on taas tarjolla. Sovelluksessa on paljon muitakin mielenkiintoisia kehityssuuntia, mutta innostus uuden oppimiseen vie ajatuksia jo muualle. Joten edessä on jokaisen kehittäjän suurin ongelma: Aloittaakko uusi projekti vai parannellakko vanhaa?

LÄHTEET

Corti, Sascha P. 2011. Browser and Feature Detection. WWW-dokumentti. http://msdn.microsoft.com/en-us/magazine/hh475813.aspx. Päivitetty: 20.10.2011. Luettu: 8.3.2014

Gartner 2014. Gartner Say Annual Smartphone Sales Surpassed Sales of Feature Phones for the First Time in 2013. WWW-dokumentti. http://www.gartner.com/newsroom/id/2665715. Päivitetty: 13.2.2014. Luettu: 16.2.2014.

Ghatol, Rohit & Patel, Yogesh 2012. Beginning PhoneGap: Mobile Web Framework for Javascript and HTML5. New York: Apress.

Giorgio, Natili 2013. PhoneGap 3 Beginner's Guide. Birmingham: Packt Publishing.

Gowel, Scot & McWherter, Jeff 2012. Professional Mobile Application Development. Indianapolis: John Wiley & Sons.

Korpela, Jukka 2009. Palvelinskriptit. WWW-dokumentti. http://www.cs.tut.fi/~jkorpela/webjulk/3.1.html. Päivitetty 11.6.2009. Luettu: 20.1.2014.

Raivio, Miikka. 2013. Alustariippumaton Mobiilisovelluskehityksen Tekniikat. Diplomityö.

Thomas, Myer 2011. Beginning PhoneGap. Indianapolis: John Wiley & Sons.