KARELIA-AMMATTIKORKEAKOULU
Tietojenkäsittelyn koulutusohjelma

Matti Mononen

HYBRIDIMOBIILISOVELLUKSIEN LAAJENTAMINEN ERI MOBIILIKÄYTTÖJÄRJESTELMILLE

Opinnäytetyö
Joulukuu 2014

Karelia AMMATTIKORKEAKOULU	**OPINNÄYTETYÖ** **Joulukuu 2014** **Tietojenkäsittelyn koulutusohjelma** Karjalankatu 3 80200 JOENSUU p. 013 260 600

Tekijä(t)
Matti Mononen

Nimike
Hybridimobiilisovelluksien laajentaminen eri mobiilikäyttöjärjestelmille

Toimeksiantaja
FastROI Oy

Tiivistelmä

Opinnäytetyössä tarkastellaan niin sanottuja hybridimobiilisovelluksia ja niiden laajentamista eri mobiilikäyttöjärjestelmille. Opinnäytetyössä selvitetään, mitä hybridimobiilisovellukset ovat sekä mitä hybridimobiilisovelluksen laajentaminen alustalta toiselle vaatii. Opinnäytetyön käytännön osuudessa tarkastellaan hybridimobiilisovelluksen laajennusprosessia esimerkkisovellusta apuna käyttäen. Lisäksi käytännön osuudessa tarkastellaan esimerkkisovelluksen tukemia mobiilikäyttöjärjestelmiä sekä sovelluksen toteutuksessa käytettyjä teknologioita. Esimerkkisovelluksena käytetään FastROI Oy:n HILKKA-sovellusta.

Opinnäytetyön teoriaosuudessa tarkastellaan hybridimobiilisovelluksia yleisellä tasolla. Teoriaosuudessa selvitetään muun muassa, mitä hybridimobiilisovellukset ovat ja kuinka ne eroavat niin sanotuista natiivisovelluksista sekä mobiililaitteille optimoiduista web-applikaatioista. Lisäksi teoriaosuudessa tarkastellaan hybridimobiilisovelluksien kehittämistä.

Opinnäytetyön tuloksena esimerkkisovelluksena käytetty HILKKA-sovellus laajentui Microsoftin Windows Phone 8 -käyttöjärjestelmälle.

Kieli suomi	Sivuja 37 Liitteet 1 Liitesivumäärä 3

Asiasanat
hybridi, mobiili, laajentaminen, Android, Windows Phone

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
December 2014
Degree Programme in Business Information Technology

Karjalankatu 3
80200 JOENSUU
p. 013 260 600

Author(s)
Matti Mononen

Title
Expanding Hybrid Mobile Applications for Different Platforms

Commissioned by
FastROI Oy

Abstract

The thesis examines the hybrid mobile applications and explains how to expand them for different platforms. The research questions were as follows: What are the hybrid mobile applications and how to expand the hybrid mobile application for different platforms. The practical part of the thesis examines the expanding process of the hybrid mobile application by using an example application. Furthermore, the mobile operating systems and technologies used by the example application are examined. The used example application is FastROI HILKKA application.

The theoretical part of the thesis examines the hybrid mobile applications in general level. The theoretical part explains what hybrid mobile applications are and how they differ from the native and mobile optimized web applications. In addition, the theoretical part examines the developing process of the hybrid mobile applications.

The result of the thesis HILKKA application expanded into Microsoft's Windows Phone 8 operating system.

Language	Pages 37
Finnish	Appendices 1
	Pages of Appendices 3

Keywords
Hybrid, Mobile, Expanding, Android, Windows Phone

Sisältö

1 Johdanto

Kiinnostus valitsemaani aihetta kohtaan heräsi alkusyksystä 2013, kun aloitin sovelluskehittäjän työtehtävät FastROI Oy:llä. Siellä keskeisimpiin työtehtäviini kuuluivat muun muassa hybridimobiilisovelluksien kehittäminen Googlen Android- ja Microsoftin Windows Phone 8 -mobiilikäyttöjärjestelmille. Opinnäytetyöni tarkoituksena on tarkastella hybridimobiilisovelluksia sekä niiden laajentamista eri mobiilikäyttöjärjestelmille mahdollisimman kattavasti. Opinnäytetyöni tutkimuskysymykset ovat seuraavat: mitä hybridimobiilisovellukset ovat ja mitä hybridimobiilisovelluksen laajentaminen alustalta toiselle vaatii. Päädyin opinnäytetyöaiheeseen työnantajan ehdotuksesta aloitettuamme yrityksen HILKKA-sovelluksen laajennusprosessin joulukuussa 2013. Laajennusprosessin aikana Googlen Android-käyttöjärjestelmälle toteutettu hybridimobiilisovellus laajennettiin Microsoftin Windows Phone 8 -käyttöjärjestelmälle yhteensopivaksi.

Opinnäytetyöni toisessa luvussa keskityn tarkastelemaan hybridimobiilisovelluksia yleisellä tasolla. Toisessa luvussa selvitetään muun muassa, mitä hybridimobiilisovellukset ovat ja kuinka ne eroavat niin sanotuista natiivisovelluksista sekä mobiililaitteille optimoiduista web-applikaatioista. Lisäksi toisessa luvussa käsitellään muun muassa hybridimobiilisovelluksien vahvuuksia ja heikkouksia sekä kehittämistä. Toisen luvun olennainen tavoite on selventää, mitä ovat hybridimobiilisovellukset ja miten ne eroavat niin sanotuista natiivisovelluksista sekä mobiililaitteille optimoiduista web-applikaatioista.

Opinnäytetyöni kolmannessa luvussa keskityn tarkastelemaan hybridimobiilisovelluksen laajentamista esimerkkiprojektin avulla. Esimerkkiprojektina käytetään FastROI Oy:n HILKKA-mobiilisovellusta ja sen laajennusprosessia. Tämän lisäksi kolmannessa luvussa käsitellään HILKKA-mobiilisovelluksen tukemia mobiilikäyttöjärjestelmiä ja sovelluksen toteutuksessa käytettyjä teknologioita sekä selvitetään, miten ja miksi kyseisiin teknologiavalintoihin on päädytty. Kolmannen luvun lopuksi teen yhteenvedon HILKKA-sovelluksen laajennusprosessista tarkastelemalla projektin aikana saavutettuja tuloksia sekä suurimpia haasteita.

2 Hybridimobiilisovellukset

Tässä luvussa keskitytään tarkastelemaan hybridimobiilisovelluksia yleisellä tasolla. Tässä luvussa selvitetään myös, miten hybridimobiilisovellukset eroavat niin sanotuista natiivisovelluksista ja mobiililaitteille optimoiduista web-applikaatioista. Lisäksi tässä luvussa tarkastellaan hybridimobiilisovelluksien vahvuuksia ja heikkouksia sekä tutustutaan hybridimobiilisovelluksien kehittämiseen.

2.1 Mikä on hybridimobiilisovellus

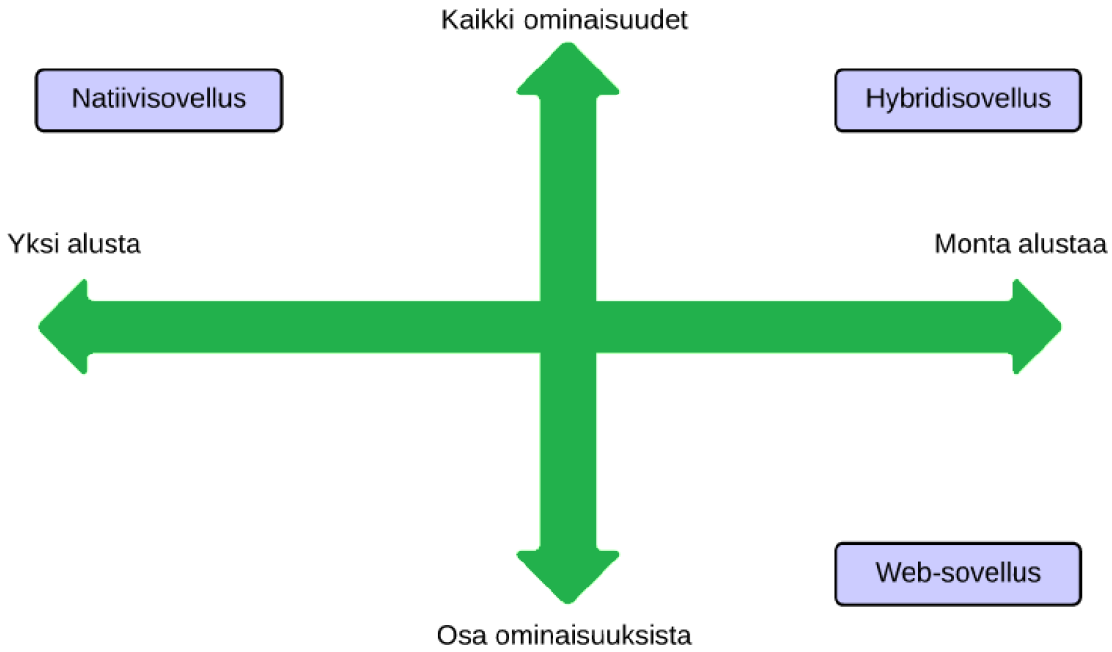
Hybridimobiilisovelluksien tai hybridisovelluksien termi on erittäin laaja käsite. Puhuttaessa mobiilisovelluksista hybridisovelluksen termillä tarkoitetaan yleensä sovellusta, joka on puoliksi niin sanottu web-applikaatio ja puoliksi niin sanottu natiivisovellus. Tämä tarkoittaa sitä, että hybridisovellukset perustuvat erittäin vahvasti erilaisiin web-tekniikoihin, kuten HTML-merkintäkieleen, CSS-tyylimäärittelyihin, JavaScript-ohjelmointikieleen tai lyhyesti HTML5-web-tekniikoihin. Puhtaista web-applikaatioista poiketen hybridisovellukset pystyvät kuitenkin hyödyntämään mobiililaitteiden tarjoamia ominaisuuksia sekä mobiilikäyttöjärjestelmien omia ohjelmointirajapintoja lähes täysin natiivisovelluksien tapaan. Tämä tarkoittaa käytännössä sitä, että hybridisovellukset pystyvät samanaikaisesti olemaan täysin riippumattomia sovelluksen alustasta ja hyödyntämään sovellusalustan ominaisuuksia, kuten esimerkiksi kameraa, paikkatietoa sekä tiedostojärjestelmää. (Korf & Oksman 2014)

Hybridisovelluksien suosio kasvaa erittäin nopeasti. Monissa sovelluskauppojen suosituimmissa mobiilisovelluksissa onkin jo päädytty erilaisiin hybridiratkaisuihin. Esimerkiksi erittäin suosittu Netflix-hybridisovellus on toteutettu kaikille sovelluksen tukemille alustoille käyttämällä täysin samaa lähdekoodipohjaa (Korf & Oksman 2014). Suurin syy hybridisovelluksien suosion nopeaan kasvuun on kustannustehokkuus. Erilaisia hybridiratkaisuja hyödyntämällä täysin sama mobiilisovellus voidaan toteuttaa usealle eri mobiilikäyttöjärjestelmälle hyvin pienel-

lä vaivalla. Tästä syystä hybridisovellus onkin täysin varteenotettava, jopa looginen ratkaisu, kun mobiilisovellus halutaan toteuttaa useammalle kuin yhdelle mobiilikäyttöjärjestelmälle. Mikäli arviot tulevat pitämään paikkaansa, vuoteen 2016 mennessä hybridisovelluksien kehitys kattaa jopa yli puolet kaikesta mobiilisovelluskehityksestä (Gartner 2013).

