LAHDEN AMMATTIKORKEAKOULU
Lahti University of Applied Sciences

ALUSTARIIPPUMATON MOBIILISOVELLUSKEHITYS

Case: Laskuavain Mobile

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikka
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2014
Seyawash Masudi

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

MASUDI, SEYAWASH:

Alustariippumaton mobiilisovelluskehitys
Case: Laskuavain Mobile

Ohjelmistotekniikan opinnäytetyö, 52 sivua

Kevät 2014

TIIVISTELMÄ

Tässä opinnäytetyössä selvitetiin, miten PhoneGapin ohjelmistokehystä voidaan hyödyntää alustariippumattomassa mobiilisovelluskehityksessä. Opinnäytetyö toteutettiin Ohjelmistotalo Koodiavaimelle.

Opinnäytetyössä käyttiin läpi alustariippumattoman mobiilisovelluskehityksen tuomia haasteita. Alustariippumaton sovelluskehys PhoneGap on yksi tekniikoista, jossa hyödynnetään selaimien vahvuuksia sovellusten tekemisessä. PhoneGap mahdollistaa sen, että sovelluskehittäjä kirjoittaa sovelluksen kerran ja se on useammalle laitealustalle sopivassa formaatissa. Opinnäytetyössä myös esiteltiin Web Servicen pääpiirteet yleisellä tasolla sekä tutustuttiin muutamiin Web Servicen protokolliin.

Opinnäytetyössä myös tutkittiin PhoneGapin tarjoamia JavaScript-kirjastoja, jotta PhoneGapin mahdollisuudet tulevat paremmin esille. Opinäytetyössä PhoneGapin JavaScript-kirjastot esiteltiin yksinkertaisten esimerkkien avulla. Opinnäytetyössä käytiin läpi jQuery Mobile, jota on käytetty tämän opinnäytetyön sovelluksen käyttöliittymänä.

Opinnäytetyön lopputuloksena valmistui käyttövalmis kuitti- ja ajopäiväkirjatietojen välitys sovellus, jolla voidaan lähettää tietoja Web Servicen läpi halutulle palvelimelle. Sovellus julkaistaan myöhemmin sovelluskaupoissa.

PhoneGapin hyödyntäminen mobiilisovelluskehityksessä osoittautui erityisen hyödylliseksi sen mahdollistaman paketoimisen ja laitteen natiiviominaisuuksien käyttöönoton vuoksi.

Asiasanat: PhoneGap, alustariippumaton sovelluskehys, Web Service, jQuery Mobile

Lahti University of Applied Sciences
Faculty of Technology

MASUDI, SEYAWASH:

Cross-platform mobile application development
Case: Laskuavain Mobile

Bachelor's Thesis in Software Engineering, 52 pages

Spring 2014

ABSTRACT

The thesis dealt with the benefits of the PhoneGap framework in cross-platform mobile application development. The application created in the thesis was carried out for Software Development Company Koodiavain.

The thesis went through challenges that come with cross-platform mobile application development. PhoneGap is a cross-platform mobile application development framework that uses the strengths of the browser in application development. PhoneGap allows the application developer to develop an application which is in suitable format for many platforms. The main features of Web Service were presented in general and a few Web Service protocols were studied in the thesis.

The JavaScript libraries of PhoneGap were also examined to show the potential of PhoneGap better. The libraries were presented in simple examples, and jQuery Mobile, which has been used as user interface in this project, was also introduced.

The final result of the project was a mobile application which allows the user to send data to the server through Web Service. The application will be released later in the application stores.

The exploitation of PhoneGap in application development showed it to be particularly useful because of its packaging feature and easy deployment of the device's native features.

Key words: PhoneGap, cross-platform development, Web Service, jQuery Mobile

SISÄLLYS

LYHENNELUETTELO

CSS	Cascading Style Sheets, verkkosivujen muotoiluun käytettävä tyylikieli.
HTML	HyperText Markup Language, verkkosivujen merkintäkieli.
HTTP	Hypertext Transfer Protocol, selaimien ja WWW-palvelimien käyttämä protokolla tiedonsiirtoon.
JavaScript	Verkkosivujen dynaamisten toiminnallisuuksien lisäämiseen tarkoitettu komentosarjakieli.
jQuery	Selainriippumaton ilmainen ja avoimen lähdekoodin Javascript-kirjasto
jQuery Mobile	UI-sovelluskehys käyttöliittymien luontiin älypuhelimissa ja tableteissa toimiville web-sovelluksille.
SOAP	Simple Object Access Protocol, yksinkertainen ja kevyt XML-muotoinen sanoma, joka voidaan lähettää sovellusten välillä verkon yli ohjelmointikielestä tai tiedonsiirtoprotokollasta riippumatta.
UDDI	Universal Description Discovery and Integration, palvelu, johon voidaan rekisteröidä Web Service -palveluita.
WEB SERVICE	Ohjelmistojärjestelmä, joka mahdollistaa keskenään yhteensopivan tietokoneiden välisen vuorovaikutuksen tietoverkon yli.
WSDL	Web Service Description Language, XML-dokumentti. WSDL:llä kuvataan web-teknologioihin perustuva Web-palvelu.

XML	Extensible Markup Language, laajennettavissa oleva rakenteinen kuvaus- ja merkintäkieli.

1 JOHDANTO

Mobiililaitteet ovat nykypäivänä yleistyneet räjähdysmäisesti ja ovat isossa roolissa ihmisten jokapäiväisessä elämässä. Erilaiset mobiililaitteet, kuten matkapuhelimet, taskutietokoneet ja kannettavat tietokoneet, tarjoavat monenlaisia palveluita opiskeluun, työelämän asioiden hoitamiseen ja esimerkiksi sosiaalisen median hyväksikäyttöön. Näin ollen nykyaikaisella matkapuhelimella ihminen pystyy tekemään lähes kaiken, mitä hän voi tehdä pöytätietokoneella.

Samalla kun mobiililaitteiden lukumäärä on kasvanut jatkuvasti, erilaiset mobiilialustat ja laitetyypit ovat myös lisääntyneet. Mobiilialustojen jakautuminen on saanut aikaan sen, että sovelluskehittäjän on yhä vaikeampi tavoittaa suurta mobiilikäyttäjäkuntaa omalla sovelluksellaan. Sovelluksen kehittäminen eri laitetyyppien ja alustojen omilla ohjelmointiympäristöillä vaatii monenkertaisen osaamisen ja työn saada sovellus toimimaan. Jos sovelluskehittäjä haluaa toteuttaa sovelluksen usealle eri mobiilialustalle perinteisellä tavalla, niin sovellus pitäisi kehittää ja toteuttaa jokaiselle alustalle erikseen.

Alustariippumaton mobiilisovelluskehys PhoneGap mahdollistaa sovelluksen toimivuuden usealla mobiilialustalla ilman, että sovelluksesta tehtäisiin eri alustalle oma versio. PhoneGap on mobiilialusta, jossa ohjelmointi toteutetaan JavaScriptiä sekä HTML:ää käyttäen. PhoneGapin avulla sovellus voidaan kehittää yleisillä web-tekniikoilla. Lisäksi PhoneGap mahdollistaa sovellukselle laitetasoisten ominaisuuksien, kuten kameran, sijainnin ja laitteen muistin käytön yleisten web-tekniikoiden avulla.

Opinnäytetyön toimeksiantajana on Ohjelmistotalo Koodiavain Oy.Yritys on perustettu vuonna 2006, ja sen toimialana on laitteisto- ja ohjelmistokonsultointi. Ohjelmistotalo Koodiavaimen päätuotteena on ohjelmistokehitys, konesali- ja integraatiopalvelut sekä konsultointi.

Tämän opinnäytetyön tavoitteena on suunnitella ja toteuttaa Ohjelmistotalo Koodiavain Oy:lle mobiilisovellus, jonka käyttötarkoituksena on kuitin- tai ajokilometritietojen välittäminen Web-palvelun läpi halutulle palvelimelle. Mobiilisovelluksella pystytään keräämään tietoja, ottamaan kuva kuitista ja lähettämään se palvelimelle.

Tässä opinnäytetyössä luvussa 2 esitellään alustariipumaton mobiilisovelluskehitys yleisella tasolla sekä käsitellään, mitä haasteita alustariipumaton mobiilisovelluskehitys tuo mukanaan. Luvussa 3 käsitellään PhoneGap-sovelluskehystä ja sen tarjoamia JavaScript-kirjastoja. Luku 3 käsittelee myös pilvipalveluna toimivaa PhoneGap Buildia. Luvussa 4 esitellään XML yleisellä tasolla sekä XML-dokumentin rakennetta kuvaillaan esimerkkien avulla. Lisäksi luvussa 4 käsitellään yleiset Web servicen -protokollat ja niiden tarjoamat rajapinnat. Luvussa 5 on kerrottu sovelluksen käytännön osuuden toteutuksesta. Luvussa 6 tehdään yhteenveto ja loppupohdinta tästä opinnäytetyöstä.

2 ALUSTARIIPUMATON MOBIILISOVELLUSKEHITYS

2.1 Määritelmä

Alustariippumaton mobiilisovelluskehitys on teknologia, jossa sovelluskehitys ei ole riippuvainen jostain tietystä ympäristöstä. Ympäristöllä tarkoitetaan esimerkiksi käyttöjärjestelmää tai laitealustaa. Alustariipumattoman mobiilisovelluskehityksen tavoitteena on tuottaa sovelluksia, jotka toimivat useammilla alustoilla. Alustariippumaton mobiilisovellus ei kuitenkaan tarkoita sitä, että se toimisi suoraan kaikilla alustoilla tai käyttöjärjestelmillä vaan, että sovelluksessa ei ole suoria riippuvuuksia laitealustaan tai käyttöjärjestelmään. Alustariippumattomassa mobiilisovelluskehityksessä sovelluksen riippuvuudet laitealustaan tai käyttöjärjestelmään voidaan joko ohjelman lähdekoodin käännön tai ajon aikana muuntaa yhteensopiviksi.

2.2 Haasteet

Mobiilikäyttöjärjestelmien suuri määrä on alustariippumattoman mobiilisovelluskehityksen suurimpia haasteita. Eri mobiilikäyttöjärjestelmät ovat ominaisuuksiltaan erilaisia, ne toimivat omalla alustallaan ja tukevat myös eri ohjelmointikieliä (Ghatol & Patel 2012). Taulukossa 1 on esitetty maailman tämän hetken kuusi suosituinta mobiilikäyttöjärjestelmää. Taulukossa 1 esitetyt käyttöjärjestelmät toimivat omalla alustallaan, jotka tukevat eri ohjelmointikieliä. Taulukosta 1 nähdään, että Androidin ja iOS:n markkinaosuus vuonna 2013 on 93,2 % maailman kaikista mobiilikäyttöjärjestelmistä.

TAULUKKO 1. Maailman tämän hetken suosituimmat mobiilikäyttöjärjestelmät (Gartner 2013)

Käyttöjärjestelmä	Markkinaosuus(%) 2013	Markkinaosuus(%) 2012
Android	79.0	64.2
iOS	14.2	18.8
Windows Phone	3.3	2.6
Black Berry	2.7	5.2
Bada	0.4	2.7
Symbian	0.3	5.9

Sovelluskehittäjän näkökulmasta tilanne on vielä haastavampi. Jokainen esitetyistä alustoista tarvitsee oman ohjelmointirajapintansa (SDK) ja oman kehitysympäristönsä (IDE). Kaikki SDK:t ja IDE:t eivät toimi kaikissa käyttöjärjestelmissä, koska ne ovat laitevaatimuksiltaan erilaisia. Esimerkiksi jos sovelluskehittäjä haluaa tehdä iOS-käyttöjärjestelmälle sovelluksia, niin hänellä on oltava Mac-tietokone sekä Xcode-kehitysympäristö asennettuna tietokoneeseen.

Taulukosta 2 nähdään, että mobiilikäyttöjärjestelmät tukevat eri ohjelmointikieliä, joiden määrä on kasvanut. Kuten taulukko 2 osoittaa, sovelluskehittäjän tulisi osata Java-, Objective-C-, C++- ja C#-ohjelmointikieliä, jotta hän voisi kehittää sovelluksia kaikille taulukossa 1 mainituille käyttöjärjestelmille.

