ALUSTARIIPPUMATON MOBIILISOVELLUSKEHITYS WEB-TEKNIIKOILLA

Case: Tiimiakatemian GAS-mobiilisovellus

Tatu Niittykangas

Opinnäytetyö
Lokakuu 2013

Tietojenkäsittelyn koulutusohjelma
Luonnontieteiden ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä(t) NIITTYKANGAS, Tatu	Julkaisun laji Opinnäytetyö	Päivämäärä 13.10.2013
	Sivumäärä 49	Julkaisun kieli Suomi
	Luottamuksellisuus ( ) saakka	Verkkojulkaisulupa myönnetty (**X**)

Työn nimi
ALUSTARIIPPUMATON MOBIILISOVELLUSKEHITYS WEB-TEKNIIKOILLA
Case: Tiimiakatemian GAS-mobiilisovellus

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Työn ohjaaja(t)
TUIKKA, Tommi

Toimeksiantaja(t)
Tiimiakatemia, Heikki Toivanen

Tiivistelmä

Työn toimeksiantajana toimi Tiimiakatemia, joka tarvitsi modernin työkalun osuuskuntien johtamisen helpottamiseen. Opinnäytteen tavoitteena oli tehdä toimeksiantajalle kartoitus mahdollisuuksista toteuttaa monialustainen mobiilisovellus käyttäen web-tekniikoita ja selvittää samalla sovellukselle sopivin jakelutapa. Viitekehyksenä työlle toimi opinnäytetyön osana toteutettu kaverijohtajuusmittari-mobiilisovellus.

Tarkempaan selvitykseen valituista tekniikoista ja työkaluista kerättiin aineistoa, jonka pohjalta nämä arvotettiin kaverijohtajuusmittari-sovellukseen soveltuvuuden perusteella. Kerätyn tiedon perusteella valittiin toteutettavaan prototyyppiin parhaiten toimeksiantajan esittämiä toiveita vastaavat työkalut ja teknologiat, ja nämä arvioitiin tarkemmin prototyyppäämällä.

Teoreettisen puolen lisäksi opinnäytetyössä toteutettiin toimeksiantajalle halutusta mobiilisovelluksesta sekä sen tarvitsemasta taustajärjestelmästä prototyypit. Itse mobiilisovelluksesta toteutettiin myös sekä natiivisovellukseksi paketoitu versio että selainpohjainen versio.

Toimeksiantaja sai opinnäytetyön tuloksena mobiilisovelluksesta toimivan prototyypin, jonka jatkokehitys aloitettiin välittömästi työn valmistuttua.

Avainsanat (asiasanat)
Alustariippumaton sovelluskehitys, HTML5, Javascript, mobiilikehitys

Muut tiedot
Liitteet, 5 sivua

Author(s)
NIITTYKANGAS, Tatu

Type of publication
Bachelor´s Thesis

Date
13.10.2013

Pages
49

Language
Finnish

Confidential

( ) Until

Permission for web publication
(**X**)

Title
PLATFORM INDEPENDENT MOBILE DEVELOPMENT USING WEB TECHNOLOGIES
Case: Team Academy's GAS mobile application

Degree Programme
Business Information Systems

Tutor(s)
TUIKKA, Tommi

Assigned by
Team Academy Jyväskylä, Heikki Toivanen

Abstract

The thesis was done for Team Academy Jyväskylä which was in need of a modern tool for making the managing of cooperatives easier. The objective of the thesis was to map the current possibilities of developing platform independent mobile applications using web technologies and the most suitable ways of distributing them. The kaverijohtajuusmittari mobile app, which was developed as a part of this thesis, was used as a framework for the thesis.

The technologies and tools chosen for closer inspection were investigated and analysed and then valued based on their applicability on the kaverijohtajuusmittari mobile app. A prototype was developed based on these findings and on how they best matched the requirements set by the client.

A prototype of the mobile app and its required backend system were developed during the creation of this thesis in addition to the theoretical side. As a proof of concept the mobile app was prototyped packaged as a native mobile application and also as a pure web application.

As a concrete result of this thesis the client received a prototype of the kaverijohtajuusmittari mobile app. Further development of the app begun right after this thesis was completed.

Keywords
Platform independent development, HTML5, Javascript, mobile development

Miscellaneous
5 pages of appendices

Sisältö

KUVIOT

TERMIT

API, Application Programming Interface	Ohjelmistojen keskinäiseen kommunikointiin tarkoitettu protokolla
C#	Microsoftin kehittämä oliopohjainen ohjelmointikieli
Compass	Sass-kielen laajennos, joka lisää siihen uusia ominaisuuksia
CSS, Cascading Style Sheets	XML-dokumenttien (mukaanlukien HTML) ulkoasun ja asettelun määrittelykieli
cURL	Suosittu komentorivityökalu, jolla voidaan siirtää dataa erinäisten protokollien kautta
Eclipse	Suosittu, useita ohjelmointikieliä tukeva IDE
Grails	Nopean iteroinnin ohjelmistokehys Groovylle
Groovy	JVM:n päällä toimiva dynaaminen ohjelmointikieli
HTML, Hypertext Markup Language	SGML:ään pohjautuva verkkosivujen merkkauskieli
IDE, Integrated Development Environment	Ohjelmistokehitysympäristö, johon on paketoitu yhteen useita työkaluja
JavaScript	Prototyyppipohjainen ohjelmointikieli, pääasiallisesti

	käytetty verkkosivuilla
JVM	Java Virtual Machine
LESS	CSS-tyylikielen laajennos, joka lisää siihen dynaamisia ominaisuuksia
Node.js	Googlen Chromen JavaScript-moottoriin pohjautuva ohjelmointialusta
Python	Suosittu dynaaminen ohjelmointikieli
REST, Representational State Transfer	Ohjelmistoarkkitehtuurimalli, de facto standardi internetin API:ssa
Ruby	Dynaaminen ohjelmointikieli, suosittu erityisesti web-kehittäjien keskuudessa
Ruby on Rails	Suosittu Ruby-kielellä toteutettu web-ohjelmistokehys
Sass	CSS-tyylikielen laajennos, joka lisää siihen dynaamisia ominaisuuksia
SDK, Software Development Kit	Ohjelmistokehitystyökalupaketti
TDD, Test-Driven Development	Ohjelmistokehityksen prosessi, jossa kirjoitetaan ensin testit tehtäville ominaisuuksille ja sitten implementoidaan ne niin, että testit menevät läpi.
V8	Googlen kehittämä JavaScript-moottori
Visual Studio	Microsoftin kehittämä IDE
XHTML, Extensible Hypertext Markup	XML:ään pohjautuva HTML-kielen

Language	laajennos
XML, Extensible Markup Language	Rakenteinen merkkauskieli

1 JOHDANTO

Älypuhelinten määrän räjähdysmäisen kasvun myötä mobiilipalvelut ovat tulleet suurten massojen taskuihin. Liikkeellä on yli 750 miljoonaa Android-laitetta (Page 2013), pelkästään iPhoneja on myyty yhden vuosineljänneksen aikana lähes 50 miljoonaa kappaletta (Apple Reports Record Results 2013) ja Windows Phonekin on onnistunut kipuamaan kolmen prosentin markkinaosuuteen (Gartner Says Worldwide Mobile Phone Sales Declined 1.7 Percent in 2012 2013) – älypuhelimesta on hyvin nopeasti tullut puhelinten de facto standardi.

Kun jossakin on noin valtava potentiaalisten käyttäjien massa, se vaikuttaa suoraan kehityksen suuntaan. Kärjistäen voidaan sanoa, että jokainen instanssi on julkaissut tai on vähintään julkaisemassa omaa mobiilisovellustaan (Austin 2013), ja pelialallakin mobiilipelit ovat suurin yksittäinen kasvaja (2012 PopCap Games Mobile Gaming Research 2012). Lähitulevaisuuden selvä trendi tulee olemaan mobiili, mutta ehkä sitäkin olennaisemmin liikkuvuus (Armano 2012); palveluiden käyttämisen on mahdollista olla yhä vähemmän sidottu aikaan sekä paikkaan, ja käyttäjät tulevat odottamaan tätä palveluilta.

