WWW.JAMK.FI

NATIIVIN ANDROID-KEHITYKSEN JA PHONEGAP-KEHITYKSEN VERTAILU

Petteri Aho

Opinnäytetyö
Tammikuu 2013

Tietojenkäsittelyn koulutusohjelma
Luonnontieteiden ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä AHO, Petteri	Julkaisun laji Opinnäytetyö	Päivämäärä 7.1.2013
	Sivumäärä 68	Julkaisun kieli Suomi
	Luottamuksellisuus ( ) saakka	Verkkojulkaisulupa myönnetty (X)

Työn nimi
NATIIVIN ANDROID-KEHITYKSEN JA PHONEGAP-KEHITYKSEN VERTAILU

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Työn ohjaaja
TUIKKA, Tommi

Toimeksiantaja
Codecenter Oy, Thomas Raehalme

Tiivistelmä

Opinnäytetyön toimeksiantajana toimi jyväskyläläinen ohjelmistoalan asiantuntijayritys Codecenter Oy. Codecenter suunnittelee tyypillisesti verkkopohjaisia tietojärjestelmiä palvelemaan liiketoiminnan tarpeita. Yksi näistä on verkkopohjainen tuntikirjanpitojärjestelmä Momento. Kehityksen myötä yritys haluaa laajentaa kasvavassa määrin mobiilisovelluksiin ja integroida näitä sovelluksia valmiisiin tuotteisiin, kuten Momentoon.

Työn tavoitteena oli tutkia, kumpi teknologia on työtehtävien kirjaamiseen mobiililaitteella parempi ratkaisu: Android-natiivikehitys vai alustariippumaton PhoneGap. Työssä verrattiin kahden sovelluksen teknologioita sekä tutkittiin, täyttävätkö nämä asiakkaan asettamat kriteerit testattavuuden, suorituskyvyn, jatkokehityksen, nykyaikaisuuden sekä laadun ja ylläpidon suhteen. Tuloksien perusteella on tarkoitus päättää, kummalla teknologialla on parempi toteuttaa mobiililaitteella toimiva ”kaukosäädin” työtehtävien kirjaamiseksi Momento-tuntikirjanpitojärjestelmään, ja antaa suosituksia yrityksen mobiilisovellusten jatkokehityksen suhteen.

Työn tuloksena saatiin tietoa siitä, voiko web-teknologioilla tehty mobiilisovellus haastaa natiivikehityksen, ja vastaus oli kyllä. Työ on hyvä esimerkki haastamaan ihmisten ennakkoluuloja web-teknologioilla toteutettuja mobiilisovelluksia kohtaan. Lisäksi se on esimerkki siitä, kuinka nykypäivänä ei ole enää pakko luottaa perinteisiin natiiviohjelmointikieliin, vaan kehittäjille on tarjolla uusia mahdollisuuksia, joiden avulla mobiilisovellusten kehittämistä pystytään helpottamaan ja nopeuttamaan. Vaikka kummassakin teknologiassa on hyvät ja huonot puolensa, on tässä tapauksessa PhoneGapilla enemmän hyviä puolia verrattuna Android-natiivikehitykseen, ja hyvät puolet kumoavat kielen pienet epäkohdat.

Työn toimeksiantaja hyötyi työstä saamalla uuden mobiilisovelluksen, jota voidaan jatkokehittää tehokkaasti ja saamalla lisää tietoa web- ja Android-teknologioista tulevaisuutta ajatellen.

Avainsanat (asiasanat)
PhoneGap, Android, Alustariippumaton sovelluskehitys, Natiivikehitys, Mobiilisovellus

Muut tiedot
Liitteet, 8 sivua

Author
AHO, Petteri

Type of publication
Bachelor's Thesis

Date
7.1.2013

Pages
68

Language
Finnish

Confidential

( ) Until

Permission for web publication
(X)

Title
COMPARISON OF NATIVE ANDROID DEVELOPMENT AND PHONEGAP DEVELOPMENT

Degree Programme
Business Information Systems

Tutor
TUIKKA, Tommi

Assigned by
Codecenter Ltd, Thomas Raehalme

Abstract

The thesis was assigned by Codecenter Ltd, Jyväskylä. Codecenter is a software company that is specialized in plan web-based information systems to serve business operation needs. One of their software is an hour accounting software, Momento. The company wants to expand their business more into mobile applications and to integrate these products in the finished application, such as to Momento.

The aim of the thesis was to examine which of the technologies has a better record in working the tasks of a mobile device: native Android development or cross- platform development for PhoneGap. In this study two technologies are compared and whether the conditions set by the client fulfill the criteria for testability, performance, further development, modernity, quality and maintenance. These results indicate the answer to the question: which technology is better to create "a remote control" application, whereby jobs can be saved into Momento hour accounting software and recommendations be made for the further development of mobile business applications.

The result of thesis was the information that it is a possible to create a web-application which is better than a native application. This study is a very good example of the fact that today there are many different options for creating an impressive mobile application. New technologies will make the development easier and faster. Of course, all technologies have pros and cons; however, in this case the PhoneGap is a better technology because of its more excellent properties compared to Android. Good properties also compensate for the cons of the language.

The client benefited from the study among other features by getting new software which is easy to further develop and by getting more information about the web and Android technology of the future.

Keywords
PhoneGap, Android, Cross platform software development, Native development, Mobile application

Miscellaneous
8 pages of attachments

SISÄLTÖ

KUVIOT

TAULUKOT

TERMIT

.apk, Android application package file	Android-käyttöjärjestelmille tarkoitettu sovellusten asennus- ja jakelupaketti
Ajax, Asynchronous JavaScript and XML	Joukko web-sovelluskehityksen tekniikoita, joiden avulla web-sovelluksista voidaan tehdä vuorovaikutteisempia.
Apache Maven	Kirjastoriippuvuuksien hallinta
API, Application Programming Interface	Ohjelmointirajapinta: ohjelmistojen välinen tiedonsiirtorajapinta
Appcelerator	Käyttöjärjestelmien välisten ohjelmointityökalujen valmistaja
C++	C-kielestä kehitetty oliopohjainen ohjelmointikieli
CSS, Cascading Style Sheets	Verkkosivujen muotoiluun käytettävä tyylikieli
Dalvik Virtual Machine	Google Android-käyttöjärjestelmän prosessivirtuaalikone
DOM, Document Object Model	Ohjelmistorajapinta, joka mahdollistaa (X)HTML-dokumenttien sisällön muokkauksen.

GWT, Google Web Toolkit	Joukko työkaluja, joiden avulla voidaan kehittää ja ylläpitää JavaScriptiä Java-kielen avulla.
HTML, HyperText Markup Language	Verkkosivujen merkintäkieli
IDE	Integroitu ohjelmointiympäristö
Java	Laitteistoriippumaton oliopohjainen ohjelmointikieli
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
JSON, JavaScript Object Notation	Yksinkertainen tiedonsiirtomuoto
MVC, Model View Controller	Ohjelmistoarkkitehtuurityyli, jonka tarkoituksena on käyttöliittymän erottaminen sovellustiedostoista.
NFC, Near Field Communication	RFID:hen pohjautuva radiotaajuisen etätunnistuksen mahdollistava tekniikka

Sandboxes	Ympäristö, jossa ohjelmaa voidaan suorittaa siten, että se ei vaikuta isäntäkoneen muun ohjelmiston toimintaan.
SDK, Software Development Kit	Kehitystyökalu, jonka avulla luodaan ohjelmistoja eri alustoille.
SVN, Subversion	Versionhallintajärjestelmä
UI, User Interface	Sovelluksen käyttöliittymä
W3C, World Wide Web Consortium	Www:n standardeja ylläpitävä ja kehittävä organisaatio
WaSP, Web Standards Project	Ryhmä ammattilaisia web-kehittäjiä
WebKit	Selainmoottori, jonka lähdekoodi on kirjoitettu C++-ohjelmointikielellä.
WHATWG, Web Hypertext Application Tecnology Working Group	Ensisijaisesti HTML- ja API-kehitykseen keskittyvä yhteisö
Widget	Mobiililaitteen ”työpöydällä” toimiva aktiivisovellus
XML, Extensible Markup Language	Laajennettavissa oleva rakenteinen kuvaus- ja merkintäkieli

1 JOHDANTO

> *”Maailmassa on tällä hetkellä enemmän mobiililaitteita kuin hammasharjoja.”* Gerard Smit

Mobiililaitteet ovat löytäneet tiensä ihmisten jokapäiväiseen elämään ja vakiintuneet terminä yleiskieleen (Mikkonen 2004, 1). Erilaiset mobiililaitteet, kuten taskutietokoneet, matkapuhelimet ja tiedonsiirron mahdollistavat kannettavat tietokoneet, ovat nousseet keskeiseen asemaan ohjelmistokehityksessä ja samalla ne tarjoavat monipuolisen ympäristön opiskeluun, työelämään ja esimerkiksi sosiaaliseen mediaan. Itse asiassa moderni matkapuhelin pystyy tekemään lähes kaiken saman kuin pöytäkone, mutta sillä on mielekkäämpi merkitys ihmisten jokapäiväisessä elämässä (Fling 2009, 3). Suurin osa laitteiden käyttäjistä ei ole tietoisia siitä, kuinka ohjelmistokehitys pystyy vastaamaan käyttäjän tarpeisiin yhä haastavammista sovelluksista. Nykyaikaisilla mobiililaitteille on ominaista laajennettavuus ja käyttäjän tarpeisiin mukautuminen. Tänä päivänä täytyy varmistaa, että esimerkiksi Android-laitteelle tehdyt sovellukset saadaan tarvittaessa toimimaan myös muilla alustoilla. Teknologian pitäisi olla nykyaikaista, laatuvaatimuksiltaan riittävän tehokasta, kustannuksiltaan halpaa ja sen pitäisi toimia monen eri valmistajan laitteissa.

Alustariippumaton sovelluskehitys on tullut viimeisempien vuosien aikana yritysten ja kolmansien osapuolten tietoisuuteen niiden kehittämisen helppouden, alustariippumattomuuden ja nopeuden ansiosta. Etuna niin kutsutuissa web-sovelluksissa on, että ne kirjoitetaan ohjelmistokehittäjille tutuilla web-teknologioilla: HTML5, CSS3 ja JavaScript. Tämä tarkoittaa nopeampaa ja helpompaa kehitysprosessia. Lisäksi koska web-teknologiat ovat standardoituja, niitä voidaan tänä päivänä käyttää monilla eri mobiilialustoilla. Näin sovelluksesta ei tarvitse välttämättä tehdä erillisiä iOS-, Android- ja Windows Phone -versioita. (Wargo 2012, 13.)

Natiivikehitys taas tarkoittaa sovelluksen kehittämistä tietylle mobiilialustalle, jolloin se myös ohjelmoidaan kehitysympäristön alustakohtaisella ohjelmointikielellä. Natiivikehityksessä käytettävät tekniikat eroavat merkittävästi alustariippumattomaan kehitykseen verrattuna. Natiivikehitys tarjoaa laajan pääsyn laitteen ominaisuuksiin, kuten kameraan ja kiihtyvyysanturiin, mutta natiivikehitys on vain yhden alustan ohjelma, ja se aiheuttaa merkittäviä lisäkustannuksia ja haasteita siirrettäessä ohjelmaa usealle eri alustalle (Wargo 2012, 13).

2 TUTKIMUSASETELMA

2.1 Taustateoriaa, tavoitteet ja rajaukset

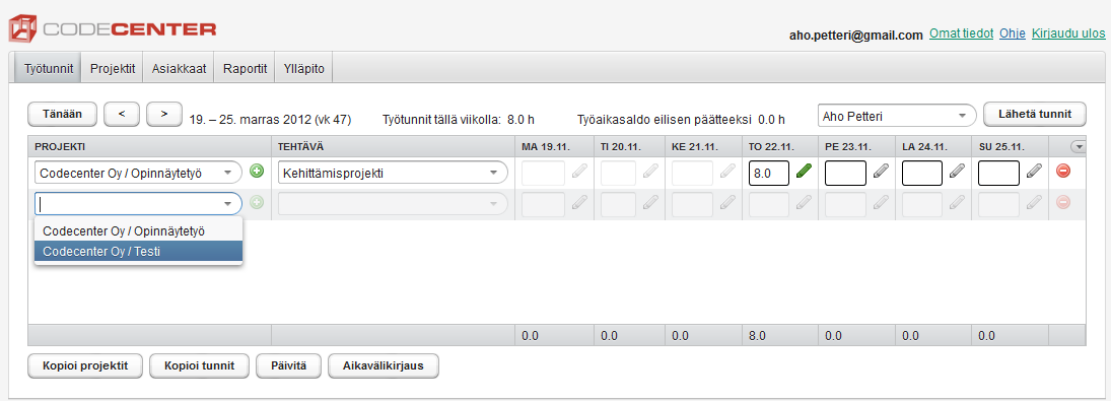
Toimeksiantaja

Opinnäytetyön toimeksiantajana toimii jyväskyläläinen Codecenter Oy ja yhteyshenkilönä yrityksessä Thomas Raehalme. Codecenter Oy on vuonna 2003 perustettu ohjelmistoalan asiantuntijayritys, jonka erityisosaamista ovat Java-teknologiat. Codecenter suunnittelee tyypillisesti verkkopohjaisia tietojärjestelmiä palvelemaan liiketoiminnan tarpeita. Yritys työllistää 10 henkilöä.

Tausta

Keväällä 2012 suoritin viiden kuukauden pituisen työharjoittelujakson Codecenter Oy:ssä. Hyvien kokemuksien pohjalta päätin ottaa opinnäytetyön esille yrityksen sisällä. Keskustelimme yrityksen teknologiajohtajan Thomas Raehalmen kanssa sopivasta aiheesta, ja esille nousi mobiilisovellus työtehtävien kirjaamiseksi verkkopohjaiseen Momento tuntikirjanpitojärjestelmään, koska kehityksen myötä yritys haluaa laajentaa yhä kasvavassa määrin mobiilisovelluksiin ja integroida näitä sovelluksia valmiisiin tuotteisiin. Alkuperäinen idea oli toteuttaa mobiilisovellus natiiviohjelmointikielellä Android-alustalle työtehtävien kirjaamiseksi Momento-järjestelmään, mutta opinnäytetyön ohjaaja suositteli tutustumaan myös alustariippumattomiin toteutus-

vaihtoehtoihin. Kuviossa 1 on esimerkkinä esitetty tuntikirjanpitojärjestelmän toimintaa. Kuviossa on näkyvissä käyttäjän tuntikirjaukset tehtäväkohtaisesti.



KUVIO 1. Momento, käyttäjän tuntikirjaukset

Momento-nimellä kutsutusta sovellusversiosta saatiin rakennettua ensimmäinen kaupallinen versio vuonna 2006. Codecenterin tavoitteena oli, että vuoden 2011 aikana järjestelmän uusi 2.0-versio saataisiin uudistettua toimimaan Vaadin-sovelluskehyksen päällä ja palvelemaan monipuolisesti käyttäjän tarpeita. Järjestelmän avulla käyttäjät voivat luoda mm. tuntikirjauksiin perustuvat laskut ja tulostaa monipuolisia raportteja, joita voidaan käyttää palkanlaskennan perusteena. Raporteista voi seurata myös työn jakautumista asiakastyölle ja sisäiselle työlle.

Teknologiat

Nykyistä teknologiaa, jolla Momenton käyttöliittymä on toteutettu, kutsutaan nimellä Vaadin. Vaadin on Java-web-sovelluskehys, joka on suunniteltu luomaan interaktiivisia sovelluksia, jotka voidaan ajaa selaimessa ilman lisäosia. Sovellus on ohjelmoitu Java-kielellä, ja se ei vaadi ohjelmoijalta HTML:n, CSS:n tai JavaScriptin käyttöä. Kaikki tarpeelliset Java-kirjastot ja työkalut ovat heti ohjelmoijan käytettävissä. (Vaadin 2012.)

Näkökulma ja perustelut

Aihe on hyvin ajankohtainen, minkä voi huomata jo kappaleen yksi ensimmäisestä lainauksesta ”maailmassa on tällä hetkellä enemmän mobiililaitteita kuin hammasharjoja”. Maailma muuttuu, ja siinä muutoksessa yritykset eivät halua jäädä jal-

koihin. Codecenter Oy haluaa tutustua uusiin teknologioihin, jotka vastaavat nykyajan tilannetta. Tähän asti yritys on tehnyt mobiilisovelluksia ensisijaisesti Android-alustoille. Ongelmia yhden alustan sovelluksessa tulee silloin, kun sovellusta halutaan laajentaa muille alustoille. Tyypillisesti noin puolet sovelluksesta joudutaan tässä tapauksessa kirjoittamaan uudelleen, koska suurin osa ohjelman lähdekoodista liittyy käyttöjärjestelmään. Tältä voidaan välttyä alustariippumattomalla kehitysympäristöllä. (Kolehmainen 2010.)

PhoneGap vaikuttaa teoreettisella tasolla muihin sovelluskehyksiin verrattuna huomattavasti kehittyneemmältä, ja sen tarjoamat mahdollisuudet hyödyntää mobiilialustan omaa natiiviohjelmointikieltä raskaampien toimintojen suorittamisessa tekevät siitä mielenkiintoisimman vaihtoehdon mobiilisovellusten kehittämisen kannalta (Kemppainen 2011, 8). Lisäksi työn tekijällä on aikaisempaa kokemusta HTML5-tekniikasta, CSS3:sta, JavaScriptistä, jQuerystä ja jQuery Mobilesta. Näin syntyi ajatus natiivin Android-kehityksen ja PhoneGap-kehityksen vertailusta. Aihetta tarkastellaan yrityksen tarpeiden sekä sovelluskehityksen kannalta.

Työn tavoite

Tämän työn tavoitteena on tutkia, kumpi teknologia on työtehtävien kirjaamiseen mobiililaitteella parempi ratkaisu: Android natiivikehitys vai alustariippumaton PhoneGap. Työssä verrataan kahden sovelluksen teknologioita sekä sitä, täyttävätkö nämä asiakkaan asettamat kriteerit testattavuuden, suorituskyvyn, jatkokehityksen, nykyaikaisuuden sekä laadun ja ylläpidon suhteen. Yrityksen ongelmakohdissa korostetaan erityisesti jatkokehityksen mahdollisuutta ja nykyaikaisuutta. Tuloksien perusteella on tarkoitus päättää, kummalla teknologialla on parempi toteuttaa mobiililaitteella toimiva ”kaukosäädin” työtehtävien kirjaamiseksi Momento-tuntikirjanpitojärjestelmään ja antaa suosituksia yrityksen mobiilisovellusten jatkokehityksen suhteen. Opinnäytetyön lopputuloksena on mobiilisovellus ohjelmointitavasta riippumatta. Työn rajaus keskittyy Android-natiivikehitykseen ja alustariippumattomaan PhoneGap-sovelluskehitykseen. Nämä kaksi asiaa pitävät sisällään kysymyksiä, joiden avulla pystytään tekemään ratkaisu ohjelman toteutustavasta.

