LAHDEN AMMATTIKORKEAKOULU
Lahti University of Applied Sciences

MOBIILISOVELLUSTEN TOTEUTUSTEKNIIKAT

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikan koulutusohjelma
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2013
Juha-Matti Viertola

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

VIERTOLA, JUHA-MATTI: Mobiilisovellusten toteutustekniikat

Ohjelmistotekniikan opinnäytetyö, 42 sivua

Kevät 2013

TIIVISTELMÄ

Tämän opinnäytetyön tarkoituksena oli tutkia erilaisia vaihtoehtoja mobiilisovellusten toteuttamiseksi. Mobiililaitteiden ja niiden käyttöjärjestelmien kirjo on valtava, ja se luokin haasteita sovelluskehittäjille.

Eri käyttöjärjestelmät vaativat natiivien sovellusten ohjelmoimiseen omat ohjelmointikielensä, minkä takia tämänkaltaisia sovelluksia on hyvin työlästä tuottaa useammalle eri alustalle. Mobiililaitteiden käytön yleistymisen myötä intressejä sovelluksien tekemiseen on kuitenkin olemassa, ja se herättääkin mielenkiintoa ratkaisuihin, joilla yhdellä lähdekoodilla voidaan kattaa useampia laitealustoja.

Tällaisia ratkaisuja ovat natiiveiksi sovelluksiksi paketoidut, web-teknologioita hyödyntävät sovellukset, sekä puhtaasti internetselaimessa toimivat sovellukset. Molemmissa vaihtoehdoissa on kuitenkin omat ongelmansa esimerkiksi suorituskyvyn suhteen.

Hybridiratkaisun hyvien puolien innoittamana tämän opinnäytetyön yhteydessä on toteutettu yksinkertainen reittiopassovellus, jonka avulla on mahdollista hakea reittiehdotuksia kotimaassa ovelta ovelle -periaatteella, julkista liikennettä hyödyntäen. Toteutuksessa on käytetty Sencha Touch- ja PhoneGap-sovelluskehyksiä.

Sovelluksen ohjelmointi paljasti sovelluskehysten käytössä olleiden versioiden olevan hieman keskeneräisen oloisia sekä myös sen, kuinka hankalaa kehitystyö voi olla, vaikka ohjelmointikielet olisivatkin tuttuja.

Asiasanat: mobiilisovellukset, mobiililaitteet, sovelluskehykset

Lahti University of Applied Sciences

Degree Programme in Information Technology

VIERTOLA, JUHA-MATTI: Mobile software development technologies

Bachelor's Thesis in Software Engineering, 42 pages

Spring 2013

ABSTRACT

This thesis focuses on different techniques of mobile application development. Versatility of mobile devices and operating systems creates challenges for software developers.

Creating native applications for different operating systems requires using of multiple programming languages, which makes it cumbersome. As the number of mobile device users grows rapidly, there is more interest in software developing, and demand for solutions that enable publishing for multiple platforms with single code base increases.

Such solutions are web-technologies utilizing native-like applications, as well as applications running purely in internet browsers. However, both of these options have their own weaknesses, for example in terms of performance.

Inspired by the merits of the hybrid solution, a simple journey planner application was developed as part of this thesis. It uses Sencha Touch and PhoneGap frameworks and it provides its users with door-to-door route suggestions by public transport in Finland.

Programming the application revealed that it would be better to wait for the frameworks develop further. Another observation was that it was difficult to master new technologies, even though the programming languages used were already familiar.

Key words: mobile software, mobile devices, framework

SISÄLLYS

1 JOHDANTO 1

2 MOBIILISOVELLUSTEN MÄÄRÄT KASVUSSA 3

2.1 Laitekanta kasvaa 4

2.2 Uudet mahdollisuudet 5

2.3 Sovellusten jakelu 5

2.4 Kehitystyö 7

3 MOBIILISOVELLUSTEN HAASTEET 9

3.1 Käyttöjärjestelmät 9

3.2 Laitteiden kehitys 11

3.3 Näyttöjen koot ja kuvasuhteet 11

3.4 Suorituskyky 13

4 VAIHTOEHTOISET TOTEUTUSTAVAT 16

4.1 Natiivit sovellukset 16

4.2 Selainpohjaiset sovellukset 16

4.3 Web-teknologioita hyödyntävät natiivisovellukset 18

5 PHONEGAP 19

5.1 Ideologia 19

5.2 Kehitys 20

6 SENCHA TOUCH 21

6.1 Ideologia 21

6.2 Kehitys 21

7 SOVELLUS 23

7.1 Ideologia 23

7.2 Teknologiat 23

7.3 Taustajärjestelmät 24

7.3.1 Koontikanta 25

7.3.2 Matka.fi API 26

7.4 Toteutus 26

7.5 Tekniikan tarjoamat mahdollisuudet 35

8 YHTEENVETO 36

LÄHTEET 38

1 JOHDANTO

Älypuhelimien ja tablettien tulo markkinoille on synnyttänyt uudenlaisen mahdollisuuden sekä niiden käyttäjille että sovelluskehittäjille, sillä käyttäjät pääsevät itse asentamaan näihin mobiililaitteisiinsa sovelluksia. Mobiilisovelluksia luodaan moniin eri tarpeisiin, mutta niiden toteutus ei kuitenkaan ole välttämättä ihan yksinkertaista. Siitä pitää huolen laitteiden kirjo. Sovelluskehittäjän on huomioitava esimerkiksi näyttöjen koot sekä prosessorien ja grafiikkasuorittimien suorituskyky. Näiden asioiden lisäksi osa laitteista sisältää muun muassa satelliittipaikannuksen ja kiihtyvyysanturin, jotka tuovat uusia mahdollisuuksia ohjelmistokehitykseen.

Laitteiden fyysisten ominaisuuksien moninaisuuden ohella eroavaisuuksia löytyy myös käyttöjärjestelmistä. Tällä hetkellä selkeästi suurimmat käyttöjärjestelmät ovat Microsoftin Windows, Applen iOS, sekä Android, jonka kehittämisestä vastaa Open Handset Alliance (Gartner, Inc. 2013). Käyttöjärjestelmän tarjoaman käyttökokemuksen lisäksi merkittäväksi asiaksi niiden suosion määrittelijänä on noussut tarjolla olevien, kolmansien osapuolien kehittämien sovellusten määrä. Näiden ohjelmien jakelu tapahtuu käyttöjärjestelmäkohtaisten sovelluskauppojen kautta, ja tätä kokonaisuutta kuvaavana terminä käytetään sanaa ekosysteemi.

Kaiken tämän takia sovelluskehittäjä joutuu tekemään valintoja: halutaanko ohjelma toteuttaa usealle eri alustalle vai riittääkö jokin yksittäinen alusta? Tämä onkin olennainen kysymys, sillä natiivisti toimivan sovelluksen tapauksessa jokainen valittu alusta tarkoittaa lisää ohjelmointityötä. Syy lisätyölle johtuu tarpeesta käyttää jokaiselle käyttöjärjestelmälle eri ohjelmointikieltä. Houkuttelevaa olisikin saada sovellus toimimaan useilla alustoilla käyttäen samaa koodipohjaa, jolloin kehitystyö ja ylläpito helpottuisivat.

Yksinkertaisempien sovellusten tapauksessa vaihtoehtona voi olla täysin käyttöjärjestelmäriippumaton web-sovellus, joka toimii internetselaimessa. Tällainen ratkaisu kuitenkin karsii mahdollisuuksia laitteiden ominaisuuksien käyttämiseksi, eikä ekosysteemien tuomiin hyötyihinkään päästä käsiksi.

Jäljelle jää vielä natiivisovellusten ja selainpohjaisten sovellusten välimaastoon asettuva malli, eli web-teknologioita hyödyntävät natiivisovellukset. Tällä

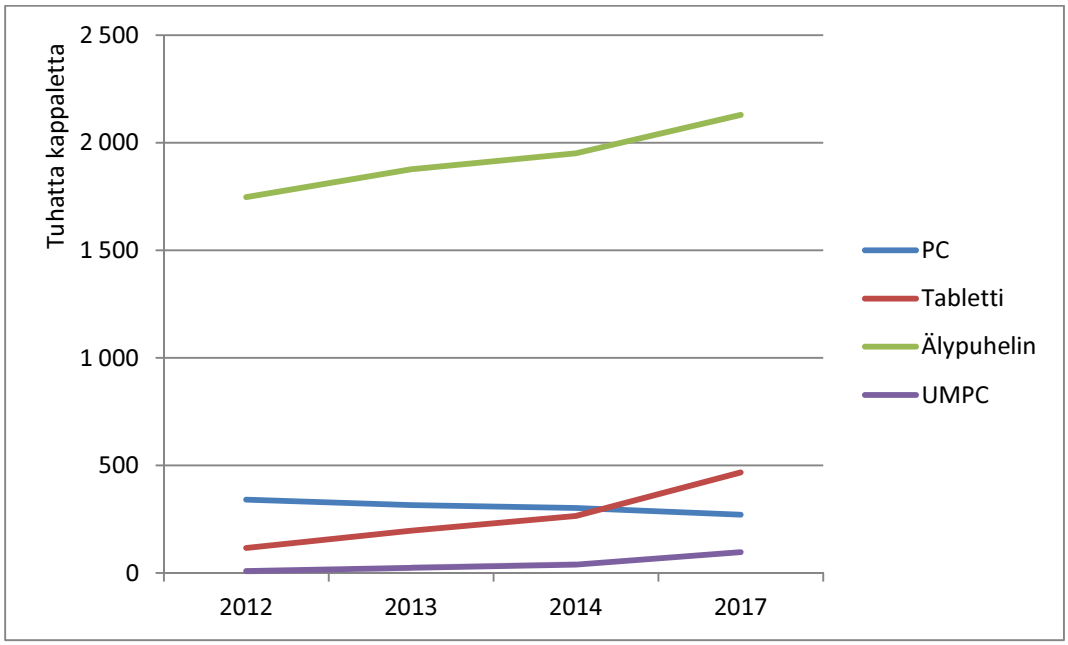
tarkoitetaan HTML-, CSS- ja JavaScript-tekniikoita käyttäviä sovelluksia, jotka paketoidaan erillisen frameworkin avulla natiivin tuntuisiksi, mutta todellisuudessa selaimessa toimiviksi sovelluksiksi. Tekniikka mahdollistaa mobiililaitteiden ominaisuuksien hyödyntämisen rajapintojen avulla sekä valmiiden sovellusten jakelemisen sovelluskauppojen kautta. Myös hybridiratkaisuissa on omat ongelmansa, kuten natiivisovelluksia heikompi suorituskyky.

Tässä opinnäytetyössä on vertailtu edellä mainittujen toteutustekniikoiden hyviä ja huonoja puolia sekä toteutettu hybriditeknologiaa hyödyntäen sovellus, jolla on mahdollista hakea reittiehdotuksia Suomessa käyttäen julkisia kulkuneuvoja. Ohjelma hyödyntää Matka.fi:n avointa ohjelmointirajapintaa muun muassa reitittämiseen. Tavoitteena oli tutkia, olisiko tällaisella ratkaisulla mahdollista toteuttaa kustannustehokkaasti ohjelmia useille eri mobiilialustoille. Eri sovelluskehysvaihtoehdoista työn toteutukseen valikoitui yhdistelmä Sencha Touch ja PhoneGap, sillä näiden yhteiskäyttö vaikutti mahdollistavan tavoitteiden täyttämisen.

Opinnäytetyö on tehty CGI:lle. CGI on globaali yritys, joka työllistää noin 69 000 ihmistä 40 maassa. Työstä saatu hyöty kohdistuu kuitenkin ensisijaisesti noin 30 hengen älyliikenteen sovelluksiin keskittyvään tiimiin.

2 MOBIILISOVELLUSTEN MÄÄRÄT KASVUSSA

Mobiililaitteet ovat kehittyneet viime vuosina vauhdikkaasti. Älypuhelimet ovat vallanneet markkinoita perinteisemmiltä kännyköiltä, ja tabletit ovat suosittuja kannettavien tietokoneiden korvaajina internetselailussa. Tämä näkyy kuviossa 1, josta on havaittavissa PC-laitteiden myynnin väheneminen samalla kun tablettien ja älypuhelimien myyntimäärät ovat kasvussa. Kuviossa PC-laitteilla tarkoitetaan sekä pöytäkoneita että kannettavia tietokoneita. Uuden teknologian tuomat mahdollisuudet ovat myös muuttaneet ihmisten haluja ja tarpeita niin laitteiden kuin sovelluksienkin osalta.