TAULUKKO 2. Sovelluskehityksen vaatimukset eri mobiilialustoissa (Ghatol & Patel 2011)

Mobiilikäyttö-järjestelmä	Tietokoneen käyttöjärjestelmä	Kehitys-ympäristö	Ohjelmointikieli
Android	Windows/Mac/Linux	Esclipe/Java + Android-lisäosa	Java
iOS	Mac	Xcode	Objective C
Windows Phone	Windows	Visual Studio	C#/.NET/Silverlight tai WPF
Black Berry	Windows	Esclipe / JDE, Java	Java
Bada	Windows	Bada	C++
Symbian	Windows/Mac/Linux	Carbide.c++	C++

2.3 Vaihtoehtoinen ratkaisu

Verkkoselainten nopea kehitys ja yhtenäisten standardien noudattaminen mahdollistaa sen, että modernit älypuhelimet sisältävät kehittyneen verkkoselaimen. Taulukosta 3 nähdään, että kuudesta suosituimmasta mobiilialustasta viidessä on WebKit-pohjainen verkkoselain. Poikkeuksena on Windows Phone, jolla on oma Internet Explorer 7 -pohjainen verkkoselain. Kuten taulukosta 3 käy ilmi, verkkoselainta voidaan hyödyntää sovelluskehityksessä ja selainsovellus voisi toimia kaikissa mobiilialustoissa.

Alustariippumaton ohjelmistokehys nimeltään PhoneGap hyödyntää verkkoselainta yhteisenä alustana sovelluskehityksessä. Sen avulla sovelluskehityksessä käytetään hyväksi HTML5-, CSS3- ja JavaScript-

ohjelmointikieliä. PhoneGap tarjoaa sovelluskehittäjälle toisena ominaisuutena mahdollisuuden käyttää laitealustan natiivirajapintoja, koska JavaScript mahdollistaa kutsut natiivikoodiin (Ghatol & Patel 2011).

TAULUKKO 3. Mobiilikäyttöjärjestelmien verkkoselaimet (Ghatol & Patel 2011)

Mobiilikäyttöjärjestelmä	Verkkoselain
Android	Webkit-pohjainen
iOS	Webkit-pohjainen
Windows Phone	IE 7-pohjainen
Black Berry	Webkit-pohjainen
Bada	Webkit-pohjainen
Symbian	Webkit-pohjainen

PhoneGapilla kehitetty sovellus on toisin sanoen verkkosivu, johon otetaan yhteyttä, kun käyttäjä aloittaa sovelluksen käytön. Sovelluksen käynnistyessä se lataa ensisijaisesti paikallisen.html-tiedoston, jolloin käyttäjälle aukeaa selaimessa toimiva sovellus. Mobiilialustojen verkkoselainmoottorien ollessa samanlaisia mobiilisovellus näyttää samalta pieniä eroavaisuuksia lukuun ottamatta kaikilla alustoilla. (Wargo 2012.)

3 PHONEGAP

3.1 Yleistä

PhoneGap on Adobe Systemsin omistama ilmainen ja avoimen lähdekoodin perustuva HTML5-sovelluskehys, joka aloitti toimintansa vuonna 2008. PhoneGap mahdollistaa mobiilisovellusten kehittämisen yleisten web-tekniikoiden (HTML, CSS, JavaScript) avulla. Tämä mahdollistaa sen, että sovellusten kehittäminen eri laitealustoille ei vaadi sovelluskehittäjältä tietämystä kaikkien laitealustojen käyttämistä ohjelmointikielistä. PhoneGap mahdollistaa sen, että sovelluksesta pystyy tekemään version usealle laitealustalle, kuten iPhonelle, Androidille ja Windows Phonelle vähäisillä ohjelmakoodin muutoksilla. (Ghatol & Patel 2012.)

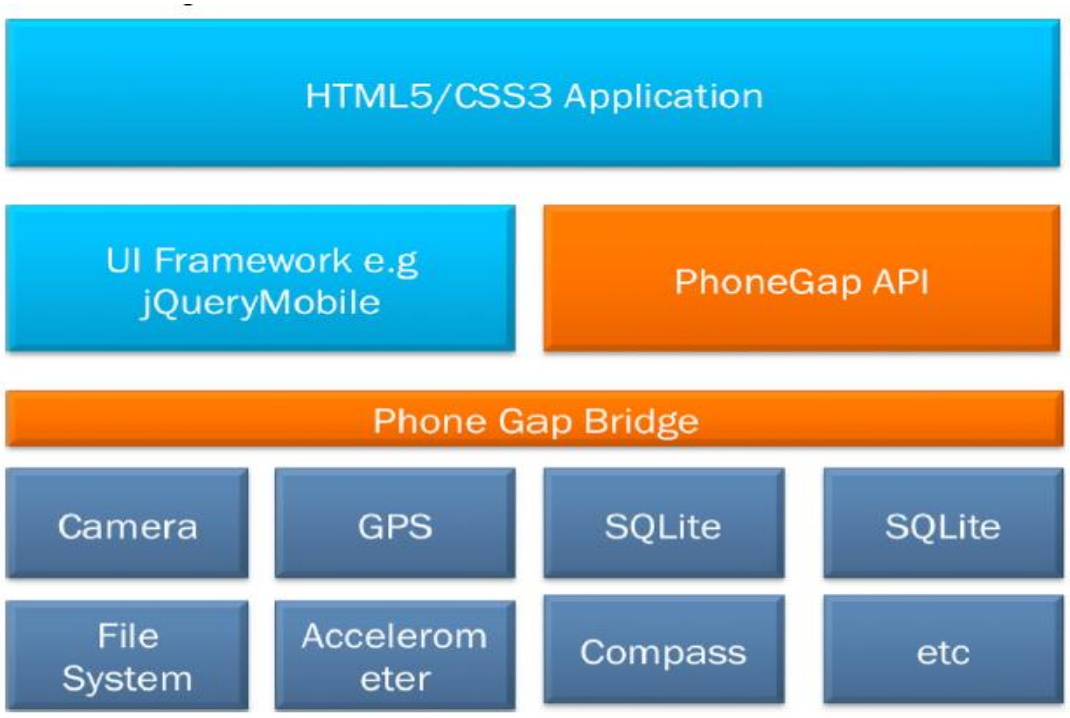
PhoneGapin tarjoaman JavaScript API -rajapinnan avulla pystytään kehittämään niin kutsuttuja hybridisovelluksia käyttäen apuna laitteen natiiviominaisuuksia miltei kaikille mobiilialustoille: iOS, Android, Windows Phone, BlackBerry, WebOS, Symbian ja Bada (PhoneGap 2014). Hybridisovellukset käyttävät laitteen selainta toimiakseen, mutta näyttävät ulospäin natiivilta sovelluksilta. Hybridisovellusten ja selainpohjaisten sovellusten ero on siinä, että hybridisovellukset ovat hyväksyttyjä sovelluskaupoissa. Sovellus saadaan toimimaan kaikille yllä mainituilla mobiilialustoilla, kun tehdään yksi sovellus HTML-kielellä, CSS:llä ja JavaScriptillä. Tämän jälkeen kaikki kooditiedostot käännetään PhoneGapin avulla jokaiselle alustalle sopivaksi.

Kuviossa 1 esitellään PhoneGap-sovelluskehyksen arkkitehtuuri. Kuviosta voidaan nähdä, että PhoneGap-arkkitehtuurissa on kaksi olennaista osaa, jotka voidaan jaotella seuraavasti:

1. Käyttöliittymä, jossa voidaan käyttää esimerkiksi jQuery Mobile -kirjastoa.
2. PhoneGap API, jonka avulla voidaan käyttää laitteen natiiviominaisuuksia.

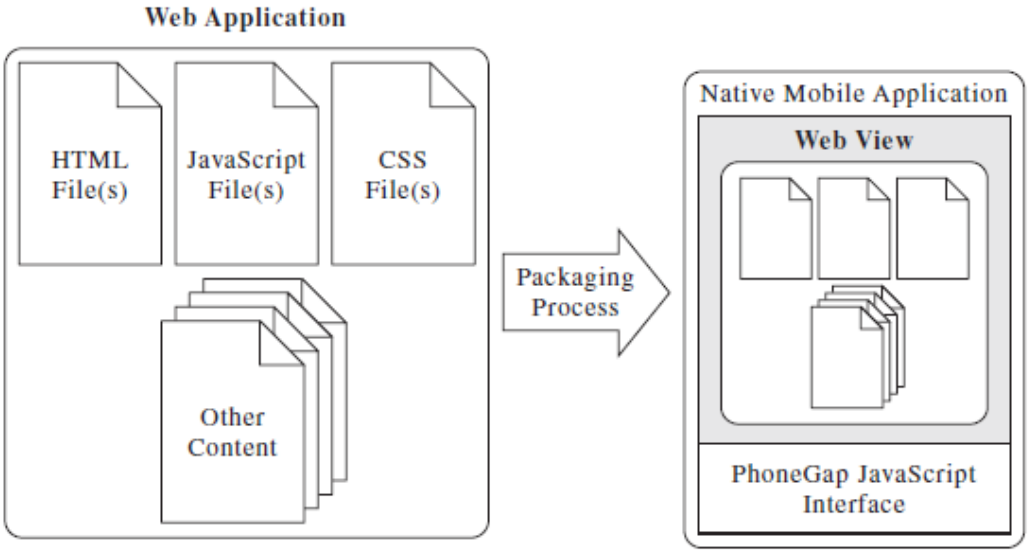
PhoneGap-sovelluksen käyttöliittymä (UI), looginen osuus sekä osa, joka kommunikoi palvelimen kanssa, ovat HTML/JavaScript-pohjaisia. Osa, joka vastaa laitteen kanssa kommunikoinnista, pohjautuu laitealustan natiivikieleen.

PhoneGap toimii puhelimen toiminnallisuuksien, kuten kameran, GPS:n tai yhteystietojen ja HTML5-, CSS- ja JavaScript-ohjelmointikielillä kirjoitetun sovelluksen välissä mahdollistaen niiden keskinäisen kommunikoinnin. PhoneGap tarjoaa kerroksen sovelluksen JavaScript-osuuden ja natiivilaiteympäristön välille (Ghatol & Patel 2012).



KUVIO 1. PhoneGapin arkkitehtuuri (Ghatol & Patel 2012)

PhoneGap tarjoaa sovelluskehittäjälle myös toisen tärkeän ominaisuuden. Aiemmin todettiin PhoneGapin mahdollistavan natiivisovellusten kehittämisen yleisten web-tekniikoiden avulla. PhoneGap paketoi sovelluskehittäjän kehittämät tiedostot kuvion 2 mukaisesti mobiilialustan natiivisovellukseksi. Tämä paketointiominaisuus mahdollistaa sen, että sovelluksen lisääminen eri mobiilialustojen sovelluskauppaan on mahdollista. PhoneGapillä kehitetty sovellus voi olla staattinen, dynaaminen tai näiden kahden sekoitus. Staattisessa sovelluksessa sisältö luodaan asennusvaiheessa ja sovelluksen sisältö löytyy kehittäjän tuottamista tiedostoista. Dynaamisessa sovelluksessa sisältö ladataan ulkopuoliselta palvelimelta, jolloin sovelluksen sisällön päivittäminen on mahdollista ilman sovelluksen uudelleen asentamista.



KUVIO 2. PhoneGap-sovelluksen paketointiprosessi (Wargo 2012)

3.2 PhoneGapin kirjastot

PhoneGap-kirjasto sisältää monia eri rajapintoja, jotka ovat tiettyihin toimintohin tarkoitettuja. Tässä luvussa tutustutaan PhoneGapin tarjoamiin kirjastoihin ja käydään läpi ne esimerkkien avulla.

Jokainen PhoneGapin tarjoama kirjasto sisältää funktioita, joilla tarvittavat toiminnot voidaan toteuttaa. Kirjastojen toimintoja käytetään kutsumalla kirjastoissa toteutettuja funktioita. Funktiolle annetaan parametreina funktio, joka käsittelee onnistuneen funktiokutsun. Toisena parametrina annetaan funktio, joka käsittelee epäonnistuneen funktiokutsun, ja kolmantena parametrina annetaan funktion valinnaiset asetukset, joilla voidaan muokata funktion toimintaa.