2.2 Hybridimobiilisovelluksien vahvuudet

Kuten luvussa 2.1 todettiin, rakentuvat hybridisovellukset puoliksi niin sanotuista web-applikaatioista ja puoliksi niin sanotuista natiivisovelluksista. Vahvuutensa (ja heikkoutensa) hybridisovellukset lainaavat tasapuolisesti molemmista leireistä (Kuva 1). Natiivisovelluksiin verrattuna hybridisovelluksien suurin vahvuus on alustariippumattomuus. Web-applikaatioihin verrattuna hybridisovelluksien suurin vahvuus on kyky pystyä hyödyntämään mobiililaitteiden tarjoamia ominaisuuksia sekä mobiilikäyttöjärjestelmien omia ohjelmointirajapintoja. Myös toimivat jakelukanavat voidaan laskea yhdeksi hybridisovelluksien vahvuuksista. (Korf & Oksman 2014)



Kuva 1. Sovellustyyppien vertailua (Kuva: Matti Mononen).

Hybridisovelluksien alustariippumattomuus saavutetaan hyödyntämällä erilaisia web-tekniikoita (Write Once, Run Everywhere). Yleensä ainakin hybridisovelluksien käyttöliittymät on toteutettu lähes kokonaan käyttämällä HTML5-web-tekniikoita. Lisäksi suuri osa sovelluksen toimintalogiikasta voi olla toteutettu esimerkiksi JavaScript-ohjelmointikielellä. Varsinaisen sovelluksen sisään lisätään selainkomponentti eli niin sanottu web-näkymä, jossa sovelluksen käyttöliittymä näytetään. Selainkomponentin hallinnasta sekä alustariippumattoman lähdekoodin tulkitsemisesta ja suorittamisesta vastaa sovellusalustan selainmoottori. (Jones 2012)

Vaikka hybridisovellukset rakentuvat hyvin pitkälle erilaisista web-tekniikoista, pystyvät ne kuitenkin hyödyntämään myös mobiililaitteiden tarjoamia ominaisuuksia sekä mobiilikäyttöjärjestelmien ohjelmointirajapintoja. Näihin ominaisuuksiin päästään käsiksi hyödyntämällä mobiilikäyttöjärjestelmien tukemia natiiviohjelmointikieliä. Natiiviohjelmointikielillä toteutetut ominaisuudet eivät ole alustariippumattomia, joten niiden kohdalla on jokaiselle tuetulle mobiilikäyttöjärjestelmälle tehtävä oma toteutuksensa halutusta toiminnallisuudesta. (Jones 2012)

Mobiililaitteille optimoitujen web-applikaatioiden kohdalla on ongelmana pitkään ollut toimivien jakelukanavien puute. Hybridisovellukset pääsevät kuitenkin nauttimaan erilaisten sovelluskauppojen palveluista täysin natiivisovelluksien tapaan (Jones 2012). Suosituimpien mobiilikäyttöjärjestelmien omat sovelluskaupat mobiilisovelluksille ovat Googlen Google Play, Applen App Store sekä Microsoftin Marketplace for Windows Phone. Edellämainittujen sovelluskauppojen lisäksi markkinoilta löytyy useita kolmansien osapuolien sovelluskauppoja.

2.3 Hybridimobiilisovelluksien heikkoudet

Hybridisovelluksien suurimmat heikkoudet liittyvät pääasiassa sovelluksien suorituskykyyn. Esimerkiksi käyttöliittymän toiminnan sulavuudessa hybridisovellukset häviävät lähes poikkeuksetta puhtaille natiivisovelluksille. Suorituskykyero natiivisovelluksiin selittyy sillä, että hybridisovelluksien web-tekniikoita hyö-

dyntävät osat toimivat sovellusalustan selainmoottorin päällä. Suorituskykyerot tulevat esille selvimmin sovelluksissa, joissa hyödynnetään paljon erilaisia graafisia ominaisuuksia. Tämä ei kuitenkaan tarkoita automaattisesti sitä, että hybridisovelluksen täytyisi olla hidas. Selaimien (selainmoottoreiden) suorituskyky paranee jatkuvasti, mikä näkyy suoraan myös hybridisovelluksien suorituskyvyssä. Suorituskykyä haettaessa on sovelluskehittäjän pystyttävä elämään eri selainmoottoreiden asettamien rajoitteiden mukaan. Mikäli sovelluksen suorituskyvyn on oltava täysin viiveetön ja sulava, ei hybridisovellus välttämättä ole oikea ratkaisu sovelluksen toteuttamiseen. (Traeg 2013)

Toinen hybridisovelluksien suorituskykyyn liittyvä ongelma on eri mobiilikäyttöjärjestelmien väliset suorituskykyerot. Suorituskykyerot johtuvat pääasiassa erilaisista selainmoottoreista eri mobiilikäyttöjärjestelmien välillä. Esimerkiksi Googlen Android-käyttöjärjestelmä käyttää selainmoottorinaan Chromiumia, Applen iOS-käyttöjärjestelmä WebKittiä ja Microsoftin Windows Phone 8 -käyttöjärjestelmä IE10:tä (Android-käyttöjärjestelmän vanhemmat versiot käyttivät selainmoottoriaan WebKittiä). Tämä aiheuttaa sovelluskehittäjälle tilanteen, jossa jokaisen tuetun mobiilikäyttöjärjestelmän selainmoottorin ominaisuudet on otettava huomioon, jotta lopputuloksesta saataisiin mahdollisimman sulava, riippumatta siitä mille mobiilikäyttöjärjestelmälle sovellusta toteutetaan. (Traeg 2013)

2.4 Hybridimobiilisovelluksien kehittäminen

Kuten aiemmissa luvuissa on todettu, rakentuvat hybridisovellukset puoliksi niin sanotuista alustariippumattomista osista. Hybridisovelluksien kohdalla alustariippumattomilla osilla tarkoitetaan sovelluksen web-koodia, jonka tulkitsemisesta vastaa sovellusalustan selainmoottori. Tämä tarkoittaa käytännössä sitä, että sovelluksen alustariippumattomat osat toimivat kaikilla mobiilikäyttöjärjestelmillä, jotka kykenevät tulkitsemaan web-standardeihin perustuvia tekniikoita (sisältävät selaimen). Oikeaoppisesti toteutetussa hybridisovelluksessa web-koodi on toteutettu siten, ettei alustakohtaista lähdekoodia ole kirjoitettu lainkaan.

Hybridisovelluksien toinen puoli rakentuu niin sanotuista natiiviosista, jotka mahdollistavat mobiililaitteiden sekä mobiilikäyttöjärjestelmien eri ominaisuuksien hyödyntämisen. Natiiviosat rakennetaan käyttämällä sovellusalustojen tukemia natiiviohjelmointikieliä. Natiiviosien kohdalla vaaditaan jokaista sovelluksen tukemaa mobiilikäyttöjärjestemää kohden oma toteutuksensa halutusta toiminnallisuudesta. Hybridisovelluksien laajentaminen tarkoittaakin käytännössä sitä, että sovelluksen natiiviosat toteutetaan kaikille halutuille kohdealustoille.

Yleensä hybridisovelluksien kehittämiseen käytetään tarkoitukseen soveltuvia sovelluskehyksiä, jotka mahdollistavat hybridisovelluksien joustavan kehityksen sovellusalustoista riippumatta. Sovelluskehykset tarjoavat muun muassa valmiin rajapinnan sovelluksen web-koodin ja natiivikoodin välille. Sovelluskehyksien hyödyntäminen mahdollistaa muun muassa sen, että sovelluksen natiiviosia voidaan kutsua yhden rajapinnan kautta täysin samalla tavalla riippumatta sovelluksen kohdealustasta. Natiiviosille voidaan antaa tietyn tyyppisiä parametrejä ja niiltä voidaan odottaa tietyn tyyppisiä paluuarvoja. Tällaisessa tilanteessa sovelluksen natiiviosien on pystyttävä pelaamaan sovelluksen web-koodin asettamien ehtojen mukaisesti. Vaikka sovelluskehyksien hyödyntäminen helpottaakin hybridisovelluksien kehittämistä, ei se ole kuitenkaan täysin välttämätöntä. Rajapinta sovelluksen web-koodin ja natiivikoodin välille voidaan toteuttaa myös itse. Rajapinta toteutetaan yleensä siten, että sovellusalustan selainkomponenttia käytetään eräänlaisena ”siltana” sovelluksen web-koodin ja natiivikoodin välillä. (Traeg 2013)

3 HILKKA-sovelluksen laajennusprosessi

Tässä luvussa käsitellään joulukuussa 2013 alkanutta, noin kolme kuukautta kestänyttä hybridisovelluksen laajennusprosessia, jossa FastROI Oy:n HILKKA-järjestelmään kuuluva hybridimobiilisovellus laajennettiin Microsoftin Windows Phone 8 -käyttöjärjestelmälle yhteensopivaksi. Laajennusprosessin ensisijaisena tavoitteena oli toteuttaa HILKKA-sovelluksesta Windows Phone 8

-käyttöjärjestelmälle yhteensopiva versio, joka noudattaisi niin sanotun web-koodin osalta täysin samaa lähdekoodipohjaa kuin sovelluksen Android-käyttöjärjestelmälle toteutettu versio. Laajennusprosessin aikana sovelluksen Android-versioon toteutetut natiiviliitännäiset kirjoitettiin C#-ohjelmointikielellä Windows Phone 8 -käyttöjärjestelmälle yhteensopiviksi. Lisäksi laajennusprosessin yhteydessä keskityttiin muun muassa muokkaamaan sovelluksen käyttöliittymien tyylimäärittelyjä sekä tekemään yleisiä parannuksia, jotka vaikuttivat kaikkiin HILKKA-sovelluksen versioihin.

