HYBRIDISOVELLUSTEN KEHITYS MOBIILILAITTEILLE

Satu Tyrväinen

Opinnäytetyö
Toukokuu 2014

Mediatekniikan koulutusohjelma
Tekniikan ja liikenteen ala
Tiivistelmä

Opinnäytetyössä tutkittiin alustariippumatonta hybridisovelluskehitystä. Työssä keskityttiin selvitämään PhoneGap-sovelluskehystä ja sen soveltamista hybridisovelluskehitykseen sekä tutkittiin hybridisovellusten mahdollisuksia, vahvuksia ja heikkouksia verrattuna web- ja natiivisovelluksiin.


Title
HYBRID APPLICATION DEVELOPMENT FOR MOBILE DEVICES

Degree Programme
Media Engineering

Tutor(s)
MANNINEN, Pasi

Assigned by
SINISALO, Natanael – MEOM Oy

Abstract
This Bachelor's Thesis studies platform-independent hybrid application development. The study focused on researching PhoneGap application framework’s suitability for hybrid application development, and studying the possibilities, strengths and weaknesses of hybrid applications compared to web and native applications.

This thesis was commissioned by MEOM Oy, an advertising agency from Jyväskylä. A Treasure Hunt application was developed for this thesis. The application has a variety of campaigns, which contain one or more treasure hunt contests. The treasure hunts were divided into three stages: in the first and second stages the user is presented with different hint images of the treasure's location and the application tells the user whether he or she is moving closer or farther away from the treasure's location. In the third stage the user types in the treasure's unique code and if the code is correct the application asks the user to fill in contact information for sending his or her prize. The application uses GPS to locate the user's current location.

In addition to PhoneGap, two user interface frameworks were compared in this thesis: Sencha Touch and jQuery Mobile, the latter of which was selected in the implementation of the Treasure Hunt application. This thesis aimed at researching ways to implement mobile applications to various devices effortlessly, that the same code could be used on all mobile platforms. Lastly the potential and future prospects of hybrid applications were analyzed.

Keywords
Mobile application, hybrid application development, PhoneGap

Miscellaneous
Sisältö

1 Opinnäytetyön lähtökohdat..........................................................5
   1.1 Toimeksiantaja......................................................................5
   1.2 Tehtävän kuvaus ja tavoitteet............................................5
2 Mobiililaitteet............................................................................6
   2.1 Mitä ovat mobiililaitteet?.......................................................6
   2.2 Käyttöjärjestelmät ................................................................6
       2.2.1 Yleistä .................................................................6
       2.2.2 Android ..................................................................7
       2.2.3 iOS ..........................................................................8
       2.2.4 Windows Phone .....................................................8
3 Mobiilisovellukset ..................................................................10
   3.1 Yleistä ..............................................................................10
   3.2 Natiivisovellukset ..............................................................10
   3.3 Web-sovellukset ................................................................11
   3.4 Natiivi- vai web-sovellus?..................................................11
   3.5 Hybridisovellukset ...........................................................13
4 PhoneGap ..................................................................................14
   4.1 Yleistä ..............................................................................14
   4.2 PhoneGapin historia ...........................................................15
   4.3 PhoneGapin sovellusarkkitehtuuri ......................................15
   4.4 PhoneGapin ohjelmointirajapinnat ....................................16
       4.4.1 Yleistä .................................................................16
       4.4.2 Ilmoitukset ................................................................18
       4.4.3 Kamera ..................................................................18
       4.4.4 Kiihtyvyysmittari .....................................................19
       4.4.5 Kompassi ..............................................................20
       4.4.6 Media .....................................................................20
       4.4.7 Muisti .....................................................................21
       4.4.8 Sijainti ....................................................................22
       4.4.9 Tiedosto ..................................................................22
4.4.10 Yhteys ............................................................................................................ 23
4.4.11 Yhteystiedot .................................................................................................. 23
4.5 PhoneGapin käyttöönotto ................................................................................. 24
4.6 PhoneGap Build ............................................................................................... 25
  4.6.1 Yleistä ........................................................................................................... 25
  4.6.2 Sovellusten pakkaaminen PhoneGap Buildin avulla .................................. 26
5 Sencha Touch ...................................................................................................... 29
  5.1 Yleistä ............................................................................................................. 29
  5.2 Sencha Touch -sovelluksen rakenne ............................................................... 30
    5.2.1 Sovellusarkkitehtuuri ................................................................................. 30
    5.2.2 Luokkajärjestelmä ..................................................................................... 31
  5.3 Sencha Touchin käyttöönotto .......................................................................... 32
  5.4 Sencha Touch ja PhoneGap ............................................................................. 33
6 jQuery Mobile ...................................................................................................... 35
  6.1 Yleistä ............................................................................................................. 35
  6.2 jQuery Mobile -sovelluksen rakenne ............................................................... 35
  6.3 jQuery Mobile -sovelluksen käyttöönotto ......................................................... 38
  6.4 jQuery Mobile ja PhoneGap ............................................................................. 38
7 Aarrejahti-sovellus ............................................................................................. 39
  7.1 Lähtökohdat ja sovelluksen kuvaus ................................................................. 39
  7.2 Sovelluksen toteutus ....................................................................................... 40
    7.2.1 Sovelluksen vaatimukset .......................................................................... 40
    7.2.2 Käytettävien teknologioiden valinta .......................................................... 40
  7.3 Arkkitehtuuri .................................................................................................. 43
  7.4 Sovelluksen esittely ........................................................................................ 46
  7.5 Sovelluksen testaus ......................................................................................... 49
    7.5.1 Sovelluksen testaus Ripple-PhoneGap-emulaattorilla ................................ 49
    7.5.2 Sovelluksen testaus älypuhelimilla ............................................................. 51
  7.6 Jatkokehitys .................................................................................................... 54
8 Pohdinta ................................................................................................................ 57
Lähteet ..................................................................................................................... 60
Kuviot

Kuvio 1. Eri Android-käyttöjärjestelmän versioiden suhteelliset osuudet ......................8
Kuvio 2. Älypuhelinten käyttöjärjestelmien jakautuminen Suomessa .........................9
Kuvio 3. Facebookin natiivisovellus (vasemmassa) ja web-sovellus (oikealla) ........11
Kuvio 4. Sovellusalueet natiivi- ja hybridisovelluksissa vasemmassa ja web-
sovelluksissa oikealla........................................................................................................14
Kuvio 5. Phonegapin sovellusarkkitehtuuri .................................................................16
Kuvio 6. PhoneGap Buildin käyttöliittymä .................................................................27
Kuvio 7. Sencha Touchin teemat iOS-, Android- ja Windows Phone 8 -
käyttöjärjestelmissä ........................................................................................................29
Kuvio 8. MVC-sovellusarkkitehtuuri ..............................................................................30
Kuvio 9. jQuery Mobile -käyttöliittymäkehysyn käyttöliittymäkomponentteja
kahdessa eri teemassa ........................................................................................................37
Kuvio 10. Themeroller-työkalun käyttöliittymä ..............................................................41
Kuvio 11. Google Chrome -selaimen kehittäjän työkalut ..........................................42
Kuvio 12. Aarrejahti-sovelluksen kansiorakenne .........................................................44
Kuvio 13. Aarrejahti-sovelluksen arkkitehtuuri ............................................................46
Kuvio 14. Aarrejahti-sovelluksen aloituskuvan päänäkymä .........................................47
Kuvio 15. Aarrejahti-sovelluksen käynnissä olevien kampanjoiden ja yksittäisen
kampanjan näkymät ............................................................................................................47
Kuvio 16. Aarrejahti-sovelluksen yksittäisen aarrejähdin 1. vaihe .........................48
Kuvio 17. Aarrejahti-sovelluksen yksittäisen aarrejähdin 3. vaihe .........................49
Kuvio 18. Ripple-PhoneGap-emulaattorin kytkeminen pääselaimessa .......................50
Kuvio 19. Aarrejahti-sovellus PhoneGap Buildissa .....................................................51
Kuvio 20. Aarrejahti-sovellus Samsung Galaxy S2-, iPhone 4- ja Nokia Lumia 820 -
älypuhelimissa .................................................................................................................54

Taulukot

Taulukko 1. Mobiilikäyttöjärjestelmien toimitusmäärät ja markkinaosuudet
kolmannella vuosineljänneksellä vuosina 2012 ja 2013 (toimitusmäärät miljoonia)....7
Taulukko 2. Natiivisovelluskehityksen vaatimukset eri käyttöjärjestelmissä ..........12
Taulukko 3. PhoneGapin tuki eri ohjelmointirajapinnoille tämän hetken suosituimmissa mobilikäyttöjärjestelmissä ................................................................. 17
Taulukko 4. Sencha Touch ja jQuery Mobile -käyttöliittymäkehysien vertailu........ 43
Taulukko 5. Aarrejahti-sovelluksen testaukseen käytetyt laitteet ......................... 53
1 Opinnäytetyön lähtökohtat

1.1 Toimeksiantaja

Opinnäytetyön toimeksiantajana toimi jyväskyläläinen mainostointimoottori MEOM Oy. MEOM Oy tuottaa pääasiallisesti verkkosivustoja Wordpress-julkaisu- ja järjestelmälle, mutta heidän tuotteisiinsa lukeutuvat myös erilaiset painotuotteet, yritysilmeet sekä sisällöntuotto. Verkkosivuissaan MEOM Oy seuraa responsiivisen web-kehityksen periaatteita, mikä mahdollistaa sivustojen selaamisen eri päätelaitteilla, kuten älypuhelimella, tabletilla tai tietokoneella.