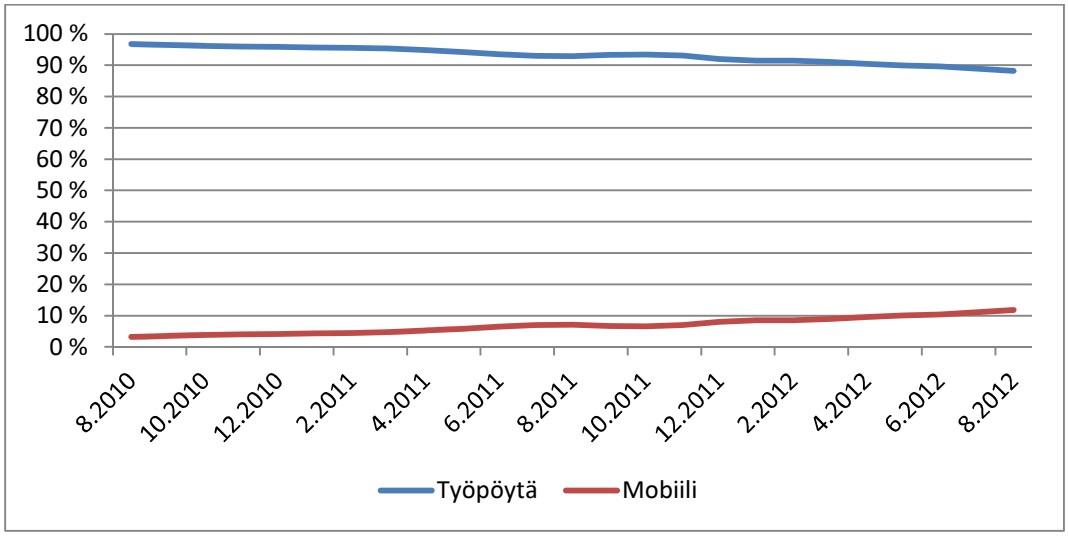
KUVIO 1. Laitemyynnin kehitys (Gartner, Inc. 2013)

Mobiililaitteisiin ja -sovelluksiin liittyvät markkinat ovat kasvaneet suuriksi. Chetan Sharma toteaa yhteenvedossaan niiden olevan jo noin kaksi prosenttia koko maailman bruttokansantuotteesta (Sharma 2011). Tämä tarkoittaa noin biljoonaa euroa reilun 51 biljoonan euron kokonaissummasta (Google 2012). Suuret markkinat houkuttelevat sovelluskehittäjiä, vaikka haasteitakin tuoreehko ala tarjoaa.

Kasvu ei myöskään ole hidastumassa. GSMA:n, mobiilioperaattoreiden ja vastaavien yritysten yhteisen järjestön tekemät ennusteet ovat huikeita. Esimerkiksi mobiiliverkoissa liikkuvan kuukausittaisen datan määrän on ennustettu kasvavan vuoden 2012 0,9 miljoonasta teratavusta 11,2 teratavuun vuoteen 2017 mennessä. Tämä tarkoittaa 66 prosentin vuotuista kasvua. (GSMA 2013.)

2.1 Laitekanta kasvaa

Markkinoiden kasvun taustalla yhtenä tekijänä on laitteiden määrän lisääntyminen. Kuviosta 2 on nähtävissä, kuinka StatCounter-sivuston keräämien maailmanlaajuisten tilastojen perusteella mobiili internetselailu syö työpöytälaitteiden osuutta noin neljän prosentin vuosivauhdilla. Tilastoja katsoessa on myös syytä huomata, että tabletit ja konsolit on laskettu työpöytälaitteiden kategoriaan, eli käytännössä muutos on vielä suurempi.



KUVIO 2. Mobiili- ja työpöytälaitteiden keskinäinen suhde internetselailussa (StatCounter 2012a)

Vaikka tässä tapauksessa ei olekaan suoraan kyse laitteiden määrästä, kertoo tilasto kuitenkin internetselailuun kykenevien mobiililaitteiden selkeästä lisääntymisestä. Tämä lisää mielenkiintoa uusien ohjelmien toteuttamiseksi.

2.2 Uudet mahdollisuudet

Tekniikan kehittyessä myös sovelluskehittäjille aukeaa uusia mahdollisuuksia, jollaisia ei ole ollut aiemmin olemassa, tai ainakaan ne eivät ole olleet itsestäänselvyyksiä. Eroavaisuuksia löytyy sekä vanhempiin mobiililaitteisiin että työpöytäkoneisiin jo toimintaperiaatteidenkin takia. Esimerkiksi liikkuminen käyttäjän mukana samalla paikkatietoa tarjoten on yksi mainitsemisen arvoisista uusista asioista, ja se luokin perustan monille hyödyllisille ja innovatiivisille ohjelmille.

Perinteisen näppäimistön ja työpöytäkäytöstä tutun hiiren korvaava kosketusnäyttö erilaisine kosketuseleineen muuttaa käyttöliittymäsuunnittelua, samalla kun kattavat yhteysratkaisut linkittävät mobiililaitteen ympäristöönsä. Esimerkkinä ”out of the box” ajattelusta toimii hyvin vaikkapa älypuhelimen käyttäminen olohuoneen viihdekeskuksen kaukosäätimenä (Kopacz 2012).

2.3 Sovellusten jakelu

Mobiilisovellukset ovat tuoneet tullessaan uudenlaiset, keskitetyt ohjelmien jakelukanavat. Nämä pyrkivät mahdollisimman helppoon ja nopeaan sovellusten jakeluun, joka käytännössä tarkoittaa ohjelmien digitaalista jakelua suoraan mobiililaitteisiin käyttöjärjestelmäkohtaisista sovelluskaupoista.

Sovelluskaupat ovat ainakin periaatteessa erittäin helppo väylä sovellusten jakelemiseksi. Ne ovat helppokäyttöisiä ja kaikkien niiden käyttäjien saavutettavissa, joilla on internetyhteys. Kuten taulukosta 1 käy ilmi, sovelluskaupat sisältävät kuitenkin valtavat määrät ohjelmia, joten massasta erottuminen voi muodostua ongelmaksi.

TAULUKKO 1. Sovelluskauppojen sovellusten ja latausten määrät (Wikipedia 2012)

Nimi	Alusta	Sovellusten lukumäärä	Päivittäisiä latauksia	Latauksia yhteensä
Google Play	Android	700 000 (10.2012)	25 miljardia	
App Store	iOS	700 000 (9.2012)	30 miljardia	
Nokia Store	Symbian, MeeGo, Maemo, S40	120 000 (8.2012)	6 miljardia	17 miljoonaa
Windows Phone Store	Windows Phone	126 530 (24.10.2012)	768 miljoonaa	
App World	Blackberry	60 000 (1.2012)		6 miljoonaa

Sovelluskauppojen suosion kasvusta kertoo kansainvälisen ICT-alan tutkimus- ja konsultointiyritys Gartnerin julkaisemasta uutisesta löytyvät ennusteet latausmääristä. Taulukosta 2 löytyvät luvut kuvaavat maailmanlaajuisia latausmääriä yksikön ollessa miljoonaa latausta.

TAULUKKO 2. Sovellusten latausmäärät, miljoonaa latausta (Gartner, Inc. 2012)

	2011	2012	2013	2014	2015	2016
Ilmaiset sovellukset	22 044	40 599	73 280	119 842	188 946	287 933
Maksulliset sovellukset	2 893	5 018	8 142	11 853	16 430	21 672
Latauksia yhteensä	24 936	45 617	81 422	131 695	205 376	309 606
Ilmaisten sovellusten osuus	88,4 %	89,0 %	90,0 %	91,0 %	92,0 %	93,0 %

Sovelluskaupat ja niiden sisältämien ohjelmien sekä pelien määrät ovat merkittävä tekijä myös laitemarkkinoilla, sillä käyttöjärjestelmien ekojärjestelmät pohjautuvat niihin. Tämän voisi olettaa olevan helpottava tekijä ohjelmistokehittäjille, sillä uusien käyttöjärjestelmien tuleminen markkinoille on hankalaa, kun myynnille oleellisten kolmansien osapuolien tekemien sovellusten määrä on aluksi väkisinkin vähäinen.

2.4 Kehitystyö

Mobiilisovelluksiin tutustuessa huomaa monien niistä olevan yhden, mahdollisesti hyvinkin spesifisen asian tekemiseen tarkoitettu. Osin syynä ovat laitteiden rajalliset resurssit, mutta osin myös sovellusten käyttötarpeet (Newman 2012). Pienten ja yksinkertaisten sovellusten kysyntä houkuttelee yritysten lisäksi myös yksittäisiä sovelluskehittäjiä, sillä pieniä sovelluksia on nopeampaa kehittää kuin suuria, eivätkä ne täten vaadi valtavia resursseja.

Ohjelmien kehittäminen pienillä resursseilla saattaa tuottaa aluksi yksinkertaisia sovelluksia, joihin lisätään ominaisuuksia jälkikäteen. Tätä tukee myös sovellusten helppo päivitettävyys sovelluskauppojen kautta. Toimivien palautekanavien myötä myös käyttäjäpalautteen saaminen uusien ominaisuuksien pohjaksi on mahdollista. Kuviossa 3 on esitetty tällaisen tilanteen mukainen päivityssykli.

KUVIO 3. Sovelluskehityksen kiertokulku

3 MOBIILISOVELLUSTEN HAASTEET

Mobiililaitteiden valtava kirjo luo sovelluskehittäjälle suuren määrän haasteita. Mille käyttöjärjestelmälle sovellus tulisi kehittää vai pitäisikö niitä valita useampikin? Kuinka huomioida erilaiset laitteet, sillä puhelin/tabletti-jaottelun lisäksi molempien kategorioiden sisällä on suurta vaihtelua esimerkiksi näyttöresoluutioiden ja suorituskyvyn suhteen. Uusia laitteita julkaistaan nopealla syklillä, joten miten olisi mahdollista varautua tuleviin uudistuksiin? Kuinka huomioida käytettävyys käyttöliittymää suunniteltaessa, kun laitteita ohjataan usein pienehköä näyttöä epätarkasti sormella painaen?

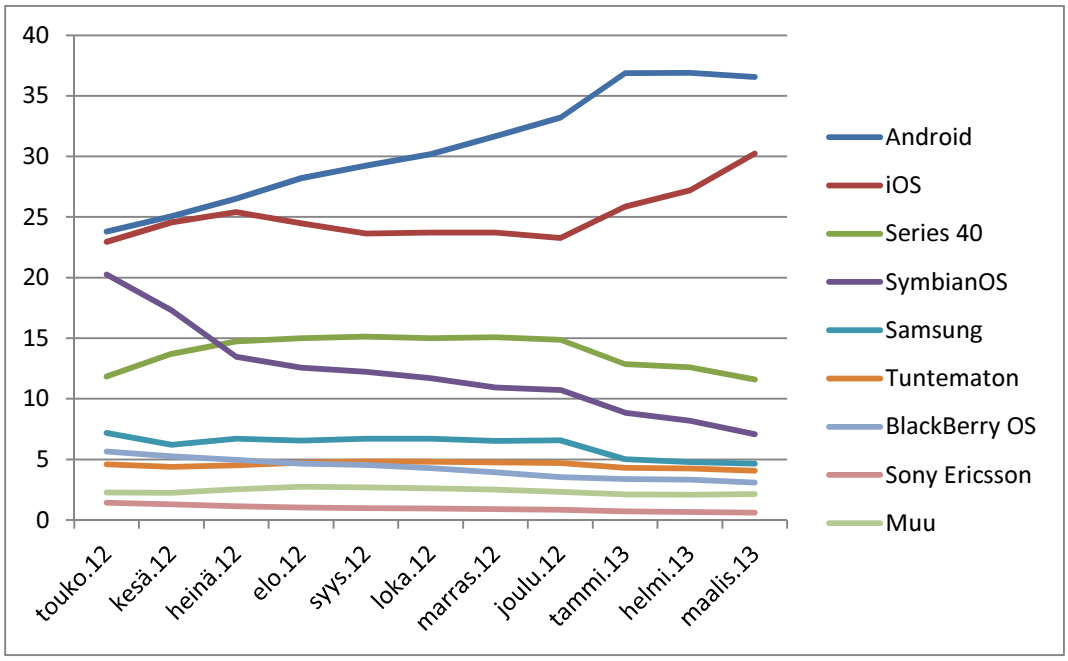
3.1 Käyttöjärjestelmät

Käyttöjärjestelmien eroavaisuudet aiheuttavat huomattavaa vaivaa mobiilisovelluksien ohjelmoinnissa. Natiivin ohjelman teko yhdellä ohjelmointikielellä usealle eri alustalle ei ole mahdollista, sillä käytettävät kielet vaihtelevat eri käyttöjärjestelmien välillä.

Android-sovellukset suositellaan ohjelmoimaan Javalla, mutta muun muassa C- ja C++-kielien käyttö on mahdollista Andoid Native Development Kitin (NDK) avulla. NDK on tarkoitettu käytettäväksi enemmän laskentatehoa tarvitsevien sovelluksen osien kanssa, joten osa sovelluksesta jää joka tapauksessa koodattavaksi Javalla. (Android Developers 2013.)