Raportointi

Työn raportoinnissa noudatetaan Jyväskylän ammattikorkeakoulun virallista raportointiohjetta. Kaikki saadut tulokset kirjataan opinnäytetyöhön, ja lisäksi niistä keskustellaan asiakkaan kanssa koko prosessin ajan. Tuloksista hyötyvät asiakkaan lisäksi muut yritykset, jotka miettivät alustariippumattomia ratkaisuja mobiilisovellusten toteuttamisessa natiivisovellusten sijasta. Tulokset tulevat olemaan hyvin luotettavia, sillä niiden pohjana käytetään tieteellisiä tutkimuksia ja alan ammattilaisten julkaisuja. Opinnäytetyö tulee olemaan kokonaisuudessaan julkinen.

2.2 Tutkimusmenetelmät

Opinnäytetyö on yrityksen mobiilisovellusten kehittämistutkimus. Kehittämistutkimuksessa on taustalla ilmiö, prosessi tai asiantila, jonka halutaan olevan kehittämisen tai muutoksen jälkeen paremmin. Kehittämisen kohde voidaan pukea ongelman muotoon. Kehittämistutkimuksessa, kuten kaikessa muussakin tutkimuksessa, on usein kyse myös olemassa olevan ratkaisun viemisestä tai soveltamisesta erilaiseen toimintaympäristöön. (Kananen 2012, 13.)

Teknologioita tutkitaan kirjallisuuden, käytänteiden, mallien ja jo tehtyjen tutkimusten avulla. Näiden avulla muodostetaan opinnäytetyön tietoperusta, joka toimii kehittäjän teknologiaoppaana ja jonka avulla pystytään kartoittamaan teknologioiden heikkouksia ja etuja. Näitä heikkouksia ja etuja verrataan yrityksen tarpeisiin sovelluksen suhteen. Työssä ilmenevät kehittämisprojektille tyypilliset piirteet, kuten ongelmalähtöisyys, tavoitteellisuus, osallistuvuus, teoriaperusta, suunnitelmallisuus, toteutus ja ajallisesti rajattu kertaluontoinen tehtäväkokonaisuus (Bister 2011, 9).

2.3 Tutkimuskysymykset

Tutkimuskysymys:

- Kumpi teknologia on työtehtävien kirjaamiseen mobiililaitteella parempi ratkaisu: Android natiivikehitys vai alustariippumaton PhoneGap?

Alakysymykset:

- Mitä vaatimuksia käytännön toiminta asettaa sovellukselle?
- Miten teknologian paremmuutta arvioidaan?
- Millainen käyttöliittymän tulisi olla?

3 PHONEGAP

3.1 PhoneGap-sovelluskehitys

PhoneGap aloitti toimintansa vuonna 2008 ja se on Adobe Systemsin omistama avoimen lähdekoodin HTML5-sovelluskehys, jolla voidaan rakentaa monen alustan natiivisovelluksia käyttäen hyödyksi web-teknologioita. Tyypillisellä web-sovelluksella ei ole kuitenkaan pääsyä mobiililaitteen kaikkiin ominaisuuksiin, kuten kameraan ja paikannukseen. PhoneGap tarjoaa tähän tarkoitukseen JavaScript API -rajapinnan, jolla pystytään rakentamaan niin kutsuttuja hybridisovelluksia käyttäen apuna laitteen natiiviominaisuuksia. Hybridisovellukset näyttävät ulospäin natiivisovellusten tapaisilta, mutta käyttävät selainta toimiakseen ilman minkäänlaista asennusta puhelimeen – voisi luulla, että selainpohjaisen sovelluksen käyttöön tarvitaan aina Internet-yhteys, mutta näin ei ole. HTML5:ssä on ominaisuus, joka mahdollistaa sovelluksen käytön ilman Internet-yhteyttä. PhoneGap tukee myös seitsemän eri laitevalmis-

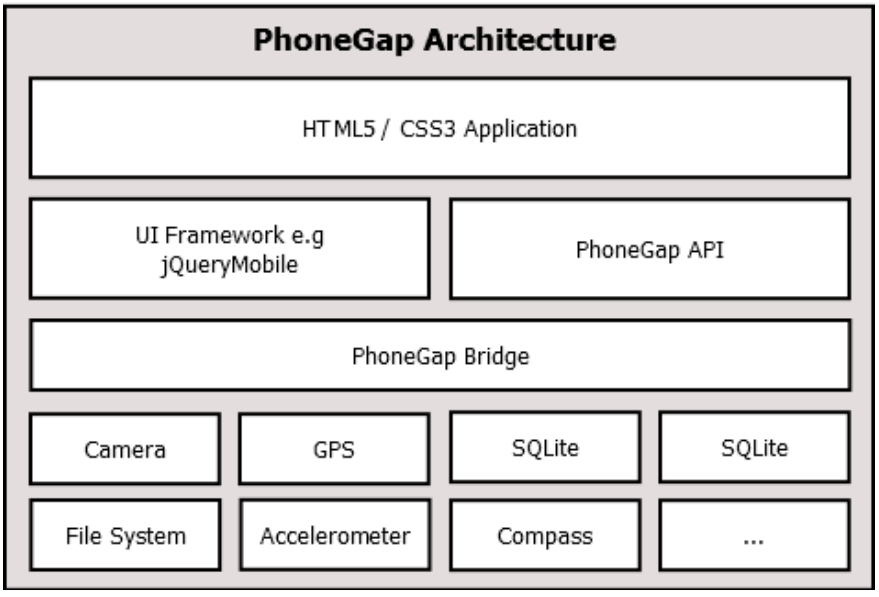
tajan tuotteita: iPhone, Android, Blackberry, WebOS, Windows Phone 7, Symbian ja Bada. (Ghatol & Patel 2012, 17.)

Kustannukset

PhoneGapin käyttö on ilmaista, kun sillä luodaan sovelluksia käyttäjän omassa kehitysympäristössä tai PhoneGap Build -palvelussa, jotka ovat kaikille käyttäjille ilmaiseksi ladattavissa. Jos sovelluksia halutaan kuitenkin tehdä yksityiseen käyttöön enemmän kuin yksi, niin silloin hinta on 9.99 dollaria kuukaudessa, mikä taas antaa oikeuden tehdä 25 sovellusta kuukaudessa. (Adobe PhoneGap Build n.d.)

3.2 PhoneGapin arkkitehtuuri

Kuviossa 2 tutkitaan PhoneGap-sovelluskehyksen arkkitehtuuria, jotta ymmärrettäisiin, mitä kerroksia se pitää sisällään.



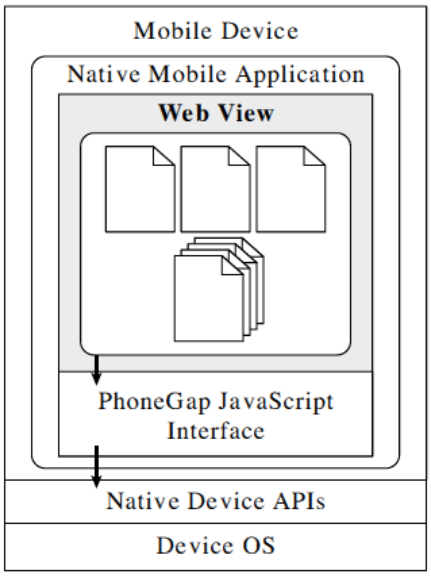
KUVIO 2. PhoneGapin arkkitehtuuri (Ghatol & Patel 2012, 18)

Kuviosta 2 on huomattavissa, että PhoneGapissa on käytännössä kaksi tärkeää osaa, jotka voidaan jaotella seuraavasti:

1. JavaScriptin liiketoimintalogiikka, joka ohjaa UI:ta ja sen toimivuutta.
2. PhoneGap API, joka pääsee käyttämään laitteen natiiviominaisuuksia.

3.3 PhoneGap-web-sovelluksen toimintaperiaate

Käyttäjän suorittaessa tyypillistä web-sovellusta ohjelma lataa ensisijaisesti aloitussivuksi index.html-tiedoston, jonka seurauksena käyttäjälle aukeaa selaimessa toimiva sovellus. PhoneGapilla tehty web-sovellus pystytään vastaavasti paketoimaan valmiiksi natiivisovellukseksi PhoneGap Build -palvelussa eri valmistajien mobiililaitteille, jolloin sovelluksen käynnistyessä se toimii kuin mikä tahansa natiivisovellus piilottaen selaimen ”kehykset” ja käyttäen halutessaan apuna laitteen natiiviominaisuuksia. Tyypilliseltä web-sovellukselta tämä ei onnistuisi, koska sillä ei ole mahdollisuutta päästä käsiksi laitteen natiiviominaisuuksiin. (Wargo 2012, 7–8.) Kuviossa 3 on esitelty PhoneGap-sovelluksen vuorovaikutusta eri osien välillä.



KUVIO 3. PhoneGap-sovelluksen vuorovaikutus (Wargo 2012, 8)

3.4 PhoneGapin ohjelmointikielet

Web-sovellukset ohjelmoidaan tyypillisesti käyttäen web-tekniikoita, mutta JavaScriptin tueksi kannattaa ottaa jQuery-kirjasto ja käyttöliittymän luonnin helpottamiseksi jokin seuraavista sovelluskehyksistä: Sencha Touch, jQuery Mobile, The M-Project tai jQTouch.

HTML5

HTML5 on iskusana tai yleisnimitys työlle, jota kehittävät W3C ja WHATWG HTML-kielen laajentamiseksi ja HTML-dokumentteihin liittyvien toimintojen määrittelemiseksi. Sitä käytetään myös yleisesti puhuttaessa webin uusista suuntauksista ja tekniikoista jopa niin, että kaikki uusi selitetään HTML5:een kuuluvaksi. Samalla HTML5 on jatkuvasti muuttuva luonnos HTML-kielen uudeksi määrittelyksi ja sen osittaisia toteutuksia. (Korpela 2011, 3.)

HTML5:n kehitystyö on ollut hidasta, ja sen voi katsoa alkaneen vuonna 2004 ja saaneen virallisen aseman vuonna 2007. W3C on ilmoittanut uudesta suunnitelmasta, jonka mukaan HTML5-spesifikaatiot siirretään ”Recommended”-tilaan vuoden 2014 loppuun mennessä ja HTML5.1-spesifikaatiot vuoden 2016 loppuun mennessä. ”Recommended”-tila merkitsee W3C:n termein valmista, lopullista versiota standardista. (Laine 2012.) Esimerkki 1 havainnollistaa HTML5-koodin perusrakenteen.

```
<!DOCTYPE html>
<meta charset="windows-1252">
<title>Uusi puhelin tulee nyt!</title>
<header>
    <nav><ul><li><a href="etusivu.html">Etusivu</a></li></ul></nav>
    <h1>Uusi Gynoid 100 - huippumallimme</h1>
</header>
<figure>
    <img alt="" src="gynoid-100.jpg">
    <figcaption>Klassista kauneutta ja modermia käytettävyyttä.</figcaption>
</figure>
<p>Uusi Gynoid 100 on markkinoiden monipuolisin puhelin.</p>
<aside>Uusi Gynoid 100 on ominaisuuksiltaa päivitetty versio vanhaan Gynoid-puhelimeen.</aside>
<footer><p>Päivitetty <time datetime="2012-10-01">1.10.2012</time>.</p></footer>
```

Esimerkki 1. HTML5-koodin perusrakenne

Esimerkissä 1 on esitelty seuraavat uudet elementit dokumentin sisäisen rakenteen kuvaamiseen, jotka poikkeavat vanhoista HTML-määrityksistä (Korpela 2011, 86):

- aside – sivuasiaa
- figcaption – kuvateksti
- figure – kuvitus
- footer – alatunniste, esimerkiksi tekijätiedot

- header – ylätunniste, esimerkiksi logo ja sisällysluettelo
- nav – navigointiosa, sivuston sisäinen navigointi

CSS3

CSS3-kieli on W3C:n ylläpitämä uuden sukupolven tyylikieli, joka mahdollistaa aikaisempaa monipuolisemman ja selkeämmän ulkoasun web-sivuille. Tyylikielellä kerrotaan sivustolle, miltä sen tulisi näyttää ja samalla myös erotellaan itse sisältö ulkoasusta selkeäksi kokonaisuudeksi. Vaikka viime vuosina on ollut puhetta ”uudesta” CSS3-kielestä, niin itse asiassa CSS3:n kehittäminen alkoi jo vuonna 1998. Vuonna 2005 kaikki CSS3-moduulit siirrettiin kuitenkin takaisin kehittämistilaan ja prosessi aloitettiin uudelleen. Tällä hetkellä monet CSS3-moduuleista ovat jo erittäin vakaita nykyaikaisissa selaimissa. (Gasston 2011, 1–3.) Esimerkki 2 havainnollistaa CSS-tyylitiedoston rakenteen.

```
valitsin {
        ominaisuus: arvo;
}

div {
        margin: 2em;
        padding: 1em;
        border: 2px solid green;
        border-radius: 15px;
        text-shadow: 1px 2px 3px red;
        box-shadow: 10px 10px 15px blue;
}
```

Esimerkki 2. CSS-tyylitiedoston rakenne

CSS:n perussyntaksi muodostuu valitsimesta, ominaisuudesta ja arvosta. Kaikki ominaisuudet ja arvot tulevat aaltosulkujen sisään. Esimerkissä on esitelty seuraavat CSS3:n uudet elementit, jotka poikkeavat vanhoista CSS-määrityksistä (Gasston 2011, 65–80):

- border-radius – kulmien pyöristys
- text-shadow – tekstin varjostus
- box-shadow – laatikon varjostus

JavaScript

JavaScriptin kehitti alun perin vuonna 1995 Netscape Communications Corporation ja sen nykymuoto on dynaamisesti tyypitetty, tulkattava oliopohjainen komentosarjakieli, jonka syntaksi perustuu löyhästi C-ohjelmointikieleen. Sen tärkein tehtävä on lisätä Internet-sivuille vuorovaikutteisuutta ja dynaamista toiminnallisuutta. JavaScript voi esimerkiksi näyttää välittömästi virheviestin, jos käyttäjä yrittää lähettää web-lomakkeen, josta puuttuvat tarvittavat tiedot. (Sawyer 2011, 1–3.)

JavaScript on tänä päivänä hyvin arvostettu ohjelmointikieli, ja se on laajentunut hurjaa vauhtia – yksi kielen hienouksista onkin sen helppo laajennettavuus. JavaScriptin laajennukset mahdollistavat esimerkiksi rakennesovelluskehysten käyttämisen MVC-mallin toteuttamiseen. Näistä yleisimpiä ovat Backbone.js ja Angular.js. Vaikka JavaScript on helposti laajennettavissa, sillä on muutama kiusallinen heikkous. Heikkouksiksi koetaan selainten pienet yksityiskohtaiset erot ja se, että monet ihmiset, kuten web-suunnittelijat kokevat ohjelmoinnin sillä vaikeaksi. Vaikka ohjelma toimisi hyvin Google Chrome -selaimella, se ei välttämättä toimi Internet Explorerilla. Tämä tilanne tekee kielestä vaikean ohjelmoida, ja voi maksaa paljon rahaa, kun sovellusta joudutaan optimoimaan monella eri selaimella. (Sawyer 2011, 3–4.) Esimerkki 3 havainnollistaa JavaScriptin toiminnan upotettuna HTML5-dokumentin sisään.

```
<!DOCTYPE html>
<html>
<body>
<button onclick="myFunction()">Kokeile!</button>
<p id="testi"></p>
<script>
function myFunction(){
    var x = "";
    var time = new Date().getHours();
    x =(time < 20) ? "Hyvää päivää!" : "Hyvää iltaa!";
    document.getElementById("testi").innerHTML = x;
}
</script>
</body>
</html>
```

Esimerkki 3. JavaScripti upotettuna HTML-dokumenttiin

Esimerkki 3 liittää tavalliseen HTML-painikkeeseen toiminnon, joka tulostaa tekstin ”Hyvää päivää!” tai ”Hyvää iltaa!” sille määrätyn ehdon ja kellonajan mukaan.

jQuery

jQuery on vuonna 2006 julkaistu selaimille tarkoitettu ilmainen ja avoimen lähdekoodin JavaScript-kirjasto, joka on tehty selaimen yhteensopivuusongelmien välttämiseksi ja jonka avulla vältetään sovelluksen optimointi eri selaimille. jQueryn syntaksi on tehty mahdollisimman helposti ymmärrettäväksi, mikä tekee siitä maailman suosituimman JavaScript-kirjaston. jQueryn käyttö mahdollistaa myös animaatioiden tekemisen, toimintojen käsittelyn, DOM-elementtien valitsemisen ja Ajax-sovelluksien toteutuksen. (Sawyer 2011, 3–4.) Toisin sanoen jQuery-kirjasto itsessään ratkaisee kaksi JavaScriptin suurinta ongelmaa: kielen monimutkaisuuden sekä selainten pienet eroavaisuudet. Esimerkki 4:ssä on perinteisellä JavaScriptillä sekä jQueryllä tehty linkkien värimuunnos.

```
/* Linkkien värien vaihto JavaScriptillä */
container = document.getElementById('container');
for (i = 0; i < container.childNodes.length; i++) {
    if (container.childNodes[i].nodeName == 'a') {
      if (container.childNodes[i].style.className == 'link') {
          container.childNodes[i].style.color = '#000000';
      }
    }
}

/* Linkkien värien vaihto jQueryllä */
$('#container a.link').css('color', '#000000');
```

Esimerkki 4. Linkkien värimuunnos

Esimerkissä 4 on huomattavissa, että jQueryllä sama värimuunnos pystytään tekemään paljon yksinkertaisemmin kuin perinteisellä JavaScriptillä. Yksinkertaisuus korostuu, mitä monimutkaisimpia hakuehtoja ja animaatioita tarvitaan. Opinnäytetyössä jQuery-kirjastoa käytetään ohjelmakoodin selkeyttämiseen.

jQuery Mobile

jQuery Mobile on helppokäyttöinen UI-sovelluskehys, jolla voidaan helpottaa käyttöliittymän luontia ja vuorovaikutusta käyttäjän kanssa mm. eri valmistajien puhelinmalleissa, pöytäkoneissa, tableteissa ja e-kirjan lukulaitteissa. jQuery Mobile käyttää ulkoasun luomiseen HTML5- ja CSS3-ominaisuuksia, ja sen toiminnot pohjautuvat jQuery-kirjastoon (Reid 2011, 1–2). Kirjasto sisältää runsaan valikoiman valmiita käyt-

töliittymäkomponentteja mobiilisovelluksen rakentamiseen. Mukana on esimerkiksi valikoita, painikkeita, lomakkeita ja työkalurivejä. jQuery Mobile osaa myös ladata sovellukseen sisältöä Ajaxilla ilman kokonaista sivun lataamista ja se sopii käyttöliittymän luontiin PhoneGap-sovelluskehyksen kanssa. (Broulik 2011, 1.)