PhoneGapin kirjastoilla on omat oikeusmääritteet, joita asetetaan sovelluksen config.xml-tiedostossa, jotta kirjastoja voidaan käyttää sovelluksessa. Syynä tähän ovat käyttäjärjestelmän asettamat tiukat säännöt ohjelmille. Kuviossa 3 on esitetty PhoneGapin config.xml-tiedosto.

```
<?xml version="1.0" encoding="UTF-8"?>
<widget xmlns     = "http://www.w3.org/ns/widgets"
        xmlns:gap = "http://phonegap.com/ns/1.0"
        id        = "com.phonegap.laskuavain"
        version   = "1.0.0">


    <preference name="phonegap-version" value="3.3.0" />

    <preference name="fullscreen"       value="true" />
    <preference name="orientation"      value="portrait" />
    <preference name="target-device"    value="universal" />

    <preference name="auto-hide-splash-screen" value="false" />
    <preference name="FadeSplashScreenDuration" value="5000"/>
    <preference name="fade-splash-screen-duration" value="5" />


    <icon src="icon.png" />

    <icon src="res/icon/android/ldpi.png"  gap:platform="android" gap:density="ldpi"  />
    <icon src="res/icon/android/mdpi.png"  gap:platform="android" gap:density="mdpi"  />
    <icon src="res/icon/android/hdpi.png"  gap:platform="android" gap:density="hdpi"  />
    <icon src="res/icon/android/xhdpi.png" gap:platform="android" gap:density="xhdpi" />
    <icon src="res/icon/ios/icon.png"          gap:platform="ios" width="57"  height="57"  />
    <icon src="res/icon/ios/icon-72_at_2x.png" gap:platform="ios" width="144" height="144" />
    <icon src="res/icon/windows-phone/icon.png"  gap:platform="winphone" />


    <gap:splash src="res/screen/android/screen-xhdpi-portait.png"  gap:platform="android" gap:density="ldpi" />
    <gap:splash src="res/screen/android/screen-xhdpi-portait.png" gap:platform="android" gap:density="xhdpi" />
    <gap:splash src="res/screen/blackberry/screen-225.png"         gap:platform="blackberry" />
    <gap:splash src="res/screen/ios/screen-iphone-portrait.png"    gap:platform="ios"     width="320" height="480" />
    <gap:splash src="res/screen/windows-phone/screen-portrait.jpg" gap:platform="winphone" />
</widget>
```

KUVIO 3. PhoneGapin cofig.xml-tiedosto

PhoneGapin kaikista kirjastoista saa lisätietoa PhoneGapin dokumentaatioista, jotka löytyvät jokaisesta PhoneGapin versiosta. Tässä opinnäytetyössä on pyritty esittelemään PhoneGapin kirjastot mahdollisimman yksinkertaisesti ja selkeästi. PhoneGapin kirjastojen tuet eri älypuhelinkäyttöjärjestelmiin muuttuvat joka versiossa hieman, niinpä kirjastoja ei käydä liian yksintyiskohtaisesti lävitse. Tässä opinnäytetyössä PhoneGapin kirjastojen esittelyssä on käytetty PhoneGap-versio 3.3.0 dokumentaatiota.

3.2.1 Kiihtyvyysmittari

Accelerometer-kirjaston avulla voidaan seurata laitteen kiihtyvyyttä x-, y- ja z-koordinaattien suhteen. Käytännössä Accelerometer-kirjastoa voi käyttää laitteen liikkeen, kallistuksen ja kiihtyvyyden mittaamiseen ja tarkkailuun. Kuviossa 4 ilmoitetaan onnistuneena funktiokutsuna laitteen sen hetkinen kiihtyvyys x-, y- ja z-suunnassa. Jos funktiokutsu epäonnistuu, näytetään käyttäjälle virheilmoitus.

```
4   navigator.accelerometer.getCurrentAcceleration(onSuccess, onError);
5
6   function onSuccess(acceleration) {
7       alert('Kiihtyvyys X: ' + acceleration.x + '\n' +
8             'Kiihtyvyys Y: ' + acceleration.y + '\n' +
9             'Kiihtyvyys Z: ' + acceleration.z + '\n');
10  };
11
12  function onError(error) {
13      alert('Tapahtui virhe' + error);
14  };
15
```

KUVIO 4. Laitteen kiihtyvyyden näyttäminen Accelerometer-kirjaston avulla

3.2.2 Kamera

Camera-kirjaston avulla saadaan yhteys laitteen oletuskamerasovellukseen. Camera-kirjasto tarjoaa toimintoja, joiden avulla voidaan hakea kuvia tai videoita suoraan sovellukseen laitteen kamerasta, kuvakirjastosta tai kuva-albumista. Tiedoston hakemiseen käytetään camera.getPicture()-metodia, joka hoitaa kuvan tai videon hakemisen sille annetuista asetuksista riippuen. Kameralle voidaan asettaa seuraavat asetukset:

- **quality**: kuvanlaatu 0 - 100 asteikolla
- **destinationType**: paluuarvona based64 enkoodattu merkkijono tai tiedoston tallennuspolku
- **sourceType**: kuvan lähde, joko kuvakirjasto, kamera tai kuva-albumi
- **allowEdit**: sallitaan yksinkertainen editointi ennen kuin kuva valitaan

- **encodingType**: kuvan enkoodaus, jpeg- tai png-muodossa
- **targetWidth**: kuvan leveys, johon kuva skaalataan
- **targetHeight**: kuvan korkeus, johon kuva skaalataan
- **savetoPhotoAlbum**: määrittää sen, tallennetaanko kuva tai video albuumin vai ei
- **correctOrentation**: kääntää kuvan laitteen asemaan nähden oikeaksi.

Kuviossa 5 laitteen kameralla otettu kuva näytetään DOM-elementissä, jos kuvan ottaminen onnistuu. Mikäli kuvan ottaminen ei onnistu, näytetään käyttäjälle virheilmoitus. Otetulle kuvalle on asetettu leveydeksi ja korkeudeksi 150 pikseliä. Lisäksi kuvaa voidaan editoida kuvan ottamisen jälkeen sekä kuva tallennetaan laitteen kuva-albumiin.

```
4   navigator.camera.getPicture(onSuccess, onFail, {
5       quality: 50,
6       destinationType: Camera.DestinationType.DATA_URL,
7       allowEdit: true,
8       targetWidth: 150,
9       targetHeight: 150,
10      savetoPhotoAlbum: true
11  });
12
13  function onSuccess(imageData) {
14      var image = document.getElementById('myImage');
15      image.src = "" + imageData;
16  }
17
18  function onFail(message) {
19      alert('Tapahtui virhe ' + message);
20  }
21
```

KUVIO 5. Kuvan ottaminen Camera-rajapintaa käyttäen

3.2.3 Kompassi

Compass-kirjastolla saadaan noudettua suunta, johon laitteen kompassi osoittaa. Laitteen osoittama suunta saadaan asteina (0 – 355.9). Compass-kirjastolla

voidaan joko tarkkailla tämänhetkistä suuntaa tai kuunnella suunnan vaihteluja määrätyllä aikavälillä. Kuviossa 6 ilmoitetaan käyttäjälle laitteen kompassin osoittama suunta, jos funktiokutsu onnistuu. Kutsun epäonnistuessa näytetään virheilmoitus.

```
3     navigator.compass.getCurrentHeading(onSuccess, onError);
4
5     function onSuccess(heading) {
6         alert('Suunta: ' + heading.magneticHeading);
7     }
8     function onError(compassError) {
9         alert('Tapahtui virhe: ' + compassError.code);
10    }
11
```

KUVIO 6. Laitteen kompassin suunnan näyttäminen Compass-kirjastoa käyttäen

3.2.4 Yhteydet

Connection-kirjastolla saadaan tietoa laitteen puhelinyhteyksien ja langattoman internetyhteyden tilasta. Kuviossa 7 on esitetty esimerkki Connection-kirjaston käytöstä, jossa käyttäjälle näytetään laitteen senhetkisen verkkoyhteyden tyyppi.

```
4     var networkState = navigator.connection.type;
5
6     var states = {};
7     states[Connection.UNKNOWN]  = 'Ei tietoa';
8     states[Connection.ETHERNET] = 'Ethernet ';
9     states[Connection.WIFI]     = 'Langaton';
10    states[Connection.CELL_2G]  = ' 2G ';
11    states[Connection.CELL_3G]  = ' 3G ';
12    states[Connection.CELL_4G]  = ' 4G ';
13    states[Connection.NONE]     = 'Ei yhteyttä ';
14
15    alert('Yhteyden tyyppi: ' + states[networkState]);
16
```

KUVIO 7. Verkkoyhteyden tyypin näyttäminen Connection rajapinnan avulla

3.2.5 Kontaktit

Contacts-kirjastolla saadaan yhteys laitteen yhteystietoihin. Sen avulla voidaan luoda uusia yhteystietoja tai hakea jo olemassa olevia yhteystietoja laitteen tietokannasta. Kuviossa 8 lisätään uusi kontakti laitteen yhteystietoihin ja ilmoitetaan käyttäjälle, jos kontaktin lisääminen onnistui.

```
5   var contact = navigator.contacts.create();
6   contact.displayName = "Matti";
7   contact.nickname = "Matti";
8
9   var name = new ContactName();
10  name.givenName = "Matti";
11  name.familyName = "Virtanen";
12  contact.name = name;
13
14
15  contact.save(onSuccess,onError);
16
17  function onSuccess(contact) {
18      alert("Tallennus onnistui");
19  };
20
21  function onError(contactError) {
22      alert("Tapahtui virhe: " + contactError.code);
23  };
24
```

KUVIO 8. Yhteystiedon tallentaminen Contacts-kirjaston avulla

3.2.6 Laitetiedot

Device-kirjastolla voidaan päästää käsiksi laitteen tietoihin, kuten laitteen käyttöjärjestelmään, laitteen malliin, PhoneGap-versioon ja laitteen uniikkiin tunnisteeseen.

Kuviossa 9 on esitetty esimerkki Device-kirjaston käytöstä ja sen tarjoamista tiedoista.

```
5
6   var element = document.getElementById('deviceProperties');
7   element.innerHTML = 'Malli: '    + device.model    + '<br />'
8           +   'Käyttöjärjestelmä: ' + device.platform + '<br />'
9           +   'ID: '     + device.uuid     + '<br />'
10          +   'Versio: '  + device.version  + '<br />';
11
12
13
```

KUVIO 9. Laitteen tietojen näyttäminen Device-kirjaston avulla

3.2.7 Tapahtumat

Events-kirjastolla käsitellään PhoneGapin sisältämät tapahtumat. Sen avulla voidaan esimerkiksi asettaa jollekin tapahtumalle kuuntelija ja reagoida siihen, kun tapahtuma laukeaa. Tapahtumiin päästään käsiksi, kun PhoneGapin deviceready-tapahtuma on tapahtunut. Tämä deviceready laukeaa, kun PhoneGap on kokonan latautunut. PhoneGap sisältää muun muassa seuraavat tapahtumankäsittelijät:

- **deviceready**: tapahtuu, kun PhoneGap on kokonaan ladattu
- **pause**: tapahtuu, kun sovellus menee taustalle
- **resume**: tapahtuu, kun sovellus palaa takaisin taustalta
- **online**: tapahtuu, kun sovellukselle aukeaa internetyhteys
- **offline**: tapahtuu, kun sovelluksen internetyhteys katkeaa
- **backbutton**: tapahtuu, kun käyttäjä painaa laitteen back-näppäintä
- **batterycritical**: tapahtuu, kun laitteen akku on saavuttanut kriittisen tason
- **menubutton**: tapahtuu, kun käyttäjä painaa laitteen menu-näppäintä
- **searchbutton**: tapahtuu, kun käyttäjä painaa laitteen search-näppäintä
- **volumedownbutton**: tapahtuu, kun painetaan näppäintä äänen voimakkuuden vähentämiseksi.
- **volumeupbutton**: tapahtuu, kun painetaan äänen nostamisen näppäintä.

Kuviossa 10 on esimerkki PhoneGapin Events-kirjaston käytöstä. Esimerkissä asetetaan deviceready tapahtumaan kuuntelija, jonka sisällä kutsutaan toista funktiota, kun PhoneGap on ladattu kokonaan.

```
9
10      // asetetaan kuuntelija devicerady tapahtumaan
11
12      function onLoad() {
13          document.addEventListener("deviceready", onDeviceReady, false);
14      }
15
16      // PhoneGap on latautunut kokonaan
17      // näytetään käyttäjälle ilmoitus siitä, että PhoneGap on kokonaan latautunut
18      function onDeviceReady() {
19          navigator.notification.alert(
20          'PhoneGap on valmis', null ,'Ilmoitus','Ok'
21          );
22      }
```

KUVIO 10. Esimerkki Events-kirjaston käytöstä

3.2.8 Tiedosto

File-kirjasto on toteutettu W3C:n File API -määräysten mukaisesti. Sen avulla voidaan lukea, kirjoittaa ja ladata palvelimelle tiedostoja. File-kirjastolla voidaan myös listata hakemistoja tai tallentaa tiedostoja tilapäiseen tai pysyvään tallennuspaikkaan.

Kuviossa 11 näytetään käyttäjälle text.txt-tiedoston sisältö File-kirjaston avulla, jos funktiokutsu onnistui. Mikäli funktiokutsu epäonnistuu, näytetään käyttäjälle virheilmoitus.

```
8
9       window.requestFileSystem(LocalFileSystem.PERSISTENT, 0, gotFS, fail);
10
11      function gotFS(fileSystem) {
12          fileSystem.root.getFile("text.txt", null, gotFileEntry, fail);
13      }
14      function fail(error) {
15          alert('Tapahtui virhe:' + error.code);
16      }
17      function gotFileEntry(fileEntry) {
18          fileEntry.file(gotFile, fail);
19      }
20
21      function readAsText(file) {
22          var reader = new FileReader();
23          reader.onloadend = function(evt) {
24
25              alert('Tiedoston sisältö : ' + evt.target.result);
26          };
27          reader.readAsText(file);
28      }
29
```

KUVIO 11. Tiedoston lukeminen File-kirjaston avulla

3.2.9 Paikannus

Geolocation-kirjasto tarjoaa tietoja laitteen sijainnista. Laitteen sijainti voidaan määrittää GPS:n, IP-osoitteen, puhelinyhteyksien, langattoman internetin tai Bluetoothin perusteella.