Applen iOS-käyttöjärjestelmälle ohjelmoitaessa suositeltava ratkaisu on käyttää Objective-C-kieltä, jolloin lopputuloksena on mahdollisimman nopeasti toimiva sovellus. iOS:lle on mahdollista ohjelmoida myös monia muita kieliä käyttäen, mutta Objective-C on ainoa virallisesti tuettu vaihtoehto. (Apple Inc 2012b)

C-pohjaiset ohjelmointikielet ovat muillakin käyttöjärjestelmillä suosittuja. Symbian-sovellukset käyttävät C++-kieltä (Nokia 2012) ja Windows Phone -sovelluksille suositellaan C#-kieltä, vaikkakin ohjelmointi on mahdollista myös Visual Basic -kielellä (Microsoft 2013).

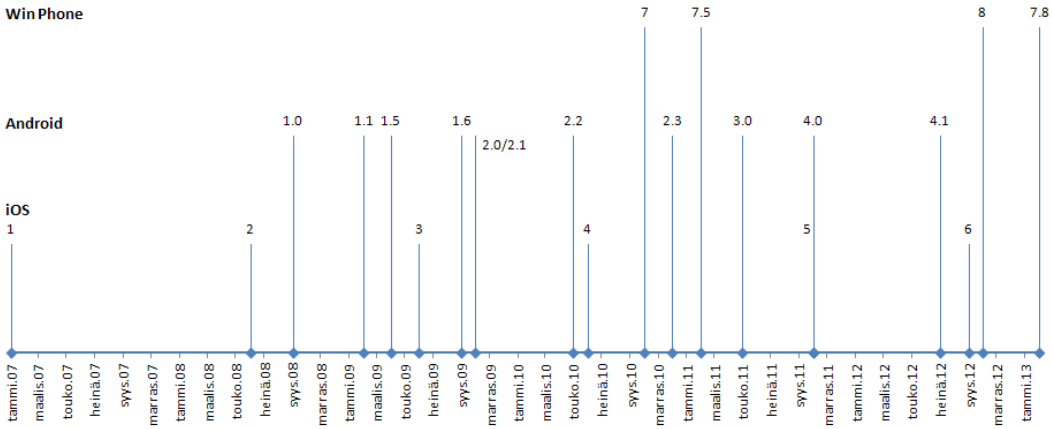


KUVIO 4. Mobiilikäyttöliittymien markkinaosuudet (StatCounter 2013)

Kuten kuviosta 4 voi havaita, StatCounter-sivuston keräämän datan mukaan Android ja iOS ovat selkeästi suosituimmat mobiilikäyttöjärjestelmät. Maaliskuussa 2013 pelkästään näiden kahden käyttöjärjestelmän osuus on noin 67 prosenttia, joten suuntaamalla sovelluksen näille alustoille saa katettua suurimman osan mobiililaitteista. Kyseisille käyttöjärjestelmille ei kuitenkaan ole mahdollista ohjelmoida samalla kielellä ja lisänä tulee vielä ulkoasuun liittyvät seikat.

Käyttöjärjestelmä vaikuttaa yleensä myös käyttöliittymäsuunnitteluun, sillä eri käyttöjärjestelmille on olemassa omat suunnitteluohjeensa, joilla sovelluksesta saadaan yleisilmeeltään ja toiminnallisuudeltaan oikeanlaisia. Esimerkiksi Applen iOS:n ohjeistus (Apple Inc 2012a) ja Androidin vastaava (Android Developers 2012) käyvät seikkaperäisesti läpi asiat, joita tulee huomioida käyttöliittymää suunniteltaessa.

3.2 Laitteiden kehitys



KUVIO 5. Käyttöjärjestelmien merkittävät versiot ([x]cube LABS 2011; [x]cube LABS 2012; Wikipedia 2013a; Wikipedia 2013b)

Yllä olevasta kuviosta 5 on nähtävissä, että mobiilikäyttöjärjestelmät saavat merkittäviä päivityksiä tiuhaan tahtiin. Suuret päivitykset tuovat tullessaan muutoksia ja uusia ominaisuuksia, jotka saattavat vaikuttaa myös laitteissa ajettaviin sovelluksiin. Ohjelmat eivät välttämättä edes toimi uudella käyttöjärjestelmäversiolla, ja tämän takia nopea päivityssykli voidaan laskea haasteeksi sovelluskehittäjille.

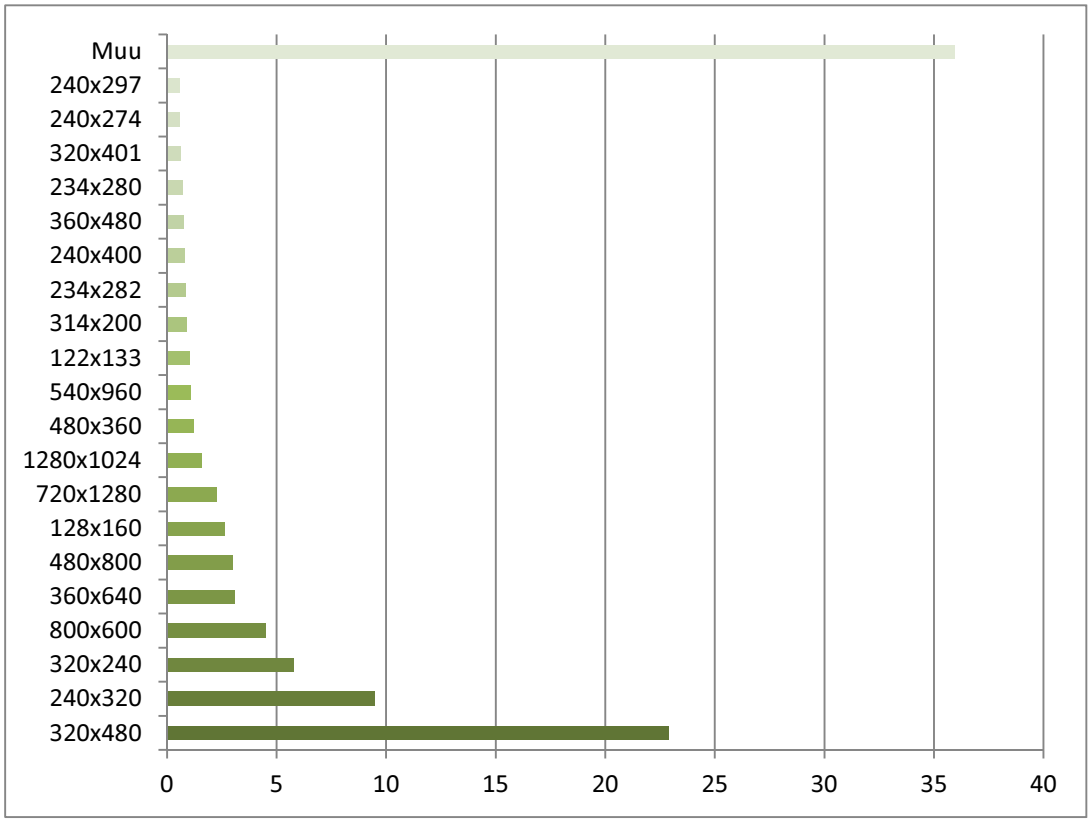
Käyttöjärjestelmien ohella itse laitteetkin kokevat uudistuksia. Yksi esimerkki tästä on Androidin versiota 3.0 ja sitä uudempia käyttöjärjestelmiä käyttävät laitteet, joista puuttuu fyysisiä nappeja vanhempiin puhelimiin verrattuna. Sovelluskehittäjien on siis pitänyt implementoida ”ActionBar”-nimikkeellä tunnettu valikkorakenne nappuloiden toiminnallisuuksien sijasta. (Main 2012.)

3.3 Näyttöjen koot ja kuvasuhteet

Mobiililaitteiden monipuolinen kirjo tekee käyttöliittymäsuunnittelusta mielenkiintoista, sillä näyttöjen koot ja kuvasuhteet vaihtelevat suuresti. Monissa laitteissa näyttö on oletuksellisesti pystyssä, jolloin kuvasuhdekin on työpöytäympäristöstä poiketen korkeampi kuin mitä sen leveys on. Suuressa

osassa laitteista on kuitenkin mahdollisuus myös vaaka-asennossa käytölle, jolloin kuvasuhde muuttuu päinvastaiseksi.

Laitteita on myös monen kokoisia aina pienistä puhelimista suurehkoihin tabletteihin. Tämä näkyy näytön fyysisessä koossa ja resoluutiossa. Eri näyttöresoluutioiden prosentuaaliset osuudet on kuvattu kuviossa 6. Asiaa sekoittaa lisäksi se fakta, ettei resoluutio aina kasva lineaarisesti näytön koon mukana.



KUVIO 6. Näyttöresoluutioiden prosentuaaliset osuudet syyskuussa 2012 (StatCounter 2012b)

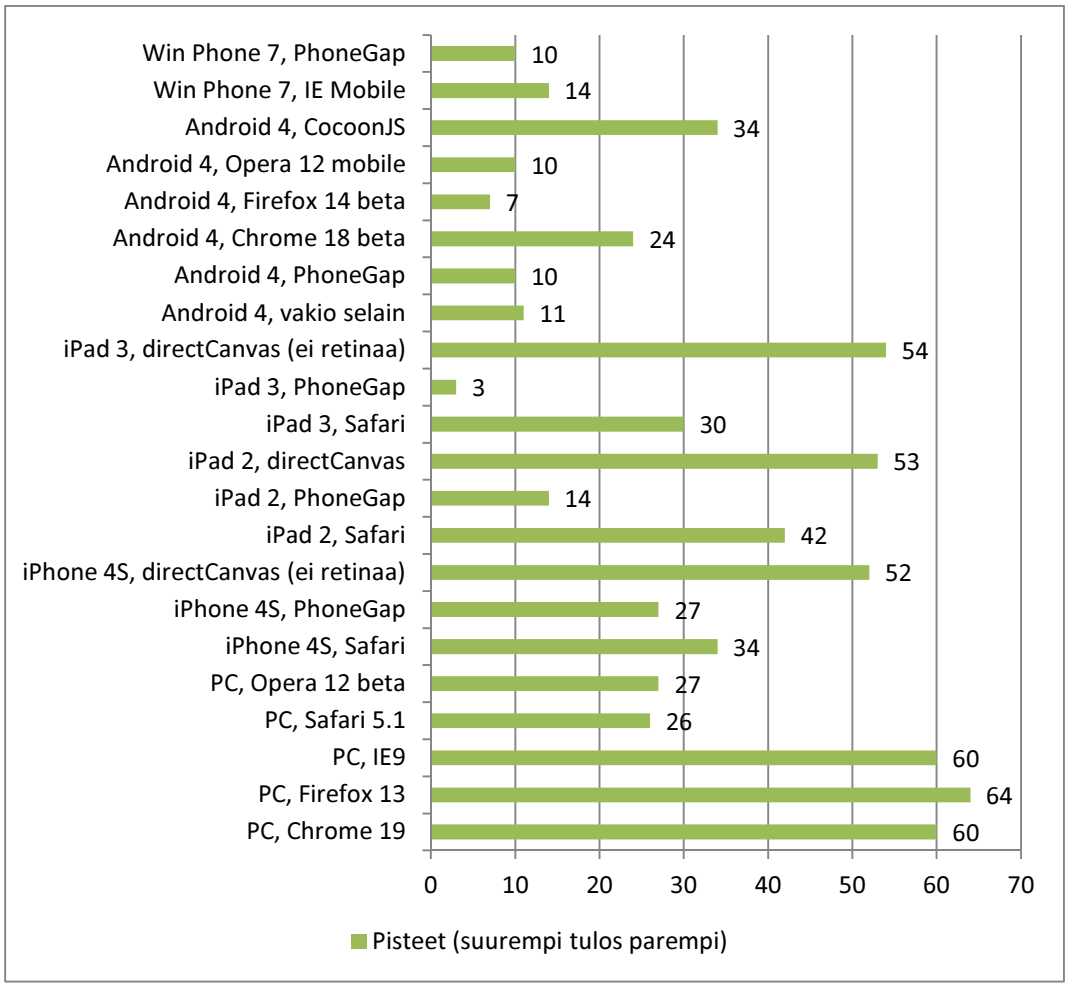
Yhteinen tekijä on joka tapauksessa pienempi kuvapinta-ala, joka on huomioitava sovellusta suunniteltaessa. Käyttäjälle ei ole mahdollista näyttää suurta tietomäärää kerralla. Tämän lisäksi painikkeet ja muut toiminnalliset elementit vaativat suhteellisen suuren koon, jotta kosketusnäytöllisillä laitteilla ohjelman käyttäminen olisi miellyttävää. Sevenval-nimisen yrityksen vuonna 2011 julkaiseman statistiikan mukaan mobiililaitteille suunnattujen internetsivujen

käyttäjistä 80 prosenttia käytti kosketusnäytöllistä laitetta (Guelle 2011). Sovellusten kohdalla osuus on oletettavasti vielä suurempi.