Samalla kun mobiililaitteiden määrä on kasvanut, se on myös vahvasti jakautunut. Aiemmin Nokian dominoimat markkinat jaettiin Applen ja erinäisten Android-valmistajien kesken – vaikka Nokia nyt onkin yrittänyt Windows Phonen myötä palata takaisin kentälle – ja jokainen uusi alusta oli teknisesti erilainen kuin aiemmat. Monen erilaisen alustan tukeminen on suuri taakka ohjelmistokehityksessä ja varsinkin pienten kehittäjien kohdalla, joiden resurssit ovat rajalliset. Tämä on johtanut siihen, että ohjelmia ja pelejä on julkaistu vain tietyille alustoille, kuten esimerkiksi Angry Birds, joka oli noin vuoden saatavilla ainoastaan Applen iOS-alustalla (Ionescu 2010).

Ihanteellinen tilanne olisi tietysti se, että jokainen mobiiliohjelmisto olisi saatavilla jokaiselle alustalle: kehittäjille se tarkoittaisi suurempaa potentiaalista käyttäjämäärää samalla työmäärällä ja loppukäyttäjille suurempaa valinnanvaraa ja tarjontaa yleisesti. Sekä alustariippumaton ohjelmistokehitys että entistä paremmat web-sovellukset ovat helpottaneet tätä ongelmaa. Muutaman vuoden aikana molemmat näistä ovat kehittyneet valtavasti eteenpäin ja useita tuotantovalmiita ratkaisuita on tuotu markkinoille (Krill 2009; Taft 2009; de Icaza 2009). Jokainen ohjelmisto ei kuitenkaan vaadi natiivia paketointia ja modernit web-teknologiat ovat mahdollistaneet hyvin rikkaiden selainpohjaisten sovellusten luonnin, jolloin erillistä mobiiliohjelmistoa ei välttämättä tarvitse ylläpitää. Useampikin alustariippumaton mobiilisovelluskehys tarjoaa myös mahdollisuuden paketoida web-teknologioilla tehty sovellus natiiviksi sovellukseksi.

Tämä opinnäytetyö ei ota kantaa natiivin mobiilisovelluskehityksen ja alustariippumattoman mobiilisovelluskehityksen välisiin eroihin, vaan keskittyy käsittelemään toimeksiantajan mobiilisovellusprojektin viitekehyksessä web-tekniikoilla toteutettuja moderneja mobiilisovelluksia. Näiden sovellusten natiivipaketoinnin hyötyjä ja haasteita käsitellään sen verran kuin se on oleellista toteutettavaan sovellusprojektiin liittyen.

2 TUTKIMUSASETELMA

2.1 Taustatietoa, tavoitteet ja rajaukset

Toimeksiantaja

Opinnäytetyön toimeksiantajana toimi Jyväskylän ammattikorkeakoulun Tiimiakatemia ja työtä toimeksiantajan päässä ohjasi Heikki Toivanen. Työ tehtiin osana Tekesin rahoittamaa Liminari – Työ ja johtajuus liminaalitilassa -tutkimushankkeen osahanketta Kaverijohtajuus osuuskunnissa. Tiimiakatemian lisäksi yhteistyötä tehtiin Silmu Softwaren kanssa mobiiliohjelmiston teknisiin ratkaisuihin ja jatkokehitykseen liit-tyen.

Tausta

Vuoden 2012 loppupuolella Tiimiakatemia oli lähestynyt tietojenkäsittelyn koulutusohjelmaa mobiiliohjelmistoprojektin merkeissä ja sattumalta olin itse samoihin aikoihin tekemässä Android-sovellusta toiseen projektiin. Kiireet kuitenkin estivät minua ottamasta Tiimiakatemian projektia hoidettavaksi syksyn aikana ja projekti saatiin alkuun vasta keväällä 2013.

Alunperin tarkoituksena oli tehdä mobiiliohjelmisto Applen iOS-käyttöjärjestelmälle ja tätä kautta mukaan tuli Silmu Software, jolta Tiimiakatemia oli aiemmin tiedustellut kyseisenlaisen mobiiliohjelmiston toteutusta. Yhteistyötä Silmu Softwaren kanssa päätettiin jatkaa niin, että he toteuttaisivat tämän opinnäytteen perusteella tehdystä prototyypistä lopullisen tuotteen; Silmu myös esitti muutamia toiveita sovelluksessa käytettävistä tekniikoista jatkokehitystä ajatellen.

Työn tavoite

Työn tavoitteena on tarkastella Tiimiakatemian tarvitseman mobiilisovelluksen kontekstissa web-teknologioilla tapahtuvaa alustariippumatonta mobiili-sovelluskehitystä ja toteuttaa optimaalisiksi todetuilla menetelmillä Tiimiakatemialle käyttökelpoinen prototyyppi sovelluksesta ja sen taustajärjestelmästä.

Tiimiakatemia halusi erityisesti modernin sovelluksen, joka sopii Y-sukupolvea edustaville käyttäjille (Toivanen 2013), joten prototyypin painopiste tulee olemaan helppokäyttöisyydessä ja interaktiivisuudessa, jotta haluttu vaikuttamisen tunne välittyy loppukäyttäjälle.

2.2 Tutkimusmenetelmät

Opinnäytetyö muotoutui lopulta kehittämistutkimukseksi. Tavoitteena on analysoida toimeksiantajalla olevaa ongelmaa ja tuottaa siihen ratkaisu, jolle perusteet esitetään keräämällä tarpeeksi kattava teoriapohja. Teoriapohja kasataan kartoittamalla käytettävissä oleva potentiaalinen teknologiavalikoima sekä analysoimalla se. Kerätyn tiedon pohjalta luodaan alustavia ratkaisusuunnitelmia toimeksiantajalla olevaan ongelmaan. Suunnitelmista valitaan parhaiten toimeksiantajan tarpeisiin soveltuva ja sen pohjalta kehitetään ratkaisu, joka on tässä tapauksessa sovelluksen prototyyppi.

2.3 Tutkimuskysymykset

Opinnäytetyön keskeinen tutkimuskysymys on kiteytettävissä seuraavasti:

- Onko web-tekniikoilla toteutetun kaverijohtajuussovelluksen kannalta parempi ratkaisu paketoida se natiiviksi ohjelmaksi vai tarjota se web-sovelluksena?

Tähän kysymykseen liittyy olennaisesti useampia alakysymyksiä:

- Saavutetaanko natiiveiksi sovelluksiksi paketoinnilla merkittäviä etuja?
- Tarvitaanko toteutettavassa sovelluksessa natiivisovellusten ominaisuuksia?
- Mitä vaihtoehtoja markkinoilla on web-tekniikoilla tehdyn sovelluksen paketoimiseen natiiveiksi sovelluksiksi?
- Kuinka paljon ylimääräistä työtä natiivisovelluksiksi paketoinnista seuraisi?

3 ALUSTARIIPPUMATON MOBIILISOVELLUSKEHITYS

3.1 Määritelmä

Alustariipumattoman ohjelmistokehityksen tarkoituksena on tuottaa ohjelmistoja, jotka joko toimivat useilla alustoilla tai ne voidaan kääntää useille alustoille, ja suoria riippuvuuksia alustojen ominaisuuksiin ei ole.

Yksi tapa saavuttaa alustariippumattomuus on käyttää sovelluksen ja käyttöjärjestelmän välissä alustariippumatonta välikerrosta, joka hoitaa kommunikaation itse käyttöjärjestelmän ja sovelluksen välillä. Esimerkkejä tällaisesta ovat esimerkiksi Java, jossa sovellukset ajetaan virtuaalikoneessa, ja internet-selaimet, jotka voidaan nykyään jo käytännössä luokitella sovellusalustoiksi; selaimet ovat nousseet merkittäväksi sovellusalustaksi viime vuosina uusien, entistä suorituskykyisempien web-teknologioiden ansiosta.