Todd Parker jQuery-projektista kertoo, että uusi jQuery Mobile 1.0 on 30–50 prosenttia nopeampi kuin aikaisempi työversio. Tästä huolimatta tekniikka on saanut nettikommentoijilta kritiikkiä suorituskyvystä. Erityistä kritiikkiä saavat animoidut sivujen väliset siirtymät, jotka usein nykivät ja pomppivat oudosti. (Sani 2011.)

Paras tapa ymmärtää jQuery Mobilea on havainnollistaa asiaa esimerkin avulla. Esimerkki 5:ssä on rakennettu yksinkertainen Internet-sivu käyttäen apuna jQuery Mobilea.

```
<!DOCTYPE html>
<html>
<head>
    <title>jQuery Mobile Application</title>
    <link rel="stylesheet" href="http://code.jquery.com/mobile/1.2.0/
    jquery.mobile-1.2.0.min.css" />
    <script src="http://code.jquery.com/jquery-1.8.2.min.js"></script>
    <script src="http://code.jquery.com/mobile/1.2.0/
    jquery.mobile-1.2.0.min.js"></script>
</head>
<body>
    <section data-role="page">
        <header data-role="header"><h1>jQuery Mobile</h1></header>
        <div data-role="content">
            <p>First page!</p>
        </div>
        <footer data-role="footer"><h4>O'Reilly</h4></footer>
    </section>
</body>
</html>
```

Esimerkki 5. jQuery Mobile-sivu

Esimerkistä 5 voi huomata, että valikko on tavanomainen HTML-lista, johon lisätään HTML5:n data-attribuutteja. jQuery Mobile muotoilee näiden perusteella listasta valikon, joka näyttää mobiilisovelluksen tyyliseltä. jQuery Mobilella luotuja sovelluksia voi käyttää myös vanhemmissa puhelimissa, joissa eivät toimi esimerkiksi animoidut tehosteet. Sovellukset näyttävät näissä puhelimissa tavanomaisilta websivuilta. (Sani 2011.)

4 ANDROID

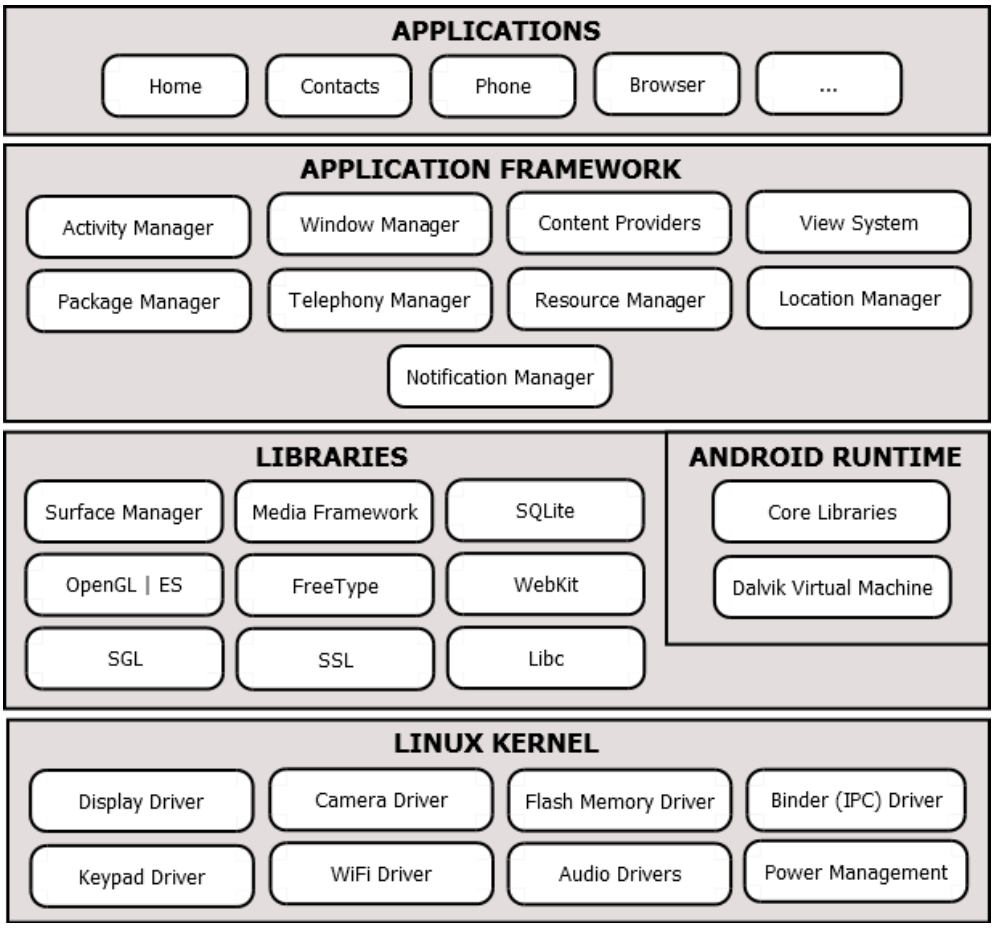
4.1 Android-natiivikehitys

Android on mobiilikäyttöjärjestelmä, joka sisältää käyttöjärjestelmän, väliohjelmistoja ja käyttäjän perusohjelmia. Siinä käytetään avoimen lähdekoodin modifioitua Linux-käyttöjärjestelmää. Järjestelmän on kehittänyt alun perin Android Inc., mutta vuonna 2005 Google osti Androidin kehitystyön sekä sen kehitystiimin osana Googlen uutta mobiilistrategiaa. Nykyisin sen kehittämisestä vastaa Open Handset Alliance, joka koostuu 84 laitteisto- ja ohjelmistovalmistajasta sekä teleoperaattorista. Google halusi Androidin olevan avoin ja maksuton, ja siksi suurin osa koodista on julkaistu avoimella Apache-Licensillä, joten kuka tahansa voi ladata Androidin-lähdekoodit ja kehittää sitä. (Lee 2012, 2.)

Androidiin tarkoitettua koodia kirjoitetaan Java-kielellä, ja se käyttää Googlen kehittämiä Java-kirjastoja. Androidille pystytään luomaa myös web-sovelluksia, jotka käyttävät hyödyksi samoja web-teknologioita ja JavaScript API -rajapintoja kuin PhoneGap. Sovellus voidaan kääntää myös natiivisovellukseksi, mutta silloin sen toiminta on sidottu vain Android-laitteeseen. Android tarjoaa myös Unified Remote -lähestymistavan sovelluskehityksessä. Unified Remote mahdollistaa puhelimen tai tabletin toiminnan kaukosäätimen tavoin tietokoneelle, jonka avulla voidaan hallita erilaisia ohjelmia, kuten elokuvasoittimien play- ja stop-painikkeita. (Lee 2012, 2.)

4.2 Androidin arkkitehtuuri

Androidin toiminnan ymmärtämiseksi on syytä katsoa kuviota 4. Kuviossa esitellään kerrokset, jotka muodostavat Androidin käyttöjärjestelmän. Android OS on karkeasti jaoteltu neljään eri kerrokseen.



KUVIO 4. Android-arkkitehtuuri (Lee 2012, 5)

- Linux Kernel – ydin, johon Androidin toiminta perustuu. Tämä kerros sisältää kaikki alhaisen tason laiteajurit Android-laitteen laitekomponentteihin.

- Libraries – kirjasto sisältää ohjelmakoodit, jotka taas sisältävät kaikki pääpiirteet Android-laitteelle. Kirjasto tarjoaa esimerkiksi SQLite-tietokannan tai WebKit-kirjaston, jossa on toiminnallisuuksia web-selailuun.

- Android Runtime – tämä kerros kuuluu samalle tasolle kirjaston kanssa. Android runtime tarjoaa joukon ytimen kirjastoja, joiden avulla kehittäjät voivat kirjoittaa Android-sovelluksia käyttäen Java-ohjelmointikieltä. Kerros sisältää myös Dalvik-virtuaalikoneen, joka mahdollistaa Android-sovellusten ajamisen omissa prosesseissa.

- Application Framework – mahdollistaa erilaisia sovelluskehyksiä, joita voidaan käyttää sovelluksen kehittämisessä.

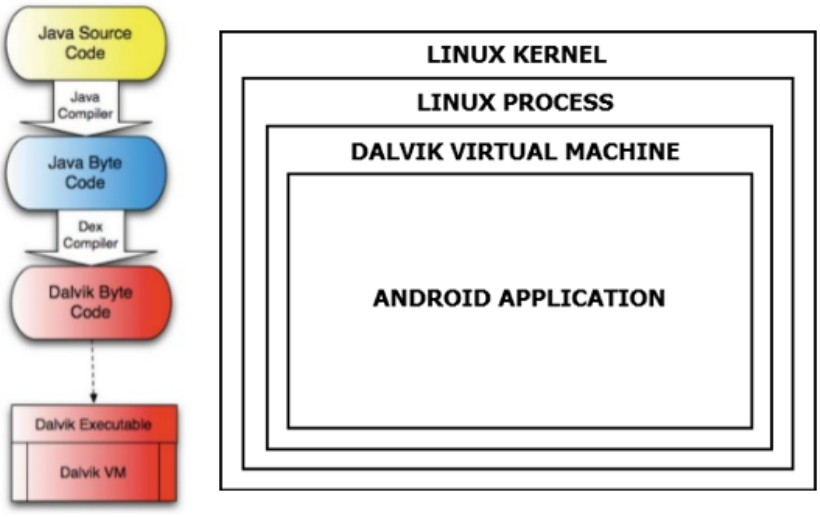
- Applications – pintakerroksella pystytään löytämään eri sovelluksia laitteelle, kuten yhteystiedot sekä sovellukset, jotka on ladattu ja asennettu Android-kaupasta. (Lee 2012, 4.)

4.3 Android-sovelluksen toimintaperiaate

On kaksi tapaa tehdä Android-sovellus: natiivisovellus, joka asennetaan käyttäjän puhelimeen .apk-päätteisenä, tai web-sovellus.

Natiivisovellus

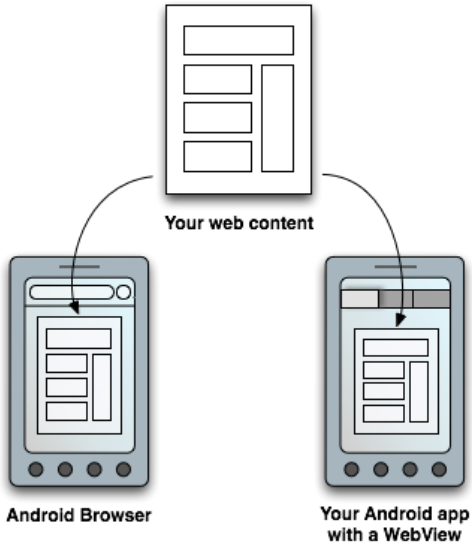
Android on rakennettu Linux Kernelin päälle, joka sitoo sisemmät kerrokset itseensä. Sovelluksen käynnistyessä Android kutsuu aluksi näitä sisempiä kerroksia, minkä seurauksena käyttäjälle aukeaa haluttu sovellus. Sovellus on tässä tapauksessa natiivisovellus, joka on ohjelmoitu Java-kielellä ja käyttää apunaan Dalvik Virtual Machinea. (Laracker 2011, 4.) Kuviossa 5 on esitetty kuvana laitteen vuorovaikutus eri osien välillä.



KUVIO 5. Android-sovellus: laitteen vuorovaikutus (Laracker 2011, 3–4)

Web-sovellus

Vaihtoehtoinen tapa luoda mobiilisovellus Androidille on web-sovellus. Web-sovellusta voidaan ajaa Androidin selaimessa tai WebView-näkymässä, jolloin sovellus toimii natiivisovelluksen omaisesti Android-laitteella. Kuvio 6 visualisoi, kuinka sovellus voi tarjota pääsyn web-selaimeen tai Android-sovellukseen. (Web Apps Overview 2012.)



KUVIO 6. Android-web-sovelluksen toiminta (Web Apps Overview 2012)

4.4 Androidin ohjelmointikielet

Android-sovelluskehitys tehdään pääasiassa Javalla, jolloin sovellukset toimivat Dalvik-virtuaalikoneen päällä. Myös suora natiivikehitys laitteistolle on mahdollista NDK:n (Native Development Kit) avulla. Tällöin käytetään C++-kieltä ja sovellukset toimivat suoraan Linux-ytimessä virtuaalikoneen sijaan. Androidin web-ohjelmointikieliä ei ole mainittu luvussa 4, koska ne perustuvat samoihin teknologioihin kuin PhoneGapissä (HTML5, CSS3 ja JavaScript).

Java

Java on Sun Microsystemsin 1990-luvun alussa kehittämä suuren suosion saavuttanut järjestelmäriippumaton oliopohjainen ohjelmointikieli, joka muistuttaa syntaksiltaan paljon C++-kieltä, mutta jossa on helpommin omaksuttava suunniteltu kielioppi. Javan virtuaalikoneen mukana tulee myös erittäin kattava standardi- ja luokkakirjasto. Myös muistienhallintaa on helpotettu ottamalla käyttöön roskienkeräin, joka vapauttaa muistia, kun sitä ei enää tarvita, ja lisäksi Java kuuluu ohjelmointikieleen, joissa on käytössä vahva tyypitys. (Kosonen ym. 2005, 9–160.) Järjestelmäriippumattomuus tarkoittaa, että Java-ohjelmia voi kehittää ja testata esimerkiksi Linuxissa, mutta sama koodi toimii myös esimerkiksi Windowsissa. Tuettavien käyttöympäristöjen määrä kasvaa jatkuvasti, ja nykyään Javalla ohjelmoidaan mm. matkapuhelimia, digitelevisioita, sulautettuja järjestelmiä ja PDA-kämmentietokoneita, ja se on käytössä noin 3,8 miljardissa laitteessa. (Kosonen, Peltomäki & Silander 2005, 2.)

Java-kielen ominaisuuksista tärkeimpiä ovat (Kosonen ym. 2005, 11):

- Oliopohjaisuus
- Yksinkertaisuus
- Järjestelmäriippumattomuus
- Turvallisuus
- Tuki verkko-ohjelmoinnille
- Tuki moniajolle (säikeet)
- Virheiden käsittely
- Ohjelmien siirrettävyys

Esimerkki 6:ssa on toteutettu Java-kielellä tyypillinen ”Hello World” -esimerkki.

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

Esimerkki 6. Java-kielen rakenne

XML

Androidin peruskirjastoon perustuvassa sovelluskehityksessä käyttöliittymät tehdään XML:llä sen erottamiseksi muusta koodista. XML on W3C:n suositus rakenteisten dokumenttien merkkauskieleksi, joka auttaa jäsentämään laajoja tietomassoja selkeämmin. XML on oikeastaan yksinkertaistettu muoto SGML-kielestä, joka on kansainvälisten asiakirjojen standardi vuodesta 1980 alkaen. XML on tekstimuotoista ja muistuttaa HTML-kieltä. Eräs epätieteellinen tapa onkin ajatella XML:ää niin, että se on HTML:stä siivottu ja yhdistynyt kieli. Tärkeä ero XML:n ja HTML:n välillä on se, että HTML on merkkauskieli, kun taas XML:llä määritellään merkkaukseen käytettäviä sanastoja. (Kosonen ym. 2005, 527.)

XML määrittää osittain myös XML-sovellusten toimintaa antamalla niille dokumentin tietosisältöön kuulumattomia prosessointiohjeita. XML:ssä ei ole ennalta luotuja tunnisteita, vaan kehittäjä voi luoda omat XML-tunnisteet ja jäsentää dokumentin vastaamaan omia tarpeitaan. On muistettava, että vaikka XML ei sisällä ennalta määrättyjä tunnisteita, se sisältää silti hyvin konkreettisia sääntöjä kielen syntaksista. (Dykes 2005, 12.)

XML-kielen tärkeimmät edut (Kosonen ym. 2005, 573–574):

- Siirrettävyys
- Sisältöjen yhdenmukaisempi tallennusmuoto
- Tyylin määrittely
- Entiteettien säilyttäminen
- Linkittäminen (integraatioiden helpottuminen)
- Tiedon hakemisen helpottuminen
- Hierarkkisuus
- Käsittelyvaiheiden automatisointi

Esimerkissä 7 on XML-kielellä toteutettu dokumentti.

```
<?xml version="1.0"?>
<note>
    <to>Juha</to>
    <from>Petteri</from>
    <heading>Muistutus</heading>
    <body>Lähetä viesti Juhalle!</body>
</note>
```

Esimerkki 7. XML-dokumentin rakenne

C++

Kun oliopohjainen analyysi, suunnittelu ja ohjelmointi saivat jalansijaa, alkoi Bjarne Stroupstrup laajentaa maailman suosituinta ohjelmointikieltä, C-kieltä, ja kehitti siihen uusia piirteitä, jotka mahdollistivat oliopohjaisen ohjelmoinnin. Hän loi C++-kielen 1980-luvulla C-kielestä lisäämällä siihen olio-ohjelmointiin ja geneerisyyteen liittyviä ominaisuuksia. C++ sisältää oliopohjaisen ohjelmoinnin peruspilarit: kapselointi, periytyvyys, polymorfia, tiedon kätkeminen, luokat ja poikkeukset, mutta sen yhtenä pääperiaatteena on ollut kaiken ylimääräisen suorituksenaikaisen koodin jättäminen pois, kuten roskienkeräyksen (Deitel & Deitel 2010, 9). Uusin C++-kielen standardi vahvistettiin vuonna 2011. Nykyisin C++:lla on kirjoitettu suurin osa maailman käyttöjärjestelmistä, sulautetuista järjestelmistä, ohjelmistoista, verkko-ohjelmista ja peliohjelmista. Myös osa C:llä kirjoitetusta ohjelmista on kelvollista C++, mutta eivät kuitenkaan kaikki mm. uusien varattujen sanojen ja tarkemman tyyppitarkastuksen vuoksi. (Liberty 1998, 5–8.)