Web-teknologioita käytettäessä ongelma on osin ratkaistavissa erillisillä tyylitiedostoilla. CSS3:n myötä on mahdollista määritellä entistä tarkemmin eri tyylitiedosto käytettäväksi erilaisissa tilanteissa. Mobiililaitteille sisältöä tuottavia kiinnostaa etenkin orientaation ja näytön vaakaresoluution mukainen tyylitiedostojen määrittely. (Rivoal, Lie, Çelik, Glazman & van Kesteren 2012.)

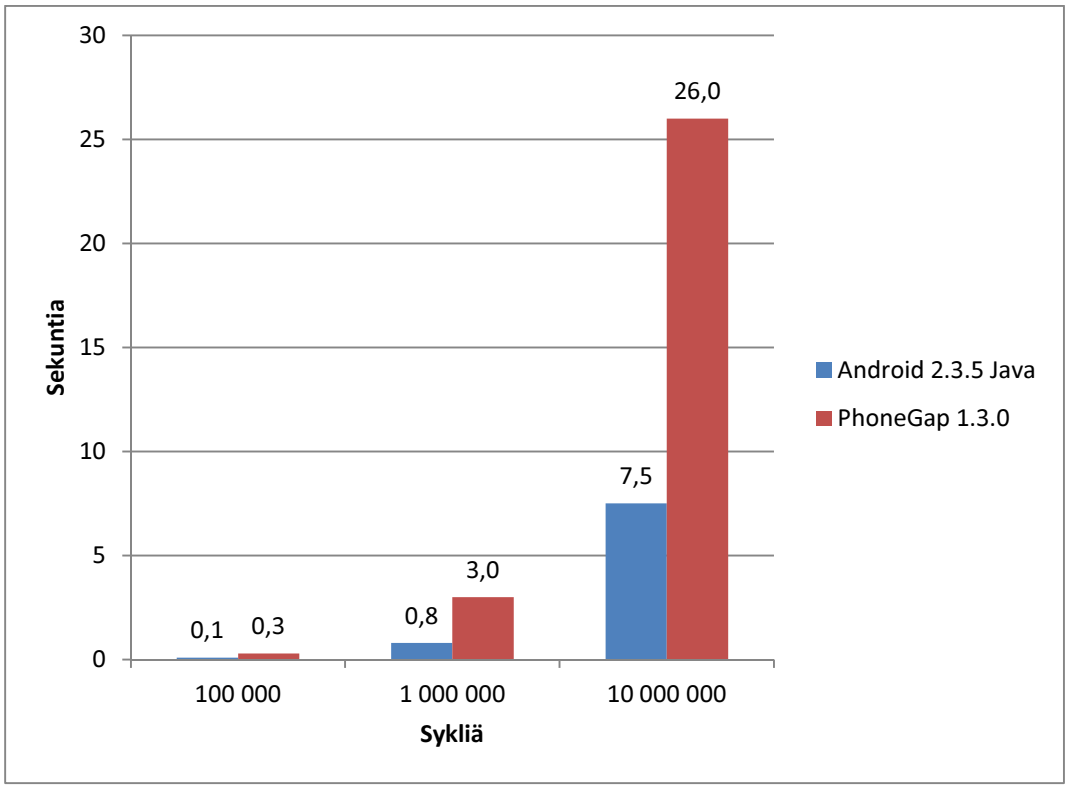
3.4 Suorituskyky

Mobiililaitteet kulkevat nimensä mukaisesti käyttäjän mukana. Tämä tarkoittaa käytännössä mahdollisimman pientä fyysistä kokoa ja akun varassa toimimista, joten tehokas ja samalla paljon virtaa kuluttava ja lämpöä tuottava tekniikka ei ole mahdollista. Suorituskyky on täten rajallista eikä suurta laskentatehoa tarvitsevia sovelluksia voida toteuttaa mobiililaitteille.



KUVIO 7. Suorituskykyvertailu (Gullen 2012)

Ashley Gullen, Scirra-nimisen yrityksen sovelluskehittäjä, on kirjoittanut yrityksen blogiin vertailun heidän oman sovelluksensa suorituskyvystä eri ympäristöissä. Kuvio 7 on koostettu vertailun yhteenvedosta, jossa suurempi pistemäärä tarkoittaa parempaa tulosta. Kuviosta on havaittavissa suurta vaihtelua suorituskyvyssä eri laitteiden ja käyttöjärjestelmien välillä, mutta myös yleinen trendi mobiililaitteiden hitaudesta työpöytätietokoneeseen nähden. Samaisesta kuviosta nähdään myös PhoneGapia käytettäessä muodostuvat suorituskykyongelmat. (Gullen 2012.)



KUVIO 8. Laskentanopeus (Pala 2012)

Kuviossa 8 on tulokset testistä, jossa PhoneGapin ja natiivin Android-sovelluksen eroja on vertailtu Margus Palan toimesta yksinkertaisella silmukassa suoritettavalla matemaattisella lausekkeella. Testi ei ole monipuolinen, mutta osoittaa kuitenkin suorituskyvyn selkeän tippumisen, kun laskenta tehdään natiivisovelluksen sijasta selaimessa JavaScript-koodilla. PhoneGap-sovellus vaatii tässä esimerkissä noin kolme kertaa enemmän laskenta-aikaa kuin natiiviohjelma. (Pala 2012.)

Yksinkertaisten sovellusten tapauksessa tällaiset erot eivät välttämättä ole merkittäviä, mutta monimutkaisemmat sovellukset alkavat erosta kärsiä. Mark Zuckerberg, Facebookin toimitusjohtaja, totesi yrityksensä suurimman virheen olleen tukeutuminen HTML5-ratkaisuihin natiivien sovellusten sijasta. Nyt yritys on siirtymässä natiivisovelluksiin. Näin suurten linjausten muutos isossa yrityksessä kielii selkeistä ongelmista uuden tekniikan kanssa. (Olanoff 2012.)

4 VAIHTOEHTOISET TOTEUTUSTAVAT

4.1 Natiivit sovellukset

Natiivit sovellukset ovat tietylle käyttöjärjestelmälle koodattuja ohjelmia, eikä niitä yleisesti ottaen kyetä käyttämään sellaisenaan toisissa käyttöjärjestelmissä (Rouse 2012). Kaikille eri alustoille oman sovelluksensa toteuttaminen on työlästä ja monesti nämä ohjelmat ovatkin vain yhdelle käyttöjärjestelmälle toteutettuja.

Natiiveilla sovelluksilla on myös selkeitä etuja. Muilla ratkaisuilla ei ole mahdollista saavuttaa samanlaista nopeutta, jonka käyttäjä havaitsee parempana käyttökokemuksena. Syynä tähän on natiivin sovelluksen suoraviivaisempi toiminta, kun itse sovelluksen ja käyttöjärjestelmän välissä ei ole selainta hidastamassa toimintaa (Rouse 2012). Lisäksi tietylle käyttöjärjestelmälle sovelluksen toteuttaminen tuottaa myös varmatoimisemman lopputuloksen, sillä sen toimintaan ei ole vaikuttamassa internetselain, eikä rajatummalla laitemäärällä testaaminen ole niin ongelmallista.

Natiivien ohjelmien jakeleminen sovelluskauppojen kautta on myös selkeä etu, sillä se on käyttäjille looginen väylä käyttötarpeiden täydentämiseen ja täten ohjelmalle on helpompaa saada tarvittavaa näkyvyyttä. Yleisesti ottaen sovelluskaupat syövät osan maksullisten ohjelmien tuloista, mutta toisaalta ne taas tarjoavat helpon tavan päivitysten jakelemiseen, vaikkakin jakelu on hitaampaa kuin puhtaasti selainpohjaisissa ratkaisuissa.

4.2 Selainpohjaiset sovellukset

Selainpohjaiset sovellukset ovat käytännössä internetsivuja, jotka pyrkivät käyttäytymään kuin natiivit sovellukset ja monesti yrittävät myös näyttää sellaisilta. Tämäntyyppisten sovellusten suurin etu on niiden käytettävyys lähes kaikilla laitteilla, joissa on internetyhteys. Asia ei ole täysin mutkaton, sillä kohdentamatta sovellusta tietylle laitekannalle, saattaa vastaan tulla erinäisiä yhteensopivuusongelmia. Verkkoselaimet eivät tunnetusti tue pilkun tarkasti

kaikkia standardeja ja esimerkiksi HTML5-tuen puute rajaa sen ominaisuuksia hyödyntävät sovellukset käyttäjän tavoittamattomiin.

HTML5:n myötä erot natiiveihin sovelluksiin ovat kuitenkin vähentyneet huomattavasti ja tulevat vähentymään vielä enemmän, kunhan HTML5-standardiin lisätään tarkat määrittelyt esimerkiksi anturitietojen käsittelemiseksi. Pelkkä standardointi ei ole vielä sovelluskehittäjän kannalta riittävä asia, vaan selaimien pitää myös tukea standardin mukaisia ominaisuuksia. Tällä hetkellä anturitietoihin on pääsy lähinnä eri käyttöjärjestelmien omien rajapintojen kautta, eikä näin ollen yksinkertaisen ja kaikkia laitteita tukevan sovelluksen luominen ole mahdollista. (Chandhok 2012.)

HTML5 tarjoaa myös mahdollisuuden offline-käyttöön, joka laajentaa web-sovellusten mahdollisuuksia. Ensimmäisellä käyttökerralla sovelluksen tarvitsemat tiedostot ladataan mobiililaitteelle, josta niitä verkkoyhteyden katketessa voidaan käyttää. Offline-ominaisuus sisältää myös tapahtumat tilan muutoksille, joita voidaan hyödyntää tapahtumakäsittelijöiden avulla. Näin esimerkiksi hetkellisen verkkokatkoksen aikana tietoa voidaan säilöä laitteen muistiin ja yhteyden taas toimiessa tallentaa tiedot palvelimelle. Myös muistin käyttöön liittyvät ominaisuudet ovat HTML5-standardin mukaisia. (Pilgrim 2010.)

Selainpohjaisille sovelluksille on kuitenkin hankalampaa saada näkyvyyttä kuin natiiveille sovelluksille. Applen AppStore tarjoaa listauksen web-sovelluksista, mutta tässäkin tapauksessa näkyvyyttä ei saa normaalien sovellusten joukossa. Sovellusta on siis markkinoitava samalla tavalla kuin verkkosivujakin. Sovelluskauppoihin pääsemättömyys ei tosin ole puhtaasti negatiivinen asia, sillä rojalteja ei tarvitse maksaa ja ohjelman päivittäminen on vielä yksinkertaisempaa, koska ohjelma ei ole tallennettuna käyttäjän laitteeseen.

4.3 Web-teknologioita hyödyntävät natiivisovellukset

Kolmas vaihtoehto on risteytys kahdesta edellisestä. Sovellukset luodaan käyttäen web-tekniikoita (HTML5, JavaScript), ja ne toimivatkin internetselaimessa. Nämä ohjelmat on kuitenkin paketoitu natiivisovelluksiksi. Ne asennetaan sovelluskauppojen kautta ja ne käynnistetään kuvakkeista, aivan kuten oikeatkin natiivisovellukset. Ohjelmat kuitenkin ajetaan verkkoselaimessa, vaikka selain itsessään onkin piilotettu käyttäjältä.

Käytännössä tämä ratkaisu vaatii sopivan frameworkin käyttämistä, joista PhoneGapin ja Sencha Touchin yhdistelmä on toimiva ja moniin ratkaisuihin sopiva. Framework pitää huolen sovelluksen paketoimisesta tietylle laitealustalle sopivaksi, niin asennuspaketin luomisen kuin laitteen ominaisuuksien hyödyntämisen osalta. Ohjelmoijan ei siis tarvitse itse huolehtia käyttöjärjestelmäkohtaisten laiterajapintojen eroavaisuuksista ja silti mobiililaitteen kaikki anturit ja muut vastaavat ominaisuudet ovat käytettävissä. Selkeästi suurin hyöty tässä toteutustavassa onkin juuri se, että samaa koodia voidaan käyttää usealla eri käyttöjärjestelmällä.

Sencha Touch liittää mukaan myös natiivin ohjelman tunnetta kosketuseleiden ja eri käyttöjärjestelmille suunniteltujen CSS-tyylitiedostojen avulla. PhoneGap paketoi sovelluksen, jolloin se voidaan jakaa sovelluskauppojen kautta ja suorittaa normaalin ohjelman tavoin. Tämänkaltaisilla ratkaisuilla ohjelma näyttää ja tuntuu natiivilta eikä sitä tarvitse myöskään ladata internetistä joka käyttökerralla, kuten web-sovelluksien tapauksessa. Ratkaisulla on myös varjopuolensa, sillä se on hitaampi kuin aidot natiivisovellukset. Yksinkertaisempien käyttötarpeiden kanssa tämä ei muodostu ongelmaksi, mutta esimerkiksi isompia datamääriä käsiteltäessä hitaus alkaa näkyä.