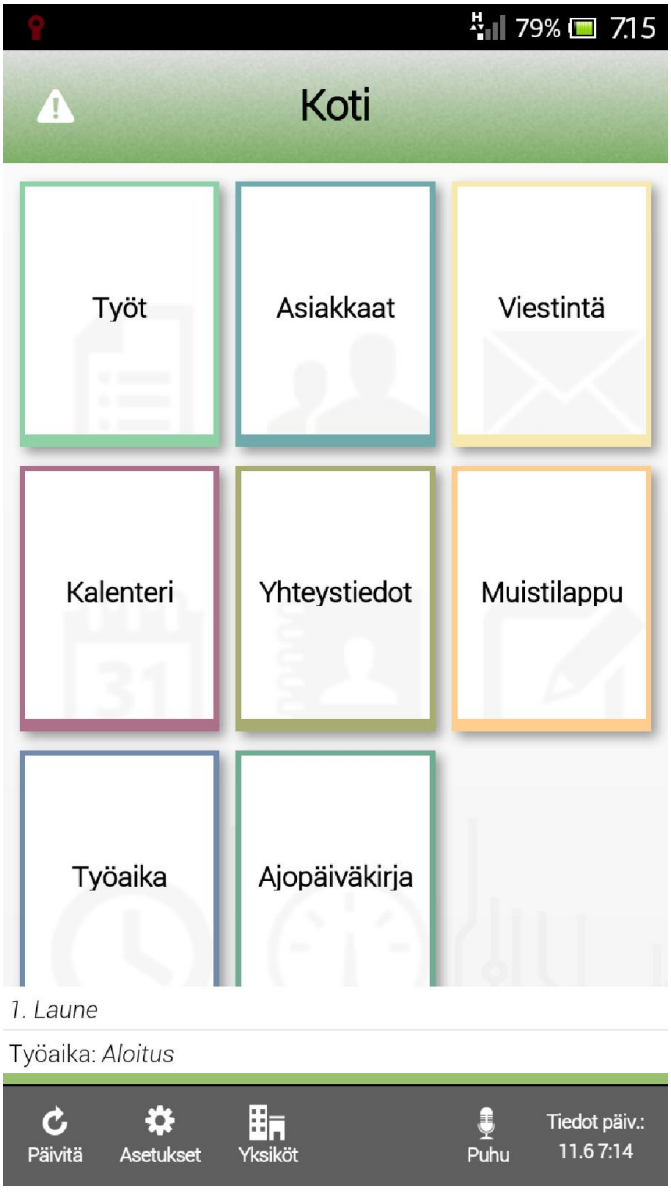
Tässä luvussa keskitytään käsittelemään HILKKA-sovellusta, sovelluksen tukemia alustoja (mobiilikäyttöjärjestelmiä), sovelluksen toteutuksessa käytettyjä teknologioita sekä itse sovelluksen laajennusprosessia. Sovelluksen laajennusprosessi käydään läpi vaiheittain ja lopuksi tarkastellaan laajennusprosessissa ilmenneitä haasteita ja saavutettuja tuloksia, eräänlaisen yhteenvedon muodossa.

3.1 Mikä on HILKKA

FastROI Oy:n HILKKA-järjestelmä on erittäin monipuolinen ratkaisu erilaisiin sosiaali- ja terveysalan työtehtäviin. HILKKA-järjestelmä soveltuu erinomaisesti muun muassa asumispalvelu- ja hoitoyksiköiden tarpeisiin: kotihoitoon, kotisairaanhoitoon sekä muihin auttamispalveluihin. HILKKA-järjestelmä toimii apuna niin hoitotyön johtamisessa, suunnittelussa, dokumentoinnissa kuin raportoinnissakin. Kokonaisuutena HILKKA-järjestelmä muodostaa niin sanotun toiminnanohjausjärjestelmän (ERP-järjestelmä), jonka avulla voidaan muun muassa hallita asiakastietoja, laskutusta sekä ohjata kentällä toimivia kotihoitajia reaaliajassa. (FastROI 2014)

Osaksi HILKKA-järjestelmää on toteutettu mobiilisovellus (Kuva 2), jonka avulla hoitajat pystyvät näkemään ja dokumentoimaan päivän suunnitellut työt. Tämän lisäksi sovellus sisältää useita eri ominaisuuksia ja toimintoja, jotka helpottavat kentällä toimivien hoitajien työskentelyä. HILKKA-sovelluksen suunnittelussa onkin huomioitu erityisesti kenttäolosuhteet ja niistä muodostuvat erikoistarpeet,

kuten esimerkiksi mahdollisuus toimia verkkoyhteyden kantaman ulkopuolella. Jos mobiililaite on verkkoyhteyden kantamalla, siirtyvät tiedot mobiililaitteesta automaattisesti HILKKA-järjestelmään. Jos taas mobiililaite on verkkoyhteyden kantaman ulkopuolella, tallentuvat tiedot sovelluksen sisäiseen tietokantaan myöhempää lähetystä varten. Tällä tavoin HILKKA-sovellus on saatu toimimaan myös alueilla, joissa verkkoyhteyttä ei ole tai verkkoyhteyden tila on erittäin huono.



Kuva 2. HILKKA-sovelluksen kotinäkymän käyttöliittymä (Android-versio) (Kuva: Matti Mononen).

Mobiilisovellukseksi HILKKA-sovellus on laajuudeltaan valtava ja toimiakseen se asettaa mobiililaitteille tietynlaisia vaatimuksia. Toimiakseen HILKKA-sovellus vaatii minimissään Android-käyttöjärjestelmän version 4.0 (Ice Cream Sandwich) tai Windows Phone -käyttöjärjestelmän version 8.0. Mobiililaitteen suorittimen osalta HILKKA-sovellus toimii niin ARM- kuin x86-arkkitehtuurin prosessoreilla. HILKKA-sovelluksen sulavaa käyttökokemusta varten on suositeltavaa, että mobiililaitteessa on vähintään tuplaydinprosessori, riittävästi keskusmuistia ja tallennustilaa. Mikäli kaikkia HILKKA-sovelluksen ominaisuuksia halutaan hyödyntää, tulisi käytössä olevien mobiililaitteiden tukea myös seuraavia ominaisuuksia:

- Sijainnin paikannus (GPS)
- Near Field Communication (NFC)
- Bluetooth.

3.2 Käyttöjärjestelmät

HILKKA-sovellus on toteutettu yhteensä kolmelle eri mobiilikäyttöjärjestelmälle: Nokian Symbian, Googlen Android sekä Microsoftin Windows Phone 8. HILKKA-sovelluksen ensimmäinen mobiiliversio toteutettiin Nokian Symbian-käyttöjärjestelmälle. Kun Symbian-käyttöjärjestelmän suosio mobiilikäyttöjärjestelmien keskuudessa alkoi hiipua, alettiin HILKKA-sovelluksesta toteuttaa täysin uutta, erilaisiin hybridiratkaisuihin perustuvaa versiota. Ensimmäiseksi uusittua HILKKA-sovellusta lähdettiin toteuttamaan Googlen Android-käyttöjärjestelmälle. Sovelluksen toteutukseen pyrittiin valitsemaan teknologioita, joita käyttämällä sovellus olisi mahdollista laajentaa toimimaan myös muilla mobiilikäyttöjärjestelmillä. Myöhemmin HILKKA-sovellus laajennettiinkin toimimaan Microsoftin Windows Phone 8 -käyttöjärjestelmällä. HILKKA-sovelluksen Android- sekä Windows Phone 8 -versiot perustuvat täysin samaan lähdekoodipohjaan.

Tässä luvussa tarkastellaan Googlen Android- ja Microsftin Windows Phone 8 -käyttöjärjestelmiä sekä selvitetään, miksi kyseiset käyttöjärjestelmät soveltuvat hyvin HILKKA-sovelluksen alustoiksi. Lisäksi tässä luvussa tarkastellaan lä-

hemmin niin Android- kuin Windows Phone 8 -sovelluskehitystä sekä tutustutaan kyseisten mobiilikäyttöjärjestelmien laajennusprosessiin vaikuttaviin tekijöihin.

3.2.1 Android

Android on Googlen kehittämä mobiilikäyttöjärjestelmä, joka perustuu muokattuun Linux-käyttöjärjestelmäytimeen. Nykyisellään Android-käyttöjärjestelmän kehitystyöstä vastaa Open Handset Alliance, johon kuuluu Googlen lisäksi useita IT-alan yrityksiä. Android-käyttöjärjestelmää kehitetään avoimen lähdekoodin GPLv2-lisenssillä. Opinnäytetyön kirjoitushetkellä viimeisin Android-käyttöjärjestelmän versio kantaa versionumeroa 5.0.1 (Lollipop), joka julkaistiin 2.12.2014. Vaikka Android-käyttöjärjestelmä onkin suunnattu lähinnä mobiililaitteille, löytyy markkinoilta muitakin Android-käyttöjärjestelmään perustuvia laitteita, kuten pelikonsoleita ja kameroita. (Wikipedia 2014a)

Android-sovelluskehitys tehdään yleisimmin Java-ohjelmointikielellä, jolloin sovellukset toimivat joko Android-käyttöjärjestelmän Dalvik- tai ART-virtuaalikoneen päällä. Virtuaalikoneen tehtävänä on kääntää Java-ohjelmointikielellä kirjoitettu lähdekoodi tavukoodiksi ja suorittaa käännettyä tavukoodia. Tämän lisäksi Android-sovelluskehitystä voidaan tehdä myös täysin natiivisti, jolloin ohjelmointikielenä voidaan käyttää C/C++-ohjelmointikieltä (Mednieks, Dornin, Meike & Nakamura 2012, 501). Android-käyttöjärjestelmän ilmaiset kehitystyökalut (Android SDK) sisältävät kaiken tarvittavan Android-sovellusten kehittämiseen. Android-käyttöjärjestelmän kehitystyökalut voidaan ladata Internetistä valmiina pakettina. Paketti sisältää Eclipse-ohjelmointiympäristön ja siihen integroidut Android-kehitystyökalut (Android developers 2014). Android-sovelluksen peruskomponentit sisältävät neljä erilaista luokkaa: aktiviteetit (activity), palvelut (services), sisältötarjoajat (content providers) ja lähetyksen vastaanottajat (broadcast receivers) (Mednieks, Dornin, Meike & Nakamura 2012, 77–83). Edellämainittujen peruskomponenttien lisäksi Android-sovellukset voivat sisältää muun muassa normaaleja Java-luokkia.

HILKKA-sovelluksen ensisijaisena alustana toimii Googlen Android-käyttöjärjestelmä. Tämä tarkoittaa käytännössä sitä, että kaikki sovelluksen uudet ominaisuudet kehitetään Android-kehitysympäristössä. HILKKA-sovelluksen Android-versio sisältääkin muutamia alustakohtaisia ominaisuuksia, jotka eivät toimi muilla sovelluksen tukemilla alustoilla. Osa näistä ominaisuuksista saattaa jäädä ainoastaan HILKKA-sovelluksen Android-versioon.