3.2 Mobiilialustojen erot ja erityispiirteet

Nykyiset, työn kannalta relevantit mobiilialustat eroavat toisistaan suuresti niin arkkitehtuurinsa, avoimuutensa kuin käytettävissä olevien työkalujen ja ohjelmointikieltenkin osalta. Relevanteiksi on määritelty tässä työssä ne alustat, joille toimeksiantaja on alunperin halunnut mobiilisovelluksensa julkaista, sekä ne joilla on vahvin asema kotimaisten käyttäjien keskuudessa. Alustojen yleisiä ominaisuuksia ei käsitellä tässä opinnäytetyössä, vaan ainoastaan niiden eroavaisuuksia sovelluskehityksen kannalta sekä sitä, mitä mahdollisia lisätöitä ja -kustannuksia monen alustan tukemisesta saattaa aiheutua.

Android

Android on alunperin Android, Inc:in kehittämä ja Googlen myöhemmin ostama käyttöjärjestelmä, joka pohjautuu Linuxin ytimeen. Vuodesta 2008 asti markkinoilla ollut käyttöjärjestelmä on nopeasti kohonnut mobiilimaailman suosituimmaksi alustaksi (Android Marks Fourth Anniversary Since Launch with 75.0% Market Share in Third Quarter, According to IDC 2012), joten sen tukeminen on lähestulkoon välttämättömyys.

Androidin sovellukset ovat pääosin Javalla koodattuja ja niitä pyörittää Googlen oma Dalvik-virtuaalikone tavanomaisen Oraclen virtuaalikoneen sijaan. Kehittäjien on myös mahdollisuus käyttää Javan sijaan tai sen rinnalla C:tä tai C++:aa sovelluskehitykseen Androidin NativeDevelopment Kitin avulla, mutta näiden käytöstä on yleensä hyötyä vain hyvin spesifeissä tilanteissa.

Google tarjoaa kehitystyökalut sovellusten tekemiseen sekä valmiina kokonaispakettina että erillisenä kirjastona, jonka kehittäjä voi ottaa käyttöön haluamassaan ympäristössä. Tämä on tärkeää huomioida siksi, että Androidin kehitystyökalut eivät ole sidottuja mihinkään tiettyyn alustaan, joten kehittäjä voi työskennellä haluamassaan ympäristössä (Developer Tools | Android Developers 2013).

Googlen Play-sovelluskauppa on Androidilla pääasiallinen sovellusten asennuskanava, mutta ei suinkaan ainoa. Amazonilla on oma sovelluskauppansa, kuten on myös Barnes & Noblella, ja käyttäjät voivat myös asentaa sovelluksia vapaasti sovelluskauppojen ulkopuolelta.

iOS

Applen iOS on älypuhelinmarkkinoilla toiseksi käytetyin käyttöjärjestelmä ja käytössä ainoastaan Applen iPhone-, iPad-, AppleTV- sekä iPodTouch -laitteissa. Vuoden 2007 julkaisun jälkeen iOS on noussut nopeasti suureen suosioon varsinkin Yhdysvalloissa.

Sovelluskehitys iOS:lle vaatii, että kehittäjä käyttää ohjelmointikielenä Objective-C:tä, jota käytetään käytännössä vain Applen iOS- ja OSX-ympäristöissä, ja kehitysympäristönään Xcodea, joka on saatavilla vain OSX:lle. Kehittäjän on käytännössä pakko omistaa Applen kannettava tai pöytäkone kehittääkseen sovelluksia Applen mobiililaitteille.

Apple ei salli sovellusten asentamista mitenkään muuten kuin heidän oman App Storensa kautta. Kehittäjien, jotka haluavat julkaista sovelluksensa AppStoressa, on maksettava vuosittainen kehittäjämaksu, joka on kirjoitushetkellä 99 dollaria.

Windows Phone

Windows Phone on Microsoftin Windows Mobilen moderni seuraaja, joka julkaistiin vuonna 2010. Ensijulkaisun jälkeen Windows Phone on päivittynyt täyden version verran ja muuttunut samalla jonkin verran.

Koska Windows Phone on Microsoftin tuote, on loogista että kehitystyökalut ovat saatavilla vain Windows-alustalle; Windows Phone 7 -kehitys vaatii vähintään Windows Vistan, ja Windows Phone 8 -kehitys vaatii Windows 8:n. Kehitykseen vaadittava Windows Phone SDK asentuu Visual Studion lisäosaksi, ja Visual Studio itsessäänkin on saatavilla vain Windows-alustalle.

Natiivisovellusten kehitys tapahtuu C#:illa tai C++:lla, ja Windows Phone 8:n kohdalla on myös mahdollista tehdä sovelluksia käyttämällä vain HTML/CSS/JavaScript-

komboa – samaan tapaan kuin myös Windows 8:lle on mahdollista tehdä Modern UI -sovelluksia. Windows Phone sallii sovellusten asentamisen vain Windows Phone Storen kautta ja sinne pääseminen vaatii vuosittaisen kehittäjämaksun maksamisen. Kirjoitushetkellä tuo maksu on 99 dollaria vuodessa.

Muut

Kolmen tällä hetkellä relevanteimman mobiilialustan lisäksi käytössä on vielä useita alustoja, joiden osuudet ovat joko laskeneet uusien alustojen tullessa markkinoille tai jotka ovat vasta hakemassa jalansijaa markkinoilla. Näiden tukeminen ei tule olemaan oleellista toimeksiantajalle tehtävän sovelluksen kannalta, mutta tulevaisuuden mahdollisuudet on hyvä kuitenkin tiedostaa.

Vaikka Nokia onkin hypännyt Microsoftin kelkkaan, on Symbian-puhelimia vielä liikkeellä jonkin verran. Nokia on kuitenkin hylännyt alustan, joten jatkossa alustan tukeminen tulee olemaan vieläkin turhempaa. Jo nyt toimeksiantajan tavoittelemat Y-sukupolven edustajat ovat pääosin jo siirtyneet käyttämään jotakin kolmesta suuresta alustasta.

Samsung on tämän hetken isoimpia tekijöitä älypuhelinmarkkinoilla, joten oli vain ajan kysymys, milloin oman mobiilialustan kehitys tuli ajankohtaiseksi. Alunperin Samsung kehitti omaa Bada-alustaansa, mutta päätyi lopulta yhdistämään voimansa Intelin kehittämän Tizen-alustan kanssa (Byford 2013). Koska Bada on hylätty ja Bada-laitteita ei voida päivittää käyttämään Tizeniä, on alustan tukeminen täysin turhaa. Tizen-laitteita ei vielä ole markkinoilla, mutta koska sen sovelluskehitys pohjautuu HTML5:een (About | Tizen 2013), on sen mahdollinen tukeminen tulevaisuudessa luultavasti helppoa.

Firefox OS on niin tuore tulokas, että markkinoilla on vasta kehittäjille tarkoitettuja laitteita, ja nekin julkaistiin kirjaimellisesti tämän kirjoitushetkellä (Wauters 2013). Alustana Firefox OS eroaa kilpailijoistaan – Tizen pois lukien – sillä, että se on tehty alusta asti HTML5-sovelluksia varten, ja niille tarjotaan suoraan pääsy laitteiden ominaisuuksiin kiinni ilman välissä olevia sovelluskehyksiä (Introduction to Firefox OS 2013). On hyvinkin mahdollista, että Mozilla on riittävän iso entiteetti teknologia-maailmassa ja alusta saavuttaa jalansijan markkinoilla, mutta tällä hetkellä mitään konkreettista ei vielä tiedetä.