Laitteen nykyinen sijainti saadaan kutsumalla geolocation.getCurrentPosition-funktiota. Tämän funktion palauttama Position-olio sisältää sijaintitiedot, kuten leveysasteen, pituusasteen, sijainnin tarkkuuden, korkeuden, korkeuden tarkkuuden, nopeuden ja suunnan.

Kuviossa 12 on esimerkki, miten Geolocation-kirjastolla voidaan näyttää nykyinen paikkatieto. Mikäli paikkatietojen näyttämisessä tapahtuu virhe, näytetään käyttäjälle virheilmoitus.

Laitteen sijainnin vaihtelua voidaan seurata geolocation.watchPosition-funktiolla tietyllä aikavälillä. Laitteen saadessa uusi sijainti geolocation.watchPosition-funktio palauttaa sen hetkisen sijainnin tiedot.

```
3
4       // Paikkatietojen haku
5       navigator.geolocation.getCurrentPosition(onSuccess, onError);
6
7       // kun haku onnistuu, näytetään paikkatiedot
8       var onSuccess = function(position) {
9           alert('Latitude: '          + position.coords.latitude          + '\n' +
10                'Longitude: '         + position.coords.longitude         + '\n' +
11                'Altitude: '          + position.coords.altitude          + '\n' +
12                'Accuracy: '          + position.coords.accuracy          + '\n' +
13                'Altitude Accuracy: ' + position.coords.altitudeAccuracy  + '\n' +
14                'Heading: '           + position.coords.heading           + '\n' +
15                'Speed: '             + position.coords.speed             + '\n');
16      };
17
18      // kun tapahtuu virhe, näytetään virheilmoitus
19      function onError(error) {
20          alert('Tapahtui virhe: ' + error.message + '\n');
21      }
22
23
```

KUVIO 12. Paikkatietojen haku Geolocation-kirjaston avulla

3.2.10 Media

Media-kirjastolla käsitellään äänitiedostoon liittyviä toimintoja. Sen avulla voidaan esimerkiksi toistaa, pysäyttää ja lopettaa äänitiedoston toistoa. Media-kirjasto tarjoaa myös äänen nauhoittamiseen tarvittavat toiminnot.

Kuviossa 13 on esimerkki Media-kirjaston käytöstä, jossa äänitiedosto haetaan palvelimelta ja toistetaan.

```
3
4       var my_media_file = null;
5
6       // haetaan palvelimetla äänitiedosto ja toistetaan se
7       function playAudio() {
8
9           my_media_file = new Media("http://audio.ibeat.org/content/p1rj1s/p1rj1s_-_rockGuitar.mp3",
10                                    onSuccess, onError);
11
12          // soitetaan haettu äänitiedosto
13          my_media_file.play();
14
15      }
16      //  Äänitiedoston haku onnistui
17      function onSuccess() {
18          console.log("Äänitiedoston toisto onnistui");
19      }
20      // Äänitiedoston haku ei onnistunut, näytetään virheilmoitus
21      function onError(error) {
22          alert(' Tapahtui virhe: ' + error.message + '\n');
23      }
24
```

KUVIO 13. Äänitiedoston toisto Media-kirjaston avulla

3.2.11 Ilmoitukset

Notification-kirjastolla ilmoitetaan käyttäjälle tapahtumista värinällä, piippauksella tai ilmoituksella. Sen avulla voidaan käyttäjälle näyttää huomio- tai vahvitusikkuna sovelluksessa. Kuviossa 14 on esimerkki Notification-kirjaston Confirm-funktiosta, jolla näytetään käyttäjälle vahvistusikkuna, jossa on kaksi nappia. Käyttäjälle näytetään hänen painamansa napin indeksi, jos funktiokutsu onnistui.

```
8
9       function onConfirm(buttonIndex) {
10          alert('You selected button ' + buttonIndex);
11      }
12
13      function showConfirm() {
14          navigator.notification.confirm(
15          'Tämä viesti näkyy käyttäjälle',
16          onConfirm,
17          'Viestin otsikko',
18          ['Nappi 1','Nappi 2']
19          );
20      }
21
```

KUVIO 14. Vahvistusikkunan näyttäminen

3.2.12 Tietojen talletus

Storage-kirjastolla voidaan tallentaa tietoja laitteen muistiin. Tallentamis-menetelmänä on kolme erilaista vaihtoehtoa, jotka ovat paikallistietokanta (SQLite Database), paikallismuisti (LocalStorage) ja istuntomuisti (sessionStorage). Paikallisotietokannan avulla tietoja voidaan tallentaa taulukoihin. Tietokannan taulukoihin voidaan lisätä tietoja, päivittää tai poista tietoja erilaisten SQL-lauseiden avulla. Paikallismuistiin tallennetaan tietoja avain-arvopareina. Istuntomuistin toiminta on sama kuin paikallismuistin, mutta eroaa paikallismuistista siinä, että tallennettu tieto poistuu istuntomuistista, kun sovellus sammutetaan. Tallennus paikallismuistiin tai paikallistietokantaan ovat taas pysyviä tallennusratkaisuja. Kuvio 15 esittää tietojen tallentamisen paikallismuistiin. Esimerkissä tallennetaan arvo BMW paikallismuistiin merkki-avaimelle ja tämän jälkeen haetaan se paikallismuistista ja näytetään se käyttäjälle.

```
8       var key = 'merkki',
9       value = 'BMW';
10
11      window.localStorage.setItem(key, value);
12
13      value = window.localStorage.getItem(key);
14
15      if (value) {
16
17          alert("Auton merkki on :" + value);
18      }
19
```

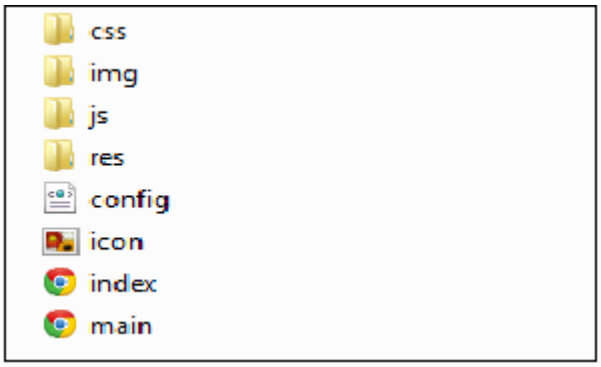
KUVIO 15. Tietojen tallentaminen paikallismuistiin Storage-kirjaston avulla

3.3 PhoneGap build

PhoneGapin pilvipalvelu tarjoaa ohjelmakoodin kääntämiseen eri mobiililaitteille sopivaksi pilvipalveluna PhoneGap Buildin. Sen avulla sovelluskehittäjän kehittämä mobiilisovellus saadaan toimimaan natiivisovelluksen mukaisesti eri mobiililaitteissa. PhoneGap Buildin ansiosta sovelluskehittäjän ei tarvitse asentaa kaikille mobiilialustoille omia kehitysympäristöjä, vaan ohjelmakoodi käännetään PhoneGapin tuetuille käyttöjärjestelmille sopivaksi pilvipalvelussa. (Wargo 2012.)

PhoneGap Build -pilvipalvelun käyttäminen vaatii sovelluskehittäjältä palveluun rekisteröitymisen, josta on sekä ilmainen että maksullinen vaihtoehto. Ilmaisella tilillä kehittäjä voi ladata sovelluksen lähdekoodit suoraan Github-versiohallintajärjestelmästä, jolloin sovelluksesta tulee julkinen ja kääntämismäärää ei ole rajoitettu. Ilmaisella tilillä voi myös ylläpitää yhtä yksityistä sovellusta, kun taas maksullinen tili nostaa yksityisten sovellusten määrän 25:een. Maksullisella tilillä avoimen lähdekoodin sovelluksien kääntämistä ei ole rajoitettu. Yksityisen sovelluksen ohjelmakoodit ladataan PhoneGap Buildiin Zip-pakettina.

Kuviossa 16 on esitetty yksityisen sovelluksen hakemistorakenne, joka voidaan ladata PhoneGap Buildiin arkistoituna Zip-pakettina. Hakemistorakenteessa on config.xml-tiedosto, jossa määritellään muun muassa, mitä PhoneGap-versiota käytetään sovelluksen kääntämisessä ja mitä laitteita sovellus tukee.



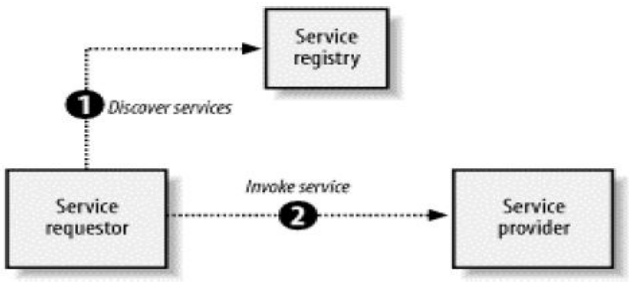
KUVIO 16. Yksityisen PhoneGap-sovelluksen hakemistorakenne

4 WEB SERVICE

4.1 Web Service yleisesti

W3C:n määritelmän mukaan Web Service on ohjelmistokokonaisuus, jonka ulkoiset rajapinnat on määritelty ja kuvattu XML-kielellä. Ohjelmistokokonaisuuden tarjoamat palvelut on löydettävissä verkosta, ja ohjelmistoa voidaan tunnistaa sen URI-osoitteen perusteella. Muut sovellukset ja ohjelmisto voivat hyödyntää ja käyttää Web-palvelun tarjoamia palveluita XML-pohjaisten protokollien avulla. XML-pohjaisten protokolliin kuuluuvat seuraavat kolme komponenttia: Simple Access Protocol eli SOAP, Web Services Description Language eli WSDL sekä Universal Description Discovery and Integration eli UDDI. (W3C 2004.)

Web Service koostuu palvelun tuottajasta, asiakkaasta eli palvelun käyttäjästä ja rekisteristä (Newcomer 2002). Tuottajan ja asiakkaan välinen kommunikointi tapahtuu SOAP-protokollan määrittelemässä muodossa, jossa tiedonvälitys verkossa toteutetaan yleensä HTTP-siirtoprotokollan avulla. Verkossa tarjolla olevat palvelut voidaan tallentaa rekisteriin, kuten UDDI (UDDI 2014). Rekisteri mahdollistaa sen, että haluttu palvelu voidaan löytää helposti miljoonien verkossa olevien palveluiden joukosta. Asiakas voi hakea palvelutarjoajan UDDI-rekisterissä julkaisemia palveluita, kuten kuvio 17 kohdassa 1 on esitetty. Asiakkaat voivat käyttää palveluita muodostamalla XML-pyyntöjä palveluntarjoajaan, kuten kuvio 17 kohdassa 2 on esitetty (Cermani 2002).



KUVIO 17. Web Servicen toiminta (Cermani 2002)

4.2 Mikä on XML

Extensible Markup Language (XML) on tiedon merkitsemistapa tai standardi, jota käytetään yleisesti verkkopalvelun ja asiakkaan välisessä tiedonsiirrossa. XML-kieli on rakenteellinen kuvauskieli, joka auttaa jäsentämään laajoja tietomassoja selkeämmin. XML-kieli muistuttaa paljon HTML-kieltä, jota käytetään www-sivujen tekemiseen. Perusajatuksena XML-formaatissa on tiedon säilyttäminen ja sen kuljettaminen. XML-kielen ero HTML-kieleen on siinä, että XML-kielessä käyttäjän ei tarvitse käyttää jotain ennalta määrättyjä elementteja, vaan elementit voidaan nimetä halutulla tavalla, joka kuvaa elementin sisäistä dataa parhaiten. Yksinkertaisesti voidaan siis todeta, että XML on yleistä tiedon kuvaamista varten, kun taas HTML on sivujen sisällön eli tietojen kuvaamista varten. XML-kielen on kehittänyt WORD Wide Web Consortium eli W3C (Rouse 2007).