Esimerkissä 8:n on C++-kielellä toteutettu tyypillinen ”Hello World” -esimerkki.

```
#include <iostream>

int main(){
    std::cout << "Hello, world!" << std::endl;
    return 0;
}
```

Esimerkki 8. C++-kielen rakenne

5 SOVELLUKSEN MÄÄRITTELY

Viidennessä luvussa käsitellään sovelluksen määrittely, joka antaa pohjatiedon teknologioiden tarkempaan vertailuun myöhemmässä vaiheessa. Pääasia on hahmottaa sovelluksen tarpeet analysoiden yrityksen nykyistä toimintatapaa ja näiden perusteella laatia vaatimusmäärittelyt. Lopuksi vaatimusmäärittelyn perustelleella hahmotetaan tarvittavan käyttöliittymän rakenne.

5.1 Esitutkimus

Toimeksiantajaa haastateltaessa kävi ilmi, että yrityksen nykyisen Android-tekniikan rinnalle kaivattaisiin mahdollisesti uusia tekniikoita, jos ne täyttäisivät yrityksen asettamat kriteerit teknologian suhteen. Nykyisen toimintatavan ongelmana on sovelluksen laajennettavuus muille alustoille. Jopa Google sanoo, ettei ole niin rikas, että voisi tukea natiivisti kaikkia mobiilialustoja.

Kehitystarpeena nähdään, että kehitetyn sovelluksen pitäisi kääntyä monelle eri valmistajan alustalle tukien nykyaikaisia teknologiaratkaisuja. Nykyisellä menetelmällä markkinakunta rajoittuu suurempiin yrityksiin, joissa tiedostetaan, mille laitteelle sovellus on kehitetty. Nämä sovellukset on optimoitu vain pienelle osalle käyttäjiä heidän puhelinmallistaan riippuen. Alustariippumaton PhoneGap-sovelluskehys mahdollistaisi sen, että sovelluksia voitaisiin markkinoida yksittäisille käyttäjille heidän puhelinmallistaan riippumatta.

5.2 Vaatimusmäärittely

Ohjelmiston vaatimusmäärittelyn tavoitteena on selvittää ohjelmistoprojektin tavoitteita ja vaatimuksia. Siinä määritellään, kuinka lopullisen ohjelmiston tulisi toimia, mitä toiminnallisuuksia ohjelmalta tarvitaan, mitä toimintoja toteuttaminen vaatii kehitysalustalta ja millä keinolla nämä toiminnallisuudet saavutetaan. (Paakki 2011, 1–9.) Selvitysten perusteella saadut toiminnalliset ja ei-toiminnalliset vaatimusmää-

rittelyt ovat kirjattu liitteeseen 1. Vaadittavat toiminnot sovellukselle kirjataan myös ylös yrityksen antamien suositusten ja vaatimusmäärittelyjen perusteella erilliseen taulukkoon (liite 2). Liitteeseen 2 etsitään vastaukset luvussa kuusi, ja tämän perusteella katsotaan, onko sovellus toteutettavissa PhoneGapilla vai Androidilla ja kuinka helposti tämä tapahtuu.

Sovelluksen vaatimusmäärityksen laaditaan yhdessä työn toimeksiantajan ja tekijän kanssa. Ennen sovelluksen varsinaista ohjelmoimista kaikki osapuolet voivat todeta vaatimusmäärityksistä, mitä osia sovellus tulee sisältämään, mitä toimintoja näissä osioissa tulee olemaan ja onko järjestelmä sellainen kuin sen tulee olla siihen tarpeeseen, johon sovellusta kehitetään.

5.3 Käyttöliittämäsuunnittelu

> *”Käytettävyyden olemassaolon huomaa parhaiten siellä, missä sitä ei ole.”* Jarkko Immonen

Käyttöliittymäsuunnittelun myötä yritykset ovat alkaneet kiinnittää enemmän huomiota käyttöliittymiin, koska hyvä käyttöliittymä tarkoittaa enemmän ostajia, säästöä, vähemmän muutoksia, vähemmän kysely- ja tukipyyntöjä ja hyvää mainosta yritykselle. Mobiilisovellusta rakennettaessa visuaalinen suunnittelu on kaiken perusta, koska hyvä muotoilu antaa käyttäjälle kuvan laadukkaasta sovelluksesta, ja näin käyttäjä haluaa viettää aikaa sovelluksen parissa useista minuuteista tunteihin. (Fling 2009, 109–110.)

Käytettävyys

Käytettävyydellä tarkoitetaan sitä, miten sovelluksen rakenteesta saadaan selkeä ja johdonmukainen, kuinka hyvin sovellus sopii tietyille käyttäjille tietyn tehtäväkokonaisuuden suorittamiseen tietyssä ympäristössä ja millaisia henkisiä ja fyysisiä ponnisteluja sen käyttäminen vaatii. Onkin tärkeää muistaa, että tuote tulee suunnitella loppukäyttäjille, ei käyttäjien esimiehille, suunnittelijalle itselleen eikä markkinointi-

osastolle. (Immonen 2012.) Käyttäjien tarpeiden määrittelemiseksi voidaan miettiä seuraavia kysymyksiä (Fling 2009, 116):

1. Keitä ovat käyttäjät? Mitä tiedämme heistä? Millaista käyttäytymistä heiltä voi olettaa tai ennustaa sovelluksen parissa?
2. Mitä tapahtuu? Mitkä ovat olosuhteet, joissa käyttäjät parhaiten pystyvät hyödyntämään sovellusta?
3. Milloin he käyttävät sovellusta? Ovatko he kotona, jossa on enemmän aikaa käyttää sovellusta? Vai ovatko he töissä kiireen keskellä? Vai onko heillä ylimääräistä aikaa heidän odotellessaan junaa?
4. Missä käyttäjät ovat? Ovatko he julkisessa vai yksityisessä tilassa? Ovatko he sisällä vai ulkona? Onko päivä vai yö?
5. Miksi he käyttävät sovellusta? Mitä he hyötyvät sisällöstä tai palvelusta?
6. Miten he käyttävät mobiililaitetta? Onko se heidän taskussaan vai kädessään? Miten he pitävät sitä kädessään? Onko se pysty- vai vaaka-asennossa?

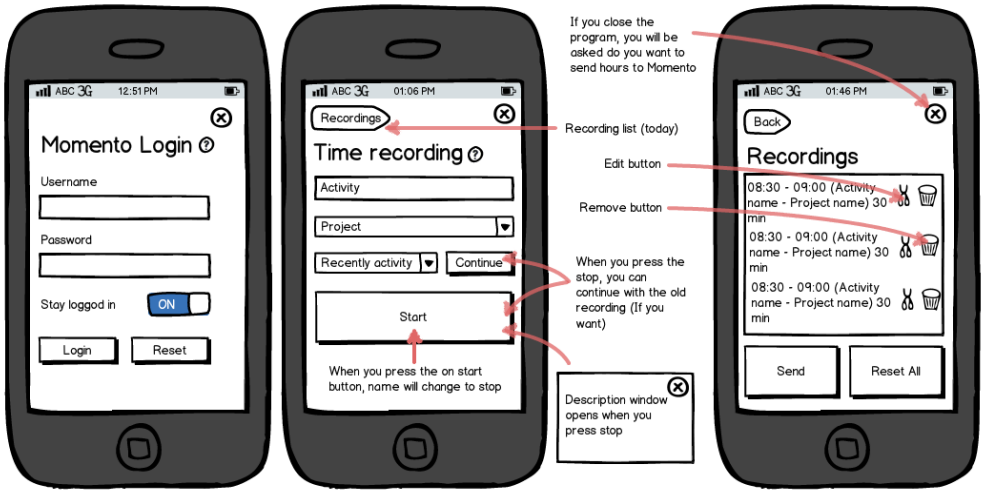
Vastaukset näihin kysymyksiin voivat tarjota käsityksen kehitettävästä käyttöliittymästä, antaa inspiraatioita ja auttaa perustelemaan, miksi jokin asia on tehtyä kyseisellä tavalla. Vastaukset edellä mainittuihin kysymyksiin löytyvät liitteestä 3.

Käytettävyyttä suunniteltaessa pitää myös miettiä seuraavaa kysymystä: mille laitteelle sovellusta ollaan kehittämässä. Tässä tapauksessa sovellusta ollaan kehittämässä mobiililaitteille, johon kuuluvat perinteisesti pienet näytöt verrattuna tietokoneiden näyttöihin. Tästä huolimatta käyttäjän tulisi pystyä toimimaan sivulla mahdollisimman tehokkaasti ja sujuvasti. Käyttöliittymä pitääkin optimoida sormella käytettäväksi, jolloin käyttöliittymäkomponenteista täytyy tehdä mobiililaitteelle tarkoitetussa sovelluksissa paljon suurempia ja visuaalisilta elementeiltään selkeämpiä ja yksinkertaisempia kuin tietokoneelle tarkoitetuissa sovelluksissa.

Prototyyppi

Joustavat prototyyppimallit auttavat hahmottaan sivun tarpeita ja ovat helposti mukautettavissa, jos esimerkiksi navigoinnin paikkaa halutaan muuttaa. Prototyyppi suunnitellaan jokaista edellä mainittua kohtaa hyväksi käyttäen. Prototyyppiä voi-

daan kutsua sivun ulkoasuksi, mutta tässä tapauksessa se tarkoittaa luonnostelmaa, jossa hahmotellaan karkeasti navigoinnin paikka, tekstikentät ja muut tärkeät elementit sovelluksen kannalta. Tarkat suunnittelumallit voivat näyttää paremmilta, mutta ne voivat olla heti ensimmäiseksi sovellukseksi brutaaleja toteuttaa. (Fling 2009, 110–121.) Kuviossa 7 on kuvattu sovelluksen ensimmäinen karkea malli siitä, miltä sovellus mahdollisesti tulee näyttämään ja mitä toimintoja siihen sisältyy. Prototyyppi on suunniteltu sen tiedon perusteella, joka tähän mennessä on saatu selvitettyä. Prototyyppi ei ota kantaa siihen, tullaanko sovellus toteuttamaan Android- vai PhoneGap-tekniikalla, koska sillä ei ole väliä tässä vaiheessa.



KUVIO 7. Sovelluksen prototyyppi

6 TEKNOLOGIOIDEN VERTAILU

Luvussa kuusi verrataan tarkemmin Android- ja PhoneGap-teknologioita ja kartoitetaan niiden ominaisuuksia. Pääasia on hahmottaa sovelluksen tarpeita luvun viisi ja siinä luodun liite 1:n ja 2:n mukaan, jossa käytiin sovelluksen määrittelyä analysoimalla mitä vaatimuksia käytännön toiminta asettaa sovellukselle. Yrityksen toiminta asettaa teknologian valinnalle ainakin seuraavat tärkeät ominaisuudet: testattavuuden, suorituskyvyn, jatkokehityksen, nykyaikaisuuden, laadun ja ylläpidon, laitteen

natiiviominaisuuksien hyödyntämisen (paikannus), debuggauksen, NFC-tuen, widgettien käytön sekä sen, pystytäänkö Maven-projektia hyödyntämään sovelluksessa.

6.1 Android vs. PhoneGap

Android

Natiiviohjelmien suurena etuna on pidetty parempaa suorituskykyä ja mahdollisuuksia mobiilialustan erikoisominaisuuksien hyödyntämiseen. Android-sovellukset ovat myös tietoturvallisia, koska jokainen .apk-tiedosto on allekirjoitettu, ja sovellukset suoritetaan turvallisessa sandboxes-ympäristössä. Natiiviohjelmien tekeminen erikseen jokaiselle alustalle on kuitenkin aikaa vievää ja erittäin kallista, ja iso osa ongelmista tuleekin silloin, kun sovellusta halutaan laajentaa muille alustoille. (Security Architecture 2012.) Androidin suosio on myös kehittäjien keskuudessa vähitellen tippumassa, sillä Appceleratorin teettämän neljännesvuosittaisen tutkimuksen mukaan Androidista ”hyvin kiinnostuneiden” kehittäjien määrä laski jo toisena neljänneksenä peräkkäin. Tällä kertaa androidin suosio laski 83,3 prosentista 78,6 prosenttiin. Appceleratorin mukaan Android-sovellusten kehittämisen suosion lasku johtuu käyttöjärjestelmän hajanaisuudesta. (Tässä on uusi merkki siitä, että Android on vaikeuksissa 2012.)

Tekniset ominaisuudet

Androidissa pystytään hyödyntämään tehokkaasti laitteen erikoisominaisuuksia ilman erillisiä lisäosia. PhoneGap tarvitsee NFC-tuen käyttöön lisäosan, mutta natiivikehityksessä se on valmiina saatavilla Javan luokkakirjastoista. Käyttäjän tarvitsee vain lisätä esimerkissä 9:n oleva ohjelmakoodi AndroidManifest-tiedostoon, mikä mahdollistaa NFC:n käytön natiivikehityksessä.

```
<!-- AndroidManifest-tiedosto -->
<uses-permission android:name="android.permission.NFC" />
<uses-feature android:name="android.hardware.nfc" android:required="true" />
<uses-sdk android:minSdkVersion="10"/>

<activity android:name=".CardActivity">
    <intent-filter>
        <action android:name="android.nfc.action.NDEF_DISCOVERED"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="text/plain" />
    </intent-filter>
</activity>
```

Esimerkki 9. NFC-lisäosan käyttöönotto (NFC Basics 2012)

Seuraava ominaisuus on Android Maven-plugin, jolla voidaan koostaa ja rakentaa sovelluksia. Koostamisella tarkoitetaan sovellusten kääntämistä ja paketoimista toimivaksi kokonaisuudeksi. Maven-työkalun lähtökohtana ovat erilaiset laajennukset, joiden avulla kaikki projektit rakennetaan standardoidulla tavalla tietyn elinkaaren mukaisesti. Sen avulla voidaan esimerkiksi kääntää, testata ja paketoida sovelluksen julkaistava versio yhdellä komentorivikomennolla. Mavenin paras puoli on sen tuki luokkakirjastoriippuvuuksien hallintaan (eng. depedency management), jolloin tarvittavia kirjastoja ei tarvitse ladata käsin verkosta, eikä niiden keskeisiä riippuvuuksia tarvitse selvittää. (Kuha 2008, 25.) Android Maven-projektiin tarvittavat pluginit ja riippuvuudet voi katsoa esimerkistä 10.

```
<!-- Android Maven-pluginin liittäminen projektin POM-tiedostoon. -->
<dependencies>
    <dependency>
        <groupId>com.google.android</groupId>
        <artifactId>android</artifactId>
        <version>2.3.3</version>
        <scope>provided</scope>
    </dependency>
</dependencies>

<!-- Aja buildaus seuraavalla komennolla. -->
mvn clean install
```

Esimerkki 10. Android Maven-pluginin käyttöönotto

Natiivikehityksessä on myös mahdollista luoda Androidille interaktiivisia widget-sovelluksia, joiden tehtävä on pääasiassa tiedon näyttäminen. Tyypillinen widget-sovellus näyttää esimerkiksi tietyn paikkakunnan säätiedot. Widgettejä kutsutaan myös työpöytäsovelluksiksi (eng. homescreen, launcher), koska termiä widget käyte-

tään myös käyttöliittymäkomponenteista, mikä voi aiheuttaa sekavuutta asiassa (Urpi 2012).

Android-natiivisovelluksen edut

- Natiivisti tuotettu koodi on suorituskykyinen suurissa laskentatehoa tai grafiikkaa vaativissa sovelluksissa.
- Java on saavuttanut suurta suosiota niin ohjelmistoteollisuudessa kuin olio-ohjelmoinnin opetuskielenäkin.
- Natiivisovellus voi hyödyntää laitteen erikoisominaisuuksia tehokkaasti.
- Android-natiivikehitykseen on saatavilla kattavat dokumentaatiot.
- Unified Remote -lähestymistapa sovelluskehityksessä
- Mahdollistaa sovelluksen kehityksen myös HTML5-, CSS3- ja JavaScript-kielellä.
- Google julkisti suurimman osan Androidin lähdekoodista avoimen ja vapaan ohjelmiston Apache-lisenssillä, joten kuka tahansa voi kehittää ja laajentaa sitä.
- Sovellus toimii varmasti sille tarkoitetulla laitteella.

Android-natiivisovelluksen heikkoudet

- Androidille tehtyjä sovelluksia ei voida helposti siirtää muille mobiilialustoille.
- Sovelluksen kehittäminen, testaaminen ja käyttöönotto ovat kallista ja aikaa vievää.
- Androidin suosio on laskussa alustojen hajanaisuuden vuoksi.
- Web-teknologioilla toteutetun sovelluksen voi kääntää ainoastaan Android-laitteelle natiivisovellukseksi.

PhoneGap

Selainpohjaisiin sovelluksiin liittyy merkittäviä hyötyjä verrattuna mobiililaitteelle asennettaviin sovelluksiin. Käyttäjän ei tarvitse asentaa web-sovellusta puhelimeen, mikä tekee sovelluksen kokeilemisesta helppoa ja vaivatonta – jos et pidä siitä, voit aina siirtyä seuraavalle sivulle jättämättä asennustiedostoja puhelimeen. Sovelluksia

on myös turvallista ajaa, koska ne suoritetaan Android-sovelluksista tutussa sandboxes-ympäristössä. Lisäksi sovelluksen päivittäminen käy helposti tallentamalla uusi versio palvelimelle, minkä jälkeen jokaisen käyttäjän sovellus päivittyy, kun he seuraavan kerran käyttävät sovellusta. (Kessin 2011, 3.) Web-sovellusten mielenkiinto on myös noussussa, sillä Appcelerator kysyi myös kehittäjiltä tutkimuksessaan, onko HTML5 osa heidän sovelluksiaan vuonna 2012 ja 79 prosenttia vastanneista oli sitä mieltä, että on. (Tässä on uusi merkki siitä, että Android on vaikeuksissa 2012.) PhoneGapin yksi tärkeimmistä ominaisuuksista on kuitenkin JavaScriptin tarjoama API -rajapinta laitteen natiiviominaisuuksiin. Kuviossa 8 on listattu rajapinnan tarjoamat natiiviominaisuudet laitekohtaisesti.

	iOS iPhone / iPhone 3G	iOS iPhone 3GS and newer	Android	OS 5.x	OS 6.0+	WebOS	WP7	Symbian	Bada
ACCELEROMETER	✓	✓	✓	✓	✓	✓	✓	✓	✓
CAMERA	✓	✓	✓	✓	✓	✓	✓	✓	✓
COMPASS	✗	✓	✓	✗	✗	✓	✓	✗	✓
CONTACTS	✓	✓	✓	✓	✓	✗	✓	✓	✓
FILE	✓	✓	✓	✓	✓	✗	✓	✗	✗
GEOLOCATION	✓	✓	✓	✓	✓	✓	✓	✓	✓
MEDIA	✓	✓	✓	✗	✗	✗	✓	✗	✗
NETWORK	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (ALERT)	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (SOUND)	✓	✓	✓	✓	✓	✓	✓	✓	✓
NOTIFICATION (VIBRATION)	✓	✓	✓	✓	✓	✓	✓	✓	✓
STORAGE	✓	✓	✓	✓	✓	✓	✓	✓	✗

KUVIO 8. Mobiililaitteiden tuetut ominaisuudet (Supported Features n.d.)