3.3 Alustariippumattomat mobiilikehitystyökalut

Alustariippumattomia kehitystyökaluja on markkinoilla suhteellisen runsaasti, mutta tämän opinnäytetyön puitteissa niistä käsitellään vain Tiimiakatemian projektiin soveltuvat. Suurin yksittäinen rajoitus oli, että työkalun tulee paketoida web-teknologioilla tehty sovellus natiiviksi sovellukseksi; osa työkaluista mahdollistaa esimerkiksi Pythonin tai C#:in käytön kaikille alustoille kehittämiseen, mutta nämä jätetään tämän työn ulkopuolelle. Toinen tärkeä rajoitus oli, että työkalun tulee olla ilmainen tai tarjota tarpeeksi pitkä testausjakso, jotta prototyypin kehitystyökalujen valintaan ei turhaan jouduta käyttämään rajallista budjettia. Tämä rajasi ulos mm. IBMWorklight-työkalut, Jemben, AMPchroman sekä Application Craftin. Kolmas rajoitus oli, että työkalulla pitää olla mahdollista paketoida sovellus kaikille relevanteiksi nähdyille mobiilialustoille, joten pelkästään Androidia ja iOS:ää tukevat työkalut, joiden tulevaisuudessa ei näkynyt tukea muillekin alustoille, rajattiin tästä ulos; tämä tarkoitti mm. hyvinkin lupaavan Titanium Appceleratorin jättämistä vertailun ulkopuolelle.

PhoneGap

PhoneGap on alun perin Nitobin kehittämä ja sittemmin Adoben ostama (Adobe Announces Agreement to Acquire Nitobi, Creator of PhoneGap 2011) ohjelmisto-kehys alustariippumattomaan mobiilisovelluskehitykseen. Nitobi oli ennen myymistään ollut aikeissa lahjoittaa PhoneGapin lähdekoodit Apache-säätiölle hallinnoitavaksi ja Adobe ei vastustanut tätä, joten lahjoitus toteutui. Tästä syntyi Apache Cordova -projekti, jonka yksi jakeluversio PhoneGap nykyään on. Tällä hetkellä ainoa ero Apache Cordovan ja PhoneGapin välillä on nimi, mutta Adobe luultavasti kehittää PhoneGapin ympärille lisää työkaluja ja mahdollistaa sen vahvemman integraation omiin tuotteisiinsa perustuvaan työnkulkuun (LeRoux 2012).

PhoneGap tukee kaikkia tällä hetkellä relevantteja mobiilialustoja, sekä muutamia, jotka ovat väistyneet uusien tieltä, ja vielä jalansijaa hakevia, kuten Intelin ja Samsungin kehittämää Tizen-alustaa.Se, mikä tekee PhoneGapista erityisen kehittäjäystävällisen, on Adoben tarjoama PhoneGap Build -pilvipalvelu, joka kääntää kehittäjän puolesta sovelluksen natiiveiksi sovelluksiksi halutuille alustoille; kehittäjän vastuulle jää silti natiivilta näyttävästä ulkoasusta huolehtiminen, mikä tietysti tuottaa lisätöitä, mutta itse kehitystyö voidaan hoitaa kokonaan yhdellä alustalla.

Käytännössä PhoneGap toimii siten, että se tarjoaa kehittäjälle natiivin projektin vaatimat kirjastot ja näiden lisäksi JavaScript-kirjaston, jonka kautta web-sovellus pääsee käsiksi laitteen ominaisuuksiin, kuten GPS:ään, tiedostojärjestelmään ja kameraan. Itse web-sovellus riippuu PhoneGapista vain tuon yhden JavaScript-tiedoston verran. Tämä mahdollistaa sen, että kehittäjä voi valikoida itselleen sopivat työkalut toisin kuin molemmissa vertailtavissa vaihtoehdoissa, jotka tarjoavat omat erilliset kehitysympäristönsä.

MoSync

MoSync on ruotsalaisen MoSync AB:n kehittämä alustariippumaton mobiilikehitystyökalupaketti, joka tukee sekä web-tekniikoiden että C++:n käyttämistä mobiilisovellusten toteuttamiseen. Alun perin vuonna 2005 julkaistu MoSync lisäsi vuonna 2010 tuen Androidille sekä iOS:lle (What's New in MoSync SDK 2.4 2010) ja vuonna 2012 Windows Phone 7:lle (What's New in MoSync SDK 3.0 2012). Versiossa 2.7 MoSynciin tuli uutena ominaisuutena JavaScript-kirjasto, Wormhole, joka mahdollisti natiivisovellusten teon web-tekniikoilla, kun aiemmin se oli mahdollista vain C++:aa käyttäen.

MoSyncin mukana tulee juuri MoSyncin käyttöä varten kustomoitu IDE, joka pohjautuu useiden muiden tapaan Eclipseen. IDE pitää sisällään valmiit projekti-

pohjat ja muita ohjelmistokehyksen käyttöönottoa helpottavia ominaisuuksia. Kääntöpuolena tämä myös sitoo kehittäjää käyttämään MoSyncin tarjoamia työkaluja sen sijaan, että hän voisi valita itselleen parhaiten sopivat.

MoSyncin ja PhoneGapin yksi suuri ero on, että MoSync on ollut olemassa pidempään ja tuki web-tekniikoilla tehdyille sovelluksille on lisätty vasta monia vuosia julkaisun jälkeen. Alun perin MoSyncin käyttö on vaatinut C++:n käyttämistä sovellusten ohjelmointiin, ja tämä vaihtoehto on säilytetty edelleen. Muuten MoSync toimii pääosin kuten PhoneGap: ohjelmistokehys tarjoaa eri alustojen tarvitsemat kirjastot ja JavaScript-tiedoston, jonka kautta web-sovellus pääsee käsiksi laitteen ominaisuuksiin.

MoSync tarjoaa suuren kirjon tuettuja alustoja, mukaan lukien tietysti halutut iOS:n, Androidin ja Windows Phonen. Lopullinen sovelluksen paketointi natiivisovellukseksi hoituu MoSyncillä muuten millä tahansa alustalla, mutta iOS-versio on pakko paketoida loppuun Applen Xcode-työkaluilla, joita ei ole saatavilla kuin Applen OSX-käyttöjärjestelmälle. Tässä mielessä MoSync häviää PhoneGapille, joka tarjoaa mahdollisuuden luoda sovelluspaketit erillisen PhoneGap Build -pilvipalvelun avulla. Oheisohjelmistojen puutteen rinnalle oikeaksi ongelmaksi nousee se, että MoSync ei vielä tue uusimpia versioita Windows Phonesta tai iOS:stä (Feature/Platform Support 2012), eikä varmuutta näiden tuesta vielä ole.

Intel XDK

Uusin tulokas web-tekniikoiden paketoimisessa natiivisovelluksiksi, vaikkakin yrityskaupan kautta, on Intel, joka osti appMobin kehittämät työkalut ja risti ne Intel XDK:ksi (Lunden 2013). Toisin kuinPhoneGap ja MoSync, Intel XDK ei tarjoa omaa ratkaisuaan alustariippumattomaan kehitykseen vaan rakentaa PhoneGapin päälle oman kehitysympäristönsä (The XDK turbocharges PhoneGap 2013). Kehittäjille XDK

tarjoaa PhoneGapin tavoin mahdollisuuden luoda mobiilisovelluksia pelkästään HTML:llä, CSS:llä sekä JavaScriptillä, jotka XDK sitten kääntää natiiviksi sovelluksiksi halutuille alustoille.

Intel XDK tuo omana lisäyksenään sovelluskehitystä helpottamaan avoimen lähdekoodin App Framework -kirjaston, joka koostuu jQueryn korvaavasta kirjastosta, käyttöliittymäkirjastosta ja mahdollisista lisäpalikoista (App Framework 2013). Intel XDK on vertailtavista helpoin ottaa käyttöön, sillä se toimii suoraan selaimessa. Kehitysympäristö on mahdollista ladata omalle koneelle, mutta se on mahdollista myös ottaa käyttöön Chrome-selaimen lisäosana; molemmissa tapauksissa se kuitenkin käynnistyy selaimessa. Intel XDK tarjoaa kehittäjälle yksinkertaisen ja harmillisen vähäominaisuuksisen koodieditorin – mutta myös mahdollisuuden käyttää itse valitsemaansa editoria – interaktiivisen testausympäristön sekä sovelluksen julkaisupalvelun. Ohjelmistokehystä on myös mahdollista käyttää erillisen kehitysympäristön kanssa (How to use the appMobi XDK with Visual Studio or other IDEs to develop cross platform mobile apps 2012), jolloin XDK:n vastuullee jää toimia vain emulaattorina ja paketoijana.