1.2 Tehtävän kuvaus ja tavoitteet

Opinnäytetyön tavoitteena oli tutkia mobiililaitteiden alustariippumatonta sovelluskehitystä, ja työssä keskityttiin tarkastelemaan eri lähestymistapoja hybridiosvellusten toteuttamiseen mobiilialustoihille. Tutkimuksen tuloksena toteutettiin aarbeitensovellus, joka hyödyntää laitteen nativialaista ominaisuuksia, mutta voidaan jakaa eri käyttöjärjestelmille mahdollisimman vaivattomasti.

2 Mobiililaitteet

2.1 Mitä ovat mobiililaitteet?


Sekä älypuhelimet että tabletit ovat yleistyneet maailmalla räjähdytsmäisesti viime vuosien aikana. BI Intelligencen vuoden 2013 joulukuussa teettämän tutkimuksen mukaan joka viidennellä ihmisellä on käytössään älypuhelin ja joka 17. omistaa tabletin. Suomessa älypuhelimia on jo 61 prosentilla väestöstä (Arki muutuu yhä mobiilikeskeisemmäksi 2013). Tablettien myynti Suomessa on kasvanut hurjaa vauhtia; vuoden 2013 tammi-syyskuussa myytiin yli 393 000 tabletta ja kasvua edelliseen vuoteen oli jopa 145 prosenttia (Kodintekniikkaindeksi 2013).

2.2 Käyttöjärjestelmät

2.2.1 Yleistä

Käyttöjärjestelmä on laitteen, kuten tietokoneen tai älypuhelimen, perusohjelmisto, joka ohjaa ja hallinnoi laitteen toimintaa. Tällä hetkellä maailmalla yli voimaisesti suosituin mobiililaitteiden käyttöjärjestelmä on Googlen kehitämä Android, toisena seuraavina iOS ja kolmantena BlackBerryn vuonna 2013 ohittanut Windows Phone (ks. taulukko 1, s. 7).
2.2.2 Android


Androidin uusin versio 4.4 Kitkat julkaistiin 31. lokakuuta 2013, mutta sitä käyttää tämän oppi- ja käytännössä vain 1,8 prosenttia Android-laitteiden omistajista, sillä kaikki vanhemmat mobiilikäyttöjärjestelmät eivät tue uusimpia Android-versioita. Suosituin Android-käyttöjärjestelmän versio on on 4.1.X Jelly Bean 35,5 prosentin osuudella (ks. kuvio 1, s. 8).


<table>
<thead>
<tr>
<th>Käyttöjärjestelmä</th>
<th>3Q12 toimitusmäärä</th>
<th>3Q12 markkinaosuus</th>
<th>3Q13 toimitusmäärä</th>
<th>3Q13 markkinaosuus</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>139,9</td>
<td>74,9 %</td>
<td>211,6</td>
<td>81,0 %</td>
</tr>
<tr>
<td>iOS</td>
<td>26,9</td>
<td>14,4 %</td>
<td>33,8</td>
<td>12,9 %</td>
</tr>
<tr>
<td>Windows Phone</td>
<td>3,7</td>
<td>2,0 %</td>
<td>9,5</td>
<td>3,6 %</td>
</tr>
<tr>
<td>BlackBerry</td>
<td>7,7</td>
<td>4,1 %</td>
<td>4,5</td>
<td>1,7 %</td>
</tr>
<tr>
<td>Muut</td>
<td>8,4</td>
<td>4,5 %</td>
<td>1,7</td>
<td>0,6 %</td>
</tr>
</tbody>
</table>
2.2.3 iOS

iOS (Internet Operating System) on Applen kehitettävä käyttöjärjestelmä. Alun perin iOS kehitettiin iPhonea varten vuonna 2007, mutta myöhemmin se on laajentunut iPod Touchille, iPadille ja Apple TV:lle. Ensimmäinen iPhone oli samalla myös ensimmäinen kosketusnäytöllinen puhelin, joka ei vaatinut styluksen eli osoitinkynän käyttöä. (iOS: A visual history 2013.)

Viimeisin versio iOS:sta, iOS 7, julkaistiin 27. syyskuuta 2013. Uuden ulkoasun lisäksi käyttöjärjestelmä sai muun muassa Control Center -työkalupalkin, joka on Applen versio Android-laitteiden vastaavasta pika-asetusvalikosta. Toisin kuin Android ja Windows Phone, iOS-käyttöjärjestelmä on lisenoitu vain Applen tuotteille eikä sitä käytä muut kuin Applen valmistamat laitteet. (Mt.)

2.2.4 Windows Phone

Taulukosta 1 voidaan todeta, että vaikka Windows Phone on maailmassa kolmanneksi käytettyin älypuhelinten käyttöjärjestelmä, sen markkinaosuus on varsin pieni (3,6 %). Suomessa taas Windows Phone on saavuttanut erityisaseman Nokian vuoksi. Market-Vision vuonna 2013 teettämän tutkimuksen mukaan se on Suomen suosituin mobiilikäyttöjärjestelmä. Windows Phonea tutkimuksen mukaan käytti 35 prosenttia suomalaisista, kun taas Androidin osuus oli 33 ja iOS:n 30 prosenttia (ks. kuvio 2).
3 Mobiilisovellukset

3.1 Yleistä


Kaikkia mobiilisovelluksia ei kuitenkaan ladata sovelluskaupista, vaan osa sovelluksista toimii web-sivujen tapaan internetselaimen väliykseellä. Näitä sovelluksia ovat ns. web-sovellukset. Sovelluskaupista ladatut applikaatiot ovat useimmiten natiiveja sovelluksia, mutta osa niistä on toteutettu samaan tapaan kuin web-sovellukset, jolloin puhutaan hybridisovelluksista.

3.2 Natiivisovellukset


Natiivisovellukset toimivat laitteesta käsin eivätkä ole riippuvaisia internetistä vaan voivat toimia offline-tilassa. Natiivisovellukset pystyvät myös hyödyntämään laitteen sisäänrakennettuja ominaisuuksia, kuten kameraa, gps-paikannusta, kiihtyvyyystmittaria, kompassia, kontaktlistaa jne. Lähä kaikki mobiilipelit ovat natiiveja sovelluksia. (Budiu 2013.)
3.3 Web-sovellukset


3.4 Natiivi- vai web-sovellus?

Natiivi- ja web-sovelluksia on nykyisin hankala erottaa toisistaan pelkän ulkonäön perusteella (ks. kuvio 3). Web-sovellukset ovat kehitetty viime vuosien aikana kilpailukykyisimmiksi ja ne saadaan näyttämään jo hyvin paljon natiiveilta sovelluksilta. Web-sovelluksien etuna on niiden helppo jakelu loppukäyttäjille; sama applikaatio toimii kaikilla laitteilla, joissa on asennettuna internetselain eikä sovellusta tarvitse tehdä erikseen eri päätelaitteille.

Kuvio 3. Facebookin natiivisovellus (vasemmalla) ja web-sovellus (oikealla)

**Taulukko 2. Natiivisovelluskehityksen vaatimukset eri käyttöjärjestelmissä (Frommel 2013)**

<table>
<thead>
<tr>
<th>Mobiilikäyttöjärjestelmä</th>
<th>Tietokoneen käyttöjärjestelmä</th>
<th>Kehitysympäristö</th>
<th>Ohjelmointikieli</th>
</tr>
</thead>
<tbody>
<tr>
<td>Android</td>
<td>Windows, Mac, Linux</td>
<td>Eclipse</td>
<td>Java</td>
</tr>
<tr>
<td>iOS</td>
<td>Mac</td>
<td>XCode</td>
<td>Objective-C</td>
</tr>
<tr>
<td>Windows Phone</td>
<td>Windows</td>
<td>Visual Studio</td>
<td>C#, Visual Basic</td>
</tr>
</tbody>
</table>

Valittaessa toteutettavan sovelluksen tyyppiä kannattaa tehdä kartoitusta sovelluksen vaatimuksista. Tarvitseeko applikaation toimia sekä Android, iOS- että Windows Phone -laitteissa, täytyykö sen kyetä käyttämään jotakin laitteen nativiiva ominaisuuutta, esimerkiksi kameraa, ja millainen on sovelluskehitykseen varattu budjetti? Websovelluksen kehittäminen on yleensä helppo, nopeampaa ja halvempaa kuin natiivisovelluksen, mutta se ei ole suorituskyvyttävän yhtä tehoksia eikä se kykene hyödyntämään laitteen sisäänrakennettuja ominaisuuksia, kuten kameraa tai notifikaatioita ilmoituksia. Web-sovelluksen päivittäminen, ylläpito ja jakelu eri mobiilialustoiille on vaivattomampaa, mutta web-sovelluksia ei saada esille eri käyttöjärjestelmien omiin sovelluskauppoihin kuten natiivisovelluksia. Natiivisovellus ja websovellus eivät kuitenkaan ole ainoat mobiilisovelluskehittäjän vaihtoehdot; näiden välimaastoon sijoittuvat hybridisovellukset, jotka yhdistelevät molempien sovellus-tyyppien ominaisuuksia. (Avola & Raasch 2013, 9-13.)
3.5 Hybridisovellukset


Vaikka hybridisovellukset toteutetaankin web-teknikoilla, ne eroavat web-sovelluksista siinä, etteivät ne toimi mobiililaitteen internetselaimesta käsin, vaan natiivin sovelluksen sisällä on oma web view (web-ikkuna), joka hyödynnetään laitteen selainmoottoria käsittäkseen HTML-sisältöä. Parhaimmillaan hybridisovellus on yhdistelmä web- ja natiivisovellusten hyvistä puolista. Niiden maine on kuitenkin kär- sinyt yritysten paketoitua tavallisia web-sovelluksia hybrideiksi saadakseen sovelluksen sovelluskauppoihin. (Riippi 2013.)