Kuviosta 8 voi huomata, että nykyään lähes kaikkien alustojen web-sovellusten käytettävissä on suurin osa samoista ominaisuuksista kuin natiivisti tehdyillä sovelluksilla, ja puuttuvia ominaisuuksia kehitetään jatkuvasti lisää. Täyden natiivituen tarjoavat kuitenkin Android-, iOS- ja Windows Phone 7 -laite. Ajantasaisen listan tuetuista ominaisuuksista näet osoitteesta: http://phonegap.com/about/feature.

Tekniset ominaisuudet

Yksi PhoneGapin parhaimmista asioista ovat sen tarjoamat lisäosat. Yksi PhoneGapin tarjoamista lisäosista on NFC-tuki. NFC-teknologia mahdollistaa esimerkiksi tilanteen, jossa puhelimen voi työpaikalle mennessä asettaa NFC-tarran päälle, ja laite täydentää automaattisesti työpaikan tiedot ja työtehtävät tuntijärjestelmään. NFC-lisäosan lisääminen on helppoa jo valmiina olevaan projektiin. Käyttäjän tarvitsee lisätä aluksi osoitteesta: http://phonegap.com/2011/09/26/building-an-nfc-enabled-android-application-with-phonegap/ löytyvät phonegap-nfc.js-tiedosto ja phonegap-nfc.jar-paketti työn hakemistorakenteeseen ja tämän jälkeen lisätä esimerkissä 11 olevat ohjelmakoodit xml-, html- ja AndroidManifest-tiedostoon, jotka mahdollistavat NFC:n käytön työssä.

```
<!-- plugins.xml-tiedosto -->
<code><plugin name="NfcPlugin" value="com.chariotsolutions.nfc.plugin.NfcPlugin"/>
</code>

<!-- html-tiedosto -->
<code><script type="text/javascript" charset="utf-8" src="phonegap-nfc.js"></script>
</code>

<!-- AndroidManifest-tiedosto -->
<code><uses-permission android:name="android.permission.NFC" />
<uses-feature android:name="android.hardware.nfc" android:required="true" />
<uses-sdk android:minSdkVersion="10" />
</code>
```

Esimerkki 11. NFC-lisäosan käyttöönotto (Add phonegap-nfc to your project 2012)

Androidin teknisissä ominaisuuksissa mainittiin Maven-plugin, jolla voitiin koostaa ja rakentaa sovelluksia. PhoneGap-sovelluskehys mahdollistaa Build-Maven-pluginin käytön projekteissa samalla periaatteella kuin Androidissa, mutta erona Androidiin PhoneGap Build-Maven-plugin vaatii toimiakseen Adobe-käyttäjätunnukset projektin kääntämisvaiheessa. PhoneGap Maven-projektin kääntämiseen tarvittavat pluginit ja riippuvuudet voi katsoa liitteestä 4 sen suuren koon vuoksi.

PhoneGap tarjoaa lisäksi mahdollisuuden kehittää sovellusta käyttämällä apuna GWT-PhoneGap-luokkakirjastoriippuvuutta. GWT:n avulla sovelluskehittäjät voivat ohjelmoida JavaScript-sovelluksia halutessaan Java-kielellä, ja käyttää apuna Javan omia kehitystyökaluja. GWT kääntää sovellusta ajettaessa Java-tiedostot JavaScript-

tiedostoiksi, jotka ovat syvästi sotkettuja ja optimoituja. (Kurka 2012.) PhoneGap GWT-luokkakirjastoriippuvuuteen tarvittavat pluginit ja riippuvuudet voi katsoa esimerkistä 12.

```
<dependency>
      <groupId>com.googlecode.gwtphonegap</groupId>
      <artifactId>gwtphonegap</artifactId>
      <version>2.0.0.0</version>
</dependency>
```

Esimerkki 12. PhoneGap GWT-luokkakirjastoriippuvuus

PhoneGapiin on lisäksi saatavilla myös laaja valikoima erilaisia käyttöjä helpottavia työkaluja, jotka ovat nähtävissä osoitteesta: http://phonegap.com/tool.

PhoneGap-web-sovelluksen edut

- PhoneGap tukee seitsemän eri laitevalmistajan tuotetta.
- PhoneGap on avoimen lähdekoodin sovelluskehys, ja siksi siihen luodaan jatkuvasti uusia parannuksia, lisäosia ja moduuleita.
- Voidaan käyttää apuna jQuery Mobile UI-sovelluskehystä, minkä ansiosta sovellukset tarjoavat visuaalisesti paremman käyttäjäkokemuksen.
- PhoneGap pystyy hyödyntämään laitteen erikoisominaisuuksia API -rajapinnan avulla.
- PhoneGap sovelluskehitykseen on saatavilla kattavat dokumentaatiot.
- Web-teknologiat ovat eniten standardoituja.
- PhoneGap mahdollistaa GWT:n käytön.
- Eri laitevalmistajat hyväksyvät PhoneGapilla tehdyt sovellukset myyntiin sovelluskauppoihin.
- Sovelluksen nopea valmistus, testaus ja käyttöönotto
- Valtaosa uusista digitaalisista innovaatioista toteutetaan nykypäivänä käyttämällä web-tekniikoita.
- Selainpohjaisten sovellusten julkaisemisesta ei tarvitse maksaa kuluja, koska näitä sovelluksia varten tarvitaan vain palvelin, jonne sovelluksen voi tallentaa.

PhoneGap-web-sovelluksen heikkoudet

- Eri valmistajien puhelinmallit eivät tue kaikkia rajapinnan tarjoamia natiiviominaisuuksia (kuvio 8).
- Ei vielä saatavilla suoraa widget-tukea.
- Web-teknologioilla tuotettu koodi ei ole suorituskyvyltään yhtä tehokasta kuin natiivisti tuotettu koodi, jos se vaatii sovellukselta suurta laskentatehoa tai sisältää paljon grafiikkaa, kuten kuvia.
- Vaikka PhoneGap tarjoaa laajan valikoiman lisäosia ja moduuleita, niin kaikkea ei silti ole vielä saatavilla, ja käyttäjän täytyy rakentaa tarvittavat asiat itse.
- Eri valmistajien mobiililaitteet eivät ole identtisiä, joten sovelluksen toiminnassa voi olla pieniä eroja havaittavissa eri mobiililaitteilla.

HTML5 ja Facebook

Pienemmät yritykset rinnastavat usein teknologian valintaa siihen, mitä kokemuksia isoilla yrityksillä on kyseisen teknologian käytöstä. Codecenter halusi tietää, miksi Internetissä toimiva yhteisöpalvelu Facebook on vaihtanut HTML5-mobiilisovelluksista takaisin natiiveihin iOS- ja Android-sovelluksiin. Zuckerberg (2012) on sanonut seuraavasti:

> *"When I'm introspective about the last few years, I think the biggest mistake that we made as a company is betting too much on HTML5 as opposed to native. Because it just wasn't there. It's not that HTML5 is bad. I'm actually, on long-term, really excited about it. One the thing that's interesting is we actually have more people on a daily basis using Facebook on the mobile Web than we have using our iOS or Android apps combined."*

Tämä ei kuitenkaan kerro syytä siihen, mikä teki HTML5:stä huonon vaihtoehdon. Facebookin edustuja Simon Cross tuli vastaamaan näihin kysymyksiin PhoneGap-päivässä vuonna 2012. Cross (2012) sanoo, että Facebook on ideana yksinkertainen kenttä tavaraa, mutta samalla se on valtava, massiivinen, täynnä kuvia, raskas ja sitä pitää rullata ylös ja alas, jolloin sovelluksen pitää ladata tekstiä ja kuvia todella paljon – nämä asiat tekivät siitä hitaan käyttää. Natiivisti toteutetut iOS- ja Android-

sovellukset ovat kaksi kertaa nopeampia kuin HTML5-sovellus, ja lisäksi tekniikan vaihdon jälkeen niiden luokitus sovelluskaupoissa nousi kahdesta tähdestä neljään tähteen. Zuckerberg ei kuitenkaan sano, että HTML5 olisi huono asia, vaan että se ei ole **vielä** täällä.

6.2 Tekniset edellytykset

Tässä luvussa testataan web-tekniikoilla toteutetun sovelluksen toimivuutta eri mobiililaitteiden selaimissa, koska sovelluksen toiminta on sidottu käytettävään selaimeen, ja niillä voi olla hyvin vaihteleva tuki eri standardien ominaisuuksille. (Vuorela 2011, 13.) Testien avulla pyritään saamaan tieto eri alustojen tukemista ominaisuuksista ja suorituskyvyistä. Testeissä käytetään kahta selaimessa toimivaa testiohjelmaa: Acid3- ja HTML5-testi. Testattavat mobiililaitteet ja selaimet on valittu yrityksen toiveiden ja sen tarjoamien testilaitteiden mukaan. Taulukossa 1 on kuvattu testilaitteet ja niissä käytettävät selaimet.

TAULUKKO 1. Testattavat mobiilialustat

Testilaite	Testattava selain
Samsung Galaxy S III	Android 4.0.4
Samsung Galaxy S III	Chrome 18
Apple iPhone 5	iOS 6.0
Google Nexus S	Android 2.3.3
Nokia Lumia 800	Windows Phone 7.5

Acid3-testi

Acid3 on www-selainten renderointitesti, jonka on suunnitellut ja toteuttanut WaSP. Testillä voidaan testata tämänhetkisiä sekä tulevaisuuden www-standardeja. Graafisen tuloksen lisäksi testi antaa numeerisen tuloksen selaimen tuesta. (WaSP: Fighting for Standards n.d.)

Testillä testataan selaimen yhteensopivuutta seuraavilla osa-alueilla (Acid3 Browser test n.d.):

- DOM2: Core, Events, HTML, Range, Style, Traversal and Views
- ECMAScript
- HTML4 (<object>, <iframe>, …)
- HTTP (Content-Type, 404, …)
- Media Queries
- Selectors (:lang, :nth-child(), combinators, dynamic changes, …)
- XHTML 1.0
- CSS2 (@font-face)
- CSS2.1 (‘inline-block’, ‘pre-wrap’, parsing, …)
- CSS3 Color (rgba(), hsla(), …)
- CSS3 UI (‘cursor’)
- data: URIs
- SVG (SVG Animation, SVG Fonts, …)

Acid3-testistä saadut alustakohtaiset tulokset on esitelty taulukossa 2.

TAULUKKO 2. Acid3-testin tulokset

Testilaite	Selain	Pisteet
Samsung Galaxy S III	Android 4.0.4	100 / 100
Samsung Galaxy S III	Chrome 18	100 / 100
Apple iPhone 5	iOS 6.0	100 / 100
Google Nexus S	Android 2.3.3	95 / 100
Nokia Lumia 800	Windows Phone 7.5	95 / 100

Acid3-testin tulosten perusteella voidaan sanoa, että www-standardien tuki uusissa mobiililaitteissa on kiitettävää tasoa, ja lähes kaikki testatuista mobiilialustoista suoriutuivatkin www-selainten renderointitestistä täysillä pisteillä. Tämä tarkoittaa sitä, että esimerkiksi kulmien pyöristysten näyttäminen uusilla ja vähän vanhemmilla mobiililaitteilla ei tuota ongelmia.

HTML5-tuen testaus

HTML5-testi näyttää luvun, joka kuvaa tuen laajuutta ja esittää analyysin eri piirteiden tuesta. Sivusto testaa kaikki HTML5-standardiin ja siihen liittyviin vaatimuksiin kuuluvat osa-alueet sekä lisäksi joukon muita HTML5-termiin liittyviä standardeja. Testissä annetaan pisteitä eri osakokonaisuuksista sekä niihin kuuluvien alatestien mukaan. Lisäksi annetaan bonuspisteitä standardeihin kuulumattomista, mutta käytännön toimivuuteen vaikuttavista osa-alueista, kuten videoformaattien tuesta. (Vuorela 2011, 13.)

HTML5-testistä saadut alustakohtaiset tulokset on esitelty taulukossa 3.

TAULUKKO 3. HTML5-testin tulokset

Ajettava testi	Android 4.0.4	Chrome 18	iOS 6.0	Android 2.3.3	Windows Phone 7.5	Maksimi
Parsing rules	10	10	10	0	0	11
Canvas	20	20	20	20	20	20
Video	21	21	21	21	21	31
Audio	20	20	20	20	20	20
Elements	29	30	29	15	15	30
Forms	106	105	102	38	7	108
User interaction	20	20	20	0	18	37
History & navigation	10	10	10	10	0	5
Microdata	15	0	0	0	0	15
Web applications	15	18	15	19	1	20
Security	10	10	15	10	0	15
Various	5	5	5	2	1	6
Location & Orientation	20	20	20	15	15	20
WebGL	25	10	10	0	0	25
Communication	31	33	33	10	5	37
Files	10	10	10	0	0	20
Storage	25	25	15	15	10	20
Workers	0	10	15	0	0	15
Local multimedia	0	0	0	0	0	20
Notifications	10	0	0	0	0	10
Other	7	10	8	5	5	7
Audio	0	0	5	0	0	4
Video & Animation	3	3	3	0	0	4
Bonus points	3	11	9	1	5	15
Yhteensä	415	390	386	200	138	515

HTML5-testin perusteella voidaan sanoa, että eri ominaisuuksien tuki vaihtelee paljon, mutta kokonaisuudessaan hajonta oli hyvin pientä uusien alustojen kanssa. Vanhemmissa alustoissa HTML5-tuki laski selvästi. Parhaiten testissä menestyi Android 4.0.4 saavuttaen testistä 415 pistettä.

Tulosten yhteenvetona voidaan sanoa, että web-teknologioiden toimivuus uusilla mobiilialustoilla on varsin hyvä ja että ne ovat nykyään kykeneviä haastamaan natiiviteknologiat monissa tapauksissa. Paras tuki saavutetaan Android-alustalla, mutta ennen HTML5-ominaisuuksien hyödyntämistä on syytä tarkistaa halutun ominaisuuden tuki kehitettävälle alustalle. HTML5-testin sivulle on koottu monipuolisesti yhteen eri laitteiden ja alustojen menestyminen HTML5-testissä. Sivulla on pistetaulukot seuraaville laitteille: tableteille, tietokoneille, pelikonsoleille, televisioille ja mobiililaitteille. Mobiililaitteiden pistetaulukkoon pääset osoitteesta: http://html5test.com/results/mobile.html.

6.3 Testaus

Sovelluksen toimivuus voidaan todeta testaamalla järjestelmän toiminnot mahdollisimman aikaisessa kehitysvaiheessa, mutta tyypillisesti testaus suoritetaan, kun vaatimukset ovat määritelty ja ohjelmointivaihe suoritettu. Testauksen tarkoitus on löytää ennalta havaitsemattomia ohjelmointivirheitä (eng. error, mistake, bug), jotka voivat vaihdella pienistä käyttäjää ärsyttävistä yksityiskohdista aina järjestelmän kaatumiseen asti. Testaamisella voidaan myös osoittaa, että ohjelmisto täyttää sille asetetut tekniset ja kaupalliset vaatimukset, ohjelmisto toimii sille tarkoitetulla tavalla ja ohjelmisto voidaan implementoida halutuilla ominaisuuksilla. (Kolehmainen 2009, 15.)

Java- ja web-sovelluksen testauksessa on merkittäviä eroja. Javaa kirjoitettaessa pääasiallinen tehtävä on testata Java-koodia, ja JavaScriptiä kirjoittaessa taas tärkein tehtävä on testata JavaScriptiä. HTML5- ja CSS3-virheitä varten ei tarvitse tehdä erillisiä testitapauksia, vaan tällöin riittää sivujen validointi. Tällä tarkoitetaan HTML5-koodin ja CSS3-tyylien oikeellisuuden tarkistamista W3C-standardin mukaisesti. Sivu-

jen validointi auttaa merkittävästi parantamaan sivujen laatua, ja se tuo kustannussäästöjä, koska standardeja noudattavat sivut toimivat parhaiten eri selaimilla – kunhan selain tukee ja näyttää standardeja oikein. Sivujen validoinnin voi suorittaa kehitysympäristöön asennettavilla apuvälineillä tai käyttäen Internetissä toimivaa W3C Markup Validation -palvelua. (About The W3C Markup Validation Service n.d.)

Yksikkötestit

Javan yksikkötestit kirjoitetaan tyypillisesti käyttäen JUnit-kirjastoa, ja JavaScriptin yksikkötestit kirjoittaa käyttäen QUnit-kirjastoa. Näiden molempien testaustapojen tehtävä on paljastaa vikoja yksittäisissä komponenteissa eli funktioissa, metodeissa tai esimerkiksi luokissa. Vaikka JUnit- ja QUnit-testit kirjoitetaan toisistaan poikkeavilla ohjelmointikielillä, niin niiden toimintaperiaate on silti sama. Molempiin kirjastoihin on saatavissa myös Maven-riippuvuus.

Testaustapa

Tämän tuntikirjanpitosovelluksen testauksessa käytetään white box -testaamista, jossa testaajalla on pääsy sovelluksen tietorakenteisiin sekä työn koodiin. Työn järjestelmätestaus suoritetaan käyttäen apuna Android-emulaattoria ja Samsung Galaxy S3 -puhelinta, jotta voidaan varmistaa, että sovelluksen rakenne ja elementit ovat sellaisia kuin niiden pitäisi olla ulkoasultaan ja toiminnallisuuksiltaan. Sovelluksen testausta suoritetaan koko ohjelmointivaiheen aikana, ja apuna testauksessa voidaan käyttää Seleniumia. Selenium mahdollistaa testauksen automatisoinnin. Yksi vaihtoehtoinen tapa testata sovellusta ilman Android-emulaattoria on The Ripple Mobile Enviroment Emulator (RMEE), joka mahdollistaa sovelluksen testauksen selaimen avulla.