Vaikka laajennusprosessin suurimmat muutokset keskittyivätkin lähinnä HILKKA-sovelluksen Windows Phone 8 -versioon, jouduttiin laajennusprosessin yhteydessä testaamaan aktiivisesti myös sovelluksen Android-versiota. Kaikki muutokset, jotka vaikuttivat sovelluksen alustariippumattomaan lähdekoodiin (web-koodiin), oli testattava myös sovelluksen Android-versiolla. Lisäksi sovelluksen käyttöliittymien tyylimäärittelyihin tehdyt muutokset oli todettava toimiviksi myös sovelluksen Android-versiolla.

3.2.2 Windows Phone 8

Windows Phone 8 on Microsoftin kehittämä mobiilikäyttöjärjestelmä, joka perustuu hybridiin Windows NT -käyttöjärjestelmäytimeen. Windows Phone 8 on Microsoftin toisen sukupolven Windows Phone -mobiilikäyttöjärjestelmä, jonka viimeisin versio opinnäytetyön kirjoitushetkellä kantaa versionumeroa 8.1 (8.10.14203.306). Kyseinen versio julkaistiin 14.11.2014. Windows Phone 8 -sovellukset eivät ole yhteensopivia Microsoftin vanhemman Windows Phone 7 -käyttöjärjestelmän kanssa. (Wikipedia 2014b)

Windows Phone 8 -sovelluskehitys tehdään yleisimmin C#-ohjelmointikielellä. Tämän lisäksi Windows Phone 8 -sovelluskehitystä voidaan tehdä myös täysin natiivisti, jolloin ohjelmointikielenä toimii C/C++-ohjelmointikieli (Windows Dev Center 2014a). Windows Phone 8 -sovelluskehitys asettaa muutamia vaatimuksia niin sovelluksien kehitysympäristön kuin kehityslaitteidenkin osalta. Windows Phone 8 -kehitysympäristö vaatii toimiakseen 64-bittisen Windows 8- tai Windows 8.1 -käyttöjärjestelmän sekä kehityslaitteen osalta rekisteröinnin Microsoftin järjestelmään. Kehityslaitteen rekisteröinti vaatii lisäksi aktiivisen Mic-

rosoft-käyttäjätunnuksen tai käyttäjätunnuksen Windwos Phone Dev Center -palveluun. Kehityslaitteen rekisteröiminen Microsoftin järjestelmään on ilmaista. Ilmaiset Windows Phone 8 -kehitystyökalut (Windows Phone SDK) voidaan ladata Internetistä valmiina pakettina. Paketti sisältää Visual Studio 2013 -ohjelmointiympäristön sekä siihen integroidut Windows Phone 8 -kehitystyökalut. (Windows Dev Center 2014b)

Windows Phone 8 on ensimmäinen mobiilikäyttöjärjestelmä, jolle HILKKA-sovellusta lähdettiin laajentamaan. Koska uusien ominaisuuksien kehittäminen tapahtuu ensisijaisesti Android-käyttöjärjestelmällä, ei sovelluksen Windows Phone 8 -versio sisällä tällä hetkellä kaikkia sovelluksen Android-versioon toteutettuja ominaisuuksia. Tämän lisäksi sovelluksen Android-versioon on toteutettu muutamia alustakohtaisia ominaisuuksia, joita ei voida laajentaa sovelluksen Windows Phone 8 -versioon. Osaksi HILKKA-sovellusta on muun muassa integroitu kolmansien osapuolien sovelluksia, jotka eivät kaikki ole yhteensopivia Windows Phone 8 -käyttöjärjestelmän kanssa. Sovelluksen Windows Phone 8 -versio sisältää kuitenkin kaikki tärkeimmät sovellukseen toteutetut ominaisuudet. Opinnäytetyön kirjoitushetkellä HILKKA-sovelluksen Windows Phone 8 -versio ei ole vielä täysin valmis.

Paperilla Microsoftin Windows Phone 8 -käyttöjärjestelmä vaikuttaa erinomaiselta vaihtoehdolta hybridisovelluksien ja erityisesti HILKKA-sovelluksen alustaksi. Windows Phone 8 -käyttöjärjestelmällä HILKKA-sovellus kärsii kuitenkin pienistä ongelmista, jotka vaikuttavat sovelluksen toimintaan negatiivisesti. Kyseiset ongelmat liittyivät lähinnä sovelluksen tiedosto-operaatioiden suorituskykyyn Windows Phone 8 -käyttöjärjestelmällä. Muutoin HILKKA-sovelluksen suorituskyky sovelluksen Android- sekä Windows Phone 8 -versioiden välillä oli käytännössä olematon. Vaikkei Windows Phone 8 -käyttöjärjestelmä olekaan aivan täydellinen, on se pieniä ongelmia lukuun ottamatta varsin toimiva alusta hybridisovelluksille.

3.3 Sovelluskehykset

HILKKA-sovelluksen toteutukseen valittuja teknologioita tarkastellessa voidaan hyvin pian todeta, että HILKKA-sovellus on alusta asti suunniteltu toimimaan useammalla kuin yhdellä mobiilikäyttöjärjestelmällä. Sovelluksen selkäranka rakentuu Apache Cordova- ja Sencha Touch -sovelluskehyksistä, joita käyttämällä sovelluksesta on saatu aikaiseksi täysiverinen hybridisovellus.

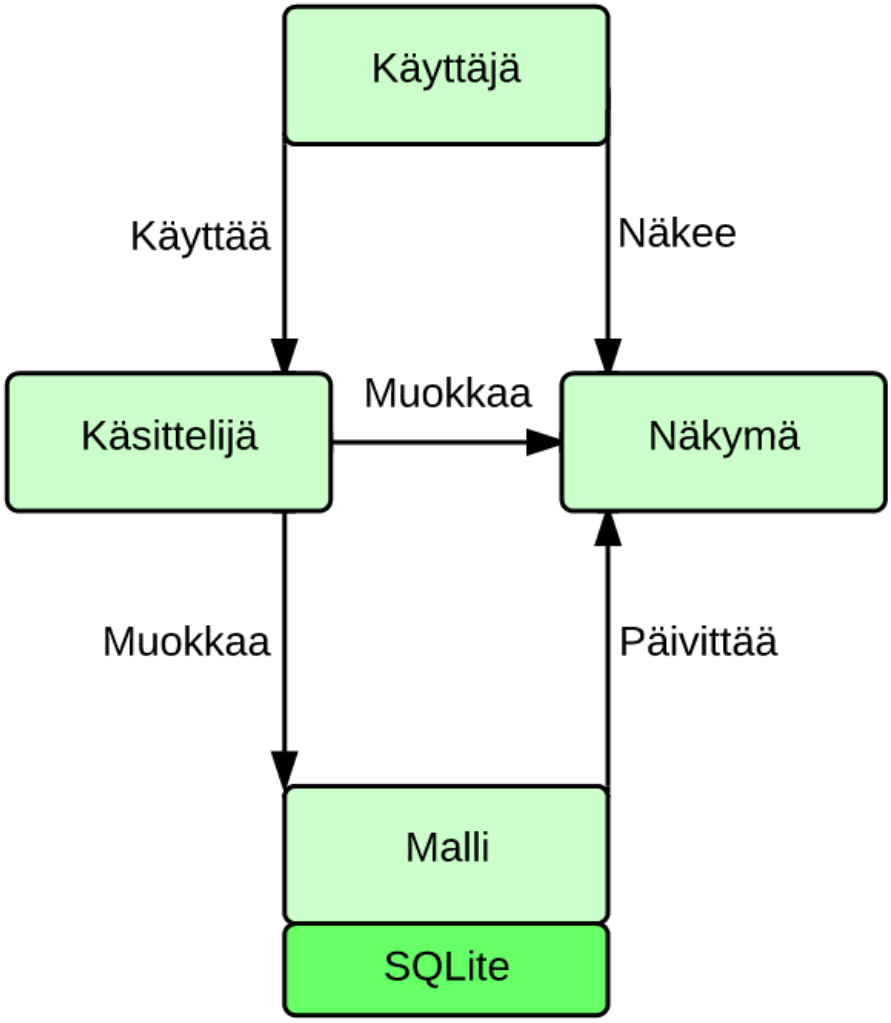
Tässä luvussa tarkastellaan HILKKA-sovelluksen toteutukseen valituista teknologioista keskeisimpiä sekä selvitetään, miten sovellus hyödyntää kyseisiä teknologioita. Lisäksi tässä luvussa selvitetään, miten HILKKA-sovelluksessa on päädytty kyseisten teknologioiden käyttöön ja mitkä seikat vaikuttivat kyseisten teknologioiden valinnoissa.

3.3.1 Sencha Touch

Sencha Touch on täysin web-standardeihin perustuva, erityisesti mobiilisovelluksien käyttöliittymien kehittämistä varten toteutettu JavaScript-sovelluskehys. Sencha Touch -sovelluskehys sisältää muun muassa yli 50 valmista käyttöliittymäkomponenttia, useita eri tyylejä sovelluskehyksen valmiille käyttöliittymäkomponenteille sekä tuen käyttöliittymäkomponenttien kustomointiin. Sencha Touch -sovelluskehyksen käyttöliittymät rakennetaan hyödyntämällä sovelluskehyksen omaa sisäänrakennettua järjestelmää, joka noudattaa MVC-arkkitehtuurin mallia. Sencha Touch -sovelluskehyksen ensimmäinen versio julkaistiin vuonna 2010, jolloin se sisälsi tuen Googlen Android- sekä Applen iOS-käyttöjärjestelmille. Opinnäytetyön kirjoitushetkellä Sencha Touch -sovelluskehyksen viimeisin versio kantaa versionumeroa 2.4.1 ja se tukee kaikkia suosituimpia mobiilikäyttöjärjestelmiä. Sencha Touch -sovelluskehys on optimoitu toimimaan erilaisten hybridisovelluksien toteutukseen tarkoitettujen sovelluskehysten kanssa, joista esimerkkinä mainittakoon Apache Cordova. Laajennusprosessin toteutushetkellä HILKKA-sovelluksessa hyödynnettiin Sencha Touch sovelluskehyksen versiota 2.3.1. (Sencha 2014)

Sencha Touch -sovelluskehys perustuu erittäin vahvasti niin sanottuun MVC-arkkitehtuuriin, jonka ideana on erottaa sovelluksen käyttöliittymä ja sen toimintalogiikka toisistaan. MVC-arkkitehtuuri rakentuu kolmesta eri osasta: malli (model), näkymä (view) sekä käsittelijä (controller). Mallin tehtävänä on toimia rajapintana sovelluksen tietovaraston sekä käsittelijän että näkymän välillä. Näkymän tehtävänä on esittää mallin tarjoama tieto ja määrittää sovelluksen käyttöliittymän ulkoasu. Käsittelijän tehtävänä on vastaanottaa käyttäjän antamat syötteet ja muokata mallia sekä näkymää niiden mukaan. (Kumar 2012, 7-12) Yksi MVC-arkkitehtuurin suurimmista eduista on komponenttien riippumattomuus toisistaan. Näkymän ja käsittelijän riippuvuus toisistaan on erittäin pieni eikä malli ole riippuvainen kummastakaan edellä mainituista. Tämä mahdollistaa muun muassa sen, että samaa mallia käyttäen voidaan toteuttaa useita eri näkymiä, käyttöliittymiä tai jopa sovelluksia. (Dalling 2009)