Intel XDK tukee pienintä määrää mobiilialustoja vertailtavista vaihtoehdoista, mutta kuitenkin kaikkia kolmea oleellista: Androidia, iOS:ää sekä Windows Phonea. PhoneGapin tukemista alustoista pois on jätetty BlackBerry, Bada, Symbian, webOS sekä Tizen.

3.4 Mobiiliselaimet

Vaihtoehtona natiiviksi sovellukseksi paketoimiselle on tarjoilla sovellus puhtaana web-sovelluksena selaimen välityksellä. Selaimet ovat nousseet yhdeksi tärkeimmistä sovellusten jakelukanavista puhtaasti sen ansiosta, että käytännössä jokainen

verkkoon kytketty tietokone sisältää vähintään yhden selainohjelman, ja mobiililaitteet eivät ole poikkeus.

Koska tavoitteena oli mahdollisimman helppokäyttöinen sovellus, on jokaisen relevantin alustan oma mobiiliselain oltava tuettuna, jotta käyttäjän ei tarvitse nähdä ylimääräistä vaivaa sovellusta käyttääkseen. Erot näiden alustojen vakioselainten välillä eivät ole suuria, sillä mitään Internet Explorer 6:en veroista myllynkiveä ei ole tuettavien listalla, mutta huomioon on otettava mahdolliset selainten eri versioiden erityispiirteet.

Android Browser

Android-mobiilialustan vakioselain on suoraan sidoksissa käytössä olevaan Androidin versioon. Koska Google tarjoaa hyvät tilastot alustansa käytöstä (Dashboards | Android Developers 2013), voimme helposti rajata tuen alarajan Androidin versioon 2.3, jolloin noin 94,3 % kaikista käyttäjistä kuuluu tuen piiriin.

Erot standardien tukemisessa ovat suuret Androidin version 2.3 ja uusimman välillä (Deveria 2013). Esimerkiksi vektorimuotoisten kuvien tuki puuttuu kokonaan, ja useita muitakin kilpailijoiden tukemia ominaisuuksia ei version 2.3 selain tue lainkaan. HTML5-standardien tukemisessa voidaan nähdä vastaava trendi, vaikka siellä version 2.3 selain silti ohittaakin Windows Phone 7.5 -version natiiviselaimen (Sights 2013). Koska versio 2.3 kuitenkin on vielä valtavan käytetty, on tuki tälle säilytettävä, vaikka se tuottaisikin ylimääräistä työtä. Heti Androidin versiosta 3 ylöspäin ominaisuustuki onneksi laajenee vastaamaan – ja joiltain osin ylittämäänkin – kilpailijoiden tarjontaa.

Android on vertailtavista alustoista ongelmallisin, koska se on hyvin vahvasti sirpaloitunut eri järjestelmäversioiden ja valmistajien omien muokkausten takia, itse fyysisten laitteiden määrästä puhumattakaan. Suorituskyvystä ei voida puhua yleisellä tasolla, kuten vaikkapa tällä hetkellä huomattavasti homogeenisemmista iPhone- ja Windows Phone -tuoteperheistä, ja laajamittainen Android-laitteiden suorituskykyvertailu olisi liian laaja alakokonaisuus tähän opinnäytetyöhön. Jonkinlaisen vertailukohteen muihin nähden antaa kuitenkin se, että Android-laitteet ovat sijoittuneet keskimäärin aina vastaavan iPhonen-mallin ja Windows Phone -mallien väliin (Klug 2012a; Klug 2012b; Klug 2012c) – Windows Phone 8 tosin rikkoo tätä kaavaa hieman uuden Internet Explorer Mobile 10 -selaimensa avulla.

Safari

Applen Safari-työpöytäselaimeen pohjautuva iOS:n Safari on sekä iPhonen että iPadin vakioselain. Selainta on päivitetty itse käyttöjärjestelmän päivitysten yhteydessä ja tällä hetkellä menossa on kuudes versio. Koska vanhin malli iPhonesta, jolle Apple tarjoaa vielä iOS6-päivityksen, on iPhone 3GS – ja tämän valmistus lopetettiin vasta uusimman mallin tultua markkinoille vuoden 2012 syyskuussa – voitaisiin periaatteessa alimmaksi tuetuksi versioksi asettaa uusin versio. On kuitenkin huomioitava, että alkuperäinen iPad ei ole tuettujen listalla, joten varmuuden vuoksi asetetaan alarajaksi iOS:n versio 5.

Tuetuilta ominaisuuksiltaan Safari on edellä kilpailijoitaan (Deveria 2013). Käytännössä kaikki oleelliset CSS3-ominaisuudet ovat tuettuja ja niitä ominaisuuksia, joille tukea ei ole, eivät myöskään kilpailijat tue yhteneväisesti. HTML5-ominaisuuksien tuessa Safarin sekä viides että kuudes versio ovat kilpailijoitaan edellä niin älypuhelimissa kuin tablet-laitteissakin (Sights 2013).

Suorituskyvyltään Safari on ollut pitkään mobiilimaailman kärkikastia (Lal Shimpi 2012), ja yhdistettynä hyvään standarditukeen tämä tekee siitä erittäin hyvän selaimen, jolla käyttää web-sovelluksia.

Internet Explorer Mobile

Mobiiliselaimista selvästi pienimmän markkinaosuuden omaa Internet Explorer Mobile. Vaikka Windows Phone ei alustana olekaan montaa vuotta ehtinyt olla markkinoilla, on Internet Explorer Mobilea ehditty kuitenkin päivittää täyden version verran. Sovelluksen kannalta alin tuettu versio voidaan asettaa Windows Phonen päivityksen 7.5 yhteydessä päivitettyyn versioon, joka pohjautuu Internet Explorer 9:n esitysmoottoriin (Microsoft Shows New Features and Future Direction as Momentum Builds for Windows Phone 7 2011). Tämä on käytännössä alin versio, joka Suomessa myydyissä puhelimissa on ollut käytössä; muun muassa kaikki Nokian julkaisemat Lumia-puhelimet ovat olleet varustettuja Windows Phonen versiolla 7.5. Vaikka Windows Phonen versiossa 8 Microsoft päivittääkin Internet Explorer Mobilen pohjautumaan Internet Explorer 10:een (Morris 2012), mikä tuo vielä paremman yhteensopivuuden standardien kanssa, niin tuki vanhemmalle versiolle on säilytettävä niiltä osin, kuin se ei tuota liikaa ylimääräistä työtä.

Ominaisuuksiltaan Internet Explorer Mobile ei jää niin pahasti jälkeen kilpailijoistaan, kuin Microsoftilta on totuttu odottamaan (Deveria 2013). Muutamia mahdollisesti tarvittavia ominaisuuksia – joita kaikki kilpailijat tukevat – ei tueta, kuten mm. syöttökenttien vihjetekstejä, CSS3-siirtymiä ja -animaatioita sekä CSS-väriliukuja. Tuki monille näistä on tuotu Internet Explorer Mobilen versiossa 10, ja kyseinen versio on monilta osin jopa edistyneempi kuin kilpailijansa. Myös HTML5-ominaisuuksissa versio 9 laahaa kilpailijoiden perässä, mutta versio 10 kilpailee jo samassa sarjassa (Sights 2013). Internet Explorer Mobile 9:n puutteet eivät onneksi ole ylitsepääsemätön este, mutta ne on tiedostettava sovellusta tehtäessä.

Suorituskyvyltään Internet Explorer Mobile 9 on keskinkertainen ja häviää selvästi kilpailijoilleen (Klug 2012a; Klug 2012b), mutta versio 10 on selvästi parempi ja yltää kilpailijoidensa tasolle ja jopa kärkisijoille osassa testejä (Klug 2012c). Kokonaisuutena Internet Explorer Mobile tulisi luultavasti olemaan täysin käyttökelpoinen sovelluksen käyttämiseen.