4 PhoneGap

4.1 Yleistä


PhoneGap on yksi tunnetuimmista ja käytetyimmistä hybridisovellusten toteuttamiseen tarkoitetuista frameworkeista. Käyttöliittymäkerros PhoneGap-sovelluksissa muodostuu web view -ikkunasta, joka on 100-prosenttisesti laitteen näytön kokoinen. Tämä tarkoittaa sitä, ettei web view -ikkunassa näy selaimelle tyyppillistä osoitettiriviä vaan sovellus täyttää koko näytön natiivisovellusten tapaan (ks. kuvio 4). (Trice 2012.)

Kuvio 4. Sovellusalueet natiivi- ja hybridisovelluksissa vasemmalla ja web-sovelluksissa oikealla
4.2 PhoneGapin historia

PhoneGapin ensimmäinen versio syntyi elokuussa 2008 San Franciscossa järjestetystä iPhoneDevCamp-tapahtumassa, jossa pieni joukko kanadalaisen Nitobi-yrityksen työntekijöitä onnistui luomaan selainikkunan natiivin iPhone-sovelluksen sisään. Alkuun Nitobin työryhmä keskittyi nimenomaan iPhoneen, sillä osaavia Objective-C-kehittäjiä oli web-kehittäjiin verrattuna hyvin vähän. Ensimmäinen vakaa versio PhoneGapista julkaistiin helmikuussa 2009, ja se sisälsi tuen iPhoneen lisäksi Androidille ja BlackBerrylle. (Taft 2009.)


4.3 PhoneGapin sovellusarkkitehtuuri

Sovellusarkkitehtuuri sisältää sovelluksen tai järjestelmän osat, niiden suhteet toisiinsa ja ympäristöön sekä sovelluksen suunnittelua ja jatkokehitystä ohjaavat periaatteet (Koskimies & Mikkonen 2005, 18). PhoneGap-sovellukset rakentuvat kuvion 5 (s. 16) mukaisesti: PhoneGapin yhteyssilta toimii JavaScript-rajapinnan ja laitteen natiivien ominaisuuksien välissä. Sovelluksien toteuttamiseen voidaan myös käyttää erillistä käyttöliittymäkehystä (esim. jQuery Mobile tai Sencha Touch), joka helpottaa käyttöliittymän rakentamista. (Ghatol & Patel 2012, 18.)


4.4 PhoneGapin ohjelmointirajapinnat

4.4.1 Yleistä

PhoneGap tarjoaa ohjelmistokehittäjille ohjelmointirajapinnan (Application programming interface, API), jonka avulla päästään käsiksi mobiililaitteen natiiveihin omi-
naisuuksiin JavaScriptillä. Rajapintojen avulla eri ohjelmat voivat niin sanotusti keskustella keskenään eli tehdä pyyntöjä ja vaihtaa tietoja. PhoneGapin tapauksessa sovelluskehittäjä rakentaa sovelluksensa toimintalogiikan JavaScriptillä ja PhoneGap toimii tavallaan siltana sovelluksen ja laitteen välillä. (Trice 2012.)

PhoneGapin nykyinen versio (3.4.0) tukee hyvin uusimpia mobiililaitteita. Taulukosta 3 voidaan todeta, että sekä iOS, Android että Windows Phone 7 ja 8 ovat hyvin tuettuja. Vain kompassille ei löydy tukea vanhemmilla iPhone-älypuhelimilla. Monia taulukosta puuttuvia ominaisuuksia, kuten kalenteria, viivakoodinlukijaa, bluetooth-yhteyttä, push-ilmoituksia tai Facebook-tilin yhdistämistä sovellukseen, voidaan hyödyntää erillisten liitännäisten avulla (Traeg 2014).

Taulukko 3. PhoneGapin tuki eri ohjelmo-intirajapinnoille tämän hetken suosituimmissa mobiilikäyt- töjärjestelmissä (PhoneGap 2014)

<table>
<thead>
<tr>
<th></th>
<th>iPhone / iPhone 3 G</th>
<th>iPhone 3 GS tai uudempi</th>
<th>Android</th>
<th>Windows Phone 7 ja 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ilmoitukset</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Kamera</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Kiihtyvyysmittari</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Kompassi</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Media</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Muisti</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Sijainti</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Tiedosto</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Yhteys</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Yhteystiedot</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

PhoneGap-versiossa 3.4.0 sovelluksessa tarvittavat rajapinnat lisätään sovellukseen liitännäisinä eli pluginina. Ohjelmointirajapintoja kutsutaan muutamaa poikkeusta lukuun ottamatta aina seuraavasti:

navigator.rajapinnannimi.rajapinnanmetodi(kutsuOnnistui, kutsuOnnistui, parametrit)

### 4.4.2 Ilmoitukset

Ilmoituksilla eli notifiikaatioilla tarkoitetaan mobiililaitteen erilaisia viestejä, joita sovellus lähettää käyttäjälle. Ilmoituksiin luetaan dialogi- eli ponnahdusikkunat, äänimerkit ja laitteen värinätoiminnot. PhoneGapin ilmoitusrajapinta käyttää ilmoitusten äänimerkkinä aina käyttäjän itse laitteelle asettamansa vakoilmoitusmerkkiääntä. Tavallisesti ponnahdusikkunan tyypissä ilmoitukset saa luotua esimerkiksi seuraavalla tavalla:

```javascript
function showAlert() {
    navigator.notification.alert(
        'Voitit pelin!',       // viesti
        alertDismissed,       // callback-funktio
        'Onneksi olkoon',    // otsikko
        'Hyväksy'             // painikkeen teksti
    );
}
```

### 4.4.3 Kamera

navigator.camera.getPicture(onSuccess, onFail, {
  quality: 75,
  destinationType: Camera.DestinationType.FILE_URI
});

// Kuva haettiin kameralta onnistuneesti
function onSuccess(imageURI) {
  var image = document.getElementById('kameranKuva');
  image.src = imageURI;
}

// Kuvan haku kameralta epäonnistui
function onFail(message) {
  alert('Tapahtui virhe: ' + message);
}

Kameran getPicture-funktiolle voidaan funktiokutsussa syöttää erilaisia parametreja. Näitä ovat muun muassa kuvan laatu, joka ilmoitetaan numeroarvolla 0-100 väliltä, jossa arvo 100 tarkoittaa parasta mahdollista laatua. Tärkeimpiä parametreja ovat kuitenkin palautettavan kuvan tyyppi (merkkijono vai kuvan osoite) ja kuvan lähde eli otetaanko kuva kameralla vai haetaanko se laitteen muistista. (PhoneGap Documentation 2014.)