4.3 XML-dokumentin rakenne

XML-tiedostossa tieto järjestetään puurakenteeseen niin, että itse tieto on jäsennelty tunnisteisiin. Tunnisteita eli elementtejä voi olla rajattomasti XML-dokumentin sisällä. Tunnisteet yleensä kertovat, minkälaista tietoa tunnisteet sisältävät. Kuviossa 18 on esitetty yksinkertainen XML-dokumentin rakenne.

```
2   <?xml version="1.0"?>
3   <esimerkki tyyppi="pakollinen">
4       <tervehdys> Hello world!</tervehdys>
5   </esimerkki>
6
```

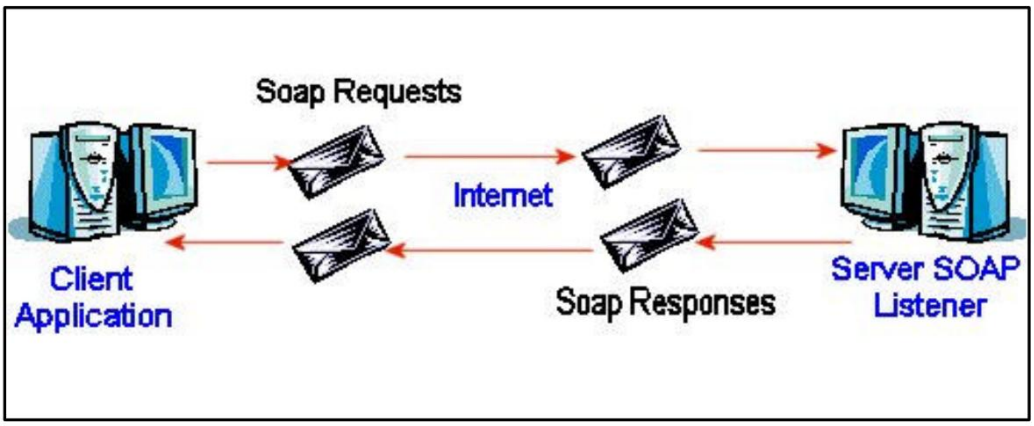
KUVIO 18. XML-dokumentin rakenne

XML-dokumentin alku on niin kutsuttu prosessointiohje. Prosessointiohjeessa on määritelty XML-dokumentin yleiset ominaisuudet, kuten XML-versio ja dokumentin käyttämä merkistö. Prosessointiohjeen on oltava XML-dokumentin ensimmäisellä rivillä, ja se kirjoitetaan <? ?>-merkintöjen väliin. Jokaisessa XML-dokumentissa saa olla vain yksi juurielementti, jonka sisällä kaikkien muiden elementtien täytyy olla. Kuviossa 18 XML-tiedoston juurielementtinä on <esimerkki>, jonka sisällä on tervehdys-niminen elementti. Elementit voivat sisältää yhden tai useamman attribuutin, jotka ovat kuvailevia ja antavat lisätietoja

kyseisestä elementistä. Kuviossa 18 esimerkki-elementin attribuuttina on pakollinen. XML-attribuutien heikkoutena on se, että ne eivät voi sisältää useita arvoja toisin kuin elementit, joilla voi olla useita lapsielementtejä. (2kmediat 2014.)

4.4 Simple Object Access Protocol

Simple Object Access Protocol on protokolla, jonka tehtävänä on kuljettaa XML-pohjaisia viestejä internetin välityksellä (Nilo 2003). SOAP-protokollan ansiosta asiakas-sovellus pystyy helposti ottamaan yhteyttä palveluntarjoajaan ja käyttämään sen tarjoamia palveluita (Cermani 2002). SOAP tarjoaa yksinkertaisen viestirakenteen, jonka sisällä voidaan kuljettaa minkä tyyppistä XML-pohjaista tietoa hyvänsä. Kuviossa 19 on esitetty, miten asiakas-sovellus kommunikoi SOAP:n välityksellä palvelutarjoajan kanssa.

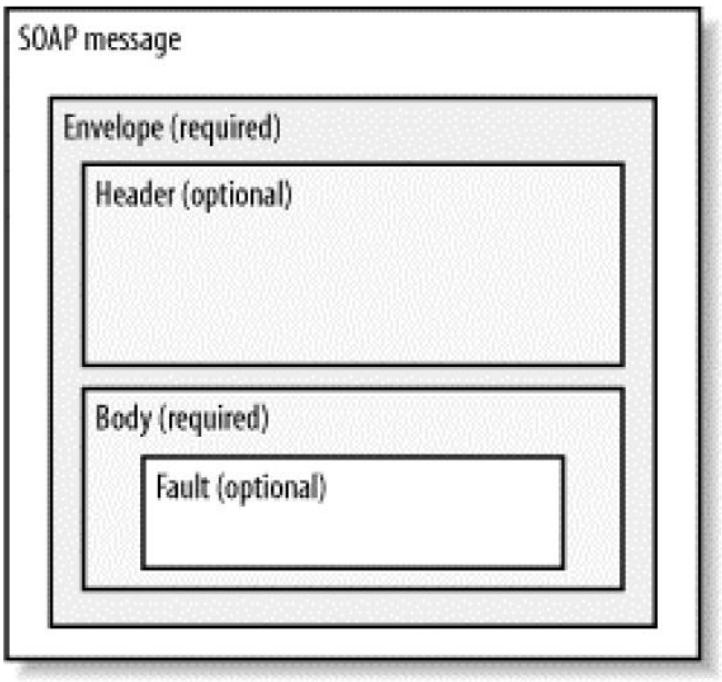


KUVIO 19. SOAP-viestit (Koftikian 2001)

SOAP-viesti sisältää juurielementin nimeltä Envelope, jonka sisällä kuljetetaan koko SOAP-viesti. Envelope-elementti sisältää Header-elementin, joka on vapaavalintainen, sekä pakollisen Body-elementin. Body-elementissä välitetään itse viesti XML-muodossa. Body-elementin sisällä voi viestin lisäksi kuljettaa vikatietoa, joka siirretään Fault-elementin sisällä. Fault-elementti ei ole pakollinen. XML-muotoisen SOAP-viestin mallirunko on esitetty kuviossa 20 sekä SOAP-viestin pääelementit kuviossa 21.

```
<?xml version="1.0"?>
    <soap:Envelope xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
    soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
        <soap:Header>
        .......
        </soap:Header>
        <soap:Body>
        .......
        </soap:Body>
    </soap:Envelope>
```

KUVIO 20. SOAP-viestin rakenne

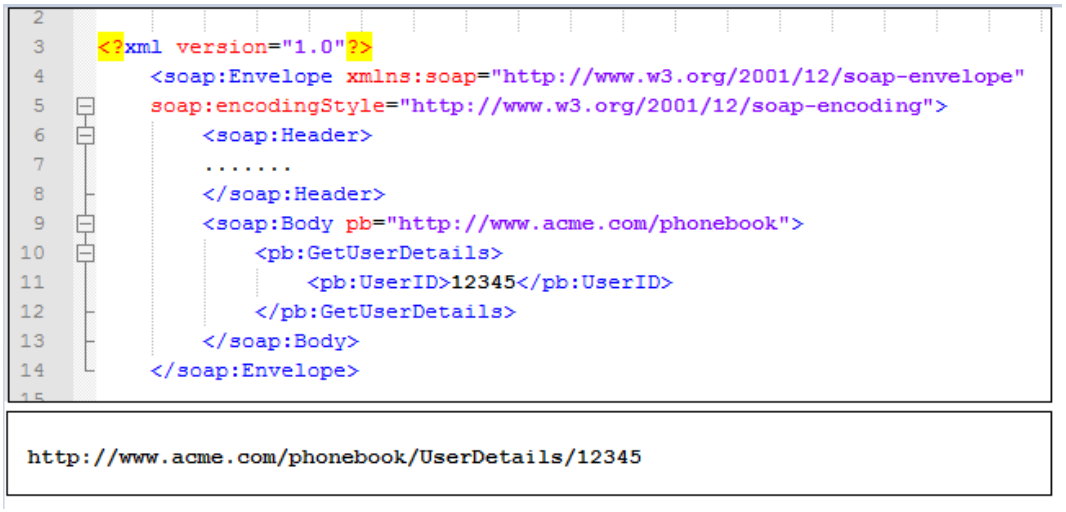


KUVIO 21. SOAP-viestin pääelementit (Cermani 2002)

4.5 REST

Web-palvelua on myös mahdollista toteuttaa muilla tekniikoilla. Yksi näistä tekniikoista on nimeltään REST eli Representational State Transfer. Palvelun toteuttaminen REST-tekniikalla on huomattavasti helpompaa kuin esimerkiksi SOAP-tekniikalla. Tämän takia REST-palveluja käytetään yhä enemmän kuluttajille suunnatuissa rajapintapalveluissa. Esimerkiksi Twitter tarjoaa mahdollisuuden käyttää REST-palvelua, joka mahdollistaa Twitterin käyttön omissa sovelluksissa (Elkstein 2008b).

REST-palvelun yksinkertaisuus on siinä, että toisin kuin SOAP-palveluissa, joissa XML-pohjainen SOAP-pyyntö lähetetään Web-palvelu palvelimelle, REST-palveluissa pyyntö on URI-pohjainen. Kuviossa 22 on esitetty SOAP-ja REST-tekniikoiden eroa. Kuviosta 22 voidaan nähdä, että ylempänä on XML-pohjainen SOAP-pyyntö, jolla haetaan tietoja 12345-nimisestä käyttäjästä palvelimelta. SOAP-tekniikan alla on esitetty sama pyyntö REST-tekniikalla, jossa käytetään HTTP-protokollan GET-kommento tietojen hakemiseen. Kun käytetään SOAP-tekniikkaa, vastauksena saadaan XML-pohjainen SOAP-viesti, kun taas REST-tekniikassa vastauksena saadaan pelkkä haettu data. (Elkstein 2008a.)



KUVIO 22. SOAP- ja REST-pyynnöt

4.6 Web Service Description Language

Web Service Description Language eli WSDL on XML-pohjainen kieli, jolla kuvataan web-teknologiaan perustuva palvelu verkossa. WSDL yksinkertaisesti kuvaa kaikki toimminallisuudet ja rakenteet tietystä web-palvelusta. WSDL-kuvaus koostuu kahdesta eri osasta, jotka ovat abstrakti osa ja konkreettinen osa. Abstrakti osa kuvaa ainoastaan kaikki ne toiminnot, joita voi palvelussa suorittaa. Konkreettisessa osassa WSDL kuvaa käytetyn yhteyden osoitteen sekä protokollan, jota toiminnoissa käytetään (W3C 2001). Kuviossa 23 on esitetty WSDL-dokumentin pääelementit.

```
<definitions name="HelloService"
   targetNamespace="http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns="http://schemas.xmlsoap.org/wsdl/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:tns="http://www.examples.com/wsdl/HelloService.wsdl"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">

   <message name="SayHelloRequest">
      <part name="firstName" type="xsd:string"/>
   </message>
   <message name="SayHelloResponse">
      <part name="greeting" type="xsd:string"/>
   </message>

   <portType name="Hello_PortType">
      <operation name="sayHello">
         <input message="tns:SayHelloRequest"/>
         <output message="tns:SayHelloResponse"/>
      </operation>
   </portType>

   <binding name="Hello_Binding" type="tns:Hello_PortType">
   <soap:binding style="rpc"
      transport="http://schemas.xmlsoap.org/soap/http"/>
   <operation name="sayHello">
      <soap:operation soapAction="sayHello"/>
      <input>
         <soap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:examples:helloservice"
            use="encoded"/>
      </input>
      <output>
         <soap:body
            encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            namespace="urn:examples:helloservice"
            use="encoded"/>
      </output>
   </operation>
   </binding>

   <service name="Hello_Service">
      <documentation>WSDL File for HelloService</documentation>
      <port binding="tns:Hello_Binding" name="Hello_Port">
         <soap:address
            location="http://www.examples.com/SayHello/">
      </port>
   </service>
</definitions>
```

KUVIO 23. WSDL-dokumentin pääelementit (Tutorialspoint 2014)

WSDL-dokumenti koostuu kuudesta pää-elementistä. Definition-elementti on WSDL-dokumentin pää-elementti, jossa määritellään Web-palvelun nimi, käytetyt nimiavaruudet ja kaikki muut elementit (Cermani 2002). Types-elementti kuvailee kaikkia tietotyyppejä, joita käytetään asiakkaan ja palvelutarjoajan välisessä kommunikoinnissa. Message-elementti määrittelee viestin nimen ja voi sisältää useampia part-elementtejä, jotka voivat viitata viestien paluuarvoihin. WSDL-dokumentissa PortTyp-elementi määrittelee abstraktin rajapinnan Web-palvelulle joukkona operaatioita, jotka ovat puolestaan annettu operation-elementin avulla. Operation-elementti määrittelee palvelun lähettämien ja vastaanottamien viestien

tyypit. Binding-elementti määrittelee WSDL-dokumentissa operaatioiden sitomista viestinvälitysmekanismeihin. Binding-elementti kuvaa, miten viestit välitetään. WSDL-dokumentin viimeinen elementti on service-elementti, joka määrittää, missä Web-palvelun tarjoamat palvelut sijaitsevat, jotta niihin voitaisiin ottaa yhteyttä.