4 TEKNIIKAT JA TYÖKALUT

Toimeksiantajan haluaman mobiilisovelluksen toteutus sekä web-sovelluksena että natiiviksi sovellukseksi paketoituna vaatii joitakin yhteisiä työkaluja. Web-sovelluksena toteutettaessa sovelluksen on skaalauduttava käyttäjän päätelaitteen mukaan, jotta erillistä mobiilisivustoa ei tarvitse ylläpitää, ja tämä dynaaminen mukautuvuus voidaan saavuttaa responsiivisella web-kehityksellä. Huolimatta siitä, toteutetaanko sovellus webissä vai natiivina, on sovelluksella oltava tehokas, selkeä ja helposti ylläpidettävä rakenne. Sovelluksen pohjaksi on tarjolla useita moderneja JavaScript-ohjelmistokehyksiä, joita esitellään myöhemmin tässä luvussa. Sovelluksen taustajärjestelmään ei opinnäytetyössä oteta kantaa syvällisesti ja sitä käsitellään myöhemmin, mutta vain pintapuolisesti ja siten, kuin se on oleellista mobiilisovelluksen kannalta.

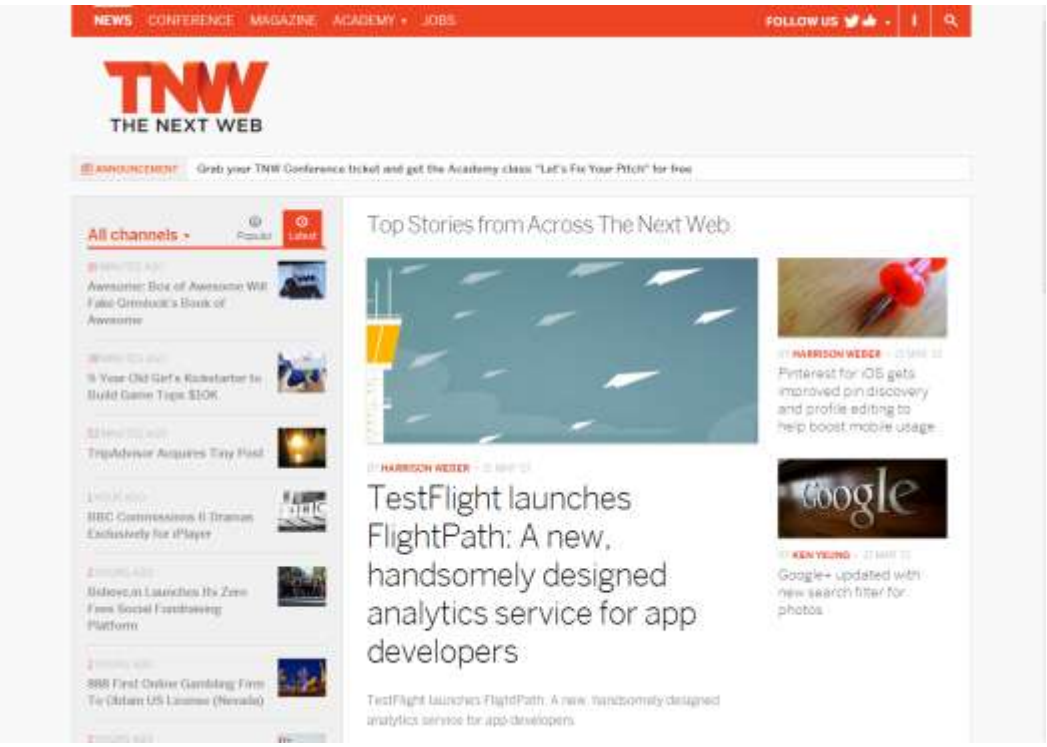
4.1 Responsiivinen web-kehitys

Responsiivinen web-kehitys on termi, joka kuvaa uudenlaista lähestymistapaa verkkosivujen suunnitteluun. Se viittaa tapaan mukautua käyttäjän päätelaitteen ominaisuuksiin ja tarjoilla näiden pohjalta mukautettu käyttökokemus (Marcotte 2010).

Erilliset mobiilisivut olivat mobiilikehityksen ensimmäistä aaltoa. Mobiililaitteita varten luotiin täysin omat ulkoasut, jotka oli suunniteltu varta vasten pieniä ruutuja ja erilaisia navigointitapoja varten. Pitkään tämä olikin toimiva strategia, sillä mobiililaitteet olivat käytettävyydeltään niin kaukana työpöytäselaimista. CSS2 toi mukanaan mediatyypin mukaan vaihtuvat tyylitiedostot, mikä oli askel oikeaan suuntaan, mutta ongelmaksi nousi pian se, että mobiiliselaimet eivät enää luokitelleet itseään CSS2:n “handheld”-mediakategoriaan (Hazaël-Massieux 2009). Tämän lisäksi erillisten tyylitiedostojen ylläpitäminen tuotti lisätyötä.

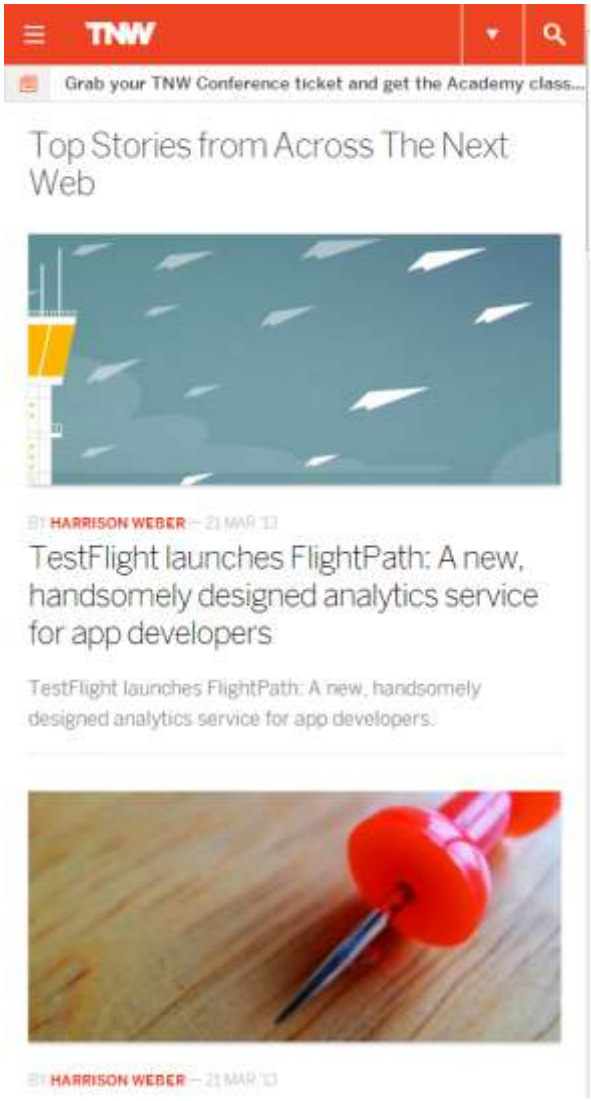
CSS3 lisäsi standardiin staattisten, eri mediatyypeille erikseen tehtyjen tyylitiedostojen rinnalle mahdollisuuden tehdä mediakyselyitä. Näiden avulla tuli mahdolliseksi mukauttaa sivustojen tyylitiedostoja päätelaitteen ominaisuuksien, kuten ruudun koon, mukaan (Media Queries 2012).Suurin osa selaimista lisäsi tuen tälle ominaisuudelle vuosien 2008 ja 2010 välillä (Deveria 2013), mutta Microsoftin Internet Explorer lisäsi tuen ominaisuudelle vasta versiossa 9 vuoden 2011 maaliskuussa. Kirjoitushetkellä ominaisuus on tuettuna kaikissa työpöytä- sekä mobiiliselaimissa.

Responsiivinen web-kehitys itsessään on Ethan Marcotten vuonna 2010 lanseeraama termi (Marcotte 2010), jolla viitataan tapaan mukauttaa verkkosivustojen tyylittelyä dynaamisesti päätelaitteen ominaisuuksien mukaan juurikin CSS3:en mediakyselyitä hyväksi käyttäen. Tyylittelyn muutos tehdään aina, kun päätelaitteen ominaisuudet muuttuvat, kuten jos käyttäjä muuttaa työpöytäselaimensa ikkunan kokoa tai kääntää kännykkänsä poikittain. Näin voidaan mukautua erilaisiin käyttötapauksiin ilman, että ylläpidettäväksi tulee useita versioita sivujen asettelusta tai tyylitiedostoista.