HILKKA-sovelluksessa hyödynnetään Sencha Touch -sovelluskehyksen näkymiä ja käsittelijöitä. HILKKA-sovelluksen mallina toimii sovelluksen SQLite-tietokanta sekä tietokantaa käsittelevät ohjelmointirajapinnat. Sencha Touch -sovelluskehyksen omaa sisäänrakennettua mallirakennetta hyödynnetään vain sovelluksen muutamassa osassa. (Kuva 3, Kuva 4, Kuva 5)



Kuva 3. MVC-arkkitehtuurin rakenne HILKKA-sovelluksessa (Kuva: Matti Mononen).

```
/* global Ext, fastroi */ /* jshint strict:false */
Ext.define('App.view.ExampleView', {
    extend: 'Ext.Container',
    alias: 'widget.exampleView',
    config: {
        helpTextFilename: 'exampleView',
        layout: {
            type: 'fit'
        }
    },
    initialize: function () {
        this.callParent(arguments);
        var self = this,
            prevDataLabel,
            inputField,
            addButton;
        fastroi.DB.getExampleViewData(function (data) {
            prevDataLabel = Ext.create('Ext.Label', {
                xtype: 'label',
                itemId: 'prevdatalabel',
                html: '<b>Data length:</b> ' + data.length + ' rows!  <br/>' +
                '<b> Previous data:</b> ' + data[data.length - 1].text
            });
            inputField = Ext.create('frtextareafield', {
                xtype: 'textfield',
                itemId: 'inputfield',
                label: 'Value'
            });
            addButton = Ext.create('Ext.Button', {
                xtype: 'button',
                itemId: 'addbutton',
                text: 'Add',
                handler: self.addClicked
            });
            self.add([prevDataLabel, inputField, addButton]);
        });
    },
    addClicked: function () {
        var field = this.down('#inputfield'),
            label = this.down('#prevdatalabel');
        this.fireEvent('addInput', field.getValue(), label);
    }
});
```

Kuva 4. Esimerkkinäkymän näkymä (Kuva: Matti Mononen).

```
/* global Ext, fastroi */ /* jshint strict:false */
Ext.define('App.controller.ExampleView', {
    extend: 'Ext.app.Controller',
    config: {
        views: 'ExampleView',
        refs: {
            exampleView: 'exampleView'
        },
        control: {
            exampleView: {
                addInput: 'onAddInput'
            }
        }
    },
    onAddInput: function (input, label) {
        fastroi.DB.addExampleViewData(input, function () {
            fastroi.DB.getExampleViewData(function (data) {
                fastroi.Dialogs.info('Add data', 'Value: ' + input +
                    ' added into database!', function () {
                        label.setHtml('<b>Data length:</b> ' + data.length +
                            ' rows! <br/><b> Previous data:</b> ' + input);
                    });
            });
        });
    }
});
```

Kuva 5. Esimerkkinäkymän käsittelijä (Kuva: Matti Mononen).

HILKKA-sovelluksen käyttöliittymät on toteutettu kokonaisuudessaan Sencha Touch -sovelluskehystä hyödyntäen. Sencha Touch -sovelluskehyksen käyttöön päädyttiin eräänlaisen sovelluskehyksien vertailuprosessin kautta. Lopullinen valinta tehtiin Sencha Touch -sovelluskehyksen sekä avoimen lähdekoodin Enyo-sovelluskehyksen välillä. Molemmat edellä mainituista sovelluskehyksistä tuntuivat päteviltä vaihtoehdoilta sovelluksen käyttöliittymien toteuttamiseen, mutta viimeistään Senchan kaupallinen tuki sai valinnan kallistumaan Sencha Touch -sovelluskehyksen puoleen. Myös yrityksen mERP-sovelluksessa oli päädytty Senchan tuotteisiin, mikä osaltaan vaikutti myös HILKKA-sovelluksen käyttöliittymäsovelluskehyksen valintaan. (Ruotsalainen 2014)

HILKKA-sovelluksessa hyödynnetään useita Sencha Touch -sovelluskehyksen valmiita käyttöliittymäkomponentteja ja lisäksi omia, eri tarkoituksiin toteutettuja käyttöliittymäkomponentteja. Sovelluksen omat käyttöliittymäkomponentit on toteutettu pääasiassa yhdistelemällä sovelluskehyksen valmiita käyttöliittymäkomponentteja toisiinsa sekä laajentamalla käyttöliittymäkomponenttien toimin-

talogiikkaa. Esimerkiksi Ext.Container-luokasta periytyvä frGenericList-komponentti rakentaa annetusta oliotaulukosta listanäkymän, johon lisätään Ext.Button-luokasta periytyviä frListItem-komponentteja. Lähes jokaista sovelluksen käyttämää käyttöliittymäkomponenttia kohden on toteutettu oma tyylimäärittely, joita käyttämällä sovelluksen käyttöliittymistä on saatu rakennettua halutun näköiset. Sencha Touch -sovelluskehyksen alkuperäisistä tyyleistä ei ole jätetty jäljelle juuri mitään, minkä ansiosta HILKKA-sovelluksen ulkoasusta on saatu uniikki.

HILKKA-sovelluksen tyylimäärittelyt on toteutettu käyttämällä dynaamista Sass-tyylikieltä, jonka ideana on helpottaa CSS-tyylimäärittelyjen toteuttamista sekä ylläpitoa. Sass-tyylimäärittelyt voivat sisältää muun muassa muuttujia ja operaattoreita sekä muita tyylimäärittelyjen toteuttamista helpottavia ominaisuuksia. Valmiit tyylimäärittelyt voidaan kääntää esimerkiksi Compass-työkalun avulla normaaleiksi CSS-tyylimäärittelyiksi. (Sass 2014)

3.3.2 Apache Cordova

Apache Cordova on hybridisovelluksien kehittämiseen tarkoitettu avoimen lähdekoodin sovelluskehys, joka sisältää joukon ohjelmointirajapintoja mobiilisovelluksien kehittämiseen eri mobiilikäyttöjärjestelmille. Apache Cordova -sovelluskehys tukee lähes kaikkia suosituimpia mobiilikäyttöjärjestelmiä ja opinnäytetyön kirjoitushetkellä sen viimeisin versio kantaa versionumeroa 4.0. (Apache Cordova 2014a) Laajennusprosessin toteutushetkellä HILKKA-sovelluksessa hyödynnettiin Apache Cordova -sovelluskehyksen versiota 3.0.

Apache Cordova -sovelluskehys tarjoaa sovelluskehittäjille valmiit rajapinnat eri mobiililikäyttöjärjestelmien tärkeimpiin ominaisuuksiin, kuten esimerkiksi kontakteihin, kameraan, paikkatietoon sekä tiedostojärjestelmään. Apache Cordova -sovelluskehyksen viimeisin versio sisältää yhteensä 19 eri tarkoitukseen sopivaa liitännäistä, joista osaa hyödynnetään myös HILKKA-sovelluksessa (Apache Cordova 2014b). Sovelluskehyksellä toteutetussa sovelluksessa, sovelluksen sisään on lisätty web-näkymä, jossa sovelluksen käyttöliittymä näyte-

tään. Apache Cordova toimii rajapintana sovelluksen web-koodin sekä natiiviohjelmointikielillä toteutettujen osien välillä (Apache Cordova 2014a). Mikäli sovelluskehyksen valmiista liitännäisistä ei löydy haluttua toiminnallisuutta, voidaan sovelluskehyksen toimintaa laajentaa kirjoittamalla omia liitännäisiä. Apache Cordova tarjoaa käyttäjilleen myös useista sadoista liitännäisistä koostuvan liitännäisrekisterin, jonne sovelluskehittäjät voivat ladata omia liitännäisiään (Apache Cordova 2014c). HILKKA-sovelluksessa ei hyödynnetä liitännäisrekisterin liitännäisiä.

HILKKA-sovelluksen natiivitoiminnallisuudet on rakennettu täysin Apache Cordova -sovelluskehyksen varaan. Sovelluskehystä päädyttiin hyödyntämään, koska sen taustalta löytyi luotettava toimija. Tämän vuoksi avoimen lähdekoodin ratkaisua ei nähty myöskään riskinä, vaan enemmänkin mahdollisuutena (Ruotsalainen 2014). HILKKA-sovelluksessa hyödynnetään osaa sovelluskehyksen valmiista rajapinnoista (Kuva 6), joita käyttämällä kohdealustojen eri ominaisuuksiin on päästy käsiksi. Lisäksi HILKKA-sovellusta varten on toteutettu useita omia liitännäisiä (Kuva 7), joita käyttämällä sovelluksen natiivitoiminnallisuuksia on pystytty laajentamaan. Esimerkiksi sovellukseen toteutettu FileSystem-liitännäinen suorittaa erilaisia tiedostojenkäsittelyyn liittyviä operaatioita, kuten esimerkiksi tiedostojen kopioimista, leikkaamista, poistamista, pakkaamista sekä tiedostojen koon kysymistä.

Liitännäinen	Käytössä HILKKA sovelluksessa
BatteryStatus	
Camera	
Console	
Contacts	
Device	X
DeviceMotion (Accelerometer)	
DeviceOrientation (Compass)	
Dialogs	X
FileSystem	
FileTransfer	
Geolocation	X
Globalization	
InAppBrowser	
Media	
MediaCapture	X
NetworkInformation	X
Splashscreen	X
Vibration	
Statusbar	

Kuva 6. Apache Cordova -sovelluskehyksen liitännäiset (versio 4.0) (Kuva: Matti Mononen).

Liitännäinen	Android	Windows Phone
BigBen	X	
Database	X	X
DatabaseDownloader	X	X
DatePicker	X	X
DeviceInfo	X	X
FileSystem	X	X
FRListView	X	X
GeoCoder	X	X
MediaPlayer	X	
MohinetPlugin	X	
NFC	X	X
NotificationUpdater	X	
ProgressDialog	X	X
SpeechListener	X	X
TextInput	X	X
Troublog	X	X
TunstallPlugin	X	
VersionChecker	X	X

Kuva 7. HILKKA-sovelluksen natiiviliitännäiset laajennusprosessin jälkeen (Kuva: Matti Mononen).