5 LASKUAVAIN MOBILE -SOVELLUS

5.1 Sovelluksen vaatimukset

Tämän opinnäytetyön tavoitteena oli suunnitella ja toteuttaa Laskuavain Mobile -mobiiilisovellus Ohjelmistotalo Koodiavaimelle. Mobiilisovellus tuli tehdä Android-, iOS- ja Windows Phone -alustoille, ja sen piti toimia yhtenäisesti palvelinpuolen Web-palvelun kanssa. Mobiilisovelluksen jakelu oli tarkoitus olla virallisten sovelluskauppojen kautta mahdollista, mutta sen julkistamisajankohdasta ei ollut sovelluksen suunnittelu- ja kehittämisvaiheessa tarkkaa tietoa. Sovellukselle asetettiin myös muita vaatimuksia.

Mobiilisovellukseen oli tarkoitus toteuttaa ominaisuus, jolla asiakas eli käyttäjä asettaa oman yrityksensä URL-osoitteensa sovellukselle. Tätä varten käyttäjän asettama URL-osoite tallennettiin sovellukseen pysyvästi. Käyttäjän asettaman URL-osoitteen perusteella sovellus lisää automaattisesti tarvittavan URL-osoitteen kuitti- ja ajopäiväkirjatietojen lähettämistä varten. Sovelluksen asetuksista tuli pystyä muokkamaan tai vaihtamaan yrityksen URL-osoite.

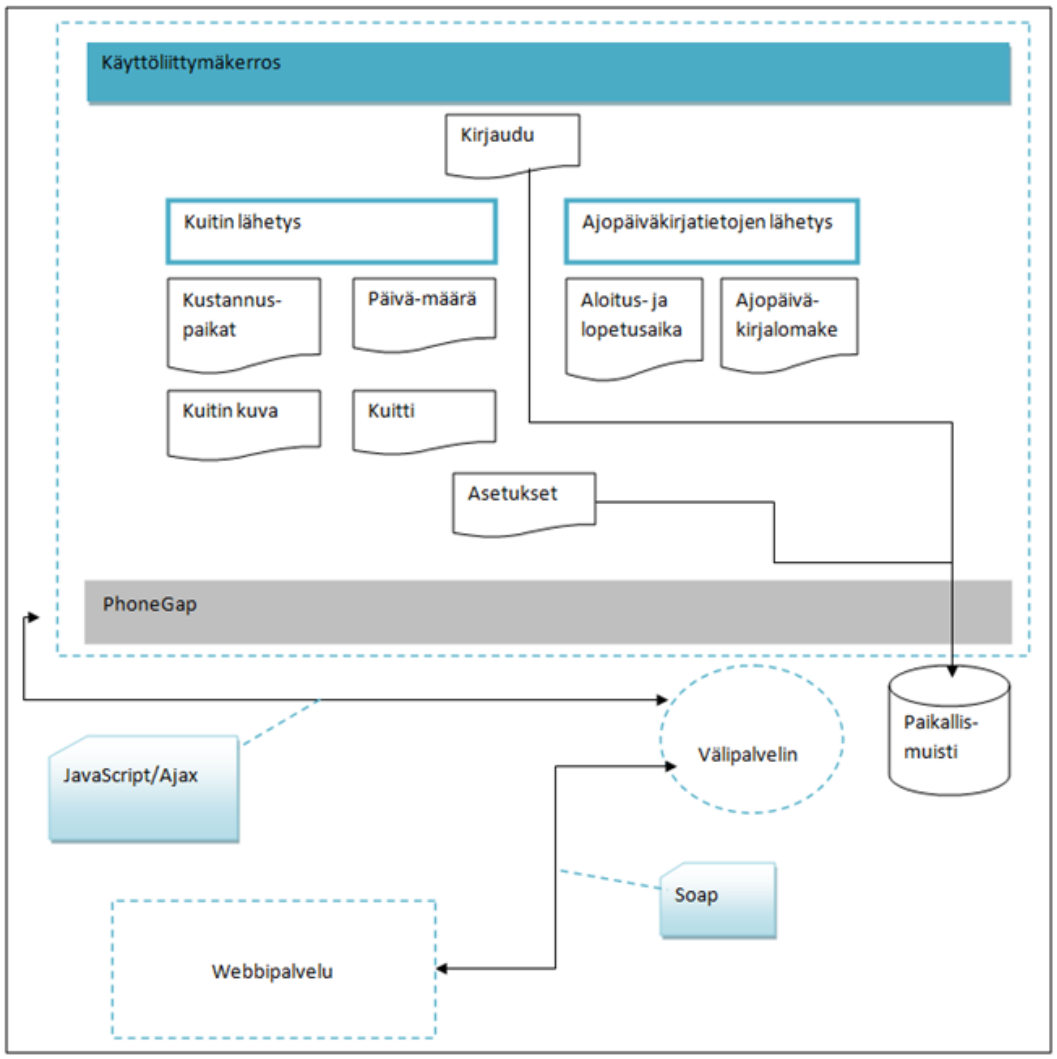
Sovellukseen kuului ominaisuus, joka ottaa kuvan ja tallentaa sen puhelimen kuva-albumiin. Sovelluksen täytyi toteuttaa ominaisuus, joka lähettää kuitti- tai ajopäiväkirjatietoja Web-palveluun. Sovelluksen vaatimuksiin kuului myös ominaisuus, joka hakee listan kustannuspaikoista Web-palvelulta ja näyttää ne sovelluksessa. Sovelluksen tuli pystyä tallentamaan käyttäjän antama käyttäjätunnus ja salasana kirjautumisvaiheessa.

Mobiilisovellus tuli suunnitella ja toteuttaa niin, että se olisi mahdollisimman helppokäyttöinen ja selkeä, jolloin sen käyttö ei vaatisi erillistä ohjeistusta. Sovelluksen ulkoasu ja värimaailma tuli myös mukailla Laskuavain-palvelun logoa.

5.2 Sovelluksen arkkitehtuuri

Laskuavain Mobile -sovelluksen toimintalogiikka on esitetty kuviossa 24. Sovellus tarjoaa kaksi erillistä toimintoa, kuitin lähetyksen ja ajopäiväkirjatietojen lähetyksen.

Sovelluksen kirjaudu-sivulla käyttäjä voi asettaa URL-osoitteen sovellukselle sekä kirjautua sisään. Käyttäjän antama käyttäjätunnus ja salasana tallennetaan paikallismuistiin, mikäli käyttäjä painaa Muista minut -painiketta. Tämä mahdollistaa sen, että kirjautumissivulla näytetään valmiiksi käyttäjätunnus ja salasana. Kuitin lähetysnäkymässä käyttäjä täyttää vaaditut kentät, ottaa kuvan kuitista ja lähettää tiedot määrättyyn web-palveluun. Kuitin kuva tallennetaan myös laitteen kuva-albumiin.



KUVIO 24. Laskuavain Mobile -sovelluksen arkkitehtuuri

Kustannuspaikat-sivulla on esitetty lista kustannuspaikoista, jotka sovellus hakee välipalvelimen kautta Web-palvelulta.

Sovelluksen kuitti-sivulla näytetään valmis kuitti ennen sen lähettämistä. Sivulla näkyy käyttäjän kuitille antamat tiedot ja kuitista otettu kuva. Käyttäjän on mahdollista muokata kuittia myös kuitti-sivulla.

Käyttäjän määrittelemää URL-osoitetta voidaan muokata myös sovelluksen Asetukset-sivulla. Lisäksi URL-osoite tallennetaan paikallismuistiin.

Ajopäiväkirjatietojen pääsivulla näytetään käyttäjälle vaaditut kentät, joita käyttäjä täyttää. Käyttäjän annettua kaikki tiedot näytetään valmis kuitti, joka on valmis lähetettäväksi. Valmista lomaketta käyttäjä voi myös muokata ennen sen lähettämistä.

5.3 Tekninen toteutus ja kehitysympäristön valinta

Sovelluksen suunnitteluvaiheessa päädyttiin yhdessä Ohjelmistotalo Koodiavaimen kanssa siihen tulokseen, että sovelluksen tekninen toteutus tehdään PhoneGap-ohjelmistokehyksen avulla. Kehitysympäristön valinnan kannalta tärkeitä ominaisuuksia olivat testaus, sovelluksen suorituskyky, yksinkertaisuus, jatkokehitys, natiiviominaisuuksien hyödyntäminen ja nykyaikaisuus.

Koska sovellus on luonteeltaan yksinkertainen eikä vaadi suurta laskentatehoa laitteelta, niin PhoneGap on täydellinen ratkaisu Laskuavain Mobile -sovellukselle. Lisäksi PhoneGapilla tehty sovellus toimii monipuolisesti eri valmistajien mobiilialustoissa. PhoneGapin JavaScript API -rajapinta tukee lähes kaikkia puhelimen natiiviominaisuuksia sekä PhoneGapin ominaisuudet riittävät sovelluksen testamiseen, jatkokehittämiseen ja debuggaukseen niin teknisesti kuin taloudellisesti. Sovelluksen testauksessa oli käytössä sekä Android- että iOS-pohjainen älypuhelin.

Mobiilisovelluksen kehityksessä on käytetty myös jQuery-kirjastoa ja jQuery Mobile -ohjelmistokehystä. jQuery on avoimen lähdekoodin JavaScript-kirjasto,

jolla pyritään helpottamaan DOM-elementtien käsittelyä ja selkeyttämään sekä yksinkertaistamaan sovelluksen JavaScript-koodia. jQueryssa on toteutettu kirjaston tarjoamat toiminnallisuudet eri selaimille erikseen. Tämä myös vähentää huomattavasti selainten välisiä ongelmia. Laskuavain Mobile sovelluksessa jQuery-kirjastoa käytettiin koodin saamiseen helppolukuisemmaksi, DOM-elementtien käsittelyyn ja helpottamaan Ajax-tekniikan käyttöä. (Bibeaut & Katz 2008.)

jQuery Mobile on kosketusnäyttöoptimoitu käyttöliittymän kehitysalusta, joka pohjautuu jQuery- ja jQuery UI -kirjastoihin. Sen avulla voidaan helposti luoda muokattavia ja responsiivisia käyttöliittymiä puhelimille ja tableteille. jQuery Mobile tarjoaa kattavan valikoiman erilaisia ”widgettejä”, kuten painikkeita, valikkoja ja sivun lataamiseen liittyviä toiminnallisuuksia. jQuery Mobile sisältää ruudukon eli ”grid”-systeemin, jonka ansiosta sivun sisältö saadaan mukautumaan laitteen näytön koon mukaan helposti ja vaivattomasti (Wikipedia 2014).

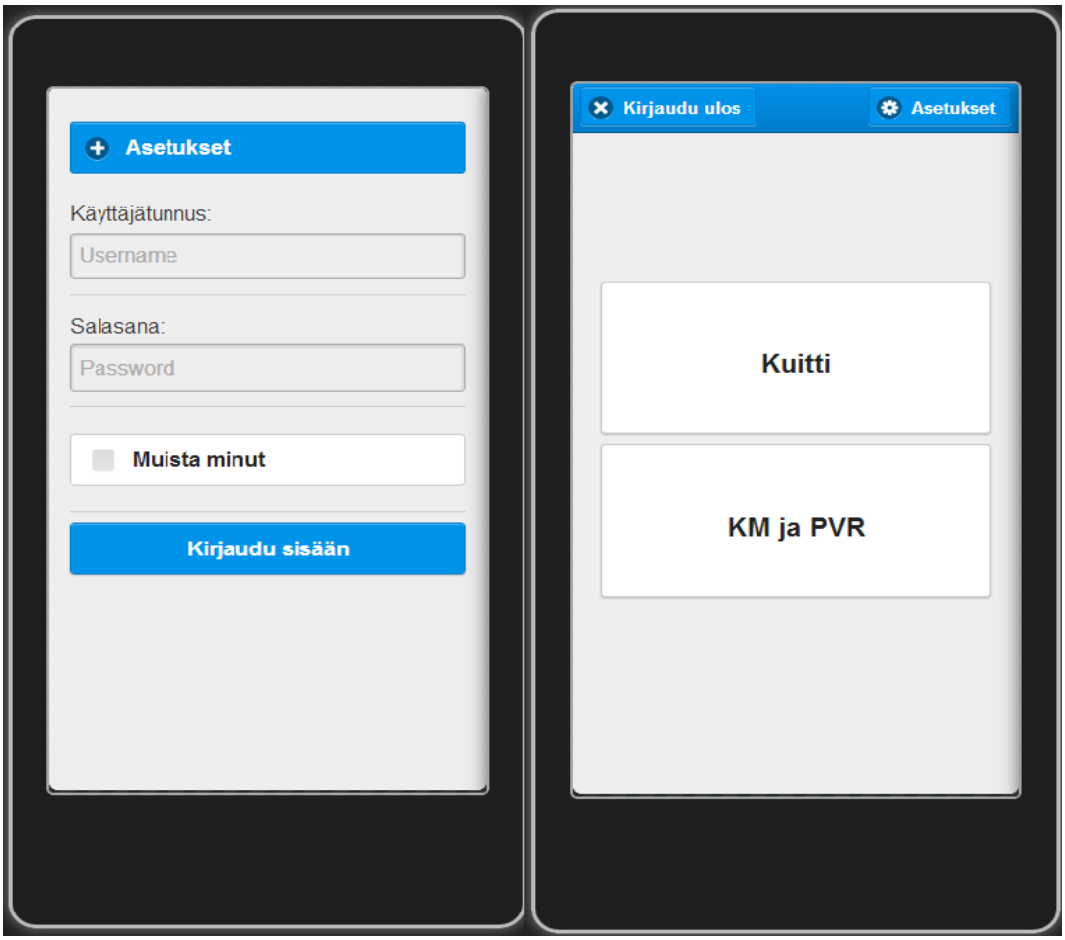
5.4 Sovelluksen kuvaus

5.4.1 Käyttöliittymä

Tämän opinnäytetyön mobiilisovelluksen käyttöliittymä mukailee natiivisovelluksille tyyppillisiä rakenteita. Sovelluksen käynnistyessä käyttäjälle näytetään ensin kuvion 25 mukainen käynnistyskuva. Kuva näkyy, kunnes PhoneGap on valmis käytettäväksi.