Debuggaus

Debuggauksen tarkoitus on löytää todellinen syy virheen aiheuttajalle. JUnit-testien ajaminen on saumatonta IDE:n toiminnallisuuksien kanssa, ja se ei vaadi erityisiä toimenpiteitä, jolloin se mahdollistaa koodin helpon debuggauksen. Tämä ominaisuus on yksi Javan ehdottomista vahvuuksista. (Kolehmainen 2009, 22–47.) JavaSc-

riptissä IDE ei tue suoraan JavaScriptin debuggausta, ja siksi siinä pitää käyttää kolmannen osapuolen debuggaus-työkaluja. Yksi näistä ”työkaluista” on Google Chromen JavaScript-konsoli, jonka avulla JavaScriptiä voidaan debugata samalla periaatteella kuin Javaa IDE:n sisällä.

Android-emulaattori ei kuitenkaan mahdollista ohjelmakoodin suoraa debuggausta omassa selaimessaan. Tämän ongelman ratkaisemiseksi on kehitetty http://debug.phonegap.com/-palvelu, joka antaa yhden rivin pituisen JavaScript-koodin käyttäjän omalla tunnisteella. JavaScript-koodi voidaan laittaa kehitettävään sovellukseen, minkä jälkeen käyttäjä voi mennä osoitteeseen http://debug.phonegap.com/client/#omaTunniste, jossa voidaan nyt debugata suoritettavaa sovellusta pöytäkoneen selaimen avulla. (Wargo 2012, 46–49.)

PhoneGappiin on olemassa myös Weinre-niminen työkalu, joka mahdollistaa sovelluksen suorittamisen fyysisessä mobiililaitteessa samalla kun ohjelmakoodi näkyy käyttäjän tietokoneen selaimessa. Näin käyttäjä voi suorittaa koodin debuggauksen tietokoneen selaimessa ja testata samalla ohjelman toimivuutta fyysisessä mobiililaitteessa. (Wargo 2012, 46–49.)

6.4 Jatkokehitysmahdollisuudet

> *”Yksi suurimmista virheistä mobiilisovelluksia suunnitellessa on halu tai tunne olla ajattelematta sovellusta pidemmällä aikavälillä.”* Brian Fling

Mobiilisovellusta pitää tarkastella pidemmällä aikavälillä, koska mobiilisovellukset eivät ole kertaluonteisia, vaan niillä voi olla monen vuoden käyttöaika. Pitkä käyttöaika tarkoittaa kuitenkin jatkuvaa ylläpitoa tekniikan ja vaatimusten kehittyessä. Jatkokehitys tarkoittaa muun muassa nykyisen järjestelmän kehittämistä paremmaksi, yhteensopivammaksi ja tehokkaammaksi.

Sovelluksen jatkokehitys asettaa tiettyjä teknisiä vaatimuksia rajapinnoissa ja koodin luettavuudessa. Sovellukseen on tarkoitus kehittää tulevaisuudessa paikannukseen perustuva automaattinen työpaikkahaku ja NFC-lukijaan perustuva työpaikan tieto-

jen automaattinen täyttötoiminto. Tämä tarkoittaa teknisesti pääsyä laitteen natiiviominaisuuksiin, joka onnistuu Android- ja PhoneGap-tekniikalla. Yksi jatkokehityksen kannalta oleellinen asia on miettiä, kumpi ohjelmakoodi on ohjelmoijan näkökulmasta ylläpidettävämpi: Java vai HTML5-, JavaScript- ja CSS3-yhdistelmä. Vastaus ei ole helppo, koska ohjelmoijan tottumukset, taustat ja tavat vaikuttavat suoraan kysymykseen. Kokenut Java-ohjelmoija voi tuomita HTML5-, JavaScript- ja CSS3-yhdistelmän sekavaksi kokonaisuudeksi, kun taas web-ohjelmoija voi sanoa, että Java on liian vaikea kieli ymmärtää.

Ohjelmointikielen valinnasta riippumatta ohjelmointikäytäntöjen avulla voidaan kompensoida ohjelmointikielessä esiintyviä puutteita ja suojautua mahdollisia tietoturvauhkia vastaan. Ohjelmointikäytänteillä tarkoitetaan tässä tapauksessa tiettyjen asioiden sopimista etukäteen. Säännöillä voidaan esimerkiksi rajoittaa muuttujien käyttöä tai käyttää tiettyä sisennystapaa koko projektin ajan – nämä asiat helpottavat koodin lukemista, ylläpitoa ja virheiden löytämistä.

Jatkokehityksen yhteenvetona voidaan todeta, että kielestä tai alustasta riippumatta molemmat teknologiat soveltuvat tasavertaisesti tämän sovelluksen jatkokehitykseen. Lisäksi uusiin teknologioihin tutustumisen ja opettelun pitäisi olla arkipäivää jokaiselle ohjelmoijalle, eivätkä Android- ja PhoneGap-alustat eroa tässä suhteessa toisistaan mitenkään.

7 SOVELLUKSEN TOTEUTUS

7.1 Teknologian valinta

Ennen sovelluksen toteutusta on syytä vastata tämän työn tutkimuskysymykseen: Kumpi teknologia on työtehtävien kirjaamiseen mobiililaitteella parempi ratkaisu: Android-natiivikehitys vai alustariippumaton PhoneGap.

Teknologian valinnan kannalta tärkeitä ominaisuuksia (testaus, suorituskyky jatkokehitys, nykyaikaisuus, laatu ja ylläpito, natiiviominaisuuksien hyödyntäminen, debuggaus, NFC-tuki ja Maven-projektin hyödyntäminen) verrataan Android-natiivikehitykseen ja PhoneGap-sovelluskehitykseen. Vertailun perusteella saadaan selvitettyä, kumpi teknologia vastaa paremmin asiakkaan asettamia vaatimuksia.

Työn tarkoitus on rakentaa yksinkertainen työtehtävien tuntikirjaamiseen tarkoitettu sovellus, joka luottaa yksinkertaiseen visuaaliseen ilmeeseen eikä vaadi suurta laskentatehoa sovellukselta. Sovellus vaatii yksinkertaistettuna vain käyttäjän sisäänkirjauksen ja erilaisia tekstikenttiä tuntien ja tehtävien syöttämiseksi. Yksinkertaisuus määrää tässä tapauksessa sovelluksen menestyksen, koska käyttäjiä ei kiinnosta teknologia, jolla sovellus on tehty, vaan se, kuinka sujuvasti he käyttävät sitä. Parempi teknologia työn toteuttamisen kannalta onkin **PhoneGap**. Alapuolelle on kerätty tarkempi lista perusteluista, miksi PhoneGap on parempi ratkaisu **tämän** sovelluksen toteuttamiseksi.

Perustelut teknologian valinnalle

Sovelluksen toteuttaminen natiivisti Androidille tuntuisi tässä tapauksessa kuin ampuisi muurahaista tykillä sen pienen koon vuoksi. Yläpuolella olevasta kuvauksesta käy ilmi, että tehtävä sovellus on hyvin pieni kokonaisuus, kun taas natiivikehitys on tarkoitettu graafisesti raskaisiin ja suurta laskentatehoa vaativiin sovelluksiin. Lisäksi natiiviohjelmoinnilla toteutettu sovellus olisi sidottu vain yhden valmistajan laitteeseen, mikä rajoittaisi sovelluksen käytön vain pienelle asiakaskunnalle. Myöhemmässä vaiheessa sovelluksen laajentaminen muille alustoille tarkoittaisi automaattisesti suuria kustannuseriä.

PhoneGap taas on täydellinen ratkaisu sovelluksille, jotka ovat yksinkertaisia luonteeltaan, eivätkä vaadi monimutkaisia toiminnallisuuksia sovellukselta, jolloin sovelluksen suorituskyky pysyy myös kiitettävällä tasolla. PhoneGapilla tehty sovellus palvelee monipuolisesti eri valmistajien mobiilialustoja ja laajempaa asiakaskuntaa. Sovellusta on myös tarkoitus jatkokehittää, jolloin selainpohjaisen sovelluksen päivit-

täminen käy helposti tallentamalla uusi versio palvelimelle, jolloin jokaisen käyttäjien sovellus päivittyy, kun he seuraavan kerran käyttävät sovellusta.

Lista perusteluista, miksi PhoneGap on parempi ratkaisu tuntikirjapitosovelluksen toteuttamiseksi mobiililaitteelle.

- PhoneGap vaikuttaa teoreettisella tasolla teknisesti nykyaikaiselta ratkaisulta.
- PhoneGap sai paremmat pisteet liitteessä 2, joka antaa myös kuva PhoneGapin paremmuudesta.
- PhoneGapilla toteutettuja hybridisovelluksia ei erota natiivisti toteutetuista sovelluksista.
- HTML5 mahdollistaa myös sovelluksen offline-käytön.
- PhoneGap ratkaisee yrityksen nykyisen ongelman, joka on sovellusten laajennettavuus muille alustoille.
- PhoneGap tuo uuden tekniikan yritykseen, jossa on jo olemassa oleva taito luoda Android-natiivisovelluksia.
- PhoneGapin JavaScript API -rajapinta tukee kaikkia Android-laitteen natiiviominaisuuksia.
- Acid3- ja HTML5-testi osoittavat, että nykyaikaiset älypuhelimet ovat valmiita web-teknologoilla toteutettuun sovellukseen.
- PhoneGap mahdollistaa laajemman asiakaskunnan.
- Muutamassa minuutissa sovellukselle saadaan luotua helppokäyttöinen käyttöliittymä jQuery Mobilen avulla.
- Vaikka on totta, että JavaScript on vaikeaselkoista, voidaan se puute tai vika korjata jQueryn tai GWT:n avulla.
- HTML5, JavaScript ja CSS3 tarkoittaa nopeampaa ja helpompaa kehitysprosessia, joka tarkoittaa taloudellisempaa ratkaisua.
- PhoneGapin ominaisuudet riittävät testausta, jatkokehitystä ja debuggausta ajatellen niin teknisesti kuin taloudellisestikin.
- Monipuoliset lisäosat mahdollistavat NFC:n ja Mavenin käytön työssä.

Tarkimmat lukijat huomasivat varmasti, että PhoneGapissa ei ole widget-tukea. Tämä ominaisuus vaikutti työn alussa tärkeältä, mutta työn edetessä on tullut selville, että se ei ole tämän sovelluksen kannalta tärkeä ominaisuus. Käyttäjän tarkoitus on tehdä vain työtehtävien kirjauksia järjestelmään ja järjestelmä ei anna tässä tapauksessa takaisin oleellista interaktiivista sisältöä työpöytäsovellukseen.

Lopuksi voidaan sanoa, että natiivisovellukset eivät palvele käyttäjää millään tavoin merkittävämmin kuin web-sovellukset. Ne vain lisäävät projektille hintaa, vaikeuttavat jakelua, ja niillä voidaan menettää runsaasti taloudellisia etuja. (Fling 2009, 150.)

7.2 Tarvittavat kehitystyökalut

Ensimmäinen askel PhoneGap-sovelluksen luomisessa on asentaa kehitysympäristö kuntoon. PhoneGap-sovelluksen luomisessa tarvitset ainakin seuraavia asioita: Android SDK, Java Development Kit (JDK), PhoneGap SDK, ja ellei ole masokisti, on syytä asentaa integroitu ohjelmointiympäristö (IDE) työssä tarvittaville ohjelmointikielille – Eclipse on erityisen hyvin tuettu ja suunniteltu juuri Android-emulaattorin käyttöön. Android SDK, JDK, PhoneGap SDK ja Eclipse ovat myös kaikki saatavilla Windows-, Mac OS- ja Linux-käyttöjärjestelmille, joten sovelluksia voidaan kehittää itselle sopivassa käyttöjärjestelmässä. (Ghatol & Patel 2012, 20–31; Meier 2012, 20–28.)

Koska sovellus on visuaalinen kokonaisuus, sen menestymisen kannalta on myös suotavaa suunnitella tarkempi käyttöliittymä prototyypin pohjalta. Tähän tarkoitukseen esimerkiksi Photoshop CS5 on hyvä apuväline, koska sen avulla värit ja tarvittavat elementit ovat helposti siirrettävissä ja muokattavissa käyttöliittymän suunnitteluvaiheessa. (Ghatol & Patel 2012, 20–31; Meier 2012, 20–28.)

Adobe Photoshop CS5

Adobe Photoshop CS5 on Adobe Systemsin kehittämä ammattitason kuvankäsittelyohjelma, joka on saavuttanut markkinajohtajuuden kaupallisessa digitaalisten kuvien muokkauksessa. Sovellus on saavuttanut niin suuren maineen, että sen nimestä on kehitetty termi ”photoshoppaaminen”, joka tarkoittaa yleisesti kuvan muokkausta

riippumatta käytetystä ohjelmasta. Vaikka Photoshop on ensisijaisesti suunniteltu muokkaamaan kuvia painotuotantoon, sitä käytetään nykyään myös paljon kuvien tekemiseen ja Internet-sivujen ja sovellusten ulkoasujen suunnitteluun. Photoshop sisältää yli 500 komentoa, ja sen tarjoamat mahdollisuudet sovelluksen ulkoasun suunnitteluun ovatkin käytännössä rajattomat. Photoshop on saatavilla Windows-, Mac OS- ja Linux-käyttöjärjestelmille. (Snider 2010, 17–18.)

Java Development Kit

Java Development Kit on Oracle Corporationin tuote, joka on suunniteltu Java-kehittäjille. JDK sisältää Java Runtime Enviromentin (JRE), joka tarvitaan käännettyjen ohjelmien ajamiseen (Java-kääntäjä), ja se sisältää myös moduulit niiden sovellusten suorittamiseen, jotka käyttävät tätä apuohjelmaa toimiakseen. Tähän ohjelmistoon kuuluvat Java-virtuaalikone, aputiedostot ja Java-lisäohjelmat. JDK on ladattavissa osoitteesta: http://java.sun.com/javase/downloads/index.jsp. (Mednieks, Dornin, Meike & Nakamura 2011, 4–5.)

Eclipse (IDE)

Eclipse on avoimen lähdekoodin ohjelmointiympäristö, joka tukee monipuolisesta erilaisia ohjelmointikieliä. IBM alkoi kehittää Eclipseä vuonna 1993, ja vuonna 2001 se julkaistiin avoimen lähdekoodin lisenssillä. Eclipse on toteutettu pääasiassa Javalla, mutta se ei kuitenkaan käytä Javan käyttöliittymäkirjastoa, vaan sille on toteutettu oma kirjasto. Tämän takia Eclipse ei ole täysin alustariippumaton. Eclipseä käytetään yleensä Java-ohjelmien kirjoittamiseen ja testaamiseen, mutta se soveltuu myös laajasti muihin ohjelmointikieliin. Eclipse on ladattavissa osoitteesta: http://www.eclipse.org/downloads/. (Mednieks ym. 2011, 5–6.)

Android Development Tools (ADT)

Android-sovellusten tekemistä varten täytyy Eclipseen ladata Android-lisäosa (ADT) käyttäen apuna plug-in-mekanismia. Plug-in-mekanismi on kevyt ohjelma Eclipsen sisällä, joka mahdollistaa lisäosien hakemisen. Lisäosan saa valitsemalla Eclipsestä Help → Install New Software ja lisäämällä osoitteen: https://dl-

ssl.google.com/android/eclipse/ sille tarkoitettuun kenttään ja tämän jälkeen valitsemalla ja asentamalla ohjelman ehdottamat Android-laajennukset.

Android SDK

Android Software Development Kit (SDK) on kokoelma tiedostoja, johon kuuluvat: API-kirjastot, ohjelmatiedostot, malliesimerkit, skriptit, dokumentaatiot ja kehitystyökalut, joiden avulla voidaan rakentaa ja testata sovelluksia. Kehitystyökaluihin kuuluu esimerkiksi Android-emulaattori, joka mahdollistaa sovellusten ajamisen ja testaamisen halutulla Android-käyttöjärjestelmän versiolla (API Level). Android SDK:n voi ladata osoitteesta: http://developer.android.com/sdk/index.html. (Get the Android SDK 2012.)

PhoneGap SDK

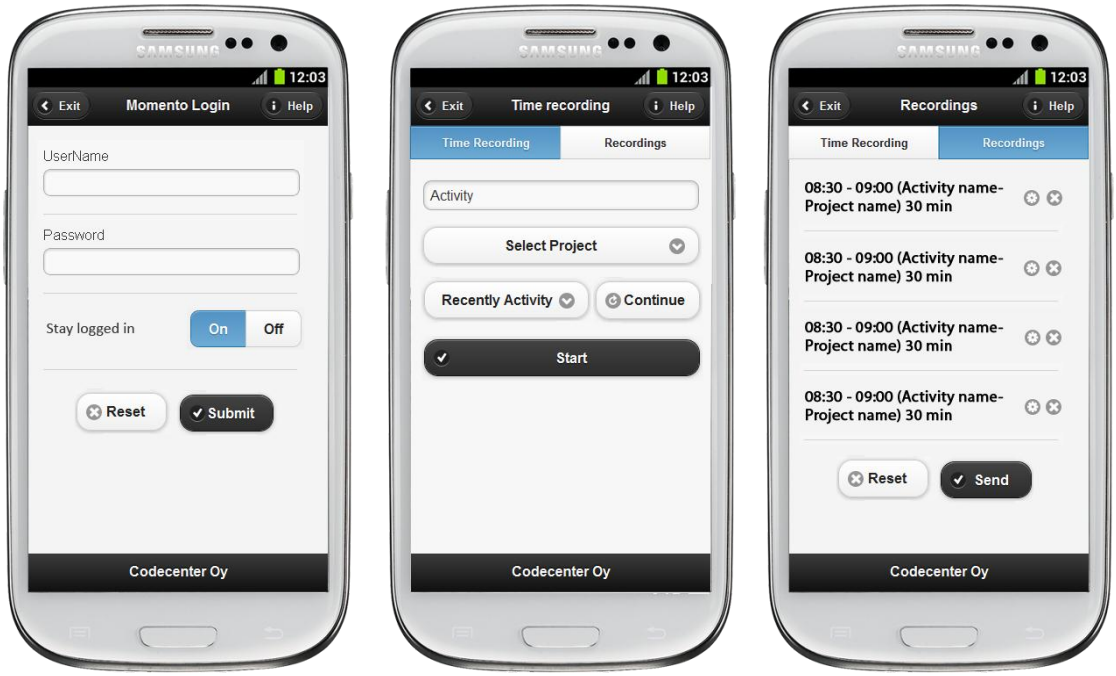
PhoneGap SDK:ta (Cordova) voi ajatella moottorina, joka antaa virtaa PhoneGapille samaan tapaan, kuin WebKit-moottori antaa virtaa Google Chrome- ja Apple Safari -selaimelle. PhoneGap SDK pitää sisällään hyvin samankaltaisia asioita, kuin Android SDK. Esimerkiksi PhoneGap SDK:n mukana tulevat dokumentaatiot, malliesimerkit ja kirjastot sovelluksen kehittämistä varten. PhoneGap SDK:n voi ladata osoitteesta: http://phonegap.com/download. (PhoneGap, Cordova, and what's in a name? 2012.)