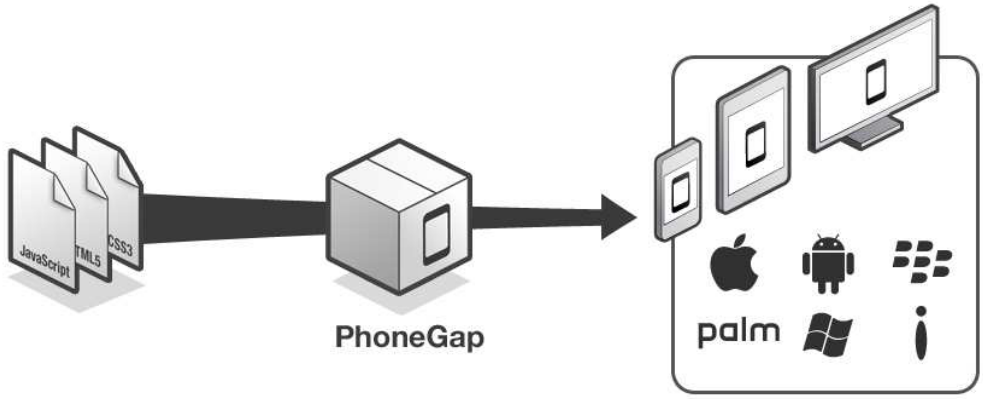
Web-teknologioihin perustuvien natiivisovellusten hyötynä on myös laaja tekninen osaaminen, vaikka JavaScript-kirjastojen käyttöönotto vaatiikin uuden opettelemista. Oppimiskynnys on kuitenkin huomattavasti matalampi kuin useiden eri ohjelmointikielien sisäistäminen.

5 PHONEGAP

PhoneGap on avoimen lähdekoodin framework, jonka avulla voi luoda useille eri käyttöjärjestelmille natiiveja sovelluksia käyttäen HTML, CSS ja JavaScript teknologioita. Adobe osti Nitobi Software -nimisen yrityksen, joka alun perin kehitti PhoneGapin. Samaan aikaan lähdekoodit lahjoitettiin Apache Software Foundationille, joten PhoneGap on nykyisin Apache Cordovan distribuutio. Tällä taataan PhoneGapin kehitys, mutta samalla myös sen pysyminen ilmaisena ja lähdekoodiltaan avoimena. (Adobe Systems Inc 2012b.)

5.1 Ideologia

PhoneGapin ajatuksena on tuoda natiivisti toimivien sovellusten mahdollisuudet web-ohjelmoijien saataville. HTML5-, CSS3- ja JavaScript-tekniikoilla luodut ohjelmat paketoidaan natiivisovelluksiksi eri käyttöjärjestelmille kuvion 9 mukaisesti. Näin sovellusta ei tarvitse ajaa käyttäjän näkökulmasta internetselaimessa, vaikka se oikeasti selaimessa suoritetaankin. Selain piilotetaan käyttäjältä, ja kun ohjelma käynnistetään pikakuvakkeesta, niin käyttäjälle toiminta vaikuttaa täysin natiivin sovelluksen kaltaiselta. Natiivipaketointi tarjoaa myös mahdollisuuden ohjelman jakelemiseen sovelluskauppojen kautta sekä ohjelman suorittamisen ilman verkkoyhteyksiä. PhoneGap ei sisällä ohjelmointiympäristöä (IDE, *integrated development environment*), vaan se on pelkkä sovelluskehys.



KUVIO 9. PhoneGap

Natiivisovellusten yksi etu on mobiililaitteen antureiden ja vastaavien ominaisuuksien tehokas hyödyntäminen. HTML5 yrittää tarjota mahdollisuuden näiden käyttämiseen, mutta käytännössä ohjelmoijan vastuulle jää käyttää käyttöliittymäkohtaisia laiterajapintoja. PhoneGapin avulla sovelluskehittäjän ei tarvitse huolehtia eri rajapinnoista, riittää kun käyttää sen tarjoamia JavaScript-funktioita. Sovelluksen paketoinnin yhteydessä PhoneGap ottaa käyttöön oikeat laiterajapinnat kohteena olevan käyttöjärjestelmän perusteella.

Näiden lisäksi PhoneGap mahdollistaa myös Ajax-kyselyiden tekemisen ulkopuolisille palvelimille. Nämä JavaScriptillä tehdyt niin sanotut ”cross-domain”-kyselyt ovat normaalisti estetty selaimissa tietoturvasyistä. Asian tärkeys korostuu hybridisovelluksessa entisestään, sillä ohjelma toimii mobiililaitteessa, eikä se siis ole minkään internetin domainnimen alaisuudessa. Täten esimerkiksi erilaisten rajapintojen käyttämiseen tarvitaan jokin ratkaisu. Mobiililaitteiden vähäisen suorituskyvyn takia raskaammat prosessoinnit on hyvä suorittaa palvelimilla, joten monipuoliset kytkeytymismahdollisuudet ovat etu.

5.2 Kehitys

PhoneGap on uudehkoa tekniikaa, joka kehittyy jatkuvasti. Uusia, kehittäjälle hyödyllisiä ja ohjelman tekoa yksinkertaistavia ominaisuuksia on julkaistu aika ajoin.

Viimeisimpänä näistä on PhoneGap Build -pilvipalvelu, jonka avulla sovelluksen voi kääntää haluamalleen alustalle. Tiedostot voi lähettää suoraan palveluun tai käyttää sitä yhteistyössä oman GIT- tai SVN-versionhallintatyökalun kanssa (Adobe Systems Inc 2012a). Palvelusta hyötyy etenkin iPhone-sovellusta tehtäessä, sillä kehitystyössä ei tarvitse käyttää Mac-tietokonetta. Windows- ja Linux-ympäristöissä iOS-sovelluksen kääntäminen ei normaalisti ole mahdollista. Pilvipalveluakaan käytettäessä kuluilta ei voi täysin välttyä, sillä iOS Developer Program -jäsenyys on pakollinen menoerä Applen laitteille sovelluksia kehitettäessä. (Hanks 2011.)

6 SENCHA TOUCH

6.1 Ideologia

Sencha Touch on JavaScript-pohjainen mobiilisovelluksiin suunnattu framework, joka sisältää käyttöliittymä- ja datakirjastot. Sen avulla on mahdollista rakentaa web-pohjaisia sovelluksia, jotka näyttävät ja tuntuvat natiivisovelluksilta. Se tukee kosketuselkeitä ja sisältää animointeja esimerkiksi sivunvaihtojen käyttökokemuksen parantamiseksi.

Sencha Touch käyttää HTML, CSS ja JavaScript teknologioita, joista etenkin JavaScript on olennainen. Totutusta web-kehityksestä poiketen sovellusten toteutuksessa käytetään paljon Sencha Touchin omia funktioita sekä käytännössä pakotetaan käyttämään MVC-mallia (Model, View, Controller). Senchan toinen suosittu tuote, Ext JS, pohjautuu samaan tekniseen ratkaisuun kuin Touch (O´Connor 2012). Tämä on huomattava etu etsittäessä esimerkkejä internetistä, sillä vaihtoehdot eivät rajoitu pelkästään Sencha Touchia käsitteleviin sivuihin. Touchin ja Ext JS:n sovelluslogiikka on myös yhtenäinen, joten ohjelman muuntaminen Touchilla toteutetusta mobiilikäyttöliittymästä Ext JS:llä toteutettuun työpöytäversioon on melko suoraviivaista.

Touch sisältää suuren määrän valmiita komponentteja, joista sovelluksen ulkoasu kootaan. Komponentit esitellään JavaScript-koodissa eräänlaisina olioina ja nämä oliot määritellään näytettäväksi halutussa näkymässä. Ohjelmoijan ei siis tarvitse kirjoittaa HTML:ää ollenkaan. Komponentit on myös muotoiltu mobiilisovelluksiin soveltuviksi ulkonäkönsä ja käytettävyytensä puolesta. (Sencha Inc 2012.)

6.2 Kehitys

Suosittua alustaa on uudistettu jatkuvasti. Uusia ja hyödyllisiä ominaisuuksia on julkaistu taajaan ja viimeaikaisista muutoksista suurin on tullut version kaksi myötä. Tuore versio toi tullessaan niin sanotun Native Packaging -tuen, eli ohjelman kääntämisen natiiviksi sovellukseksi. PhoneGap tukee useampaa

laitealustaa, mutta Touchin eduksi voidaan kuitenkin lukea mahdollisuus kääntää iOS-sovellukset myös Windows-tietokoneella. PhoneGapia käytettäessä käännöksen joutuisi tekemään heidän pilvipalvelussaan. (Nguyen 2012.)

Toinen uusi asia on metodit laiterajapintojen käyttämiseksi. Mobiililaitteiden antureiden ja vastaavien laitteiden hyödyntäminen ei ole ainakaan vielä yhtä laaja-alaista kuin PhoneGapia käytettäessä, sillä Touch tukee ainoastaan iOS- ja Android-käyttöjärjestelmäpohjaisia laitteita ja niilläkin mahdollisuudet ovat rajallisemmat. Tähän on oletettavissa kuitenkin muutoksia uusien päivitysten myötä.

7 SOVELLUS

7.1 Ideologia

Sovelluksen tarkoituksena oli tutkia yksinkertaisinta mahdollista tapaa toteuttaa julkisen liikenteen reittiopas mobiililaitteille, rajaamatta sen käyttöä toteuttamalla sovellus vain yksittäiselle käyttöjärjestelmälle. Web-teknologioiden käyttö olisi myös etu, sillä CGI:n muut reittioppaat on toteutettu niitä käyttäen ja vanhojen lähdekoodien hyödyntäminen helpottaisi kehitystyötä. Selainpohjainen ratkaisu ei tullut kuitenkaan kysymykseen, sillä tarkoituksena oli luoda natiivin sovelluksen kaltainen käyttökokemus ja mahdollistaa hienompien tekniikoiden käyttäminen, joista esimerkiksi paikannus toisi suurta lisäarvoa sovellukselle.

Näillä kriteereillä sovellusta alettiin toteuttaa web-tekniikoilla, ajatuksena paketoida se lopulta natiiviksi sovellukseksi. Sovellus itsessään tarjoaisi mahdollisuuden hakea reittejä ovelta ovelle -periaatteella maanlaajuisesti, joukkoliikennettä hyödyntäen. Reitittämiseen käytettäisiin Matka.fi:n avointa rajapintaa, jolloin sovelluksen osuudeksi jää käyttöliittymän lisäksi rajapintojen käyttöön ja datan visualisointiin liittyvien asioiden toteutus.

7.2 Teknologiat

Sencha Touchia on käytetty käyttöliittymän luomiseen eri komponentteineen. Sen avulla ohjelman ulkoasu ja käyttökokemus soveltuvat kosketusnäytöllisille mobiililaitteille. Käyttöliittymän lisäksi Touchin osuuteen kuuluu iso osa datan käsittelystä, sillä siihen on rakennettuna tähän tarkoitukseen soveltuvia olioita. Näitä ovat esimerkiksi XML-lukijat ja -kirjoittajat sekä dataproxyt ja -säiliöt.

PhoneGapin avulla kierretään internetselaimien käyttämä niin sanottu ”same origin policy” –tietoturvakonsepti, jotta dataa saadaan siirrettyä XML-muotoisena Ajax-kyselyillä sovelluksen ja palvelimella toimivan Matka.fi:n API:n välillä. Lisäksi PhoneGap paketoi selainsovelluksen natiiviksi.

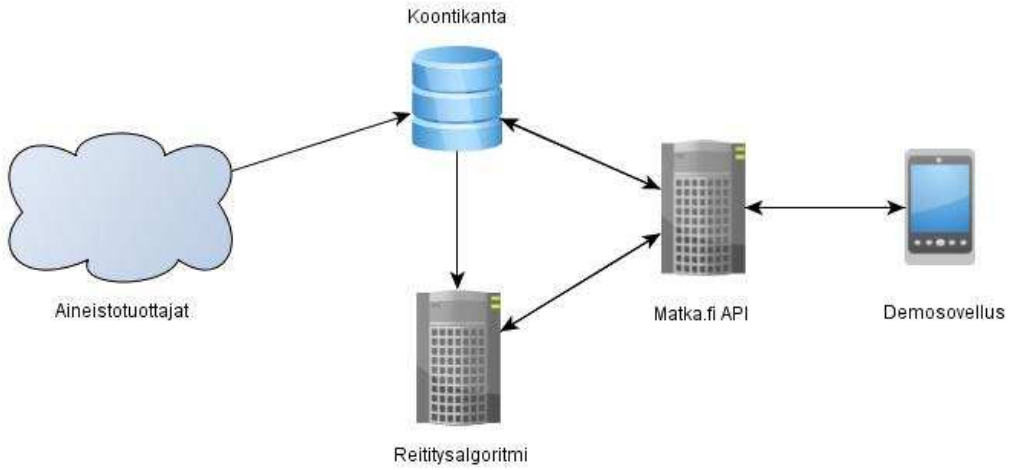
Kehitystyössä on käytetty NetBeans-ohjelmointiympäristöä, joka on NBAndroid-liitännäisen avulla yhdistetty AndroidSDK:n kanssa, jonka tehtävä on huolehtia

esimerkiksi mobiililaitteen kanssa kommunikoimisesta. Andoidin ohjelmistokehitysympäristö tarjoaa myös mahdollisuuden emulaattorin käyttämiseen fyysisen laitteen sijasta, mutta emulaattorin hitauden takia mobiililaitteen käyttäminen kehityskäytössä on miellyttävämpi ratkaisu.