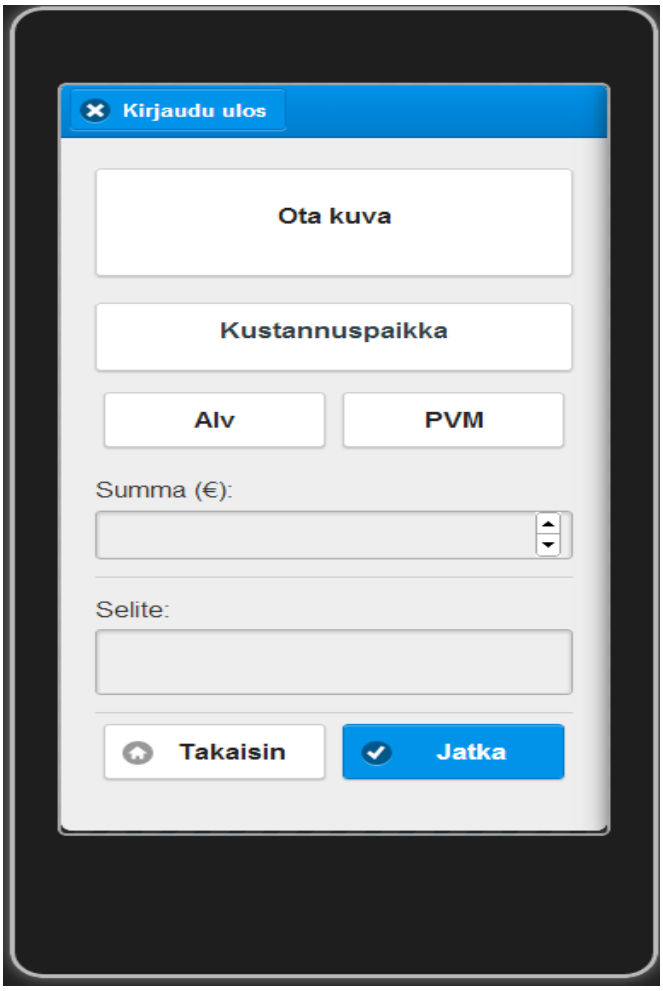
KUVIO 25. Sovelluksen käynnistyskuva

Tämän jälkeen käyttäjälle näytetään kuvion 26 esitetty kirjautumissivu. Sovellusta käytettäessä ensimmäistä kertaa käyttäjän on annettava URL-osoite kirjautumissivun Asetukset-kohdalla. Kun URL-osoite, käyttäjätunnus ja salasana on annettu oikein, käyttäjälle näytetään kuvion 26 kaltainen pääsivu. Pääsivulla käyttäjä voi muokata tai uudelleen asettaa URL-osoitteen painamalla Asetukset-painiketta. Pääsivusta käyttäjä voi siirtyä joko Kuitti- tai Ajopäiväkirja-osioon.



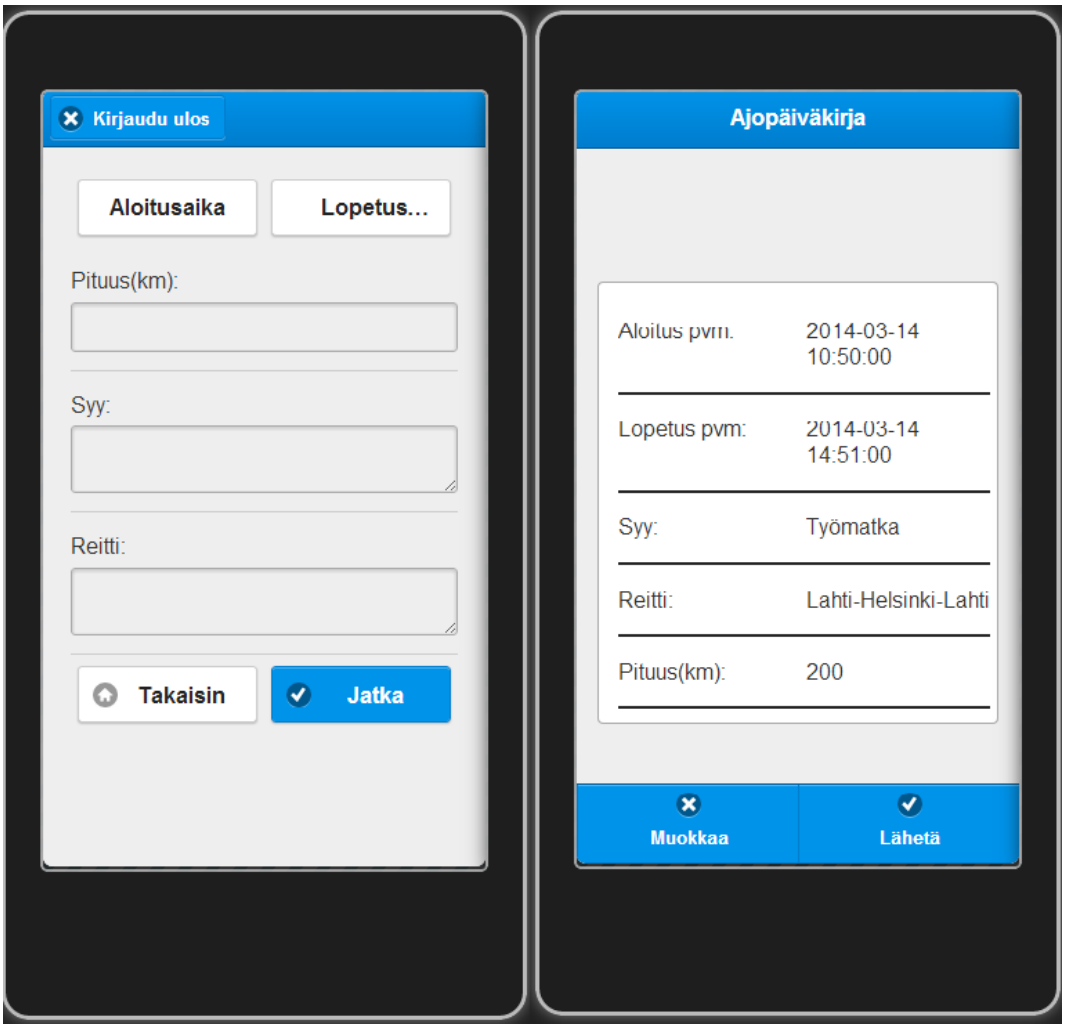
KUVIO 26. Sovelluksen kirjautumis- ja pääsivu

Sovelluksen kaikki sivut kirjautumissivua lukuun ottamatta ovat rakenteeltaan samanlaisia. Jokaisen sivun yläosassa on palkki, jossa on sivun otsikko. Palkin lisäksi sivulla on sivun varsinaisen sisällön sisältävä alue. Kuviossa 27 on esitetty kuitti-sivu, jossa käyttäjä täyttää kuittiin tarvittavat tiedot sekä ottaa kuvan kuitista. Kun kaikki tiedot on annettu, käyttäjä siirtyy painamalla Jatka-painiketta sivulle, jossa näytetään täytetty kuitti lähetettäväksi. Kuitti-sivulta käyttäjä voi myös siirtyä takaisin sovelluksen pääsivulle painamalla takaisin-painiketta. Käyttäjä voi myös kirjautua ulos sovelluksesta painamalla sivun yläosassa olevaa Kirjaudu ulos-painiketta.



KUVIO 27. Sovelluksen kuitti-sivu

Sovelluksen pääsivulla käyttäjä voi siirtyä myös ajopäiväkirja-sivulle, joka on esitetty kuviossa 28 vasemmanpuoleisessa kuvassa. Ajopäiväkirja-sivulla käyttäjä täyttää tarvittavat tiedot, kuten matkan pituus, matkan syy, reitti sekä aloitus- ja lopetusaika. Kun kaikki tietdot on annettu ja käyttäjä painaa Jatka-painiketta, näytetään täytetty ajopäiväkirja-kuitti käyttäjälle kuvion 28 oikeanpuoleisen kuvan mukaisesti. Ajopäiväkirja-kuitin käyttäjä voi lähettää painamalla Lähetä-painiketta. Lähetyksen onnistuessa näytetään käyttäjälle ilmoitus onnistuneesta lähetyksestä.



KUVIO 28. Sovelluksen ajopäiväkirja-sivu ja valmis ajopäiväkirja-kuitti

Sovelluksen käyttöliittymä on toteutettu jQuery Mobilen avulla, joka mahdollistaa sen, että sovelluksen kaikki sivut upotetaan yhteen HTML-dokumenttiin. Laskuavain Mobile-sovelluksessa kaikki sivut on upotettu yhteen HTML-dokumenttiin, jossa näkyvissä on yksi sivu kerrallaan. Navigointisivujen välille on toteutettu jQuery Mobilen avulla tarjoamalla liukutehostetta. Kuviossa 29 on esitetty sovelluksen yksittäisen sivun rakennetta.

```
<!-- Pääsivu -->
<div data-role="page" id="home_page">

    <div data-role="header" data-position="fixed">
        <a href="#" data-role="button" data-icon="delete"
          class="logout" data-mini="true">
          Kirjaudu ulos
        </a>
        <a href="#settings_page" data-role="button"
          data-icon="gear" data-mini="true">
          Asetukset
        </a>
    </div>
     <!-- /header -->

    <div data-role="content" id="home_page_content" >
        <div class="ui-grid-solo" >
            <div class="ui-block-a">

                <a data-role="button" href="#main_page"
                    class="medium_btn height_" >
                    Kuitti
                </a>
            </div>
        </div>
        <div class="ui-grid-solo" >
            <div class="ui-block-a">

                <a data-role="button" href="#km_main_page"
                    class="medium_btn height_">
                    KM ja PVR
                </a>
            </div>
        </div>

    </div>
    <!-- /content -->

</div>
<!-- /page -->

```

KUVIO 29. Sovelluksen yksittäisen sivun rakenne

Kuviossa 29 on esitetty sovelluksen pääsivu. Sivu on laitettu div-elementin sisään. Elementin data-role-attribuutissa määritellään kyseisen elementin rooli, joka tässä tapauksessa on page. Muut käytetyt elementtien roolit ovat header ja content. Header-rooli märittelee tämän sivun tapauksessa div-elementin toimimaan navigointipalkin tapaisesti sivun yläosassa. Content-rooli määrittelee sivun varsinaisen sisällön. Sisältö skaalautuu koko ruudulle header-elementin alaosasta aina sivun alaosaan asti.

5.4.2 Sovelluksen käyttämät PhoneGap-kirjastot

Laskuavain Mobile sovellus käyttää PhoneGapin tarjoamia Notification-, Camera, File- ja Storage-kirjastoa. Sovelluksessa käytetään tietojen tallentamiseen paikallismuistia, johon saadaan yhteys PhoneGapin Storage-kirjaston avulla. Camera-kirjastolla voidaan laitteen kameraa käyttää kuvan ottamisessa ja File-kirjastolla voidaan laitteen kameralla otettu kuva lähettää palvelimelle. Notification-kirjastolla ilmoitetaan käyttäjälle toiminnoista sekä vahvistusilmoituksien toteuttamisesta.