Apache Cordova -sovelluskehyksen liitännäiset rakentuvat kahdesta eri osasta, joita ovat liitännäisen alustariippumaton rajapinta sekä liitännäisen varsinainen natiivitoteutus. Liitännäisen natiivitoteutus on toteutettava erikseen jokaista sovelluksen tukemaa sovellusalustaa kohden. Alustariippumattoman rajapinnan tehtävänä on kutsua liitännäisen varsinaista natiivitoteutusta. Liitännäisten natiivitoteutuksia kutsutaan käyttämällä sovelluskehyksen cordova.exec-funktiota. Tällä tavoin sovellus osaa automaattisesti kutsua oikeaa toteutusta halutun liitännäisen sopivasta toiminnosta. Seuraavista kuvista (Kuva 8, Kuva 9, Kuva 10) voi havainnoida, kuinka FileSystem-liitännäisen alustariippumaton rajapinta kutsuu liitännäisen varsinaisia natiivitoteutuksia. Kaikki Apache Cordova -sovelluskehyksen liitännäiset toimivat asynkronisesti. (Apache Cordova 2014c)

```
//! Delete a file.
this.deleteFile = function (filePath, win, fail) {
    exec(win, fail, pluginName, "deleteFile", [filePath]);
};
```

Kuva 8. FileSystem-liitännäisen deleteFile-funktion alustariippumaton rajapinta (Kuva: Matti Mononen).

```
/**
 * @param args
 *            0:An absolute path of the file to delete.
 * @param callbackContext
 */
private void deleteFile(CordovaArgs args, CallbackContext callbackContext) {
    File f = new File(getAbsPath(args.optString(0)));
    if (f.delete()) {
        callbackContext.success();
    } else {
        callbackContext.error("deleteFile failed for " + f.getName());
    }
}
```

Kuva 9. FileSystem-liitännäisen deleteFile-funktion Java-toteutus (Kuva: Matti Mononen).

```
// exec(win, fail, pluginName, "deleteFile", [filePath]);
public async void deleteFile(string options)
{
    string cbId = CbId.extractFromOptions(options);
    string[] args = JsonConvert.DeserializeObject<string[]>(options);
    string filePath = args[0].Replace('/', Path.DirectorySeparatorChar);
    if (filePath != null)
    {
        var theFile = await getLocalRelativeFile(filePath);
        if (theFile != null)
        {
            await theFile.DeleteAsync(StorageDeleteOption.PermanentDelete);
        }
    }
    DispatchCommandResult(new PluginResult(PluginResult.Status.OK), cbId);
}
```

Kuva 10. FileSystem liitännäisen deleteFile-funktion C#-toteutus (Kuva: Matti Mononen).

3.4 Laajennusprosessi

Tässä luvussa tarkastellaan hybridisovelluksen laajennusprosessia, jossa opinnäytetyön esimerkkiprojektiksi valittu HILKKA-sovellus laajennettiin Microsoftin Windows Phone 8 -käyttöjärjestelmälle yhteensopivaksi. Laajennusprosessissa keskitytään tarkastelemaan sovelluksen Windows Phone 8 -version natiiviliitännäisten toteutusta, sovelluksen rakenteeseen tehtyjä muutoksia sekä sovelluksen käyttöliittymien tyylimäärittelyihin tehtyjä muutoksia. Lisäksi tässä luvussa tarkastellaan sovelluksen laajennusprosessissa ilmenneitä haasteita ja saavutettuja tuloksia.

3.4.1 Liitännäiset

Kuten edellisissä luvuissa on todettu, on HILKKA-sovellusta varten toteutettu useita natiiviliitännäisiä, joita käyttämällä sovelluksen kohdealustojen tärkeimpiin ominaisuuksiin on päästy käsiksi. Yksi laajennusprosessin keskeisimmistä tavoitteista oli toteuttaa sovelluksen natiiviliitännäiset Windows Phone 8 -käyttöjärjestelmälle siten, että liitännäiset sisältäisivät täysin samat toiminnalli-

suudet kuin vastaavat Android-versioon toteutetut liitännäiset. Toinen laajennusprosessin keskeisimmistä tavoitteista oli toteuttaa Windows Phone 8 -käyttöjärjestelmän natiiviliitännäiset siten, ettei vastaaviin Android-käyttöjärjestelmän liitännäisiin tai liitännäisten alustariippumattomiin rajapintoihin tarvitsisi tehdä muutoksia. Laajennusprosessin aikana kolmetoista HILKKA-sovelluksen toiminnan kannalta tärkeintä natiiviliitännäistä toteutettiin Windows Phone 8 -käyttöjärjestelmälle C#-ohjelmointikieltä hyödyntäen.

Natiiviliittänäisten laajentaminen sovelluksen Windows Phone 8 -versioon aloitettiin HILKKA-sovelluksen toiminnan kannalta kriittisimmistä liitännäisistä. Kriittisimpiin liitännäisiin voidaan laskea ainakin sovelluksen tietokantaa käsittelevät liitännäiset, kuten Database ja DatabaseDownloader. Koska HILKKA-sovellus perustuu erittäin vahvasti tietokantojen käyttöön, ei sovelluksen laajennusprosessia voitu viedä eteenpäin ennen kuin sovelluksen tietokantoja käsittelevät liitännäiset oli toteutettu myös sovelluksen Windows Phone 8 -versioon. Kun sovelluksen toiminnan kannalta kriittisimmät liitännäiset oli toteutettu, ei loppujen liitännäisten toteutusjärjestyksellä ollut enää juurikaan väliä.

3.4.2 Behemoth

HILKKA-sovelluksen alkuperäisessä toteutuksessa kaikki sovelluksen web-koodi pakattiin suojattuun tiedostoon, joka lisättiin sovelluksen asennuspakettiin. Kun sovellus asennettiin mobiililaitteeseen, purettiin web-koodi sovelluksen suojattuun hakemistoon sovelluksen ensimmäisen käynnistyskerran yhteydessä. Kyseisen prosessin ideana oli estää HILKKA-sovelluksen lähdekooditiedostoihin käsiksi pääseminen, joka muutoin olisi voitu toteuttaa purkamalla sovelluksen asennuspaketti auki. Windows Phone 8 -käyttöjärjestelmällä ongelmaksi muodostui kuitenkin tiedostojen hidas purkuprosessi, mikä kesti useita minuutteja. Sovelluksen Android-versiolla täysin vastaava prosessi vei ainoastaan murto-osan Windows Phone 8 -version kuluttumasta ajasta. Tästä syystä johtuen kyseiseen prosessiin oli tehtävä suuria muutoksia. Lähdekooditiedostojen purkamisen hitauden syyksi selvisi Windows Phone 8 -käyttöjärjestelmän tie-

dostojärjestelmä, joka ei kyennyt suoriutumaan purkuprosessista riittävän nopeasti.

HILKKA-sovelluksen kannalta yksi laajennusprosessin suurimmista muutoksista oli lähdekooditiedostojen pakkaus- ja purkurutiineihin tehty muutos. Kaikkien lähdekooditiedostojen pakkaamisen sijaan sovelluksen web-koodista alettiin generoida behemothiksi (behemoth.js) nimettyä tiedostoa, joka sisältää kaiken HILKKA-sovelluksen web-koodin. Tämä tarkoittaa sitä, että myös sovelluksessa käytettyjen JavaScript-kirjastojen lähdekoodit sisältyvät behemoth-tiedostoon. Prosessin lopputuloksena syntyy massiivinen JavaScript-tiedosto (pitkälle yli 100 000 koodiriviä), joka lisätään sovelluksen asennuspakettiin. Behemoth-tiedosto generoidaan sovelluksen kääntämisen yhteydessä ajettavien scriptien avulla.

Kun HILKKA-sovelluksesta tehdään niin sanottua julkaisuversiota, ajetaan sovelluksen lähdekoodi vielä Googlen Closure Compiler -työkalun lävitse. Closure Compiler poistaa sovelluksen lähdekoodista muun muassa kaikki kommentit sekä debug-notaatioiden sisällä olevan koodin (//<debug> console.log("This is debug message!"); //</debug>). Tämän lisäksi behemoth-tiedoston sisältämä lähdekoodi kompressoidaan (minifioidaan), jottei lähdekoodi olisi ihmiselle luettavassa muodossa. Lähdekoodin kompressoinnilla saavutetaan muitakin hyötyjä, kuten lähdekooditiedostojen koon pieneneminen sekä mahdollisesti pieni suorituskyvyn paraneminen. Siirtymällä behemoth-tiedoston käyttöön saatiin ensimmäisen käynnistyskerran vaatima aika tiputettua sovelluksen Windows Phone 8 -versiolla useista minuuteista muutamiin sekunteihin. Siirtyminen behemoth-tiedoston käyttöön nopeutti ensimmäisen käynnistyskerran vaatimaa aikaa myös sovelluksen Android-versiolla.

3.4.3 Tyylimäärittelyt

HILKKA-sovelluksen laajennusprosessin yksi keskeisimmistä tavoitteista oli toteuttaa sovelluksen käyttöliittymien tyylimäärittelyt siten, että sovelluksen ulkoasu olisi yhtenevä kaikilla sovelluksen tukemilla alustoilla. Laajennusprosessin

alkumetreiltä asti oli selvää, että sovelluksen käyttöliittymien tyylimäärittelyt tulevat vaatimaan muutoksia, jotta käyttöliittymät saataisiin toimimaan halutulla tavalla myös sovelluksen Windows Phone 8 -versiolla. Laajennusprosessin alkumetreillä oli kuitenkin täysin mahdotonta arvioida, kuinka paljon muutoksia sovelluksen tyylimäärittelyt tulevat vaatimaan. Laajennusprosessin edetessä myös sovelluksen tyylimäärittelyihin vaadittavien muutoksien määrä alkoi hahmottua tarkemmin. Suurin osa laajennusprosessin aikana ilmenneistä tyylimäärittelyongelmista johtui siitä, ettei tyylimäärittelyistä ollut toteutettu Windows Phone 8 -käyttöjärjestelmän selainmoottorille yhteensopivia toteutuksia. Tämän vuoksi osaa sovelluksen tyylimäärittelyistä ei voitu tulkita lainkaan, mistä johtuen sovelluksen käyttöliittymät eivät toimineet oikein. Tyylimäärittelyistä johtuvat ongelmat esiintyivät pääasiassa käyttöliittymäkomponenttien väreissä sekä mittasuhteissa. Ennen tyylimäärittelyjen korjaustoimenpiteitä suuri osa sovelluksen Windows Phone 8 -version käyttöliittymistä oli täysin käyttökelvottomassa kunnossa.