7.3 Taustajärjestelmät

Tähän opinnäytetyöhön liittyvä sovellus tarkoittaa ainoastaan mobiilisovellusta, joka sellaisenaan ei olisi vielä toimiva. Sovelluksen taustalla tarvitaan ratkaisuja aikataulutietojen keräämiseksi, tallentamiseksi ja hyödyntämiseksi. Suuren tietomäärän vuoksi se säilytetään palvelinklusterissa sijaitsevassa tietokannassa.

Mobiililaitteeseen tallennettavaksi liian suuren datamassan lisäksi reititysalgoritmi olisi liian raskas tehottomalle laitteistolle. Näistä syistä dataa hyödynnetään rajapinnan avulla. Vastaavaa ratkaisua käytetään myös monissa muissa reittiopassovelluksissa, joita sovelluskehittäjät ovat toteuttaneet esimerkiksi Tampereen ja pääkaupunkiseudun julkisen liikenteen käytön helpottamiseksi.



KUVIO 10. Taustajärjestelmien ja sovelluksen hierarkiset suhteet

Kuvio 10 esittää taustajärjestelmien ja sovelluksen hierarkisia suhteita. Yksinkertaistettuna aineistotuottajien palvelimilta noudetaan ja prosessoidaan

tiedot Koontikantaan. Matka.fi:n API toimii välikätenä sovelluksen, Koontikannan ja reititysalgoritmin välillä.

7.3.1 Koontikanta

Koontikanta on CGI:n Liikennevirastolle toteuttama ja ylläpitämä tietokanta, johon tallennetaan eräajoina Suomen eri joukkoliikenneoperaattoreiden aikataulutiedot niiden esittämistä ja reitin neuvontaa varten. Tietokannan sisältämää dataa käytetään Matka.fi-palvelun taustalla.

Käytännössä koontikanta on MySQL-tietokanta, joka koostuu 30 taulusta. Taulut sisältävät kaiken tarpeellisen tiedon, josta reitit voidaan koostaa. Reitit eli linjat kulkevat pysäkkien kautta tietyssä järjestyksessä. Linjoja liikennöidään tiettyinä päivinä ja tiettyinä kellonaikoina ennalta määriteltyjen operaattorien toimesta tietyn tyypisellä kalustolla. Lähtöihin saattaa sisältyä myös rajoitteita ja selitteitä, ja lisäksi pysäkeillä voi olla esimerkiksi eri vaihtoaikoja.

Koontikanta kokoaa nimensä mukaisesti eri lähteistä useassa eri formaatissa saatavan aikatauludatan yhtenäiseen muotoon, yhdestä paikasta saatavaksi. Se kattaa tällä hetkellä bussien, junien, paikallisliikenteen ja sisämaan lentoliikenteen aikatauluja ja reittejä seuraavilta toimijoilta ja alueilta:

- Kaupunkiseutujen paikallisliikenne
    - Pääkaupunkiseutu: Helsinki, Espoo, Kauniainen, Vantaa, Kerava, Kirkkonummi
    - Turku ja Oulu seutukuntineen
    - Tampere, Joensuu, Kotka, Lappeenranta, Mikkeli, Jyväskylä, Lahti, Kuopio, Hyvinkää, Hämeenlinna, Imatra, Pori, Seinäjoki, Varkaus
    - VALLU-tietokannan sisältämä vakiovuoroliikenne
- Kaukoliikenne
    - VR:n junat
    - Valtion luparekisterin sisältämä pika-, erikoispika- ja vakiovuoroliikenne
- Yhteyslautat
- Sisämaan lentoliikenne
    - Finavian lentoasemien (yli 20 asemaa) sisämaan lennot.

Tietokannassa oli perjantaina 24.8.2012 esimerkiksi

- 76 458 pysäkkiä
- 34 596 lähtöä samaiselle päivälle.

7.3.2 Matka.fi API

Matka.fi:hin kuuluu osana myös avoin rajapinta. Rajapinnan tarkoituksena on mahdollistaa aikataulutiedon käyttäminen kolmansien osapuolien sovelluksissa.

Rajapinta on toteutettu PHP-ohjelmointikielellä, ja se toimii CGI:n palvelimilla. Periaatteessa se on web-sivu, jota käytetään HTTP-protokollan mukaisilla GET-pyynnöillä. Rajapinnan verkko-osoitteen yhteydessä lähetetään tarvittavat parametrit, joiden perusteella API koostaa vastauksen tarpeen mukaisesti tietokantaa ja reititysalgoritmia hyödyntäen. Lopulta vastaus palautetaan kyselijälle XML-muotoisena. Kyseessä on siis niin sanotun REST (Representational State Transfer) -arkkitehtuurimallin mukainen rajapinta.

Parametreilla välitetään tunnistautumistiedot, pyynnön tyyppi sekä pyyntöön liittyvät tarvittavat tiedot, kuten koordinaatit. Rajapinnan tarkempi määrittely on tarjolla Matka.fi:n kehittäjäsivustolla (CGI 2012).

Sovelluksessa rajapintaa käytetään etsimään sijainteja käyttäjän antaman hakusyötteen perusteella (geokoodaus). Sijaintien tiedoista poimitaan koordinaatit, joita käytetään reitityksessä käyttäjän valitseman ajankohdan kanssa. Näiden lisäksi API osaa muuntaa koordinaatit osoitteiksi (käänteinen geokoodaus) ja etsiä aikatauludataa useilla eri kriteereillä.

7.4 Toteutus

Vaikka sovellus onkin käytännössä web-sivu, poikkeaa sen toteutus totutusta melko paljon. Index.html-tiedosto sisältää ainoastaan viittaukset tarvittaviin JavaScript- ja CSS-tiedostoihin sekä tapahtumakuuntelijan, joka kutsuu pääohjelman launch()-funktiota, kun laite on saanut ladattua sovelluksen muistiinsa.

Pääohjelman launch()-funktio on yksinkertainen. Se alustaa Viewport-olion, joka on tärkeä osa Sencha Touchin käytön yhteydessä suositeltavaa ohjelmarakennetta.

Viewport-olion ominaisuuksissa määritellään ohjelman rakenne, eli tässä tapauksessa sen kolme erillistä välilehteä. Tämä osuus on esitetty kuviossa 11.

```
app.views.Viewport = Ext.extend(Ext.TabPanel, {
    fullscreen: true,
    layout: 'card',
    cardSwitchAnimation: 'slide',
    ui: 'light',
    initComponent: function() {
        //put instances of cards into app.views namespace
        Ext.apply(app.views, {
            routeSearch: new app.views.RouteSearch(),
            showRoutes: new app.views.ShowRoutes(),
            showLegs: new app.views.ShowLegs()
        });

        //put instances of cards into viewport
        Ext.apply(this, {
            items: [
                app.views.routeSearch,
                app.views.showRoutes,
                app.views.showLegs
            ]
        });

        app.views.Viewport.superclass.initComponent.apply(this, arguments);
    }
});
```

KUVIO 11. Viewport-olion määrittely

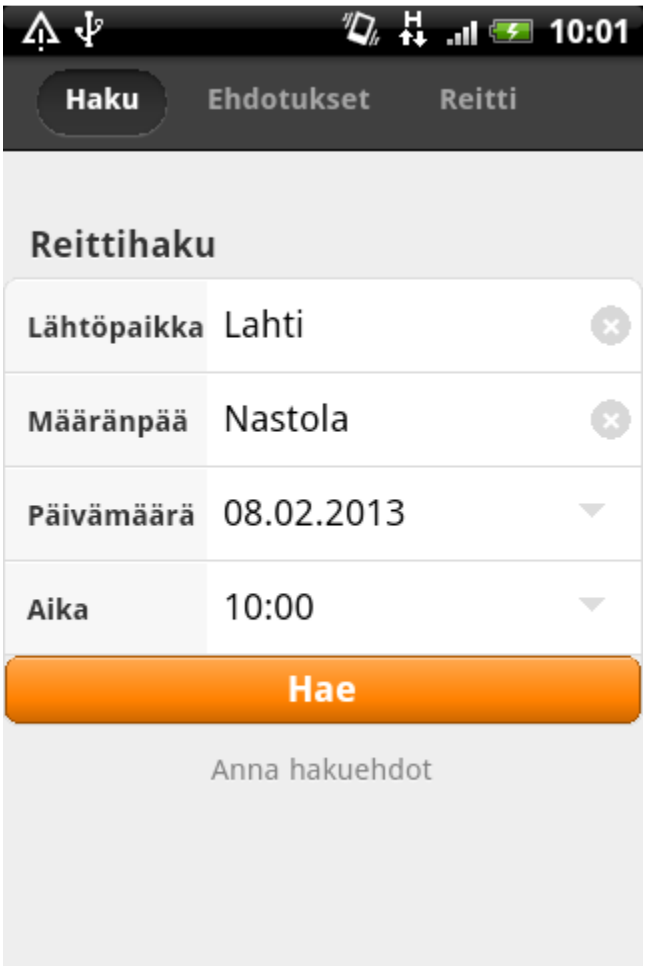
Ohjelman käyttämät MVC-mallin (Model-View-Controller) mukaiset näkymät (views) on määritelty näytettäväksi välilehtinä, tai kortteina, kuten Sencha Touchissa asia ilmaistaan. Korttien sisältö on koodattu käytetyn frameworkin mukaisesti, joten sekin on perinteisestä JavaScriptistä poikkeavaa.

```
app.views.RouteSearch = Ext.extend(Ext.Panel, {
    id: 'routeSearchForm',
    title: Dictionary['routeSearchFormTabTitle'],
    layout: 'fit',
    items: [{
        xtype: 'fieldset',
        title: Dictionary['routeSearchFormTitle'],
        instructions: Dictionary['routeSearchFormInstructions'],
        defaults: {
            xtype: 'textfield',
            useClearIcon: true,
            labelWidth: '32%'
        },
        items: [
            {
                id: 'fromInput',
                name: 'from',
                label: Dictionary['routeSearchFormLabelFrom']
            },
            {
                id: 'toInput',
                name: 'to',
                label: Dictionary['routeSearchFormLabelTo']
            },
            {
                xtype: 'datepickerfield',
                id: 'dateInput',
                name: 'date',
                label: Dictionary['routeSearchFormLabelDate'],
                value: new Date(),
                picker: {
                    yearFrom: new Date().getFullYear()
                }
            },
```

KUVIO 12. Osa hakulomakkeen lähdekoodista

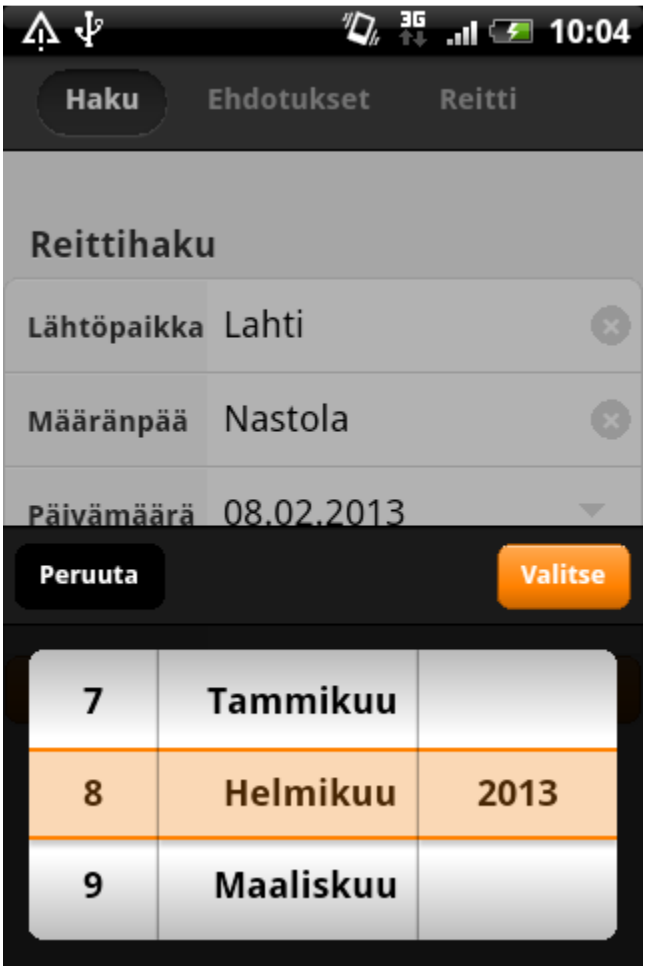
Yllä olevassa kuviossa 12 on osa hakulomakkeen lähdekoodista, jossa sille on määritelty sen ulkoasuun liittyvät asiat sekä tarvittavat elementit. Tiedosto sisältää kuviossa näkyvien asioiden lisäksi loppujen elementtien määrittelyt sekä kaksi tapahtumakäsittelijää, joista toinen liittyy välilehden aktivoitumiseen ja toinen hakupainikkeen painallukseen.