KUVIO 1. The Next Web -sivusto työpöytäresoluutiolla

KUVIO 2. The Next Web -sivusto tablettilaiteresoluutiolla



KUVIO 3. The Next Web -sivusto mobiililaiteresoluutiolla

Vuotta 2013 on tituleerattu responsiivisen web-kehityksen vuodeksi (Cashmore 2012). Tablet-tietokoneet ovat nopeasti yleistymässä Euroopassa (Husson & Reitsma 2013), ja älypuhelimenkin omistaa jo 55% Euroopan suurten markkinoiden asukkaista (EU5 Smartphone Penetration Reaches 55 Percent in October 2012 2012), eikä kasvu näytä hidastuvan (Smartphones Reach Majority in all EU5 Countries 2013). On siis selkeä tarve tarjota mobiilipäätelaitteita käyttäville käyttäjille käyttökokemus, joka tarjoaa sivun täyden sisällön, mutta on käytettävyydeltään optimoitu erikokoisille

kosketusnäytöille. Responsiivisen web-kehityksen noususta kertoo myös se, että kaksi suosittua responsiivista frontend -ohjelmistokehystä löytyy GitHubin suosituimpien listalta, Twitter Bootstrap peräti ensimmäiseltä sijalta (Popular Starred Repositories 2013).

Vertailuun on otettu kaksi suosittua ohjelmistokehystä, Twitter Bootstrap ja ZURB Foundation. Nämä valittiin sen perusteella, että molempien kehitys on aktiivista ja ne tarjoavat molemmat laajan määrän ominaisuuksia sekä tarvittaessa runsaasti valmiita laajennoksia. Molempia näistä myös tuetaan laajalti, mikä on jatkokehityksen kannalta tärkeää.

Twitter Bootstrap

Twitter Bootstrap on vuonna 2011 julkaistu responsiivinen frontend-ohjelmistokehys. Se kehitettiin alunperin Twitterin omaan sisäiseen käyttöön, mutta julkaistiin vuoden 2011 elokuussa yleiseen käyttöön Apachen avoimen lähdekoodin lisenssin alaisena.

Alunperin Bootstrap ei sisältänyt responsiivisia ominaisuuksia, mutta niitä lisättiin vain puoli vuotta alkuperäisen julkaisun jälkeen versiossa 2.0 (Otto 2012). Kolmas versio Bootstrapista on juuri valmistumassa ja siinä ohjelmistokehys on muutettu olemaan ensisijaisesti mobiililaitteille suunnattu, kun frontend -ohjelmistokehykset yleensä ovat keskittyneet tukemaan ensisijaisesti työpöytäselaimia. Siinä, missä aiemmin on ensin tehty työpöytäselaimille versio ja riisuttu siitä ominaisuuksia, tehdään nyt ensin versio, joka toimii mobiililaitteilla ja lisätään siihen ominaisuuksia työpöytäversiota varten.

Bootstrap tarjoaa kehittäjille yksinkertaisen ruudukkopohjaisen responsiivisen sivurakenteen, valmiit geneeriset tyylittelyt ja lisensoidun ikonipaketin sekä toistakymmentä valmista käyttöliittymäkomponenttia. Nimensä mukaisesti se on

tehty helpottamaan projektin nopeaa käyttöliittymäprototyyppäämista, tosin kääntöpuolena Bootstrapin tyylittelyiden ylikirjoittaminen täysin omanlaisiksi on työläämpää. Itse tyylien kustomointiin Twitter on valinnut LESS-kielen, joka voidaan kääntää puhtaaksi CSS:ksi.

Bootstrapin selaintuki on kattava ja nykyinen versio tukee jopa Internet Explorer 7:ää, mutta tuki tälle ollaan jättämässä pois tulossa olevassa kolmosversiossa.

ZURB Foundation

Toinen iso pelaaja responsiivisten frontend-ohjelmistokehysten areenalla on ZURB Foundation, jonka ZURB kehitti alunperin omaksi sisäiseksi työkalupakikseen asiakasprojekteja varten. Vuoden 2011 lokakuussa ZURB julkaisi Foundationin MIT-lisenssin alaisuudessa yleiseen käyttöön versionumerolla 2.0. Tämä versio sisälsi suoraan responsiivisia ominaisuuksia ja vuoden 2013 helmikuussa julkaistu versio 4.0 on ensisijaisesti optimoitu mobiiliselaimille, eli se on ns. ”mobile first”.

Foundation tarjoaa Bootstrapin tavoin kehittäjille responsiivisen ruudukkopohjaisen sivurakenteen ja kevyesti tyyliteltyjä käyttöliittymäkomponentteja, jotka mahdollistavat nopean prototyyppäyksen. Foundationin kustomointi tehdään Sass-kielellä käyttäen Compass-työkalun tuomia lisäominaisuuksia ja näiden käyttäminen vaatii kehittäjältä Rubyn asennuksen.

Foundation on luopunut Internet Explorer 7:n tukemisesta ja Internet Explorer 8:kaan ei ole täysin tuettu ilman JavaScript-apukirjastojen käyttöä, mutta muuten kaikkia yleisiä selaimia tuetaan.

4.2 Web-sovellusten JavaScript-ohjelmistokehykset

Aiemmin täysin palvelimen päässä tapahtuneita operaatioita on vähä vähältä pystytty siirtämään asiakaspuolelle JavaScript-moottoreiden kehityksen myötä. Ärsyttävistä ponnahdusikkunoista ja lomakkeiden dynaamisesta validoinnista on tultu pisteeseen, jossa kokonainen sovellus voi pyöriä asiakaspuolella. Useita ohjelmistokehyksiä on kehitetty helpottamaan näiden uudentyyppisten sovellusten tekoa, ja näistä vertaillaan kahta tällä hetkellä aktiivisesti kehitettävää ja suosittua vaihtoehtoa.

Valittujen vaihtoehtojen lisäksi markkinoilla on useita muitakin vaihtoehtoja, mutta mobiilisovellusta varten haluttiin ohjelmistokehys, joka tarjoaisi kaiken tarvittavan toiminnallisuuden yhdessä paketissa. Esimerkiksi Backbone on laajalti käytetty vaihtoehto, mutta se vaatii ympärilleen useita kirjastoja, ennen kuin se vastaa ominaisuuksiltaan vertailuun valittuja ohjelmistokehyksiä. Yksi merkittävä vertailusta pois jätetty on Meteor, koska sitä ei olisi mahdollista käyttää, jos sovellus paketoitaisiin natiivisovellukseksi.

Ember.js

Alun perin Sproutcoresta omaksi ohjelmistokehyksekseen haarautunut Ember.js on Yehuda Katzin perustaman Tilde Inc:n hallinnoima avoimen lähdekoodin JavaScript-ohjelmistokehys.

Ember.js on MVC-mallin mukainen ohjelmistokehys, joka painottaa konventioita enemmän kuin konfiguraatiota. Toisin kuin esimerkiksi AngularJS, Ember.js on tarkoitettu kokonaisratkaisuksi, joka hallinnoi koko sivua eikä vain pientä osaa siitä.

Kehittäjätiimin vetäjä, Yehuda Katz, on tunnettu Ruby on Rails- ja jQuery -kehittäjä ja tämän vaikutus näkyy myös Ember.js:ssä. Ember.js vaatii toimiakseen jQueryn, ja itse ohjelmistokehyksen suunnittelussa on nähtävissä Ruby on Railsin piirteitä. Ember.js tarjoaa myös suoraan koodisapluunan Ruby on Rails -integraatioon. Ember.js tukee kehittäjien mukaan käytännössä kaikkia selaimia, mutta vanhempien selainten suorituskyvystä ei anneta takeita (Wagenet 2012).