HILKKA-sovelluksen Windows Phone 8 -version käyttöliittymien tyylimäärittelyjen korjaustoimenpiteet aloitettiin sovelluksen toiminnan kannalta kriittisimmistä osista. Ensimmäiseksi korjattiin kaikki tyylimäärittelyt, joista johtuvat ongelmat estivät sovelluksen käytön tai haittasivat käyttöä merkittävästi. Tyylimäärittelyjen korjaustoimenpiteet osoittautuivat odotettua suuremmaksi urakaksi, koska vaadittujen muutoksien määrä oli odotettua suurempi. Tämän lisäksi tyylimäärittelyt oli testattava kaikilla sovelluksen tukemilla alustoilla, joka omalta osaltaan lisäsi työn määrää. Useimmissa ongelmatapauksissa tyylimäärittelyihin tarvitsi toteuttaa ainoastaan Windows Phone 8 -käyttöjärjestelmän selainmoottorille yhteensopiva toteutus, kuten kuvasta (Kuva 11) voi havainnoida.

```
.fr-menu-button {
  min-height: 2.5em !important;
  height: auto !important;
  width: 100% !important;
  background-color: #ffffff !important;
  border: 2 solid #e5e5e5;
  border-radius: 0px;
  margin: 0 !important;
  background-image: -webkit-gradient(
    linear, left bottom, left top,
    color-stop(0.1, rgba(255,255,255, 0)),
    color-stop(0, rgba(255,255,255, 0))) !important;
  background-image: -ms-linear-gradient(
    top,
    rgba(255,255,255, 0) 100%,
    rgba(255,255,255, 0) 0%) !important;
}
```

Kuva 11. HILKKA-sovelluksen fr-menu-button-komponentin tyylimäärittely (Kuva: Matti Mononen).

3.4.4 Laajennusprosessin haasteet

Vaikka HILKKA-sovelluksen laajennusprosessi etenikin alusta loppuun melko suoraviivaisesti, ei laajennusprosessista selvitty kuitenkaan täysin ongelmitta. Laajennusprosessin suurimmat ongelmat keskittyivät lähinnä sovellusalustojen välisiin eroavaisuuksiin, kuten esimerkiksi selainmoottoreihin ja tiedostojärjestelmiin. Myös sovelluksen toteutuksessa käytettyjen teknologioiden toimivuus eri sovellusalustoilla aiheutti omat haasteensa.

Laajennusprosessin ensimmäiset suuret haasteet kohdattiin jo ennen varsinaisen laajennustyön aloittamista. Koska HILKKA-sovellus perustuu erittäin vahvasti tietokantoihin ja koska sovelluksen käytössä oleva tietokanta sisältää asiakkaita koskevaa tietoa, on tietokannan sisältö salattava huolellisesti. HILKKA-sovelluksen Android-versiossa sovelluksen sisältämä SQLite-tietokanta salataan käyttämällä avoimen lähdekoodin SQLCipher-laajennusta (256-bittinen AES-salaus). Laajennusprosessin aloitushetkellä SQLCipher-laajennuksesta ei

kuitenkaan ollut saatavilla Windows Phone 8 -käyttöjärjestelmälle yhteensopivaa versiota. Laajennuksen kääntäminen Windows Phone 8 -käyttöjärjestelmälle olisi kuitenkin ollut mahdollista, joskin se olisi ollut todella paljon aikaa vievä prosessi. Laajennuksen kehityksestä vastaava Zetetic LLC julkaisi kuitenkin Windows Phone 8 -käyttöjärjestelmälle yhteensopivan version SQLCipher-laajennuksesta laajennusprosessin alkupuolella (Zetetic 2014).

Yksi laajennusprosessin suurimmista haasteista oli Windows Phone 8 -käyttöjärjestelmän tiedostojärjestelmä. Laajennusprosessin aikana tehdyissä testeissä sovelluksen Windows Phone 8 -versio hävisi sovelluksen Android-versiolle kaikissa tiedostojenkäsittelyyn liittyvissä operaatioissa. Kuten aiemmassa luvussa 3.4.2 todetaan, jouduttiin HILKKA-sovelluksen lähdekooditiedostojen pakkaus- ja purkurutiineihin tekemään suuria muutoksia, jotta sovelluksen ensimmäisen käynnistyskerran vaatima aika saatiin tiputettua järkeviin lukemiin. Myös tietokantakyselyt olivat selvästi hitaampia sovelluksen Windows Phone 8 -versiolla (Ruotsalainen 2014). Sovelluksen normaalissa käytössä tiedostojärjestelmän hitautta ei pysty kuitenkaan juuri havaitsemaan.

HILKKA-sovelluksen Android-versio on suunniteltu siten, että sovellus säilyy aktiivisena, vaikka käyttäjä siirtyisi käyttöjärjestelmän kotinäkymään tai kokonaan toiseen sovellukseen. Sovelluksen Android-versiossa taustalla pysymistä on parannettu hyödyntämällä Android-käyttöjärjestelmän taustapalveluita (services). Windows Phone 8 -käyttöjärjestelmä tukee ainoastaan yhtä aktiivista sovellusta kerrallaan. Siirryttäessä pois aktiivisesta sovelluksesta siirtyy Windows Phone 8 -sovellus normaalisti niin sanottuun ”keskeytetty” tilaan. HILKKA-sovelluksen kohdalla ongelmaksi muodostui se, että käyttöjärjestelmä ”tappoi” sovelluksen välittömästi sovelluksen siirtyessä taustalle. Tämä ongelma pystyttiin kuitenkin kiertämään merkitsemällä sovellus niin sanotuksi ”paikannussovellukseksi”. Tällä tavoin myös sovelluksen Windows Phone 8 -versio saatiin säilymään taustalla, joskaan sovelluksen taustalla pysyminen ei toimi aivan yhtä varmasti kuin sovelluksen Android-versiossa. (Windows Dev Center 2014c)

Vaikka HILKKA-sovelluksen toteutukseen valitut teknologiat mahdollistavatkin hybridisovelluksien kehittämisen usealle eri mobiilikäyttöjärjestelmälle, ei laa-

jennusprosessi toteutuksessa käytettyjen teknologioiden osalta sujunut kuitenkaan täysin ongelmitta. Suurimmat haasteet käytettyjen teknologioiden kohdalla ilmenivät Sencha Touch sovelluskehyksen kanssa. Esimerkiksi Sencha Touch -sovelluskehyksen sisältämä päivämääränvalintadialogi (Ext.field.DatePicker) oli käytännössä käyttökelvoton sovelluksen Windows Phone 8 -versiolla. Tämän vuoksi kyseisen käyttöliittymäkomponentin tilalle jouduttiin toteuttamaan liitännäinen, joka hyödyntää sekä Android- että Windows Phone 8 -käyttöjärjestelmien omia natiiveja päivämääränvalintamenetelmiä. Lopuksi sovelluksen lähdekoodi jouduttiin käymään läpi, jotta toimimaton käyttöliittymäkomponentti saatiin korvattua uudella toteutuksella.

Vaikka natiiviliitännäisten toteuttaminen Widows Phone 8 -käyttöjärjestelmälle olikin pääasiassa melko mutkatonta, koettiin laajennusprosessin aikana muutamia ongelmia Windows Phone 8 -käyttöjärjestelmän ohjelmointirajapintojen kanssa. Yhtenä HILKKA-sovelluksen ominaisuutena on, että se pystyy lukemaan muun muassa NFC-tageja (Near Field Communication). NFC-tagien lukemiseen perustuva ominaisuus hyödyntää NFC-tagien uniikkia ID-numerointia tunnistaakseen, mistä NFC-tagista on kysymys. NFC-tagien lukemista varten Windows Phone 8 -käyttöjärjestelmälle toteutettiin liitännäinen, jolla NFC-tagien lukeminen olisi mahdollista. Liitännäisen toteutuksessa jouduttiin kuitenkin päätymään Android-versiosta poikkeavaan ratkaisuun, koska NFC-tagien ID-numeron lukeminen Windows Phone 8 -käyttöjärjestelmän omien ohjelmointirajapintojen puitteissa ei ole mahdollista.

3.4.5 Laajennusprosessin tulokset

Noin kolme kuukautta kestäneen laajennusprosessin lopputuloksena HILKKA-sovelluksen Android-käyttöjärjestelmälle toteutetusta versiosta saatiin laajennettua viimeistelyä vaille valmis sovellus Windows Phone 8 -käyttöjärjestelmälle. Laajennusprosessin aikana HILKKA-sovelluksen Android-versioon toteutetuista natiiviliitännäisistä kolmetoista tärkeintä liitännäistä kirjoitettiin C#-ohjelmointikielellä Windows Phone 8 -käyttöjärjestelmälle yhteensopiviksi. Liitännäisten toteuttamisen lisäksi laajennusprosessissa keskityttiin muokkaamaan

HILKKA-sovelluksen käyttöliittymien tyylimäärittelyjä siten, että sovelluksen ulkoasu näyttäisi samalta niin sovelluksen Android- kuin Windows Phone 8 -versioilla. Edellä mainittujen vaiheiden lisäksi laajennusprosessin yhteydessä toteutettiin lukuisia parannuksia, jotka vaikuttavat molempiin HILKKA-sovelluksen versioihin.

Opinnäytetyön kirjoitushetkellä HILKKA-sovelluksen jatkokehitys on siirtynyt käytännössä kokonaan takaisin Android-version pariin ja Windows Phone 8 -version tulevaisuus onkin hieman epäselvä. Laajennusprosessin lopputuloksena syntynyt Windows Phone 8 -sovellus sisältää kaikki tärkeimmät HILKKA-sovelluksen ominaisuudet ja niin sanotun web-koodin osalta se noudattaa täysin samaa lähdekoodipohjaa kuin sovelluksen Android-versiokin. Vaikka sovelluksen jatkokehitys onkin siirtynyt käytännössä kokonaan takaisin Android-version pariin, ei uusien ominaisuuksien laajentaminen Windows Phone 8 -versioon ole tässä vaiheessa enää kovinkaan vaikeaa. Vaikka HILKKA-sovelluksen Windows Phone 8 -version tulevaisuus onkin vielä hieman epäselvä, saattaa HILKKA-sovellus laajentua Applen iOS-käyttöjärjestelmälle lähitulevaisuudessa (Ruotsalainen 2014).

4 Pohdinta

Tässä luvussa tarkastelen opinnäytetyöprosessia, opinnäytetyön esimerkkiprojektin kehitysideoita sekä tuntemuksia, joita opinnäytetyön tekeminen minussa sai aikaan. Opinnäytetyöprosessin osalta luvussa keskityn tarkastelemaan opinnäytetyön sisältöä, etenemistä sekä oppimisprosessia. Kehittämisideoiden osalta luvussa käsittelen HILKKA-sovelluksen jatkokehitystä sekä sovelluksen Windows Phone 8 -version viimeistelyä.