4.4.4 Kiihtyvyysmittari

Kiihtyvyysmittari toimii liiketunnistimen tapaan: se reagoi laitteen kallistumiseen, sijainnin muutoksiin ja kiihtyvyyteen vertaamalla laitteen nykyistä sijaintia ja suuntaa kolmessa ulottuvuudessa x-, y- ja z-akselen suhteen. Kiihtyvyysmittarirajapinnalla on kolme metodia: getCurrentAcceleration, joka palauttaa laitteen nykyisen sijainnin koordinaatit x-, y- ja z-akselien suhteen, watchAcceleration, joka tarkkailee laitteen kiihtyvyyden muutoksia ja palauttaa kiihtyvyyden arvoja tasaisin väliajoin sekä clearWatch, jota käytetään keskeyttämään kiihtyvyyden tarkkailua käyttämällä kiihtyvyydelle annettua tunnusta eli id:tä. Seuraavassa esimerkissä käyttäjälle ilmoitetaan laitteen senhetkinen kiihtyvyys x-, y- ja z-akseleiden suhteen:

```javascript
function onSuccess(acceleration) {
  alert('Kiihtyvyys X: ' + acceleration.x + '

  ' + acceleration.y + '

  ' + acceleration.z + '

  ' + '

  Aikaleima: ' + acceleration.timestamp + '

);
function onError() {  
    alert('Tapahtui virhe');  
};

navigator.accelerometer.getCurrentAcceleration(onSuccess, onError);

4.4.5 Kompassi

Kompassirajapinta mahdollistaa laitteen suunnan tarkkailun. Kompassi tunnistaa, mihin suuntaan laite osoittaa. Suunta palautetaan astelukuna väliltä 0-359.99, jossa 0 tarkoittaa pohjoista. Laitteen suuntaa voidaan tarkkailla tasaisin väliajoin samaan tapaan kuin kiihtyvyyttä. Tämän lisäksi voidaan asettaa raja, jonka ylittävää käyttäjälle näytetään ilmoitus sijainnin muuttumisesta. Laitteen suunnan seuranta kolmen sekunnin välein saadaan aikaan seuraavasti:

function onSuccess(heading) {  
    var element = document.getElementById('suunta');  
    element.innerHTML = 'Suunta: ' + heading.magneticHeading;  
};

function onError(compassError) {  
    alert('Tapahtui virhe: ' + compassError.code);  
};

var options = {  
    frequency: 3000  
}; // Tarkkaillaan 3 sekunnin välein

var watchID = navigator.compass.watchHeading(onSuccess, onError, options);

4.4.6 Media

Mediarajapinta tarjoaa mahdollisuuden tallentaa ja toistaa äänitiedostojaa sovelluksessa. Mediarajapinnalla on paljon erilaisia metodeja: mediasoittimille yleiset play-, stop- ja pause, äänenvoimakkuuden säätö, uuden äänitiedoston nauhoitus sekä nykyisestä äänitiedostosta voidaan palauttaa sen kesto tai kohta, jossa ollaan menossa toistettaessa ääntä. Esimerkki äänitiedoston toistamisesta sovelluksessa:

// Aloita äänitiedoston toistaminen
function playAudio(url) {
    // Toista äänitiedosto annetusta osoitteesta
    var aanitiedosto = new Media(url,  
    // callback-funktio
    function () {

console.log("Äänitiedoston toisto onnistui");

// callback-funktio virhetilanteissa
function (err) {
    console.log("Tapahtui virhe: " + err);
}

// Toista äänitiedosto
aanitiedosto.play();

4.4.7 Muisti

Muistirajapinnan avulla sovelluksessa voidaan tallentaa erilaisia tietoja muistiin; sovelluksessa voi olla vaikkapa mahdollisuus säättää joitakin asetuksia ja näitä asetuksia voidaan sitten tallentaa, jotta sovellus muistaa valitut asetukset seuraavalla kerralla. Menetelmä tiedon tallentamiseen on neljä: LocalStorage, WebSQL, IndexedDB ja erilaiset liitäntä- eli plugin-ratkaisut. (PhoneGap Documentation 2014.)


    // Haetaan lomakekentistä muuttujien arvot
    var nameVal = $('#name').val();
    var zipcodeVal = $('#zipcode').val();
    var cityVal = $('#city').val();

    // Tallennetaan tiedot paikallismuistiin
    window.localStorage.setItem('name', nameVal);
    window.localStorage.setItem('zipcode', zipcodeVal);
    window.localStorage.setItem('city', cityVal);
4.4.8 Sijainti


```javascript
// Laitteen sijainti paikallistettiin onnistuneesti
var onSuccess = function(position) {
    alert('Leveysaste: ' + position.coords.latitude + '
    Pituusaste: ' + position.coords.longitude + '
');
}

// Laitteen sijainnin paikallistaminen epäonnistui
function onError(error) {
    alert('Virhekoodi: ' + error.code + '
    Tapahtui virhe: ' + error.message + '
');
}

navigator.geolocation.getCurrentPosition(onSuccess, onError);
```

4.4.9 Tiedosto


```javascript
// Hakemistonsiirto onnistui
function success(entry) {
    console.log("Uusi polku: " + entry.fullPath);
}

// Hakemistonsiirto epäonnistui
function fail(error) {
```
alert("Tapahtui virhe: " + error.code);
}

// Hakemistonsiirtofunktio
function moveDir(entry) {
    var parent = document.getElementById('parent').value,
        parentName = parent.substring(parent.lastIndexOf('/')+1),
        newName = document.getElementById('newName').value,
        parentEntry = new DirectoryEntry(parentName, parent);

    // Siirtää ja uudelleennimeää hakemiston
    entry.moveTo(parentEntry, newName, success, fail);
}

4.4.10 Yhteys

Yhteysrajapinta tarjoaa tietoa mobiililaitteen puhelinverkon- ja wifi-yhteyksistä ja kykenee tarkistamaan onko laite kytkeytyneen internetiin. Yhteysrajapinnan avulla voidaan selvittää käytetyn yhteyden tyyppi ja tarkkailla internetyhdyksen tilaa; sovellus voi esimerkiksi ilmoittaa käyttäjälle, jos internet-yhteys katkeaa seuraavasti:

    // Tarkkaillaan yhteyttä
    document.addEventListener("offline", onOffline, false);

    // Ilmoitetaan käyttäjälle, että laite on offline-tilassa
    function onOffline() {
        alert("Laite on offline-tilassa.");
    }

4.4.11 Yhteystiedot

Yhteystietorajapinnalla sovellus voi käyttää laitteen kontaktelistaa; listaan voidaan luoda uusia kontakteja, vanhoja voidaan poistaa tai kopioida ja kontakteja voidaan näyttää sovelluksessa. Yhteystietolistaan voidaan lisätä uusi yhteystieto seuraavasti:

    // Kontaktin tallentaminen onnistui
    function onSuccess(contact) {
        alert("Kontakti tallennettu.");
    };

    // Kontaktin tallentaminen epäonnistui
    function onError(contactError) {
        alert("Tapahtui virhe: " + contactError.code);
    };}
// Luodaan uusi kontaktiolio
var contact = navigator.contacts.create();
contact.displayName = "Esimies";
contact.nickname = "Pomo";
var name = new ContactName();
name.givenName = "Matti";
name.familyName = "Meikäläinen";
contact.name = name;
// Tallennetaan luotu kontakti kontaktilistaan
contact.save(onSuccess,onError);

4.5 PhoneGapin käyttöönotto

PhoneGapin dokumentaatiosta käy ilmi, että ennen PhoneGapin asentamista tulee
asentaa kaikkien haluttujen kohdealustojen omat SDK:t. SDK (Software Development
Kit) on ohjelmistopaketti, joka mahdollistaa sovelluskehityksen tietyllä alustalla. Tyypillisesti SDK sisältää yhden tai useamman ohjelmointirajapinnan, erilaisia ohjelmointityökaluja sekä dokumentaation. Kehitetäessä PhoneGap-sovelluksia lokaalisti, tulee kehitysympäristö valita kunkin kohdealustan mukaan. PhoneGapin oma Com-