Kuvion 13 mukainen hakulomake näytetään käyttäjälle heti ohjelman avautuessa, ja sille on mahdollista palata myöhemmässäkin vaiheessa. Lomakkeelta löytyy kentät lähtöpaikalle, määränpäälle, päivämäärälle ja kellonajalle. Päivämäärä ja kellonaika ovat oletuksena sovelluksen käynnistämisajankohdan mukaiset.



KUVIO 13. Sovelluksen alkunäkymä lähtöpaikka ja määränpää syötettynä

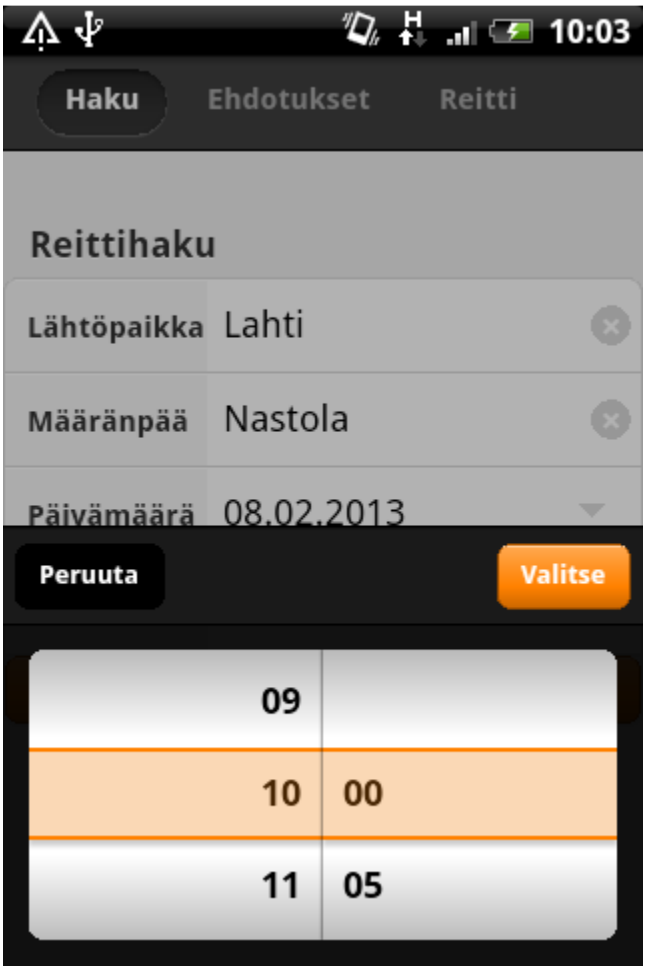
Käyttäjän on mahdollista muuttaa päivämäärä ja kellonaika haluamikseen. Tätä varten sovelluksessa on kosketusnäyttöystävälliset pyöritettävät rullat, joilla halutut hakuehdot valitaan. Esimerkkinä kuviossa 14 näytetään päivämäärän valinta.



KUVIO 14. Päivämäärän valinta

Päivämäärän valintaa varten Sencha Touch tarjoaa valmiin elementin, joka on lokalisoitavissa tarpeiden mukaan. Lisäksi se on alustettavissa tarvittavalle aikavälille. Tämän sovelluksen tapauksessa tätä on hyödynnetty vuoden valinnan osalta, sillä Matka.fi:n rajapinta tarjoaa aikatauluja noin kuukaudeksi eteenpäin.

Valinta tapahtuu pyörittämällä rullia ylös- tai alaspäin kosketusnäytöllä sormea liikuttaen. Ratkaisu on helppokäyttöinen verrattuna esimerkiksi kolmeen erilliseen pudotusvalikkoon tai kalenteriin, jossa päiviä kuvaaviin laatikoihin voisi pienellä näytöllä olla hankalaa osua.



KUVIO 15. Kellonajan valinta

Kellonaikaa varten Sencha Touchin sovelluksen käyttämästä versiosta ei löydy valmista ratkaisua, mutta ongelma on kierretty hyödyntämällä internetissä jaettua valmista JavaScript-laajennusta. Toimintaperiaatteeltaan se on päivämäärävalintaa vastaava, kuten kuviosta 15 on nähtävissä.

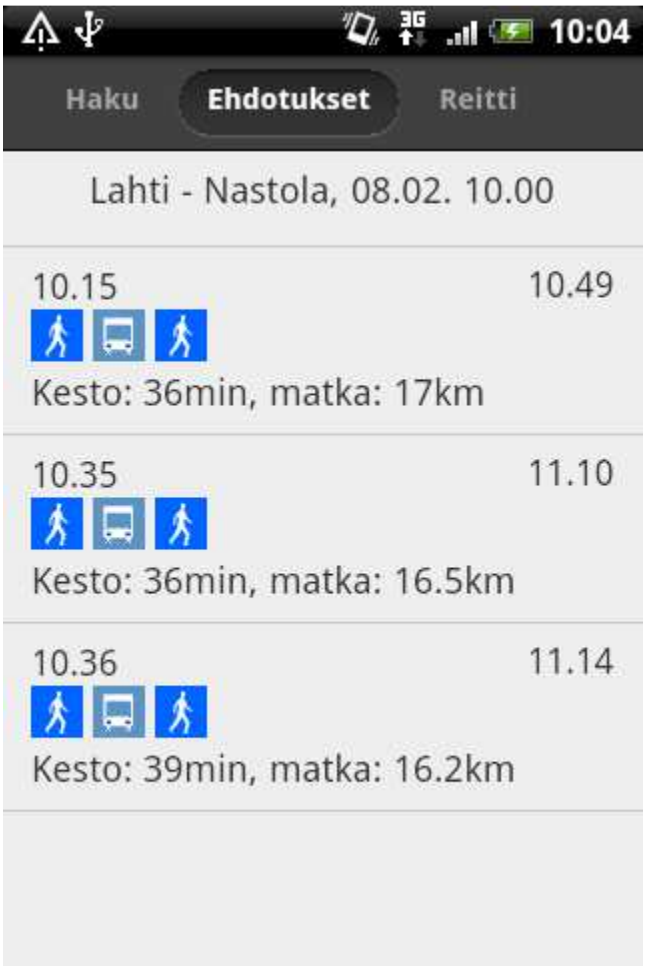
Kun käyttäjä on valinnut sopivat hakukriteerit, reittiehdotusten etsintä aloitetaan Hae-painikkeella. Painiketta painaessa sovellus tallentaa hakukriteerit ”SearchParams”-mallin tyyppiseen varastoon (store), josta ne ladataan takaisin käyttöön käyttäjän palatessa lomakkeelle. Tämän jälkeen sovellus käyttää Matka.fi:n rajapinnan geokoodaustoimintoa lähtöpaikan ja määränpään etsimiseksi.

Kun sijainnit löytyvät, koostetaan niiden koordinaateista sekä muista hakukriteereistä reittihakupyyntö API:lle. Rajapinta palauttaa XML-muotoisen vastauksen, josta parsitaan tarpeelliset tiedot. Nämä tiedot tallennetaan omiin varastoihinsa, jaoiteltuna reitteihin ja reittien etappeihin.

```
23      {
24          xtype: 'list',
25          store: app.stores.Routes,
26          width: '100%',
27          itemTpl : new Ext.XTemplate(
28              '<tpl if="this.printDates(departureDate, arrivalDate)">',
29                  '<span>{departureDate} {departureTime}</span>',
30                  '<span class="arrival">{arrivalDate} {arrivalTime}</span>',
31              '</tpl>',
32              '<tpl if="this.printDates(departureDate, arrivalDate) == false">',
33                  '<span>{departureTime}</span><span class="arrival">{arrivalTime}</span>',
34              '</tpl>',
35              '<div>',
36              '{routeId:this.printTransportTypes}',
37              '</div>',
38              '<div>',
39              Dictionary["showRoutesDuration"] + ' {duration:this.formatDuration}, ',
40              Dictionary["showRoutesDistance"] + ' {distance}km</div>',
41              {
42                  printDates: function(departureDate, arrivalDate){
43                      var date = app.stores.SearchParamsStore.getDate('d.m.');
44
45                      if (date !== departureDate || date !== arrivalDate)
46                          return true;
47                      return false;
48                  },
```

KUVIO 16. Osa Ehdotukset-välilehden lähdekoodista

Tämän jälkeen reittiehdotukset näytetään käyttäjälle Ehdotukset-välilehdellä. Kyseisellä välilehdellä käytetään hakulomakkeesta poiketen myös mallipohjaa, jonka mukaisesti kaikki reittivaraston tietueet esitetään automaattisesti sovelluksessa. Mallipohja on määritelty kuviossa 16 Ext.XTemplate-tyyppisenä. Varasto sisältää siis tietyn tyyppisiä olioita, joiden tiedot näytetään mallipohjaan määritellyllä tavalla. Tässä tapauksessa jokainen reittiolio tulostetaan omana rivinään ja jokaisella rivillä on kerrottu tärkeimmät reittiin liittyvät tiedot.



KUVIO 17. Ehdotukset-välilehti

Lopputuloksena on nähtävissä kuvion 17 mukainen näkymä, jossa esitellään kolme reittihakutulosta yksinkertaisessa muodossa, näyttäen vain tärkeimmät tiedot, joiden perusteella käyttäjä voi tehdä reittivalintansa. Yksittäisen reittiehdotuksen tarkemmat tiedot käyttäjä näkee valitsemalla haluamansa reitin listalta.

KUVIO 18. Reitti-välilehti

Kuviossa 18 esitetyllä Reitti-välilehdellä näytetään yksittäisen reittiehdotuksen tarkka rakenne. Ehdotus on jaoteltu eri kulkumuodoilla liikuttaviin osiin eli etappeihin, joista kerrotaan kulkumuoto, lähtöaika, kesto ja matkan pituus. Teknisesti Reitti-välilehden rakenne on hyvin samankaltainen Ehdotukset-välilehden kanssa. Sovelluksen yläosasta löytyvällä navigaatiolla voi missä tahansa vaiheessa palata hakulomakkeelle tai siirtyä Ehdotukset- ja Reitti-välilehtien välillä.

Sovellus on paketoitu Phonegapin avulla natiiviohjelmaksi, joka tässä tapauksessa tarkoittaa Android-puhelimeen asentuvaa apk-tiedostoa. Sovellusta olisi mahdollista myös jaella Google Playn kautta, mutta käytännössä se on vain siirretty datakaapelilla testilaitteelle. Testeissä on käytetty HTC Wildfire S -

puhelinta, jonka heikohko suorituskyky mahdollistaa riittävän kriittisen arvostelun teknisien ratkaisuiden osalta.

7.5 Tekniikan tarjoamat mahdollisuudet

Nykyisessä muodossaan sovellus ei hyödynnä läheskään kaikkia tekniikan tarjoamia mahdollisuuksia. Sovelluksesta puuttuu esimerkiksi paikkatiedon hyödyntäminen, vaikka mobiililaitteiden kulkiessa käyttäjiensä mukana sijaintitiedon hyödyntäminen olisi helppoa ja hyödyllistä.

GPS-paikannuksen avulla olisi mahdollista seurata etenemistä reaaliajassa piirtäen puhelimen sijaintia karttapohjalle. Tähän olisi mahdollista lisätä myös ohjeistus kävelyosuuksilla navigoimiseksi. Karttapohjan sijaan reitin etenemistä voisi seurata myös reitille kuuluvien pysäkkien listalla. Näin käyttäjä näkisi, mitkä pysäkit on jo ohitettu, mitä seuraavaksi on vastaan tulossa ja milloin on oikea hetki poistua kyydistä. Reaaliaikaisen seurannan etuna olisi myös automaattinen uudelleen reititys tarvittaessa, esimerkiksi käyttäjän myöhästyessä alkuperäiseen reittiehdotukseen kuuluvasta linja-autosta. Ilman ajoneuvoseurantaa tämä ei kuitenkaan olisi täysin aukotonta, mutta automaattisuuden sijasta voisi tarjota myös manuaalista uudelleen reitittämistä. Sijaintitietoa voisi käyttää myös reittihaun parametreissä lähtöpaikkana. Tämän hyödyntäminen vähentäisi kirjoittamisen määrää, joka mobiililaitteilla ei ole kovin käytännöllistä.