7.3 Sovelluksen ulkoasu

Sovelluksen ulkoasun Photoshop-versio on suunniteltu lukua viisi ja kuusi apuna käyttäen ja se antaa prototyypistä poiketen varsinaiselle sovellukselle suunnan myös värimaailman suhteen. Tarkemmassa käyttöliittymäsuunnittelussa on hyvä huomioida laitteiden fyysiset ominaisuudet, koska esimerkiksi Android-puhelimissa on fyysinen back-painike, kun taas iOS-laitteissa ei sellaista ole. Web-sovelluksessa kannattaakin miettiä tarkasti, mitä painikkeita käyttäjä tarvitsee navigoidessaan sovelluksessa. Tarpeen vaatiessa varsinaisessa sovelluksessa voidaan käyttää alustakohtaisesti mukautuvia tyylejä ja toiminnallisuuksia. Tarvittavat painikkeet voidaan myös leika-

ta Photoshopin avulla irti luodusta käyttöliittymäkuvasta ja siirtää ne lopulliseen sovellukseen.

Sovelluksen runkorakenne perustuu sisäänkirjautumis-, työtehtävien kirjaus- ja katselemisnäkymään. Näkymät ovat erillisiä kokonaisuuksia, koska sisäänkirjautumisen jälkeen kyseistä näkymää ei enää tarvita ja se voidaan piilottaa. Sisäänkirjautumisen jälkeen käyttäjälle näytetään sovelluksen varsinainen päänäkymä, joka mahdollistaa työtehtävien nauhoituksen. Kuviossa 9 on esitelty kirjautumisnäkymä, työtehtävien nauhoitusnäkymä ja käyttäjän tekemät tuntikirjaukset. Photoshopilla suunniteltu käyttöliittymäkuva voi poiketa lopullisesta versiosta, koska suunnittelu auttaa ainoastaan hahmottamaan ja ideoimaan tehtävän sovelluksen kokonaiskuvaa.



KUVIO 9. Sovelluksen ulkoasun Photoshop-versio

7.4 Kooditiedostot

Kun tarvittavat ohjelmat ja kehitysympäristö on saatu asennettua tietokoneelle, sovelluksen vaatimukset määritettyä, ulkoasu hahmotettua ja teknologia valittua, voi itse sovelluksen toteuttaminen alkaa.

PhoneGap-projektin aloitukseen Eclipsellä löytyy ohjeita mm. PhoneGapin kotisivulta osoitteesta: http://docs.phonegap.com/en/2.2.0/guide_getting-started_index.md.html, jossa esitellään yksityiskohtaisesti kehitys eri valmistajien alustoille. Hyviä oppaita PhoneGap-sovellusten tekemiseen ovat myös alan kirjallisuus ja opetusvideot YouTube-palvelussa.

Ohjelmointivaihe

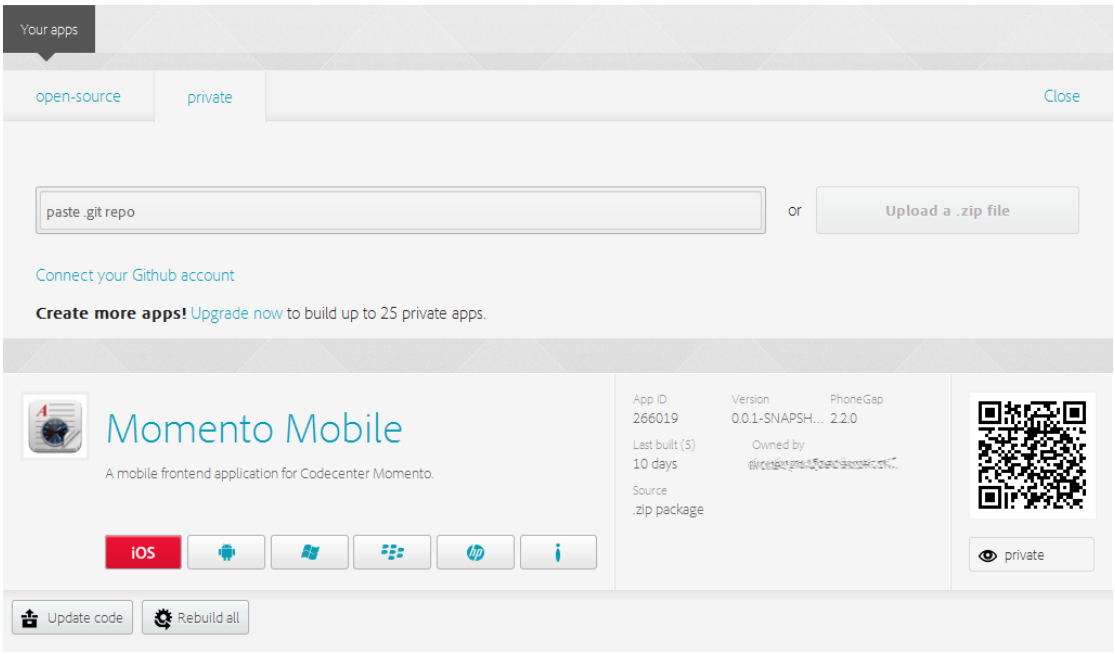
Ohjelmointivaiheessa käytetään apuna hyviä ohjelmointikäytänteitä ja testauskappaleessa esille tulleita asioita, jotta sisältöä pystytään jatkokehittämään ja kontrolloimaan sujuvasti koko projektin ajan. Varsinaista ohjelmointivaihetta ei käydä yksityiskohtaisesti työssä läpi, vaan osa lopullisista tiedostoista sijoitetaan työn liitteisiin, kuten myös kuva sovelluksen valmiista ulkoasusta. Osalla tiedostoja tarkoitetaan tässä tapauksessa pom.xml- ja config.xml -tiedostoja, mutta ei esimerkiksi HTML- ja CSS3 -tiedostoja. Tiedostojen on tarkoitus helpottaa lukijan käsitystä valmiista sovelluksesta ja PhoneGap-projektin luonnista.

Momenton ja mobiilisovelluksen välisessä tiedonsiirrossa tietoa lähetetään JSON-muodossa HTTP:n yli. JSON:a on helppo käyttää JavaScriptin kanssa, mutta nimestään huolimatta sitä voidaan käyttää myös muilla ohjelmointikielillä, kuten Javalla.

7.5 Paketoiminen natiivisovellukseksi

PhoneGapin yksi vahvuus on ohjelmakoodin kääntäminen eri valmistajien mobiililaitteille, ja samalla sovellus saadaan toimimaan myös natiivisovelluksen omaisesti mobiililaitteessa. Kääntämistä varten käyttäjä tarvitsee Adobe- tai Github-tunnukset, jolla voidaan kirjautua PhoneGapin käännöspalveluun osoitteessa: https://build.phonegap.com/plans/free. PhoneGap Build -palvelussa käyttäjä voi valita ohjelman lähdekoodit suoraan Github-palvelusta, jolloin sovellus luokitellaan julkiseksi sovellukseksi, ja niiden kääntämismäärää ei ole rajoitettu. Toinen vaihtoehto on ladata palveluun yksityinen sovellus .zip-tiedostona, joka voidaan kääntää samalla periaatteella kuin Github-palvelusta ladattu tiedosto, mutta silloin sovelluksien käännösmäärä on rajoitettu: yksi sovellus yhtä käyttäjää kohden ilmaiseksi. Kuviossa 11

on esitelty Momento Mobile -sovelluksen lataus käännöspalveluun ja se, miltä sovellus näyttää käännöksen jälkeen.



KUVIO 10. PhoneGap Build-palvelun käyttö

PhoneGap Build on palveluna hyvä, koska se tarjoaa vaihtoehtoisia tapoja ohjelman lähdekoodien lisäämiseen ja niiden kääntämiseen. Rajoittavia tekijöitä ovat ohjelmien tekeminen yksityiskäyttöön ja sovelluksen kääntämiseen ja paketointiin kuluva aika, joka voi kestää sovelluksen koosta riippuen usean minuutin.

Käännöksen jälkeen valmiin sovelluksen voi ladata yllä olevan kuvan sinisistä painikkeista pöytäkoneelle tai suoraan fyysiseen mobiililaitteeseen ja asentaa sen siihen. Vaihtoehtoinen tapa sovelluksen kääntämiselle on käyttää Maven-projektia, jolloin sovelluksen paketoiminen onnistuu komennolla: mvn clean install -Dphonegap-build.username=[USERNAME] -Dphonegap-build.password=[PASSWORD].

Kaiken kaikkiaan Build-palvelu tuntuisi toimivan hyvin, mutta iOS-laitteelle paketoiminen ei onnistunut, koska paketoiminen vaatii toimiakseen kehittäjän allekirjoituksen, jonka saa rekisteröitymällä iOS Developer Centeriin. Lisäksi Bada-laitteelle paketoimista ei ollut saatavana.

8 TUTKIMUSTULOKSET

Tämän opinnäytetyön tutkimustuloksia ovat raportti projektista, toimeksiantajan mobiilisovellusten luonnin tehostuminen uuden teknologian avulla sekä uusi mobiilisovellus. Tarkemmat tulokset ajankäytöstä, teknologian tehokkuudesta ja sen sopivuudesta toimeksiantajalle havaitaan kuitenkin vasta pidemmällä aikavälillä monipuolisten ja erityyppisten projektien avulla.

Tutkimuksesta hyötyvät monipuolisesti erilaiset ohjelmistotalot, jotka ovat aikoneet tutkia natiivikehityksen ja alustariippumattoman sovelluskehityksen hyötyjä ja haittoja. Tutkimuksen ensisijainen hyöty tulee kuitenkin toimeksiantajalle, koska mobiilisovellus kehitettiin juuri tämän yrityksen tarpeiden mukaan valmiiseen tuntikirjanpitojärjestelmään.

Kumpi teknologia on työtehtävien kirjaamiseen mobiililaitteella parempi ratkaisu: Android natiivikehitys vai alustariippumaton PhoneGap?

Tutkimuksen pääkysymykseen on saatu vastaus alakysymyksien tuloksista. Alakysymysten vastaukset on taas saatu jakamalla tutkimus loogisesti eteneviin osiin ja vaiheisiin, jotta se olisi helpompi toteuttaa. Tällainen vaihejaottelu on tyypillinen myös ohjelmistoprojekteille, jossa ennen ohjelman toteutusta seuraavat esimerkiksi asiakkaan haastattelu ja esitutkimus, jolloin osataan hahmottaa, mitä vaatimuksia käytännön toiminta asettaa sovellukselle.

Näiden alakysymysten perusteella tutkimuksen pääkysymykseen on saatu vastaukseksi PhoneGap. Luvussa seitsemän ovat tarkemmat perustelut sille, miksi PhoneGap on tässä tapauksessa parempi ratkaisu työn toteuttamisen kannalta.

Mitä vaatimuksia käytännön toiminta asettaa sovellukselle?

Ensimmäiseen alakysymykseen saadaan vastaus luvusta viisi, jossa selvitettiin asiakasta haastattelemalla yrityksen nykyiset toimintatavat ja ongelmat. Tämän haastattelun jälkeen sovellukselle voitiin luoda tarkemmat vaatimusmääritellyt liitteeseen 1, ja niistä muodostettiin erillinen taulukko vaadittavista toiminnoista liitteeseen 2. So-

vellusta mietittiin lisäksi käyttäjän näkökulmasta (liite 3), mikä auttoi hahmottamaan myöhemmässä vaiheessa sovelluksen käytettävyyttä ja käyttöliittymän suunnittelua.

Miten teknologian paremmuutta arvioidaan?

Toiseen alakysymykseen saadaan vastaus luvusta kuusi, jossa tutkittiin tarkemmin teknologioiden eroja ja niiden teknisiä ominaisuuksia. Android- ja PhoneGap teknologioiden tekniset ominaisuudet otettiin vertailuun liitteen 1 ja 2 mukaan, jolloin saatiin selvitettyä tieto siitä, mitä vaadittavia teknisiä ominaisuuksia kummallakin teknologialla voidaan tehdä. Tämän jälkeen teknologioita verrattiin kehitettävään sovellukseen niiden heikkouksien, etujen, teknisten edellytyksien, testauksen ja jatkokehityksen perusteella, jolloin luvussa 7 voitiin tarkasti perustella, kumpi teknologia on työtehtävien kirjaamiseen mobiililaitteella parempi ratkaisu.

Millainen käyttöliittymän tulisi olla?

Kolmanteen alakysymykseen saadaan vastaus liitteestä 1 ja 3. Näissä liitteissä määriteltiin sovelluksen toiminnalliset vaatimukset ja käyttäjän tarpeet sovelluksen suhteen. Nämä kaksi liitettä vastaavat siis kysymykseen: millainen käyttöliittymän tulisia olla. Kun sovelluksesta tiedetään toiminnalliset vaatimukset ja käyttäjän tarpeet, voidaan sovelluksen käyttöliittymästä luoda prototyyppi ja PhotoShop-versio.

9 POHDINTA

Lähtökohdat

Lähtökohdat olivat hyvät teknologioiden vertailun kannalta, koska työn tekijällä oli aikaisempaa kokemusta web-teknologioista ja Android-natiivikehityksestä. Vieraampi käsite oli PhoneGap, joka toi työhön oman haasteensa. Käytännössä teknologioiden vertailu oli kuitenkin todella raskasta, koska kahdesta eri teknologiasta piti etsiä vastaavat tiedot. Tämä tarkoitti useiden eri alan teosten lainaamista kirjastosta sekä niiden ostamista omaksi muutoin heikon saannin vuoksi. Asiaan perehdyttiin myös

kokeellisten tutkimusten perusteella, joissa testattiin web-tekniikoilla toteutetun sovelluksen toimivuutta eri mobiililaitteiden selaimissa.

Työn luotettavuus

Tutkimuksen tulosten kannalta on otettava huomioon, että tulokset käsittelevät tilannetta pääasiassa syksyllä 2012, ja tulkinnat valittavista menetelmistä ja ratkaisuista perustavat sillä hetkellä käytettyyn aineistoon ja tieteellisiin tiedonkeruu- ja analyysimenetelmiin. Vaikka lähdeaineisto on vahvasti tutkimusta tukevaa, siinä on käytettyä alan uusinta kirjallisuutta ja Internetistä löytyvää sisältöä on otettu vain luotettavista lähteistä, on kuitenkin huomioitava, että ala on jatkuvasti muuttuva, joten tulosten paikkansapitävyys voi muuttua nopeasti. Työn kannalta olisi ollut myös parempi, jos työn tekijällä olisi ollut aikaa toteuttaa sovellus molemmilla teknologioilla, jolloin työtä olisi voinut lähestyä teoriapohjan lisäksi käytännön kokemuksen pohjalta. Työn lopullisen tuotoksen, teknologioiden vertailun, luotettavuutta voidaan kuitenkin arvioida tavoitteiden täyttymisellä. Työ pystyi vastamaan tutkimuskysymykseen ja sen alakysymyksiin todella yksityiskohtaisesti ja kattavasti. Myös työn toimeksiantaja piti työtä laadukkaana.

Tutkimusmenetelmien soveltuvuus

Työssä käytettävän kehittämistutkimuksen avulla taustalla oleva ilmiö, prosessi ja asiantila saatiin kehittämisen ja muutoksen jälkeen paremmaksi. Työn toimeksiantaja hyötyi työstä saamalla uudella teknologialla tehdyn mobiilisovelluksen, jota voidaan jatkokehittää tehokkaasti ja saamalla lisää tietoa web- ja Android-teknologioista.

Työn toteutus

Itse sovelluksen kehittäminen aloitettiin yhdessä työn toimeksiantajan kanssa, jolloin sovelluksesta luotiin toimiva Maven-projekti ja mahdollisuus käyttää yrityksen versionhallintajärjestelmää (SVN). PhoneGap Build -palveluun luotiin myös yrityksen omistama käyttäjätili.

Sovellus toteutettiin kokonaisuudessaan luvussa 3.4 mainittavilla teknologioilla. Nämä teknologiat osoittautuivat hyviksi vaihtoehdoiksi työn toteutuksen kannalta, ja itse sovelluksessa on päästy vaiheeseen, jossa on toteutettu vaatimusmäärittelyn mukaiset ”pakollinen” -toiminto kohdat (liite 1). Käytännön toteutuksen perusteella näyttäisi siltä, että tutkimuksessa tehdyt johtopäätökset PhoneGapin eduista ja haitoista pitävät muuten paikkansa, mutta PhoneGap tukee todellisuudessa vain kuutta eikä seitsemän laitevalmistajan tuotetta. Tutkimuksen liitteissä 5 ja 6 on kuvattu sovelluksen pom.xml- ja config.xml -tiedostoja. Sovelluksen lopullinen ulkoasu on nähtävissä liitteestä 7.

Jatkotutkimukset

Tämän tutkimuksen jälkeen voisi tulevaisuudessa olla aiheellista tehdä alustariippumattomille sovelluksille suorituskykytestauksia ja selvittää, missä vaiheessa niiden suorituskyky tippuu käyttäjää häiritsevälle tasolle. Lisäksi olisi mielenkiintoista tutkia enemmän web-teknologioilla tehdyn sovelluksen käyttäjäkokemusta ja niiden hyötyjä liiketoiminnan näkökannalta.