4.1 Opinnäytetyöprosessi

Opinnäytetyöni tutkimuskysymyksinä olivat, mitä ovat hybridimobiilisovellukset sekä mitä hybridimobiilisovelluksen laajentaminen alustalta toiselle vaatii. Opinnäytetyöni yhtenä tavoitteena oli käsitellä kyseisiä aiheita niin teorian kuin käytännön kautta. Luvussa 2 (Hybridimobiilisovellukset) keskityin käsittelemään aihetta teoriapainotteisesti, tarkastelemalla hybridisovelluksia yleisellä tasolla. Luvussa 3 (HILKKA-sovelluksen laajennusprosessi) tarkastelin opinnäytetyön esimerkkiprojektin toteutuksessa käytettyjä teknologioita sekä sovelluksen laajennusprosessia.

Opinnäytetyöprosessi eteni alusta loppuun melko mutkattomasti, eikä opinnäytetyön suunnitelmaan tai sisältöön tarvinnut tehdä suurempia muutoksia prosessin aikana. Opinnäytetyöprosessin mutkattomaan etenemiseen vaikutti osaltaan ainakin se, että opinnäytetyön esimerkkiprojektina toiminut HILKKA-sovellus ja sen laajennusprosessi toteutettiin käytännössä lähes kokonaan ennen varsinaista opinnäytetyöprosessin alkamista. Tämä helpotti huomattavasti opinnäytetyön suunnittelua sekä työn laajuuden arviointia.

Ennen HILKKA-sovelluksen laajennusprosessia minulla ei ollut minkäänlaista kokemusta Windows Phone 8 -sovelluskehityksestä. Tämä ei kuitenkaan tuottanut suurta ongelmaa, koska hallitsin C#-ohjelmointikielen vähintäänkin kohtuullisesti. Vaikka hybridisovelluksien kehittäminen eroaakin hieman natiivien Windows Phone 8 -sovelluksien kehittämisestä, antoi laajennusprosessi minulle valmiudet Windows Phone 8 -sovelluksien kehittämiseen. Myös hybridisovelluksien kehittäminen usealle eri mobiilikäyttöjärjestelmälle oli minulle entuudestaan tuntematonta aluetta. Vaikka projektin aikana kohdattiin lukuisia eri haasteita, olin jopa hieman yllättynyt siitä, miten helposti sovelluksen Windows Phone 8 -versio lopulta saatiin aikaiseksi.

4.2 Kehittämisideat

Hybridisovelluksilla on omat vahvuutensa ja heikkoutensa. Pienistä ongelmistaan huolimatta näen hybridisovelluksille oman paikkansa ja käyttötarkoituksensa mobiilimarkkinoilla. Hybridisovellukset ovat ehdottomasti vahvimmillaan silloin, kun sama sovellus halutaan toteuttaa useammalle eri kohdealustalle. Koska HILKKA-sovelluksen toteutukseen valitut teknologiat mahdollistavat sovelluksen toteuttamisen usealle eri mobiilikäyttöjärjestelmälle, näkisin sovelluksesta erittäin mielelläni myös version Applen iOS-käyttöjärjestelmälle. HILKKA-sovelluksen kohdalla sovelluksen laajentaminen ei ole kuitenkaan aivan näin yksinkertaista, koska sovellus on laajuudeltaan valtava ja se sisältää todella paljon natiivikoodin tasolle siirrettyjä toimintoja. Tämän lisäksi sovelluksen laajentaminen tekee sen ylläpidosta astetta haastavampaa.

Laajennusprosessin aikana HILKKA-sovelluksen Windows Phone 8 -versio saatiin käytännössä viimeistelyä vaille valmiiksi. Projekti pistettiin kuitenkin jäihin, koska sovelluksen Windows Phone 8 -versiolle ei löytynyt ottajia. Laajennusprosessiin käytetyn ajan nimissä haluaisin henkilökohtaisesti nähdä Windows Phone 8 -version valmistumisen. Järjellä ajatellen sovelluksen viimeistely tuntuisi kuitenkin olevan tällä hetkellä täysin ajan hukkaa. Sovelluksen Android-versio on jatkanut omaa elämäänsä laajennusprosessin jäätyä tauolle. Niinpä sovelluksen Windows Phone 8 -version viimeistely tulisi vaatimaan vielä melkolailla työtä, ennen kuin se olisi täysin julkaisukelpoisessa kunnossa. Nyt HILKKA-sovelluksen Windows Phone 8 -versiosta on olemassa kuitenkin toimiva pohja, johon perustuen Windows Phone 8 -versio voidaan viimeistellä, mikäli se joskus koetaan tarpeelliseksi.

Lähteet

Android developers. 2014. http://developer.android.com/sdk/index.html. 25.9.2014.
Apache Cordova. 2014a. Apache Cordova Overview. http://cordova.apache.org/docs/en/4.0.0/guide_overview_index.md.html#Overview. 1.11.2014.
Apache Cordova. 2014b. Apache Cordova Plugin APIs. http://cordova.apache.org/docs/en/4.0.0/cordova_plugins_pluginapis.md.html#Plugin%20APIs. 1.11.2014.
Apache Cordova. 2014c. Apache Cordova Plugin Development Guide. http://cordova.apache.org/docs/en/4.0.0/guide_hybrid_plugins_index.md.html#Plugin%20Development%20Guide. 1.11.2014.
Dalling, T. 2009. Model View Controller Explained. http://www.tomdalling.com/blog/software-design/model-view-controller-explained/. 29.10.2014.
FastROI. 2014. FastROI HILKKA. http://www.fastroi.fi/2014/hilkka. 23.9.2014.
Gartner. 2013. By 2016, More Than 50 Percent of Mobile Apps Deployed Will be Hybrid. http://www.gartner.com/newsroom/id/2324917. 15.9.2014.
Jones, A. 2012. Native, Hybrid or Web Apps. http://www.sitepoint.com/native-hybrid-or-web-apps. 17.9.2014.
Korf, M. & Oksman, E. 2014. Understanding Your Mobile Application Development Options. https://developer.salesforce.com/page/Native,_HTML5,_or_Hybrid:_Understanding_Your_Mobile_Application_Development_Options. 15.9.2014.
Kumar, A. 2012. Sencha MVC Architecture. Birmingham: Pact Publishing Ltd.
Mednieks, Z., Dornin, L., Meike, G. & Nakamura, M. 2012. Programming Android, Second Edition. Sebastopol: O'Reilly Media.
Ruotsalainen, J. Projektipäällikö. FastROI Oy. Haastattelun muistiinpanot. 11.9.2014.
Sass. 2014. Sass Basics. http://sass-lang.com/guide. 29.10.2014.
Sencha. 2014. Sencha Touch Features. http://www.sencha.com/products/touch/features/. 29.10.2014.
Traeg, P. 2013. Best of Both Worlds: Mixing HTML5 and Native Code. http://www.smashingmagazine.com/2013/10/17/best-of-both-worlds-mixing-html5-native-code/. 1.11.2014.
Zetetic. 2014. Introducing SQLCipher for Windows Phone 8 and Windows Runtime. https://www.zetetic.net/blog/2014/1/13/introducing-sqlcipher-for-windows-phone-8-and-windows-runtim.html. 20.9.2014.
Wikipedia. 2014a. Android (operating system). http://en.wikipedia.org/wiki/Android_(operating_system). 24.10.2014.
Wikipedia. 2014b. Windows Phone 8. http://en.wikipedia.org/wiki/Windows_Phone_8. 24.10.2014.

Windows Dev Center. 2014a. Native code for Windows Phone 8. http://msdn.microsoft.com/en-us/library/windows/apps/jj681687(v=vs.105).aspx. 26.9.2014.
Windows Dev Center. 2014b. Getting started with developing for Windows Phone 8. http://msdn.microsoft.com/en-us/library/windows/apps/ff402529(v=vs.105).aspx. 26.9.2014.
Windows Dev Center. 2014c. Running location-tracking apps in the background for Windows Phone 8. http://msdn.microsoft.com/en-us/library/windows/apps/jj681691(v=vs.105).aspx. 21.9.2014.

Haastattelun muistiinpanot

Miten ja miksi HILKKA sovelluksessa päädyttiin hybridiratkaisuun?

- Ei haluttu sitoutua yhteen mobiilialustaan
- Hybridisovelluksen kohdalla sama sovellus on nopeasti toteutettavissa myös muille sovellusalustoille (= vähemmän alustakohtaista koodia)

Oliko Windows Phone versio suunnitelmissa alusta asti?

- Kyllä

Kääntyykö HILKKA tulevaisuudessa myös iOS:lle?

- Todennäköisesti kyllä

Miten ja miksi HILKKA sovelluksessa päädyttiin käyttämään Sencha Touchia?

- Työkaluja vertailtiin ennen niiden lopullista valintaa
  - Enyo (Open Source)
  - Sencha Touch
- Sencha Touchissa on kaupallinen tuki
- FastROI mERP-järjestelmässä päädyttiin Ext Js / Sencha Touch ratkaisuun

Miten ja miksi HILKKA sovelluksessa päädyttiin käyttämään Apache Cordovaa?

- Apache Cordovan takaa löytyy ns. ”luotettava toimija”
- Apache Cordovan kohdalla OpenSourcessa ei nähty riskejä
- Ei ollut ollut tarvetta kokeilla mutia vastaavia sovelluskehyksiä

Mitkä olivat mielestäsi käännösprosessin suurimmat haasteet? onnistumiset?

- SQLChipher
    - Käytetään sovelluksen tietokannan kryptauksen purkuun
    - Windows Phone 8 yhteensopiva versio olisi voitu kääntää itse
        - Olisi ollut paljon aikaa vievä prosessi
    - Zetetic julkaisi maksullisen version Windows Phone 8 -käyttöjärjestelmälle
- Windows Phone (+ IE) ongelmat yleensä
    - Sencha Touch ongelmat
    - NFC
- Hidas tiedostojärjestelmä (by design)

Millä tapaa käännösprosessi vaikutti HILKKA sovelluksen Android versioon?

- Useita yleisiä parannuksia jotka vaikuttivat molempiin sovelluksen versioihin
    - Hidas käynnistyminen (tiedostojärjestelmä)
        - behemoth.js

Mitkä ovat HILKKA sovelluksen suurimmat erot Android ja Windows Phone 8 versioiden välillä?

- Tiedostojärjestelmän nopeus
    - Käytännössä kaikki tiedostojenkäsittelyyn liittyvät operaatiot toimivat nopeammin Android-käyttöjärjestelmällä

Mitkä ovat Windows Phone 8 käyttöjärjestelmän suurimmat vahvuudet? heikkoudet?

- HILKKA-sovelluksen kohdalla Android kaikilta osin parempi vaihtoehto
- Windows Phone 8 -laitteiden marginaalinen osuus mobiilimarkkinoilla
    - Vaikea löytää apua ongelmatilanteisiin yms.

Millaisena näet HILKKA sovelluksen Windows Phone 8 version tulevaisuuden?

- Tällä hetkellä hiljaista