Määränpäänä voisi käyttää kalenteritapahtumalle tallennettua sijaintia. Kalenteria voisi hyödyntää myös toisinpäin viemällä haetun reitin kalenterimerkinnäksi ja lisäämällä sille tarvittaessa hälytyksen. Näin muistutus tietystä menosta sisältäisi helposti saatavan reittiohjeen.

Reititykseen tarvittavia sijainteja voisi myös hakea puhelimeen tallennetuista kontakteista. Esimerkiksi asiakkaan luona pidettävään palaveriin voisi hakea reitin käyttämällä määränpäänä asiakkaan käyntikortin tietoja.

Mobiililaitteen muistia olisi mahdollista käyttää reittien ja sijaintien tallentamiseen. Myös muita reititysparametreja olisi mahdollista tallentaa myöhempää käyttöä varten.

8 YHTEENVETO

Mobiilisovellusten toteutukseen on olemassa käytännössä kolme erilaista lähestymistapaa. Jokaisella näistä ratkaisuista on kuitenkin omat hyvät ja huonot puolensa, eikä kaikkia tarpeita kyetä nykyisin täyttämään vain yhtä toteutustekniikkaa hyödyntäen.

Vaihtoehdot ovat natiivit ja selainpohjaiset sovellukset sekä näiden hybridi, eli natiiviksi paketoitu web-tekniikoita käyttävä ratkaisu. Natiivien ohjelmien etuna on esimerkiksi paras mahdollinen suorituskyky ja jakelu sovelluskauppojen kautta. Selkeästi suurimpana ongelmana tämmöisillä sovelluksilla on kuitenkin laitetuki, sillä yhtä ja samaa lähdekoodia ei pysty hyödyntämään useilla eri käyttöjärjestelmillä.

Toisen ääripään, eli selainpohjaisten sovellusten tilanne on täysin päinvastainen. Niiden suorituskyky on heikkoa, jolloin sulavan käyttökokemuksen tarjoaminen käyttäjälle on hankalaa, ellei jopa mahdotonta. Niitä ei myöskään pysty jakelemaan sovelluskauppojen kautta, joten näkyvyyden saaminen voi olla hankalaa, aivan kuten perinteisilläkin internetsivuilla. Merkittävänä etuna on kuitenkin tuki käytännössä kaikille laitteille, joissa on internetselain.

Näiden väliin putoava hybridiratkaisu hyödyntää web-tekniikoita, eli käytännössä siis HTML-kuvauskieltä, CSS-tyylejä ja Javascript-ohjelmointikieltä. Sovellus toimii web-selaimessa, mutta selain on piilotettu käyttäjältä. Lisäksi ohjelma on paketoitu siten, että se vaikuttaa käyttäjän näkökulmasta natiivilta sovellukselta. Se käynnistetään pikakuvakkeesta, ja jakelu tapahtuu sovelluskauppojen kautta. Tällaisia ratkaisuita on tarjolla useita, mutta tässä opinnäytetyössä on perehdytty Sencha Touch- ja PhoneGap-sovelluskehysten yhteiskäyttöön. Sencha Touchia käytetään käyttöliittymän rakentamiseen ja PhoneGapia lähinnä sovelluksen paketoimiseen.

Hybridiratkaisu kuulostaa hyvältä. Se muun muassa tarjoaa mahdollisuuden mobiililaitteen antureiden hyödyntämiseen, mahdollistaa sovelluskauppojen käytön ja tekee tämän kaiken ilman erillisiä lähdekoodeja jokaiselle käyttöjärjestelmälle. Laitetuki onkin sen suurimpia etuja, mutta myös huonoja

puolia löytyy. Haittapuolet tulevat vastaan sovelluskehityksen myötä. Ensimmäisenä vastaan tulee kehitystyön hankaluus, kun joutuu tutustumaan itselle uuden sovelluskehyksen ominaispiirteisiin. Tiedon löytäminen on myös hankalampaa puhtaaseen ohjelmointikieleen verrattuna, sillä käyttäjäkunto on rajallisempi. Kun sovelluksesta saa toimivan, kiinnittyy huomio helposti heikkoon suorituskykyyn.

Ajan kanssa hybridiratkaisut tulevat varmasti kehittymään, osin HTML:n ja Javascriptin kehityksen ansiosta. Myös mobiililaitteiden määrän räjähtävä kasvu lisää mielenkiintoa ratkaisuihin, joilla sovelluksia voi luoda yksinkertaisesti ja tehokkaasti.

LÄHTEET

Adobe Systems Inc. 2012a. Adobe PhoneGap Build [viitattu 14.12.12]. Adobe Systems Inc. Saatavissa: http://build.phonegap.com/

Adobe Systems Inc. 2012b. PhoneGap | About [viitattu 23.11.2012]. Adobe Systems Inc. Saatavissa: http://phonegap.com/about

Android Developers. 2012. User Interface [viitattu 2.11.2012]. Android Developers. Saatavissa: http://developer.android.com/guide/topics/ui/index.html

Android Developers. 2013. Android NDK [viitattu 18.1.2013]. Android Developers. Saatavissa: http://developer.android.com/tools/sdk/ndk/index.html

Apple Inc. 2012a. iOS Human Interface Guidelines [viitattu 2.11.2012]. Apple Inc. Saatavissa: http://developer.apple.com/library/ios/#documentation/userexperience/conceptual/mobilehig/Introduction/Introduction.html#//apple_ref/doc/uid/TP40006556-CH1-SW1

Apple Inc. 2012b. Programming with Objective-C [viitattu 22.2.2013]. Apple Inc. Saatavissa: http://developer.apple.com/library/ios/#documentation/cocoa/conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html

CGI. 2012. Matka.fi API Developer's Guide. Liikennevirasto. Saatavissa: http://developer.matka.fi

Chandhok, R. 2012. HTML5: Friending the Mobile Web [viitattu 15.11.2012]. Qualcomm. Saatavissa: http://www.qualcomm.com/media/blog/2012/02/27/html5-friending-mobile-web

Gartner, Inc. 2012. Gartner Says Free Apps Will Account for Nearly 90 Percent of Total Mobile App Store Downloads in 2012 [viitattu 7.10.2012]. Gartner, Inc. Saatavissa: http://www.gartner.com/it/page.jsp?id=2153215

Gartner, Inc. 2013. Gartner Says Worldwide PC, Tablet and Mobile Phone Combined Shipments to Reach 2.4 Billion Units in 2013 [viitattu 29.5.2013]. Gartner, Inc. Saatavissa: http://www.gartner.com/newsroom/id/2408515

Google. 2012. World Development Indicators [viitattu 26.10.2012]. Google. Saatavissa: https://www.google.fi/publicdata/explore?ds=d5bncppjof8f9_&met_y=ny_gdp_mktp_cd&tdim=true&dl=en&hl=en&q=global%20gdp

GSMA. 2013. The Mobile Economy 2013 [viitattu 1.3.2013]. GSMA.Saatavissa: http://www.gsmamobileeconomy.com/read/

Guelle, R. 2011. Device Fragmentation 2011 / Metrics of the Mobile Web [viitattu 6.7.2012]. Sevenval GmbH. Saatavissa: http://www.slideshare.net/sevenval/device-fragmentation-2011-metrics-of-the-mobile-web

Gullen, A. 2012. The Great HTML5 Mobile Gaming Performance Comparison [viitattu 18.1.2013]. Scirra Ltd. Saatavissa: https://www.scirra.com/blog/85/the-great-html5-mobile-gaming-performance-comparison

Hanks, L. 2011. iOS Development on Windows w/ PhoneGap Build [viitattu 14.12.12]. CodeOutlaw. Saatavissa: http://www.codeoutlaw.com/2011/10/ios-development-on-windows-w-phonegap.html

Kopacz, B. 2012. Smart TV Remote Control + DLNA [viitattu 1.2.2013]. AppBrain. Saatavissa: http://www.appbrain.com/app/smart-tv-remote-control-dlna/com.SmartRemote

Main, S. 2012. Say Goodbye to the Menu Button [viitattu 8.3.2013]. Android Developers Blog. Saatavissa: http://android-developers.blogspot.fi/2012/01/say-goodbye-to-menu-button.html

Microsoft. 2013. Getting started with developing for Windows Phone [viitattu 1.3.2013]. Microsoft. Saatavissa:

http://msdn.microsoft.com/library/windowsphone/develop/ff402529%28v=vs.105%29.aspx

Newman, J. 2012. How Mobile Apps Are Changing Desktop Software [viitattu 14.9.2012]. PCWorld. Saatavissa: http://www.pcworld.com/article/259110/app_invasion_coming_soon_to_your_pc.html

Nguyen, T. 2012. Difference of Native packaging between Sencha touch2 and PhoneGap [viitattu 14.12.12]. Stack Overflow. Saatavissa: http://stackoverflow.com/questions/10725591/difference-of-native-packaging-between-sencha-touch2-and-phonegap?rq=1

Nokia. 2012. How do I start programming for Symbian OS? [viitattu 1.3.2013]. Nokia. Saatavissa: http://www.developer.nokia.com/Community/Wiki/How_do_I_start_programming_for_Symbian_OS%3F

O´Connor, B. 2012. Top 10 Reasons to Use the Sencha HTML5 Framework [viitattu 14.12.12]. Universal Mind. Saatavissa: http://www.universalmind.com/mindshare/entry/top-10-reasons-to-use-the-sencha-html5-framework

Olanoff, D. 2012. Mark Zuckerberg: Our Biggest Mistake Was Betting Too Much On HTML5 [viitattu 18.1.2013]. Aol Tech. Saatavissa: http://techcrunch.com/2012/09/11/mark-zuckerberg-our-biggest-mistake-with-mobile-was-betting-too-much-on-html5/

Pala, M. 2012. PhoneGap performance measurement results [viitattu 18.1.2013]. Pala, M. Saatavissa: http://marguspala.com/phonegap-performance-measurement-results/

Pilgrim, M. 2010. Dive Into HTML5 [viitattu 15.11.2012]. Pilgrim, M. Saatavissa: http://diveintohtml5.info/offline.html

Rivoal, F., Lie, H. W., Çelik, T., Glazman, D. & van Kesteren, A. 2012. Media Queries [viitattu 17.8.2012]. W3C. Saatavissa: http://www.w3.org/TR/css3-mediaqueries/

Rouse, M. 2012. What is native app? [viitattu 25.1.2013]. TechTarget. Saatavissa: http://searchsoftwarequality.techtarget.com/definition/native-application-native-app

Sencha Inc. 2012. API Documentation [viitattu 14.12.12]. Sencha Inc. Saatavissa: http://docs.sencha.com/touch/2-1/#!/api

Sharma, C. 2011. State of the Global Mobile Industry - Half Yearly Assessment 2011 [viitattu 26.10.2012]. Chetan Sharma Consulting. Saatavissa: http://www.chetansharma.com/globalmobileupdate1H2011.htm

StatCounter. 2012a. StatCounter Global Stats, Mobile vs. Desktop from Jan 2010 to Jan 2012 [viitattu 2.11.2012]. StatCounter. Saatavissa: http://gs.statcounter.com/?#mobile_vs_desktop-ww-monthly-201001-201201

StatCounter. 2012b. StatCounter Global Stats, Top 14 Mobile Screen Resolutions on Sept 2012 [viitattu 5.10.2012]. StatCounter. Saatavissa: http://gs.statcounter.com/?#mobile_resolution-ww-monthly-201209-201209-bar

StatCounter. 2013. StatCounter Global Stats, Top 8 Mobile Operating Systems from May 2012 to Mar 2013 [viitattu 1.3.2013]. StatCounter. Saatavissa: http://gs.statcounter.com/?#mobile_os-ww-monthly-201205-201303

Wikipedia. 2012. List of mobile software distribution platforms [viitattu 2.11.2012]. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/List_of_digital_distribution_platforms_for_mobile_devices

Wikipedia. 2013a. iOS version history [viitattu 8.3.2013]. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/IOS_version_history

Wikipedia. 2013b. Windows Phone version history [viitattu 8.3.2013]. Wikipedia. Saatavissa: http://en.wikipedia.org/wiki/Windows_Phone_version_history

[x]cube LABS. 2011. The Evolution of Mobile Operating Systems [viitattu 8.3.2013]. [x]cube LABS. Saatavissa: http://www.xcubelabs.com/evolution-of-mobile-operating-systems.php

[x]cube LABS. 2012. The Android Story [viitattu 8.3.2013]. [x]cube LABS. Saatavissa: http://www.xcubelabs.com/the-android-story.php