Yhteenveto

Opinnäytetyön tuloksena saatiin tieto siitä, voiko web-teknologioilla tehty mobiilisovellus haastaa natiivikehityksen, ja vastaus oli kyllä. Työssä tutkittavissa teknologioissa on hyvät ja huonot puolensa, mutta hyviä puolia on tässä tapauksessa PhoneGapilla enemmän, ja hyvät puolet kumoavat kielen pienet epäkohdat. Työ on hyvä esimerkki haastamaan ihmisten ennakkoluulot web-teknologioilla toteutetuista mobiilisovelluksista ja siitä, kuinka nykypäivänä ei ole enää pakko luottaa perinteisiin natiiviohjelmointikieliin, vaan kehittäjille on tarjolla uusia mahdollisuuksia, joiden avulla mobiilisovellusten kehittämistä pystytään helpottamaan ja nopeuttamaan. On kuitenkin muistettava, että käytettävät teknologiat pitää arvioida sovelluskohtaisesti ja että laadukas koodi on laadukasta koodia alustasta ja ohjelmointikielestä riippumatta.

LÄHTEET

About The W3C Markup Validation Service. N.d. W3C Markup Validation Service. Viitattu 18.11.2012. http://validator.w3.org/about.html.

Acid3 Browser test. 2012. The Web Standards Project. Viitattu 3.9.2012. http://www.webstandards.org/action/acid3/.

Add phonegap-nfc to your project. 2012. PhoneGap Blog. Viitattu 8.11.2012. http://phonegap.com/2011/09/26/building-an-nfc-enabled-android-application-with-phonegap/.

Adobe PhoneGap Build. N.d. Choose your plan. Viitattu 4.10.2012. https://build.phonegap.com/.

Bister, T. 2011. Tutkimuksesta ja sen tekemisestä. Viitattu 28.9.2012. https://optima.jamk.fi/learning/id2/bin/doc_show?id=531640&ws=2194.

Broulik, B. 2011. Pro jQuery Mobile. New York:Apress L. P. ISBN: 978-1-4302-3966-6.

Cross, S. 2012. Facebook State of Mobile. Viitattu 9.11.2012. http://www.youtube.com/watch?v=A7gNig-2iy0&feature=player_embedded.

Deitel, P. & Deitel, H. 2010. C How to program. 6. ed. New Jersey:Person Education. ISBN: 978-0-13-612356-9.

Dykes, L. 2005. XML for Dummies. 4. ed. Indianapolis, Indiana:Wiley Publishing. ISBN: 978-0-07645-8845-7.

Fling, B. 2009. Mobile Design and Development. Sebastopol:O'Reilly Media. ISBN: 978-0-596-15544-5.

Gasston, P. 2011. The book of CSS3. A developer's guide to the future of web design. 1. ed. Canada:No Starch Press. ISBN: 978-1-59327-286-9.

Get the Android SDK. 2012. Download the SDK for Windows. Viitattu 19.10.2012. http://developer.android.com/sdk/index.html.

Ghatol, R. & Patel, Y. 2012. Beginning PhoneGap. Mobile Web Framework for JavaScript and HTML5. New York:Apress L.P. ISBN: 978-1-4302-3903-1.

Immonen, J. 2012. Käytettävyyttä lyhyesti. Viitattu 16.10.2012. http://batman.jamk.fi/~imjar/usabweb/kaytettavyys.html.

Kananen, J. Kehittämistutkimus opinnäytetyönä. Kehittämistutkimuksen kirjoittamisen käytännön opas. Jyväskylän ammattikorkeakoulun julkaisuja 134. Tampere:Juvenes Print.

Kemppainen, T. 2011. Tv-Compass. Monikanavainen median toisto. Viitattu 20.9.2012. http://urn.fi/URN:NBN:fi:amk-2011112114916.

Kessin, Z. 2012. Programming HTML5 Applications. Sebastopol:O'Reilly Media. ISBN: 978-1-449-39908-5.

Kolehmainen, A. 2009. Ohjelmistotestaus Opas ohjelmistokehittäjille. Viitattu 13.11.2012. http://urn.fi/URN:NBN:fi:amk-200912168016.

Kolehmainen, A. 2010. Mobiilisovellus joka sormelle. Viitattu 24.9.2012. http://www.tietoviikko.fi/taustat/mobiilisovellus+joka+sormelle/a538313?service=mobile.

Korpela, J. 2011. HTML5 uudet ominaisuudet. Porvoo:Docendo. ISBN: 978-951-0-0-38137-3.

Kosonen, P., Peltomäki, J. & Silander, S. 2005. Java 2 Ohjelmoinnin peruskirja. Porvoo:Docendo. ISBN: 951-846-237-2.

Kuha, J. 2008. Tehokas Java EE-sovellustuotanto. Porvoo:Docendo. ISBN: 978-951-0-34166-7.

Kurka, D. 2012. A GWT Wrapper for the PhoneGap JS Library. Viitattu 2.12.2012. http://code.google.com/p/gwt-phonegap/wiki/GettingStarted.

Laine, P. 2012. W3C: HTML5 viralliseksi suositukseksi 2014. Viitattu 25.9.2012. http://muropaketti.com/w3c-html-5-viralliseksi-suositukseksi-2014.

Laracker, H. 2011. Android basic principles. Viitattu 6.10.2012. http://www.slideshare.net/HenkLaracker/android-8165957.

Lee, W. 2012. Beginning Android 4 Application Development. Indianapolis, Indiana:Wiley & Sons. ISBN: 978-1-118-19954-1.

Liberty, J. 1998. C++ Trainer. Helsinki:IT Press. ISBN: 951-826-010-9.

Mednieks, Z., Dornin, L., Meike, B. & Nakamura, M. 2011. Programmin Android. Java Programming for the New Generation of Mobile Devices. Sebastopol:O'Reilly Media. ISBN: 978-1-449-38969-7.

Meier, R. 2012. Professional Android 4 Application Development. Indianapolis, Indiana:Wiley & Sons. ISBN: 978-1-118-10227-5.

Mikkonen, T. 2004. Mobiiliohjelmointi. 2. uud. p. Helsinki:Talentum. ISBN: P9521408618.

NFC Basics. 2012. Android Developers. Viitattu 11.11.2012. http://developer.android.com/guide/topics/connectivity/nfc/nfc.html.

Paakki, J. 2010. Ohjelmistojen vaatimusmäärittely. Viitattu 14.10.2012. http://www.cs.helsinki.fi/u/paakki/Vaatimus-11-Luentokalvot-1.pdf.

PhoneGap, Cordova, and what's in a name? 2012. Viitattu 20.10.2012. http://phonegap.com/2012/03/19/phonegap-cordova-and-what%E2%80%99s-in-a-name/.

Price, C. 2012. Using PhoneGap Build with Maven. Viitattu 10.11.2012. http://www.scottlogic.co.uk/2012/06/using-phonegap-build-with-maven/.

Reid, J. 2011. jQuery Mobile. Building Cross-Platform Mobile Applications. Sebastopol:O'Reilly Media. ISBN: 978-1-449-30668-7.

Sani, I. 2011. jQuery Mobile uunista ulos. Viitattu 26.9.2012. http://www.tietoviikko.fi/kehittaja/jquery+mobile+uunista+ulos/a724861?service=mobile.

Sawyer, D. 2011. JavaScript & jQuery. The missing manual. 2. ed. Sebastopol:O'Reilly Media. ISBN: 978-1-449-3-9902-3.

Security Architecture. 2012. Android Developers. Viitattu 10.11.2012. http://developer.android.com/guide/topics/security/permissions.html.

Snider, L. 2010. Photoshop CS5 the missing manual. Sebastopol:O'Reilly Media. ISBN: 9781449381684.

Supported Features. N.d. PhoneGap Docs. Viitattu 24.10.2012. http://phonegap.com/about/feature.

Tässä on uusi merkki siitä, että Android on vaikeuksissa. 2012. Artikkeli Talous-elämä-lehdessä. Viitattu 29.10.2012. http://www.talouselama.fi/nokialandia/tassa+on+uusi+merkki+siita+etta+android+on+vaikeuksissa/a2092715#commentsList.

Urpi, T. 2010. Vaihtoehtoiset työpöytäsovellukset. Viitattu 12.11.2012. http://blog.androidsuomi.fi/2010/12/vaihtoehtoiset-tyopoytasovellukset/.

Vaadin. 2012. Vaadin Faq. Viitattu 30.9.2012. https://vaadin.com/faq.

Varisvirta, K. 2011. Mobiilisovellus-alustat ja tausta-järjestelmät. Viitattu 21.9.2012. http://www.slideshare.net/exove/mobiilisovellukset-ja-sivustot.

Vuorela, A. 2011. Web-teknologiat mobiililaitteiden ohjelmistokehityksessä. Viitattu 15.10.2012. http://www.scribd.com/doc/55349538/Web-teknologiat-mobiililaitteiden-ohjelmistokehityksessa-in-Finnish.

Wargo, J. 2012. PhoneGap Essentials. Building Cross-Platform Mobile Apps. New Jersey:Person Education. ISBN: 978-0-321-81429-6.

WaSP: Fighting for Standards. N.d. The Web Standards Project. Viitattu 3.9.2012. http://www.webstandards.org/about/mission/.

Web Apps Overview. 2012. Web apps overview. Viitattu 10.10.2012. http://developer.android.com/guide/webapps/overview.html.

LIITTEET

Liite 1. Vaatimusmäärittely

Tässä liitteessä on listattu toteutettavan tuntikirjanpitosovelluksen toiminnalliset ja ei-toiminnalliset vaatimukset. Vaatimukset on jaoteltu pakollisiin ja valinnaisiin vaatimuksiin. Valinnaisia vaatimuksia ei toteuteta opinnäytetyön yhteydessä.

Toiminnalliset vaatimukset

1. Käyttäjä voi kirjautua sovelluksen sisään. Pakollinen.
2. Käyttäjä voi kirjautua ulos sovelluksesta. Pakollinen.
3. Käyttäjä voi käyttää sovellusta offline-tilassa. Valinnainen.
4. Käyttäjä voi aloittaa työtehtävän nauhoituksen. Pakollinen.
5. Käyttäjä voi pysäyttää työtehtävän nauhoituksen. Pakollinen.
6. Käyttäjä voi lähettää kirjatut työtunnit Momentoon. Valinnainen.
7. Käyttäjä voi muokata kirjattuja työtunteja. Pakollinen.
8. Käyttäjä voi valita projektin. Pakollinen.
9. Käyttäjä voi valita työtehtävän. Pakollinen.
10. Käyttäjä voi lisätä kuvauksen työtehtävästä. Pakollinen.
11. Käyttäjä voi selata tuntikirjauksia. Pakollinen.
12. Käyttäjä voi poistaa tuntikirjauksia. Valinnainen.
13. Käyttäjä voi vaihtaa työtehtävän päivämäärää. Valinnainen.
14. Käyttäjä voi lisätä työtehtävän tiedot NFC-lukijaa käyttäen. Valinnainen.
15. Käyttäjä voi palauttaa käyttäjätunnuksen ja salasanan. Pakollinen.
16. Automaattinen kielivalinta. Pakollinen.
17. Sovelluksen tulee automaattisesti katsoa työtehtävän aloitus- ja lopetusaika. Valinnainen.

Ei-toiminnalliset vaatimukset

1. Sovellukseen voi tehdä yksikkötestejä. Valinnainen.
2. Sovelluksen pitää olla ajettavissa debuggerin avulla. Pakollinen.
3. Teknologian pitää olla nykyaikaista ja laadukasta. Pakollinen.
4. Sovelluksen pitää olla silmämääräisesti suorituskykyinen uusissa mobiililaitteissa. Pakollinen.
5. Sovelluksen pitää olla taloudellisesti ja teknisesti jatkokehitettävissä. Pakollinen.
6. Projektista pitää pystyä tekemään Maven-projekti. Pakollinen.

Liite 2. Sovelluksen vaadittavat toiminnot

Luvussa kuusi on tarkemmin määritelty liitteessä 2 olevat asiat, kuten jatkokehitys. Sovellukselle tärkeistä toiminnoista on annettu pisteitä 0–1 välillä, jossa nolla tarkoittaa, että toiminto on huono, vanhanaikainen tai se ei ole saatavilla PhoneGap-sovelluskehityksessä tai Android-natiivikehityksessä. Numero yksi tarkoittaa, että toiminto on saatavilla.

Toiminto	PhoneGap-sovelluskehitys	Android-natiivikehitys
Testaus (yksikkötestit)	1	1
Suorituskyky	1	1
Maven plugin	1	1
Jatkokehitys	1	1
Nykyaikaisuus	1	0
Laatu ja ylläpito	1	1
Natiiviominaisuudet (paikannus)	1	1
Debuggaus	1	1
NFC-tuki	1	1
Widget-tuki	0	1
Monen alustan tuki	1	0
Yhteensä	10	9

Liite 3. Käyttäjien tarpeet

Liitteessä 3 on kirjattu luvussa 5.3 esille tulleisiin kysymyksiin vastaukset. Vastaukset on jaettu numeroiden mukaan, jolloin niissä noudatetaan samaa vastaus- ja kysymysjärjestystä kuin luvussa 5.3.

1. Käyttäjät ovat kaiken-ikäisiä, tietoteknisiltä taidoiltaan eritasoisia ja työssä käyviä ihmisiä, joilla on käytössä Momento-tuntikirjanpitojärjestelmä.
2. Sovellusta käytetään, kun työtehtävän tekeminen aloitetaan, keskeytetään tai kun se on saatu valmiiksi.
3. Käyttäjät käyttävät sovellusta todennäköisesti töissä kiireen keskellä.
4. Tuntikirjaukset voidaan tehdä missä tahansa, mihin kellonaikaan tahansa ja sijainnista riippumatta.
5. Sovellus auttaa helpottamaan tuntikirjausten tekoa, jolloin tuntikirjausten tekeminen esimerkiksi Post-it-lappuihin katoaa.
6. Mobiililaite on käyttäjän kädessä pystyasennossa.

Liite 4. PhoneGap Maven-pluginin käyttöönotto (Price 2012)

```
<!-- PhoneGap Maven-pluginin liittäminen projektin POM-tiedostoon. -->
<plugin>
    <groupId>com.github.chrisprice</groupId>
    <artifactId>phonegap-build-maven-plugin</artifactId>
    <version>0.0.3</version>
    <executions>
        <execution>
            <id>phonegap-build</id>
            <!-- the goals are lifecycle bound by default -->
            <goals>
                <goal>clean</goal>
                <goal>build</goal>
            </goals>
            <configuration>
                <platforms>
                    <platform>ios</platform>
                    <platform>android</platform>
                    <platform>winphone</platform>
                    <platform>webos</platform>
                    <platform>symbian</platform>
                    <platform>blackberry</platform>
                </platforms>
            </configuration>
        </execution>
    </executions>
</plugin>

<!-- Config.xml-tiedosto. Tiedosto sisältää keskeisiä asioista sovelluksesta. -->
<?xml version="1.0" encoding="UTF-8" ?>
<widget xmlns="http://www.w3.org/ns/widgets" xmlns:gap="http://phonegap.com/ns/1.0"
        id="fi.codecenter.momento.momento-mobile" version="0.0.1-SNAPSHOT">
    <name>Momento Mobile</name>
    <description>A mobile frontend application for Codecenter Momento.</description>
    <author href="http://codecenter.fi" email="">Codecenter Oy</author>
</widget>

<!-- Aja buildaus seuraavalla komennolla. -->
mvn clean install -Dphonegap-build.username=[USERNAME] -Dphonegap-build.password=[PASSWORD]
```

Liite 5. Momento Mobile pom.xml

```xml
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>fi.codecenter</groupId>
        <artifactId>codecenter</artifactId>
        <version>3.0.2</version>
    </parent>
    <groupId>fi.codecenter.momento</groupId>
    <artifactId>momento-mobile</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <packaging>war</packaging>
    <build>
        <plugins>
            <plugin>
                <groupId>com.github.chrisprice</groupId>
                <artifactId>phonegap-build-maven-plugin</artifactId>
                <version>0.0.3</version>
                <executions>
                    <execution>
                        <id>phonegap-build</id>
                        <!-- the goals are lifecycle bound by default -->
                        <goals>
                            <goal>clean</goal>
                            <goal>build</goal>
                        </goals>
                        <configuration>
                            <configFile>src/main/resources/config.xml</configFile>
                            <platforms>
                                <platform>ios</platform>
                                <platform>android</platform>
                                <platform>winphone</platform>
                            </platforms>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-war-plugin</artifactId>
                <configuration>
                    <failOnMissingWebXml>false</failOnMissingWebXml>
                </configuration>
            </plugin>
        </plugins>
    </build>
    <properties>
        <phonegap-build.server>momento.phonegap-build</phonegap-build.server>
    </properties>
</project>
```

Liite 6. Momento Mobile config.xml

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<widget xmlns="http://www.w3.org/ns/widgets" xmlns:gap="http://phonegap.com/ns/1.0"
        id="fi.codecenter.momento.momento-mobile" version="0.0.1-SNAPSHOT">
    <name>Momento Mobile</name>

    <description>A mobile frontend application for Codecenter Momento.</description>

    <author href="http://codecenter.fi" email="">Codecenter Oy</author>

    <preference name="phonegap-version" value="2.2.0" />
    <preference name="orientation" value="portrait" />
    <preference name="target-device" value="universal" />
    <preference name="fullscreen" value="true" />
    <preference name="permissions" value="none"/>

    <cordova>
        <access origin="*" subdomains="true" />
        <preference name="useBrowserHistory" value="true" />
    </cordova>

    <!-- Plugins -->
    <gap:plugin name="BarcodeScanner" /> <!-- latest release -->
    <!-- Plugins end -->

    <icon src="res/icon/icon.png" />
    <gap:splash src="res/splash/splash.png" />

    <!-- iOS Specific -->
    <preference name="prerendered-icon" value="true" />
    <preference name="exit-on-suspend" value="true" />
    <!-- iOS Specific end -->

    <!-- Android Specific -->
    <preference name="android-minSdkVersion" value="10" />
    <preference name="android-maxSdkVersion" value="17" />
    <preference name="android-installLocation" value="auto" />

    <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" />

    <gap:splash src="res/splash/android/ldpi-portrait.png" gap:platform="android" gap:density="ldpi" />
    <gap:splash src="res/splash/android/mdpi-portrait.png" gap:platform="android" gap:density="mdpi" />
    <gap:splash src="res/splash/android/hdpi-portrait.png" gap:platform="android" gap:density="hdpi" />
    <gap:splash src="res/splash/android/xhdpi-portrait.png" gap:platform="android" gap:density="xhdpi" />
    <!-- Android Specific end -->

</widget>
```

Liite 7. Momento Mobile-sovelluksen ulkoasu