Sovelluksen Kuitti-osuudella kustannuspaikkojen nimet ladataan etäpalvelimelta käyttäen Ajax-tekniikkaa. Tässä sovelluksessa Asynchronous JavaScript- ja XML -tekniikka mahdollistaa HTTP-pyynnön lähettämisen asynkronisesti taustalla. Kun pyyntö on lähetetty onnistuneesti etäpalvelimelle, etäpalvelin lähettää Soap-kutsun Web Serviceen. Tämän jälkeen Web Service palauttaa vastauksena kustannuspaikkojen nimet etäpalvelimelle. Lopulta Ajax-tekniikkaa käyttäen parsitaan etäpalvelimelta kustannuspaikkojen nimet ja tulostetaan se sovelluksen kustannuspaikat-sivulle. Kuviossa 30 on esitetty, miten kustannuspaikkojen nimet haetaan Web Serviceltä etäpalvelinta käyttäen.

```php
<?php


    /*Sovelluksen URL-osoite */
    $laskuavain_url = "https://firma.laskuavain.fi";

    $url = $laskuavain_url . '/soap/kuitit.asmx?wsdl';

    /*Soap Client-kutsu*/
    $client = new SoapClient($url);

    /*Soap vastaus Web Service:lta.

        Vastauksena saadaan taulukko,
        jossa on kustannuspaikkojen nimet
    */
    $response = $client->Kustannuspaikat();

    /* Tulostetaan Web Service:n palauttama vastaus */
    var_dump($response);


?>
```

KUVIO 30. Kustannuspaikkojen haku ja tulostus etäpalvelimella

Laskuavain Mobile sovelluksessa Camera-kirjastoa käytetään kuvan ottamiseen kuitista. Kun kuva on otettu, se tallennetaan myös laitteen kuva-albumiin. Lisäksi otettu kuva lähetetään File-kirjaston avulla etäpalvelimella sijaitsevaan tmp-nimiseen kansioon. Lopulta kuitin lähetysvaiheessa otettu kuva leimataan osaksi kuitin tietoa ja lähetetään Web Servicen läpi. Kuviossa 31 on esitetty, miten laitteen kameralla otettu kuva lähetetään etäpalvelimelle PhoneGapin File-kirjastoa käyttäen.

```
1
2   // Otetaan kuva laitteen kameraa käyttäen
3   function capturePhoto() {
4       navigator.camera.getPicture(uploadPhoto, onFail, {
5           quality: 75, // kuvalaatu
6           destinationType: Camera.DestinationType.FILE_URI, // paluarvon formaatti
7           saveToPhotoAlbum: true, // tallennetaan otettu kuva laitteen kuva-albumiin
8           targetWidth: 1200, // kuvan leveys
9           targetHeight: 1200 // kuvan korkeus
10      });
11  }
12
13  // ja lähetetään kuva etäpalvelimelle uploadPhoto-function avulla
14  function uploadPhoto(imageURI) {
15      var options = new FileUploadOptions();
16      options.fileKey = "file";
17      options.fileName = imageURI.substr(imageURI.lastIndexOf('/') + 1); // kuvan nimi
18      options.mimeType = "image/jpg"; // kuvan tyyppi
19      var params = new Object();
20      params.imageURI = imageURI; // kuvan polku laitteessa
21      options.params = params; // valinnaiset parametrit
22      options.chunkedMode = false;
23      var url = "http://www.example.com/getPhoto.php"; // palvelin, joka vastaanottaa kuvaa
24
25      var ft = new FileTransfer();
26      ft.upload(imageURI, url , onSuccess, onFail, options); // lähetetään kuva palvelimelle
27
28  }
29
```

KUVIO 31. Kuvan ottaminen laitteen kameralla ja sen lähettäminen etäpalvelimelle

Notification-kirjastoa on käytetty ilmoitusten ja vahvistusikkunoiden näyttämisessä käyttäjälle. Seuraavassa kuviossa 32 on esimerkki Notification-kirjaston käytöstä, jolla käyttäjälle näytetään vahvistusikkuna, kun kuva on otettu onnistuneesti ja tallennettu puhelimen kuva-albumiin.

```
3
4   //Kuvan ottaminen onnistui
5   function onSuccess() {
6       navigator.notification.alert(
7           'Kuva on valittu ja tallennettu puhelimen kuva-albumiin.', // viesti
8               alertOk, // functio
9               'Kuva on valittu!', // otsikko
10              'OK'                // näppäimen nimi
11      );
12
13  }
14
15  function alertOK() {
16      // suoritetaan jotain toimintaa
17  }
18
```

KUVIO 32. Vahvistusikkunan näyttäminen Notification-kirjaston avulla

5.5 Sovelluksen testaus ja jatkokehitys

Laskuavain Mobile-sovelluksen testaus suoritetiin sovelluskehityksen aikana tapahtuvan oman testauksen ohella Ohjelmistotalo Koodiavaimen henkilökunnan avulla. Henkilökunnalla oli käytössään IPhone 5 -matkapuhelin, IPad 2 -tabletti ja muutama Android-pohjainen matkapuhelin, joihin ladattiin aina uusin sovellusversio testattavaksi. Testaajien antamien palautteiden pohjalta sovellukseen tehtiin kehitysvaiheessa tarvittavia muutoksia. Testauksessa keskityttiin muun muassa sovellukseen toimivuuteen Web Servicen tarjoamien palveluiden kanssa sekä jQuery Mobilen yhteensopivuuteen PhoneGapin kanssa. Lisäksi testattiin sovelluksen JavaScript-toiminnallisuutta. Sovelluksen kehitysvaiheessa testiympäristönä käytettiin myös Ripple-Emulatoria, jota saadaan Chrome-selaimeen lisäosana. Ripple-Emulatorilla testattiin muun muassa miltä sovellus näyttää esimerkiksi IPhone 4S -puhelimessa.

Sovelluksen mahdollisena jatkokehityksenä voisi olla sovelluksen lisäominaisuus, jolla voitaisiin lähetetyt kuitit tallentaa puhelimeen paikallismuistiin. Tämä mahdollistaisi sen, että käyttäjällä olisi lista kuiteista, joita on lähetetty Web-palveluun. Tämä lisäominaisuus mahdollistaisi myös sen, että lähetettyjä kuitteja voidaan listata näytettäväksi aiheen tai päivämäärän perusteella.

Sovelluksen ajopäiväkirja-osiolla PhoneGapin Geolocation-kirjaston käyttäminen reitin selvittämiseksi on myös yksi mahdollinen jatkokehityksen suunnitelma. Tämä lisäominaisuus toteutettaisiin niin, että sovellus ehdottaisi reittejä laitteen paikkannustietojen perusteella.

Käyttöliittymän parantaminen sekä laitteen kameralla otetun kuittikuvan auttomaattinen muokkaus maksimikokoon voisivat olla myös sovelluksen jatkokehityksen aiheita tulevaisuudessa. Lisäksi sovelluksella voisi olla ominaisuus listata kaikki lähetetyt kuitit käyttäjälle. Sovelluksen ulkoasua voisi myös tulevaisuudessa parantaa näyttävämmäksi.

6 YHTEENVETO

Opinnäytetyön tavoitteena oli suunnittella ja toteuttaa mobiilisovellus, jonka käyttötarkoituksena on kuitin- tai ajokilometritietojen välittäminen Web Servicen läpi halutulle palvelimelle. Mobiilisovelluksella pystyy keräämään tietoja, ottamaan kuitista kuvan ja lähettämään sen palvelimelle. Ohjelmistotalo Koodiavaimelle toteutettu mobiilisovellus oli ensimmäinen PhoneGapin sovelluskehityskehyksellä toteuttamani sovellus. Sovelluksen kehitysvaiheessa kohdattiin muutamiin PhoneGapiin liittyviin ongelmiin, jotka puolestaan johtuivat lähinä siitä, ettei ollut aikaisempaa kokemusta PhoneGapistä.

Yksi niistä haasteista, johon kehitysvaiheessa kohdattiin, oli laitteen kameralla otetun kuvan lähettäminen etäpalvelimelle. Toisen haasteen PhoneGap aiheutti testauksen osalta. HTML-, CSS- ja JavaScript-ohjelmoitikielillä kirjoitettujen sovelluksen toiminnallisuuksia voidaan testata tietokoneen verkkoselaimiin ja niihin lisäosina asennetuissa testaustyökalujen avulla, PhoneGapin JavaScript-kirjastoja voidaan testata ainoastaan laitteessa. Tämä prosessi hidastaa huomattavasti sovelluksen toiminnallisuuksien kehittämistä ja virheiden korjaamista.

Opinnäytetyön teoriaosuudessa tutustuttiin PhoneGapin tarjoamiin JavaScript-kirjastoihin sekä niiden toiminnallisuuksiin. Lisäksi työssä tutustuttiin Web Serviceen yleisellä tasolla sekä siihen, miten voidaan käyttää Web Servicen tarjoamia palveluita verkon yli PhoneGap-sovelluksessa.

Koska PhoneGapin avulla toteutettu sovellus käyttää laitteen WebWiev-komponenttia toimiakseen ja käyttöliittymä ohjelmoidaan HTML- ja CSS-ohjelmointikielellä, sovellus ei vältämättä toimi samalla tavalla erilaisten mobiilialustojen verkkoselainmoottoreiden HTML- ja CSS-toteutuksien takia. Esimerkiksi eri Android-versioiden kesken verkkoselainmoottorien tuki HTML- ja CSS-kieliin on erilainen.

Vaikka PhoneGapin sovelluskehitysympäristö tarjoaa mahdollisuuden kehittää sovellusta uusimpien teknologioiden, kuten HTML5:n ja CSS:n sekä JavaScriptin avulla, se ei tarkoita, että sovelluksen kehittäminen yhdelle mobiilialustalle olisi nopeampaa. PhoneGapin vahva puoli tulee esiin, kun sovellusta halutaan tehdä

useammalle alustalle. Tämän mahdollistaa PhoneGapin pilvipalveluna toimiva PhoneGap Build, jossa kooditiedostot käännetään useammalle mobiilialustalle sopivaksi.

Tämän opinnäytetyön mobiilisovelluksen käyttöliittymänä käytetiin jQuery Mobile -ohjelmistokehystä, joka alkuun tuntui oikealta valinnalta. Sovelluskehityksen edetessä tuli haasteita saada jQuery Mobile ja PhoneGap yhteensopiviksi, jotka hidastivat hieman työnkulkua. Lisäksi jQuery Mobilen animaatioiden toimivaksi saaminen erityisesti Android-laitteissa toi lisähaastetta.

PhoneGap mahdollistaa siis alustariippumattoman sovelluskehityksen, jossa voidaan päästä käsiksi laitteen natiiviominaisuuksiin PhoneGapin tarjoamien JavaScript-kirjastojen avulla. PhoneGapin ansiosta sovelluskehitys on entistä nopeampaa. Sovelluskehittäjn näkökulmasta PhoneGap tarjoaa oivan mahdollisuuden saada oma sovellus eri mobiilialustoille ja näin suurelle mobiilikäyttäjäkunnalle.

LÄHTEET

Bibeault, B. & Katz, Y. 2008. jQuery in action. Stamford:
Manning Publications Co.

Cermani, E. 2002. Web Service Essentials, Distributed Application With XML-RPC, SOAP, UDDI & WSDL. California: O'Reilly Media

Elkstein, M. 2008b. What is REST? [viitattu 2.2.2014].
Saatavissa: http://rest.elkstein.org/2008/02/what-is-rest.html

Elkstein, M. 2008a. How Simple is REST? [viitattu 2.2.2014].
Saatavissa: http://rest.elkstein.org/2008/02/how-simple-is-rest.html

Ghatol, R. & Patel, Y. 2012. Beginnig PhoneGap. New York: Springer Sience + Business Media

Gartner. 2013. [viitattu 11.2.2014]. Saatavissa:
http://www.gartner.com/newsroom/id/2573415

Jack Koftikian 2001. Simple Object Access Protocol [viitattu 7.2.2014].
Saatavissa: http://www.sts.tu-harburg.de/pw-and-m-theses/2001/Koft01.pdf

Newcomer E.2002 Understanding web services – XML, WSDL, SOAP, and UDDI. Boston: Addison-Wesley.

Nilo, M. 2003. "SOAP Version 1.2 Part 0: Primer" [viitattu 16.1.2014].
Saatavissa: http://www.w3.org/TR/soap12-part0

PhoneGap 2014. [viitattu 9.2.2014]
Saatavissa: https://build.phonegap.com/

Rouse, M. 2007. XML (Extensible Markup Language) [viitattu 1.2.2014].
Saatavissa: http://searchsoa.techtarget.com/definition/XML

Tutorialspoint. 2014. WSDL [viitattu 5.4.2014].
Saatavissa: http://www.tutorialspoint.com/wsdl/wsdl_example.htm

Wargo, J. M. 2012. PhoeGap Essentials Building Cross-Platform Mobile Apps. Boston: Addison-Wesley. [viitattu 8.2.2014].

Wikipedia 2014. [viitattu 14.3.2014].
Saatavissa: http://en.wikipedia.org/wiki/JQuery_Mobile

W3C. 2001. Web Services Description Language [viitattu 7.2.2014]
Saatavissa: http://www.w3.org/TR/wsdl

W3C. 2004. World Wide Web Consortium, Web Service Architecture
[viitattu 1.2.2014]. Saatavissa: http://www.w3.org/TR/ws-arch/

UDDI. 2014. Universal Description, Detection and Integration [viitattu 2.2.2014].
Saatavissa: http://www2.amk.fi/mater/tietotekniikka/nimipalvelut/6_uddi.html