mand-Line Interface (CLI) eli komentorivikäyttöliittymä tukee seuraavia kombina-
tioita:

- iOS (Mac)
- Android (Mac, Linux, Windows)
- Windows Phone (Windows)

Kehitysympäristön lisäksi PhoneGapin asennus vaatii Node.js:n. Node.js on kevyt
JavaScript-sovellusalusta web-palvelujen kehitykseen palvelinpäässä. Node.js:n voi
ladata ja asentaa ilmaseksi Node.js:n virallisilta verkkosivuilta tai suoraan tietoko-
nen komentoriviltä. Kun Node.js on asennettu onnistuneesti, voidaan asentaa itse
PhoneGap komentoriviltä seuraavasti:

    npm install -g phonegap

Asennuskomento voi tarvita toimiakseen etuliitteen "sudo". Tällöin asennus suorite-
taan järjestelmänvalvojana. Kun PhoneGap on onnistuneesti asennettu, voidaan en-
simmäinen sovellus luoda komentoriviltä käskyllä
phonegap create hello com.example.hello HelloWorld

Sovelluksen luomiskäskyssä “hello” on projektille luotavan kansion nimi. Loput kaksi parametria ovat vapaaehtoisia mutta suositeltavia: ensimmäinen on projektin käänteisen domainin -tyylin tunniste ja toinen sovelluksen nimi. Kumpaakin vapaaehtoista parametria voidaan muokata myös jälkeenpäin config.xml-tiedostosta. Kaikki sovellusta käsittelevät komennot täytyy tämän jälkeen ajaa komenriviiltä sovelluksen hakemistosta käsin. Sovellus ”builddataan” eli kootaan ajoa varten valmiiksi seuraavasti:

phonegap build android

Tässä käsyyssä sanan “android”-tilalle laitetaan haluttu kohdealusta, jolle sovellus käännetään. Buildauksen jälkeen sovellusta voidaan testata emulaattorissa komennolla

phonegap install android
phonegap run android

Ensimmäisellä komennolla asennetaan halutun kohdealustan emulaattori ja toisella ajetaan sovellus tässä emulaattorissa. (PhoneGap Documentation 2014.)

4.6 PhoneGap Build

4.6.1 Yleistä

PhoneGap tarjoaa sovelluskehittäjille mahdollisuuden luoda laitteen natiiveja ominaisuuksia hyödyntävien mobiilisovelluksia web-tekniikoilla ja paketoida nämä sovellukset sovelluskauppoihin jaettavaksi. Kuitenkin jokainen eri päätelaite asettaa omat vaatimuksensa sovelluksen kehitysypäristölle ja näin ollen saman sovelluksen monistaminen eri päätelaiteille hankaloituu vaikka suurimman osan koodista voisin käyttää uudelleen. Tätä prosessia helpottamaan kehitetty PhoneGap Build tarjoaa vaihtoehtoisen laiteriippumattoman tavan sovellusten paketoimiseen.
PhoneGap Build on Adoben kehitämä pilvipalvelu PhoneGap-sovellusten paketointiin. Sovelluksen tiedostot voidaan joko ladata PhoneGap Buildiin zip-pakettina tai ne voidaan hakea sovelluskehittäjän Git- tai SVN-talletuspaikasta (repository). PhoneGap Build pakaa tiedostot valmiiksi sovellukseksi. PhoneGap Build tukee Android-, iOS- ja Windows Phone 8-laitteita. (PhoneGap Build Documentation 2014.)

4.6.2 Sovellusten pakkaaminen PhoneGap Buildin avulla


Kuvio 6. PhoneGap Buildin käyttöliittymä


Windows Phone 8-sovelluksia varten täytyy rekisteröityä Windows Phone-sovelluskehittäjäksi ja avata Windows Phone 8-mobiililaitte sovelluskehittäjän tunnuksilla sovelluskaupan ulkopuolisia applikaatioita varten. Tämän jälkeen PhoneGap Build:lla pakattua sovellusta voidaan testata Windows Phone 8-laitteella. Sovelluskauppajakelu varten sovellus tarvitsee vielä julkaisijatunnuksen. Tätä varten täytyy olla sovelluskehittäjän tunnukset: pienten yritysten ja yksityishenkilöiden tunnus maksaa 14 euroa ja suurempien yritysten tunnus 75 euroa vuodessa. Kun sovelluksen
julkaisijatunnus on ladattu PhoneGap Buildiin, voidaan Windows Phone 8 -sovellus julkaista Windows Phone Marketplace -sovelluskaupassa. (PhoneGap Build Doc-umentation 2014.)
5 Sencha Touch

5.1 Yleistä

Sencha Touch on mobiililaitteille suunnattujen web-sovellusten toteuttamiseen kehitetty JavaScript-koodikirjasto ja -framework. Sencha Touchilla toteutetut käyttöliittymät näyttävät ja tuntuvat hyvin pitkälti kullekin päätelaitteelle yksilöllisesti suunnitelluilta nativisovelluksilta; Sencha Touchilla on omat teemansa muun muassa Androidille, iOS:lle ja Windows Phone 8:lle (ks. kuvio 7). (Sencha Touch Build Mobile Web Apps with HTML5 2014.)

Kuvio 7. Sencha Touchin teemat iOS-, Android- ja Windows Phone 8-käyttöjärjestelmissä (Sencha Touch Build Mobile Web Apps with HTML5 2014)

Ensimmäinen versio Senha Touchista julkaistiin heinäkuussa 2010, kun ExtJS-, jQTouch- ja Raphaël-JavaScript-kirjastot yhdistettiin yhdeksi paketiksi, ja se sisälsi tuen Androidille ja iOS:lle. Uusin versio, Sencha Touch 2.3.1, julkaistiin marraskuussa 2013 ja se on suunniteltu toimimaan Androidin ja iOSin lisäksi Windows Phone 8 ja Windows 8-käyttöjärjestelmissä. (Kosmaczewski 2013, 2-4.)
5.2 Sencha Touch -sovelluksen rakenne

5.2.1 Sovellusarkkitehtuuri

Sencha Touch perustuu MVC-arkkitehtuuriin. Lyhteneen MVC tulee sanoista Model eli malli, View eli näkymä ja Control eli ohjain. MVC-sovellusarkkitehtuurin perusajatukseen on erotettu käyttöliittymäkerros varsinaisesta sovelluslogiikasta ja sovelluksessa käytettävästä datasta. Kun käyttöliittymä on erillään muusta sovelluksesta, sitä voidaan helposti muuttaa tai se voidaan vaihtaa jopa täysin uuteen. (Koskimies & Mikkonen 2005, 142.)

MVC-sovellusarkkitehtuuria käyttävissä sovelluksissa sovellus jaetaan kolmeen osaan: malleihin, näkymiin ja ohjaimiin (ks. kuvio 8). Malli käsittää sovelluksessa käytettävän dataan liittyvät toiminnot: mitä dataa sovelluksessa hyödynnetään ja miten sitä käsitellään. Näkymät määrittävät käyttöliittymän eli sovelluksen ulkoasun ja sen, miten mallien tieto esitetään sovelluksessa. Ohjain toimii tavallaan siltana mallien ja näkymien välissä; se ottaa kässkyjä käyttäjältä ja muuttaa näkymää tai mallia niiden mukaan. (Mt.)

Sencha Touch -sovellusten arkkitehtuuri eroaa tavallisesta MVC-mallista siinä, että mallien, ohjainten ja näkymien lisäksi se voi sisältää myös profiileja (profile) ja varastoja (store). Profiileilla voidaan määrätä miltä sovellus näyttää eri päätelaitteilla, kuten tableteilla ja älypuhelimilla. Varastot vastaavat puolestaan datan lataamisesta sovellukseen ja datan järjestämisestä mallin avulla. (Vino 2014.)
5.2.2 Luokkajärjestelmä

Sencha Touchia kirjoitetaan hyvin samaan tapaan kuin olio-ohjelmointikieliä; sen syntaksi perustuu olio-ohjelmoinnin tapaan luokille ja niiden instansseille eli esiintymille. Sencha Touchin luokkajärjestelmä perustuu kahdelle pääfunktiolle: Ext.define() ja Ext.create(), joista ensimmäisen avulla voidaan määrittää uusia luokkia ja kaikki niiden määrittämiseen tarvittavat tiedot, ja toisella luodaan olemassa olevista luokista uusia instansseja (Kosmaczewski 2013, 23-27). Yksinkertainen kontakttilistaluokka voitaisiin luoda esimerkiksi seuraavasti:

```javascript
Ext.define('HelloWorld.view.ContactList', {
    extend: 'Ext.dataview.List',
    xtype: 'contactlist',
    config: {
        /* Luokan määritykset */
    }
});
```

Yllä olevassa koodissa määriteltiin uusi luokka HelloWorld.view.ContactList, joka luodaan laajentamalla Sencha Touchin olemassa olevaa Ext.dataview.List-luokkaa. Tästä kontakttilistaluokasta voidaan luoda uusi instanssi seuraavasti:

```javascript
var list = Ext.create('HelloWorld.view.ContactList', {
    items: [
        /* Listaelementit */
    ],
    title: 'Yhteystiedot',
    /* Ym. parametrit eroteltuina pilkulla */
});
```

Yksi Sencha Touchin luokkajärjestelmän tyypillisimmistä elementeistä on luokkien xtype-ominaisuus. Xtype-nimi on lyhyempi ja helpompi muistaa ja käyttää kuin luokan nimi ja se helpottaaakin ohjelmistokehittäjän työtä. Xtype-ominaisuutta voidaan käyttää Ext.create()-funktion sijaan luodessa uusia instansseja luokista. Esimerkiksi äsken luotuun kontakttilistaan saadaan helposti lisättyä yläpalkki, jossa on otsikko ”Yhteystiedot” ja yksi painike, xtype-ominaisuuksien avulla seuraavasti:

```javascript
Ext.define('HelloWorld.view.ContactList', {
    extend: 'Ext.dataview.List',
    xtype: 'contactlist',
    config: {
```

5.3 Sencha Touchin käyttöönotto


Sencha Touch tarvitsee myös Rubyn luomaan Sencha Touchin käyttämän käännetyyn CSS-tyylitiedostoon. Ruby on dynaaminen avoimen lähdekoodin ohjelmointikieli, joka keskittyy selkeyteen ja tuottavuuteen (Ruby Programming Language 2014). Kun kaikki vaadittavat ohjelmistot on asennettu, voidaan ladattava itse Sencha Touch sen virallisilta verkkosivuilta. Pakatut tiedostot pureetaan sovelluskehittäjän projekteille varat-
tuun työtilaan, joka on yleensä lokaalissa web-kehityksessä käytettävän web-serverin www-kansio. Sencha Cmd-työkalun asennuksen voi tarkistaa menemällä komentorivillä kansioon, jonne Sencha Touch purettiin ja ajamalla komennon ”sencha”. Tämä tulostaa käytössä olevan Sencha Cmd:n versionumeron ja tiedot. Mikäli Sencha Cmd on asennettu oikein, voidaan ensimmäinen Sencha Touch sovellus luoda seuraavasti:

```bash
sencha generate app HelloWorld ../hello
```

Jotta komento toimisi, täytyy sovelluskehittäjän olla kansiossa, johon Sencha Touch purettiin. Komento luo hello-projektikansion HelloWorld-nimiselle sovellukselle Sencha Touch -kansion ulkopuolelle. (Sencha Docs 2014.)

### 5.4 Sencha Touch ja PhoneGap

Sencha Touch on pääasiassa laaja käyttöliittymäkehys, joka voidaan yhdistää PhoneGapin kanssa mobiililaitteen natiivien ominaisuuksien hyödyntämiseksi. Kun Sencha Touch -sovellus on luotu käyttäen Sencha Cmd-ohjelmistoa, se voidaan laajentaa käsittämään myös PhoneGap seuraavalla komennolla:

```bash
sencha phonegap init com.example.hello HelloWorld
```

Komennon kaksi viimeistä parametria ovat vapaaehtoisia. Jotta komento toimisi, täytyy komentorivillä ensin navigoida Sencha Touch -sovelluksen hakemistoon. Komento lisää sovelluksen projektikansioon phonegap-hakemistoon, joka sisältää PhoneGapin tarvitsemat tiedostot. (Gerbasi 2013.)

Sencha Touch -sovelluksen työnkulku ei muutu PhoneGapin liittämisensä jälkeen juuri lainkaan. Sovellus voi käyttää PhoneGapin tarjoamia rajapintoja hyödyntääkseen laitteen natiiveja ominaisuuksia, mutta rakentuu muuten samalla tavalla kuin mikä tahansa Sencha Touch -sovellus. Sovellus voidaan paketoida jakelua varten joko Senchan omalla Sencha Mobile Packager -työkalulla, Apache Cordovalla tai PhoneGap Build -pilvipalvelussa. Phonegap.local.properties-tiedostoon voidaan määrittää kohdealistat ja valita pakkausmenetelmäksi PhoneGap Build seuraavasti:

```bash
phonegap.platform=android ios wp8
```
Jos ei haluta käyttää PhoneGap Build -pilvipalvelua sovelluksen pakkaamiseen, on huolehdittava kunkin kohdealustan vaatimasta kehitysympäristöstä. (Boonstra 2013.)
6 jQuery Mobile

6.1 Yleistä

jQuery Mobile on yhden maailman tunnetuimman ja suosituimman JavaScript-kirjaston, jQueryn, liitännäinen. jQueryn ensimmäinen versio julkaistiin vuonna 2006 ja se on tällä hetkellä käytettyin JavaScript-kirjasto web-sivustoilla (Usage of JavaScript libraries for websites 2014). jQuerylla voidaan vaivattomasti toteuttaa muun muassa animaatioita tai ajax-kyselyjä sekä se helpottaa html-dokumentin eri elementtien käsittelyä. jQueryn syntaksi on tehty mahdollisimman yksinkertaiseksi ja selkeäksi, mikä on osaltaan vaikuttanut sen suosioon web-kehittäjien keskuudessa. (jQuery 2014.)

jQuery Mobile on käyttöliittymäkehys, jonka avulla voidaan toteuttaa web-sovelluksia mobiililaitteille vaivattomasti; se tukee muun muassa iOS-, Android- ja Windows Phone 7.5 ja 8 -laitteita (jQuery Mobile 1.4 Supported Platforms 2014). jQuery Mobilen ensimmäinen versio julkaistiin loppuvuodesta 2010 ja uusin vakaa versio julkaistiin helmikuussa 2014. jQuery Mobile -sovellukset rakentuvat jQuery-ytimen päälle joten ne vaativat jQueryn toimia. (Ortiz 2011.)

6.2 jQuery Mobile -sovelluksen rakenne


```html
<!doctype html>
<html>
<head>
    <title>Hei Maailma!</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
```

Koska jQuery Mobile on käyttöliittymäkehys, se sisältää runsaasti erilaisia valmiita käyttöliittymäkomponentteja (ks. kuvio 9, s. 37), joiden avulla sovelluksen rakentaminen sujuu vaivatta. Näihin komponentteihin lukeutuvat muun muassa painikkeet eli buttonit, paneelit, listanäkymät, popup- eli ponnahdusikkunat, lomake-elementit sekä erilaiset työkalupalkit.
Erilliset näkymät luodaan kaikki samaan index-html-tiedostoon div-elementteinä, joille annetaan datarooliksi page eli sivu.Navigointi eri näkymien väillä tapahtuu yksinkertaisimmin ankkurilinkkien avulla: painikkeen tai linkin osoitteen (href) asetetaan kohdenäkymän tunnus eli id ankkurilinkkinä seuraavasti:

`<a href="#page2" data-role="button">Seuraava</a>`

Yllä olevalla koodilla luodaan painike “Seuraava”, jota painamalla sovellus navigoi näkymään, jonka id:ksi on asetettu “page2”. Käyttöliittymäkomponenttien lisäksi jQuery Mobile sisältää useita eri metodeja. Esimerkiksi `.mobile.activePage`-metodilla voidaan selvittää, missä näkymässä käyttäjä juuri siltä hetkellä on ja `.mobile.changePage`-metodin avulla voidaan navigoida sovelluksessa haluttuun näkymään ilman, että käyttäjän tarvitsee painaa linkkiä. (Ortiz 2011.)
6.3 jQuery Mobile -sovelluksen käyttöönotto


6.4 jQuery Mobile ja PhoneGap

7 Aarrejahti-sovellus

7.1 Lähtökohdat ja sovelluksen kuvaus

Usealla jyväskyläläisen mainostoimiston, MEOM Oy:n, asiakkaalla on ollut käynnissä erilaisia aarteenetsintäkampanjoita. Näissä kampanjoissa kampanjan järjestäjä on piilottanut Jyväskylän alueelle ”aarteen”, joka aarteenetsikilpailun osallistujien on tarkoitus löytää. Sosiaalisessa mediassa, esimerkiksi kilpailun järjestäjän Facebook-sivulla, on jaettu vihjekuva, jonka perusteella kilpailijat ovat lähteneet etsimään aarretta. Kilpailijoiden etsimät aarteet ovat olleet joko julisteita tai tarroja, joissa oleva koodi aarteen löytäneen kilpailijan on käytävä syöttämässä kampanjalle tehdyn websivun lomakkeeseen. Lomakkeeseen syötetty aarteen koodi tarkistetaan ja mikäli koodi on oikein ja palkintoa ei ole kukaan muu kilpailija vielä voittanut, kilpailuun osallistunut täyttää lomakkeeseen myös yhteystietonsa palkinnon lähettämistä varten.

7.2 Sovelluksen toteutus

7.2.1 Sovelluksen vaatimukset


7.2.2 Käyttöavuisten teknologioiden valinta

Hybridisovelluksen toteutukseen valittiin PhoneGap-ohjelmistokehys, jonka sijaan geolocation-rajapintaa käytettiin käytävän sijainnin määrittämiseen. PhoneGap-sovelluksen kehittämiseen on olemassa kaksi eri lähestymistapaa: PhoneGapin oma sisäänrakennettu buildaus eli paketointi, jolloin jokaiselle eri kohdekokoonpanolle täytyy olla niiden vaatimat sovelluskehitysympäristönsä (ks. taulukko 2, s. 12), ja PhoneGap Build-pilvipalvelun käyttäminen sovelluksen paketoinmiseen, jolloin sovellus kirjoitetaan kerran ja paketoidaan pilvipalvelussa kaikille kohdevesiin. Jotta sovelluskehitys olisi mahdollisimman vaivatonta, Aarrejahti-sovelluksen toteutukseen päätettiin käyttää Phonegap Build-palvelua sovelluksen paketointiin Androidille, iOSille ja Windows Phone 8:ille. Opinnäytetyössä tutustuttiin myös kahteen websovellusten toteuttamiseen tarkoitetuun käyttöliittymäkehitykseen: Sencha Touchiin ja jQuery Mobileen, joista jQuery Mobile valittiin sovelluksen käyttöliittymän toteutukseen.

Sencha Touch ja jQuery Mobile eroavat toisistaan monin tavoin: Sencha Touch sisältää käyttöliittymäkomponenttien lisäksi oman sisäänrakennetun MVC-arkkitehtuurinsa, jota jQuery Mobilesta ei valmiina löydy. Halutessaan sovelluskehitäjä voi yhdistää jQuery Mobileen jonkin MVC-arkkitehtuuriin sisältävän sovelluskehyksen, kuten Backbone.js:stä tai Angular.js:stä. Kokeneelle jQuery-kehittäjälle jQuery Mobilen omaksuminen tulee lähes luonnostaan kun taas Sencha Touchin luokkajär-
jestelmä ja täysin JavaScriptillä kirjoitettu sovelluslogiikka voivat olla hankalia sisäis- tää. JQueryyn suuren suosion myötä myös jQuery Mobile on saanut taaksensa paljon käyttäjiä ja vahvan yhteisön: mm. ohjelmistokehittäjien suosimalta Stackoverflow-sivustolta löytyy runsaasti tukea jQuery Mobile -sovelluksen tekoon. Sencha Touch -sovelluskehittäjän on huomattavasti vaikeampi löytää tukea ja apua ongelmiin Senchan oman virallisena sivustona ulkopuolelta. Senchan oma tukipalvelukin on maksullinen lisäpalvelu.


Kuvio 10. Themroller-työkalun käyttöliittymä

Sencha Touchin debuggaaminen on jQuery Mobileen verrattuna vaikeaa: jos koodissa on virhe, se ei välttämättä näy kehittäjän työkalujen konsolissa ollenkaan. Pienetkin virheet Sencha Touch -sovelluksen koodissa voivat aiheuttaa koko sovelluksen toimimattomuuden ja kokemattoman JavaScript-kehittäjän on vaikea löytää virhettä. Sencha Touchin dokumentaatio puolestaan on hyvinkin laaja ja kattava: käyttöliittymäkomponenttien lisäksi dokumentaatio käy läpi muun muassa metodit, luokkajärjestelmän, datan käsittelyn ja näkymien luonnin. JQuery Mobilen dokumentaatio...
keskittyvät lähinnä esittämään esimerkkejä käyttöliittymäkomponenteista ja muu osuus jää lähes käsittelemättä.


Taulukko 4. Sencha Touch ja jQuery Mobile -käyttöliittymäkehyksen vertailu

<table>
<thead>
<tr>
<th>Sencha Touch</th>
<th>jQuery Mobile</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nopea &amp; sulava</td>
<td>Helppo &amp; nopea käyttöönotto</td>
</tr>
<tr>
<td>Tuntuu ja näyttää natiivilta</td>
<td>Paljon ulkopuolista tukea (mm. Stackoverflow)</td>
</tr>
<tr>
<td>MVC-arkkitehtuuri</td>
<td>Themeroller-teemat</td>
</tr>
<tr>
<td>Kattava &amp; laaja dokumentaatio</td>
<td>Helppo virheiden debuggaus</td>
</tr>
<tr>
<td>Monimuutkainen → hidas käyttöönotto</td>
<td>Voi olla melko hidas</td>
</tr>
<tr>
<td>Ei juurikaan apua virallisten sivujen ulkopuolelta, tukipalvelu maksullinen</td>
<td>Sovellukset näyttävät kaikki usein samanlaisilta</td>
</tr>
<tr>
<td>Virheiden debuggaus vaikeaa</td>
<td>Dokumentaatio vaillinainen</td>
</tr>
</tbody>
</table>

7.3 Arkkitehtuuri

Aarrejahti-sovellus on arkkitehtuurillisesti melko yksinkertainen. Koska sovelluksen paketointiin käytetään PhoneGap Build -pilvipalvelua, ei PhoneGapista tarvita kuin www-kansio ja config.xml-tiedosto, johon määritellään sovelluksen asetukset PhoneGap Buildin pakkaustta varten. Koska sovelluksen käyttöliittymän toteutukseen valittiin jQuery Mobile, tarvittiin sekä jQueryn että jQuery Mobilen vaatimat tiedostot. Sovelluslogiikka on kirjoitettu yhteen JavaScript-tiedostoön ja käyttöliittymä rakennettu index.html-tiedostoon. Sovelluksen vaatimat kunkin eri kohdealaustan kuvak-
keet ja aloitusnäyttökuvat (splashscreen) sijoitettiin res-kansioon. Sovelluksen saamiseksi helposti PhoneGap Buildiin sovelluksessa käytettiin Git-versionhallinann GitHub-työkalua, joka lisää sovelluskansion juureen omat tiedostonsa. Koko sovelluksen kansiorakenne on kuvattu kuviossa 12.

Kuvio 12. Aarrejahti-sovelluksen kansiorakenne

Itse sovellus koostuu kampanjoista ja niihin kuuluvista aarrejahdeista. Sekä kampanjat että aarrejahdit haetaan sovellukseen JSON-tiedostosta palvelimelta. JSON (JavaScript Object Notation) on muun muassa JavaScript-ohjelmissa käytettävä yksinkertainen tiedonsiirtomuoto, jota käytetään nykyään erittäin laajasti kaikessa asiakkaan ja palvelimen välisessä liikenteessä. Aarrejahti-sovelluksessa käytettävä JSON-tiedosto on rakenteeltaan seuraavanlainen:

```json
{
    "campaigns": [
    {
        "campaignID": "campaign01",
```
"campaignName": "Kampanja1",
"campaignLogo": "kampanja1logonosoite.jpg",
"campaignDate": "22.03.2014",
"campaignDescription": "Kampanja1:n kuvaus.",
"treasurehunts": [ { "treasurehuntID": "campaign01treasurehunt01",
  "treasurehuntName": "Aarrejahti1",
  "treasurehuntLat": "62.24162",
  "treasurehuntLon": "25.75973",
  "treasurehuntStatus": "0",
  "treasurehuntCode": "KAMPANJAN1AARREJAHDIN1KOODI",
  "treasurehuntStep1": "http://vihjekuvan1osoite.jpg",
  "treasurehuntStep2": "http://vihjekuvan2osoite.jpg"
} ]
]
}
]

JSON-tiedostossa campaigns-sisältää kaikki sovelluksen kampanjat. Kampanjoilla on aina tunnus eli ID, joka on muotoa campaignXX, jossa XX on kampanjan numero. Tunnuksen lisäksi kampanjoille asetetaan nimi, logo-kuvan osoite, kampanjan lisäyspäivämäärä, kuvaus ja kaikki kampanjaan kuuluvat aarrejahdit. Aarrejahdeilla on myös tunnus, joka on aina muotoa campaignXXtreasurehuntXX, jossa campaignXX on sen kampanjan tunnus, johon ko. aarrejahti kuuluu ja treasurehuntXX on aarrejahdin oma tunnus. Aarrejahdeille annetaan nimi, aartein sijainnin leveys- ja pituusasteet (latitude ja longitude), aarrejahdin tila eli status (0 tarkoittaa käynnissä olevaa aarrejahtia ja 1 suljettua), aarrein koodi sekä kahden vihjekuvan osoitteet.

7.4 Sovelluksen esittely

Yksittäinen aarrejahti alkaa aarrejahdin 1. vaiheella, jossa aarteensijalle näytetään ensimmäinen vihjekuva (ks. kuvio 16). Vihjekuvan yllä on aarrejahdin etenemistä kuvaava edistymispalkki (progress bar), joka täytyy sitä mukaa kun aarteensijä lähenee aarten sijaintia, sekä otsikko, jossa oletuksena lukee ”Etsitään sijaintia”.

Kun paikallistaminen onnistuu, otsikon sisältö riippuu aarteensijan sijainnista aarteeseen nähden: ”Et ole lähellä aarretta, etsiskelehän vielä”, ”Olet vielä melko kaukana aarteesta”, ”Olet melko lähellä aarretta” ja ”Olet lähellä aarretta!”. Vihjekuvan alla on myös edistymisestä tiedottava vihje, joka aarteensijän sijainnista riippuen on joko ”kylmenee”, ”lämpenee”, ”ei kylmene eikä lämpene” tai ”polttaa”. Aarteensijällä on mahdollisuus tarkistuttaa sijainnista vihjekuvan alta löytynytä paikankeesta. Oletuksena sovellus tarkistaa käyttäjän sijainnin 5 sekunnin välein ja päävittää tarvittaessa vihjeet ja edistymispalkkin. Kun aarteensijä on tarpeeksi lähellä aarten sijaintia, vihjekuvan alle ilmestyy painike, josta pääsee seuraavaan vaiheeseen (ks. kuvio 16 oikealla).

**Kuvio 16. Aarrejahti-sovelluksen yksittäisen aarrejahdin 1. vaihe**

Aarrejahdin 2. vaiheen käyttöliittymä on täysin identtinen 1. vaiheen kanssa: vain vihjekuva on eri. Toisen vihjekuvan on tarkoitus olla tarkempi kuin ensimmäisen, joka on suurpiiriteisempi vihje siitä alueesta Jyväskylää, jolle aarre on kätetty. 2. vaiheessa on samat vihjeet ja edistymispalkki kuin 1. vaiheessakin ja kun aarteensijä on aivan aarten lähetyvillä, pääsee hän siirtymään kolmanteen ja viimeiseen vaiheeseen.

![Kuvio 17. Aarrejahti-sovelluksen yksittäisen aarrejahdin 3. vaihe](image)

### 7.5 Sovelluksen testaus

#### 7.5.1 Sovelluksen testaus Ripple-PhoneGap-emulaattorilla


Ripple-emulaattori tukee osaa PhoneGapin ohjelmointirajapinnoista, joten sillä voidaan hyvin demota esimerkiksi sijainnin määrittämistä. Oman sijainnin voi emulaattorin määrittää joko tarkoina leveys- ja pituusasteina tai oikeassa laidassa olevan kartan avulla. Aarreensinnän edistymistä pystyttiin simuloimaan siirtelemällä kartan osoitinta lähemmäs tai loitommas aarten sijaintiin nähden. Emulaattorissa on
myös mahdollisuus valita kohdealusta pienestä määrästä erilaisia mobiililaitteita. Kuviossa 18 (s. 50) laitteeksi on valittu Samsung Galaxy Nexus -älypuhelin. Sovellusalueen koko ja muoto muuttuvat valitun kohdealustan mukaan.

7.5.2 Sovelluksen testaus älypuhelimilla

Jotta PhoneGap Build -sovellusta voitaisiin testata mobiililaitteella, täytyy se ensin pakata eli buildata. Pakkaamista varten sovellustiedostot täytyy ladata PhoneGap Buildiin joko zip-pakettina tai vaihtoehtoisesti hakea sovelluskehittäjän GitHub-talletuspaikasta, kuten Aarrejahti-sovelluksen tapauksessa tehtiin. Tällöin PhoneGap Buildiin kirjaudutaan sisään GitHubin tunnuksilla, sovellustyypiksi valitaan avoimen lähdekoodin (open source) sovellus ja sovelluksen pakattavien tiedostojen hakemisto valitaan sovelluskehittäjän GitHub-tunnukseen alla olevista talletuspaikoista. Kun talletuspaikka on valittu PhoneGap Build pakkaa sovelluksen kaikille niille kohdealustoielle, jotka on määritelty sovelluksen config.xml-tiedostoon (ks. kuvio 19). iOS-sovellukset vaativat sovelluksen pakkausta varten iOS-allekirjoitusavaimen, joka ladataan erikseen PhoneGap Buildiin.

Kuvio 19. Aarrejahti-sovellus PhoneGap Buildissa

Android-laitteella voi paketoitua sovellusta testata lukemalla PhoneGap Buildista sovelluksen QR-koodin (ks. kuvio 19) tai vaihtoehtoisesti avamaalla laitteen internetselaimella sovelluksen latausosoitteen. Molemmissa tapauksissa Android-mobiililaitte


Windows Phone 8 -älypuhelimet täytyy ennen testausta rekisteröidä sovelluskehityskäyttöön. Tätä varten tarvitaan Windows Phone 8 SDK ja joko Microsoft-tunnus (ent. Windows Live) tai Windows Phone -kehittäjätunnus. Windows Phone 8 SDK vaatii Windows 8 -käyttöjärjestelmän eikä sitä voi asentaa vanhemmille Windowsin versioille. SDK sisältää Windows Phone -laitteiden rekisteröintityökalun (Windows Phone Developer Registration Tool), jonka avulla puhelin rekisteröidään sovelluskehitystä varten. Kun puhelin on rekisteröity, voidaan PhoneGap Buildissa pakattu sovellus asentaa siihen Windows Phone 8 SDK:n sovellusten käyttöönottoon tarkoitetulla työkalulla (Application Deployment tool). Tätä varten puhelin on kytkettävä tietokoneeseen USB-kaapeliolla.

<table>
<thead>
<tr>
<th>Ominaisuus</th>
<th>iPhone 4</th>
<th>Samsung Galaxy S2</th>
<th>Nokia Lumia 820</th>
</tr>
</thead>
<tbody>
<tr>
<td>Käyttöjärjestelmä</td>
<td>iOS 7</td>
<td>Android 4.1.2 Jelly Bean</td>
<td>Windows Phone 8</td>
</tr>
<tr>
<td>Näyttö (tuumaa)</td>
<td>3,5</td>
<td>4,27</td>
<td>4,3</td>
</tr>
<tr>
<td>Resoluutio (pikseliä)</td>
<td>960 x 640</td>
<td>800 x 480</td>
<td>800 x 480</td>
</tr>
<tr>
<td>Prosessori</td>
<td>Apple A4 800 MHz</td>
<td>ARM Cortex-A9 1,2 GHz tuplaydin</td>
<td>Snapdragon S4 1,5 GHz tuplaydin</td>
</tr>
<tr>
<td>RAM-muisti</td>
<td>512 Mb</td>
<td>1 Gt</td>
<td>1 Gt</td>
</tr>
</tbody>
</table>


iPhone 4:n näyttö on kooltaan testilaitteista pienin, mutta sen resoluutio on suurin. Sovellus ei mahtunut iPhonen näytölle aivan yhtä hyvin kuin Galaxy S2:n ja Lumia 820:n ja värit olivat haaleammat kuin muilla testilaitteilla (ks. kuvio 20, s. 54). iPhoneella sovelluksen alanäyttöjen oikeaan yläreunaan sijoitettu takaisin-painike toimi vaihtelevasti: iOS-sovelluksille olisi voinut tehdä oman takaisin-painikkeen sovelluksille.

Taulukko 5. Aarrejahti-sovelluksen testaukseen käytetyt laitteet

Kuvio 20. Aarrejahti-sovellus Samsung Galaxy S2-, iPhone 4- ja Nokia Lumia 820 -älypuhelimissa

7.6 Jatkokehitys

Opinnäytetyötä varten tehdystä Aarrejahti-sovelluksesta jätettiin tarkoituksella kokonaan huomioimatta sovelluksen ylläpitopuoli, joka on yksi ensimmäisistä jatkokehityksessä tehtävistä asioista. Ylläpidon tulisi olla mahdollisimman vaativata ja helppoa, sillä kampanjaiden ylläpitäjät ovat usein tietämättömiä ohjelmoinnista ja ohjelmointikäsistä. Aarrejahtien sijainnit tallennetaan JSON-tiedostoon leveys- ja pituusasteina: aarrejahdin ylläpitäjän täytysi kyetä helposti tallentamaan uuden aarrejahdin sijainnin, kun aarteen kätköpaikka on selvillä. Tämän voisi mahdollisesti hoitaa joko omalla erillisellä sovelluksella, joka kehitettäisiin täysin Aarrejahdin ylläpitoa varten tai nykyiseen sovellukseen voitaisiin tehdä ylläpitäjien alue, jonne he voisivat kirjautua omilla tunnuksillaan. Ylläpitäjä voi aarrejulistetta tai -tarraa kiinnittäessään
tallentaa sen hetkisen sijaintinsa ylläpitosovelluksella aarteen sijainniksi. Samalla ylläpitäjä voisi ottaa mobiililaitteensa kameralla aarrejahdin molemmat vihjekuvat ja tallentaa ne aarteen sijainnin koordinaattien yhteydessä.


Aarrejahtien koodin voisi jatkokokehtyksessä korvata QR-koodilla, joka sitten luettaisiin sovelluksessa QR-koodilukijalla. Käyttäjälle voitaisiin myös ilmoittaa sovelluksen uu-
sista aarrejahdeista ja kampanjoista push-notifikaatiolla eli työntöilmoituksella, jolloin sovelluksen kuvakkeen yhteyteen tulee pieni ilmoitusikoni, kun uusia aarrejahdeja tai kampanjoita on käynnistynyt. Tällä hetkellä sovellus toimii vain Jyväskylän alueella, mutta tulevaisuudessa se voitaisiin laajentaa eri kaupunkeihin: sovelluksen alussa voisi valita minkä kaupungin kampanjoihin tai aarrejahteihin haluaa osallistua tai halutut kaupungit voisi asettaa oman tilin tietoihin ja muokata niitä tarvittaessa. Sovellus voisi myös hakea lähimmät aarrejahdit käyttäjän sijainnin perusteella, jolloin ei tarvitsisi erikseen määritellä kaupunkia.
8 Pohdinta

Opinnäytetyössä tutkittiin alustariippumattomia hybridisovelluskehitystä mobiililaitteille ja siinä keskityttiin PhoneGap-sovelluskehitykseen ja sen hyödyntämiseen hybridisovelluskehityksessä. Opinnäytetyössä selvitettiin PhoneGap-sovelluskehysten soveltumista hybridisovelluskehitykseen sekä tutkittiin hybridisovellusten mahdollisuuksia, vahvuksia ja heikkouksia verrattuna web- ja nativiisovelluksiin.


PhoneGap-sovelluskehityksen yksi haasteellisimmista seikoista on sovelluksen testaus. PhoneGap-sovellusta voidaan kyllä web-sovelluksen tapaan testata internetseilaimella, mutta selaimella ei pysty kokeilemaan nativiivien ominaisuuksien toimivuutta. Opinnäytetyössä tehdyn Aarrejahti-sovelluksen sijaintirajapinnan tarkkuutta testattiin pääasiassa Ripple-emulaattorilla, joka soveltui testaukseen oikein hyvin. Ripplellä ei kuitenkaan saada testattua kaikkia PhoneGapin Javascript-rajapintoja: se tukee vain sijainti-, kihtyyysmittari- ja yhteysrajapintoja sekä joitakin PhoneGapin tapahtumia eli eventtejä (esimerkiksi deviceready). Testaaminen oikeilla laitteilla eroaa valitusta buildaus- eli paketointimenetelmästä: mikäli käytetään PhoneGap Build - pilvipalvelua, sovelluksen projektitiedostot tulee joka testauskerralla hakea uudelleen GitHubista, mikäli koodiin on tullut muutoksia, ja sovellus on pakattava uudel-
leen, jonka jälkeen se asennetaan testilaitteelle. Natiivia sovelluskehitystä muistuttaavana lähdejärjestelmissä, jossa jokaisella eri kohdealustalla on oma kehtisyydistäntä, testaaminen mobiililaitteilla on nopeampaa: sovelluskehitysympäristöstä valitaan usb-kaapelillä kytketty laite sovelluksen testauskohteeksi, jolloin sovellus ajetaan paikallisesti laitteessa.


Tämän opinnäytetyön tekohetkellä kaikkein vaivattomin tapa sovelluskaupoissa jael- tavien alustariippumattomien sovellusten tekoon lienee Aarrejahti-sovelluksessa käytetty PhoneGap Build -pilvipalvelu. PhoneGap Build ei tosin sovellu joka tilanteeseen eikä sillä voida korvata nativjeja sovelluksia: parhaiten se sopii yksinkertaisten web-sovellusten paketointiin natiiiveiksi. PhoneGapin nativiivia muistuttava lähestymis- tapa mobiilisovelluskehitykseen mahdollistaa jo monimutkaisempien sovellusten toteutuksen ja siinä voidaan huomioida kuka eri kohdealusta erikseen, mikä taas parantaa käyttäjäkokemusta. Sovelluskehittäjällä täytyy tosin olla tällöin kohdealus- tojen vaatimat kehitysympäristöt ja laitteet käytössä, mutta hän pystyy kierrättämään melko vaivattomasti kerran kirjoitettua koodia eri sovellusten välillä.

Lähteet