AngularJS

AngularJS on Googlen ylläpitämä avoimen lähdekoodin JavaScript-ohjelmistokehys. Projekti alkoi JSON-säilytysjärjestelmää varten kehitettynä ohjelmistokehyksenä (Get Angular 2009), mutta kehittäjät hylkäsivät alkuperäisen liiketoimintamallinsa ja julkaisivat kehittämänsä Angular-ohjelmistokehyksen avoimena lähdekoodina vuoden 2010 lokakuussa (angular/angular.js at v0.9.0 2010).

Ohjelmistokehyksenä AngularJS on sekoitus MVVM- ja MVC-malleja. Kehyksessä on eriytetty tietomallit, bisneslogiikka ja esityskerros, mutta kehittäjille on jätetty paljon liikkumavaraa näiden toteutuksen suhteen; mallia on kuvattu nimellä Model-View-Whatever (Sanderson 2012). AngularJS ei myöskään vaadi yksinoikeutta sovelluksen toteutuksessa, vaan sitä voidaan käyttää niin yksittäisten komponenttien kuin koko sovelluksenkin pohjana.

Selaimista AngularJS tukee kaikkia moderneja, ja raja on vedetty Internet Explorer 8:aan. Myös Androidin ja iOS:än mobiiliselaimet ovat tuettuina, ja vaikka sitä ei erikseen mainitakaan, pitäisi myös Internet Explorer Mobilen olla tuettujen listalla, koska se pohjautuu tarpeeksi uuteen Internet Exploreriin.

5 TOTEUTUS JA TYÖPROSESSI

5.1 Taustajärjestelmä

Sovelluksen taustajärjestelmään ei tässä opinnäytetyössä paneuduta syvällisemmin, mutta jonkinlainen yleiskuvaus siitä on tarpeen vähintäänkin kontekstin takia. Korkean tason kaavio taustajärjestelmän arkkitehtuurista löytyy liitteestä kaksi.

Vaatimusmäärittely

Taustajärjestelmä tulee toimimaan mobiilisovelluksen taustalla, joten siltä vaaditut ominaisuudet ovat suoraan sidottuja mobiilisovelluksen ominaisuuksiin. Tärkeimpänä toiminnallisuutena on kyselyiden ja kysymysten hallinta: näiden luominen, kyselyiden tarjoaminen käyttäjille ja vastausten näyttäminen toivotun laisessa muodossa. Näihin olennaisesti liittyviä vaatimuksia ovat myös käyttäjien hallinta sekä uusista kyselyistä ilmoittaminen.

Vaatimukset kerättiin haastattelemalla toimeksiantajaa ja ennakoimalla niitä asioita, joita toimeksiantaja ei välttämättä ollut tullut ajatelleeksi. Osa vaatimuksista jätettiin suoraan prototyyppivaiheesta pois ja kirjattiin jatkokehitykseen kuuluviksi. Taustajärjestelmän alustava toiminnallisten vaatimusten määrittely löytyy liitteestä yksi.

Käytettävät tekniikat

Taustajärjestelmä rakennettiin Play-ohjelmistokehyksen päälle. Taustajärjestelmän oli tarkoitus olla kevyt ja vain sivuosassa tätä projektia, joten Play valittiin sen poh-

jaksi, koska se mahdollistaa erittäin nopean iteroinnin ja sen kanssa voidaan käyttää Javaa, joka oli allekirjoittaneen vahvin osaamisalue.

Ennen Play:hin päätymistä tehtiin myös pienimuotoiset prototyypit Node.js:llä, jota Silmu Software käyttää, sekä Grailsilla, joka on toinen nopean iteroinnin ohjelmistokehys. Node.js oli lupaava, ja ihan syystäkin se on yksi kuumimmista nimistä alalla tällä hetkellä, mutta sen ekosysteemi on vielä nuori, ja mm. ORM-kirjastot olivat vielä huomattavan epävakaita ja ominaisuuksiltaan köyhiä verrattuna muiden alustojen vastaaviin. Grails sen sijaan on jo vakaa tuote, mutta lopulta sen käyttämä Groovy-ohjelmointikieli ei ollut opettelemisen vaivan arvoinen; varsinkin, kun vaihtoehtona oli vähintään samalla viivalla puhtaalla Javalla käytettävä vaihtoehto.

Itse taustajärjestelmä toteutettiin, kuten aiemmin todettiin, Play-ohjelmistokehyksen päälle käyttäen ohjelmointikielenä Javaa. Play tarjoaisi myös mahdollisuuden käyttää kielenä Scalaa, ja Play:n uudet versiot on tehty Javan sijaan Scalalla, mutta uuden ohjelmointikielen opettelu nähtiin tässä tapauksessa turhaksi. Play:n itse käyttämien kirjastojen lisäksi ei taustajärjestelmän teossa käytetty muita ulkoisia kirjastoja kuin Deadbolt2-kirjastoa käyttäjien oikeuksien hallintaan, bcrypt-kirjastoa tietokannan kriittisten tietojen kryptaamiseen sekä kirjastoa MySQL-tietokannan kanssa kommunikointiin.

Taustajärjestelmän tietojen varastointiin valittiin tietokannaksi MySQL. Vaihtoehtoina tälle puntaroitiin muun muassa MongoDB:tä ja PostgreSQL:ää, mutta molemmat näistä karsittiin pois ennen prototyypin teon aloittamista. Taustajärjestelmä ei olisi hyötynyt merkittävästi MongoDB:n vapaamuotoisemmasta tiedonvarastoinnista, koska kysymykset ja vastaukset, jotka ovat taustajärjestelmän tärkeimpiä osia, olisivat kuitenkin olleet aina yhdenmuotoisia. Tämän lisäksi MongoDB ja muut NoSQL-tietokannat eivät olleet tarpeeksi tuttuja, että niitä olisi ollut mahdollista käyttää tehokkaasti ilman pidempää perehtymisaikaa ennen toteutuksen aloittamista.

PostgreSQL olisi ollut MySQL:än tapaan perinteinen SQL-tietokantaratkaisu ja MySQL:ää parempi ominaisuuksiltaan, mutta se ei olisi prototyyppivaiheessa tuonut mitään merkittävää etua. Päinvastoin, PostgreSQL:än ja MySQL:än eroavaisuuksien opetteluun olisi mennyt turhaan aikaa, joka voitiin käyttää itse prototyypin tekoon. Tulevaisuudessa pohjalla olevan tietokannan vaihto PostgreSQL:ään ei ole suuri työ, jos se halutaan tehdä esimerkiksi suorituskyvyn tai skaalautuvuuden parantamiseksi. Taustajärjestelmän arkkitehtuurikuvaus löytyy liitteestä kaksi.

Rajapinnat

Taustajärjestelmän tärkein ominaisuus on tiedonvaihto mobiilisovelluksen kanssa, joka toteutettiin REST-rajapinnan avulla. Taustajärjestelmä tarjoilee mobiilisovellukselle rajapinnan kautta tiedot käyttäjälle kuuluvista avoimista kyselyistä ja antaa käyttäjän lähettää vastauksia kyselyihin taustajärjestelmälle päin. Koska rajapintaa tullaan käyttämään vain mobiiliohjelmiston kanssa, sen käyttö suljettiin tunnistautumisen taakse.

Tiedonvaihto mobiilisovelluksen ja taustajärjestelmän välillä salattiin SSL:llä ja kirjautumistiedot mobiilisovelluksesta taustajärjestelmään myös obfuskoitiin. Vaikka obfuskaatio ei olekaan millään tavalla tietoturvallinen ratkaisu, on se yhdistettynä SSL:ään kuitenkin tarpeeksi hyvä suojataso prototyypille; samasta bcrypt-kryptauskirjastosta, kuin mitä käytettiin taustajärjestelmässä, on myös JavaScript-implementaatio, jota voitaisiin käyttää, mutta sen vaikutusta suorituskykyyn ei ehditty tutkia tarpeeksi eikä se ollut kriittinen prototyypin kannalta.

Itse rajapinta oli prototyypissä sen verran yksinkertainen – se sisälsi vain kirjautumisen sekä käyttäjän kaikkien tietojen haun yhdellä kutsulla – että sen kuvaaminen tarkemmin ei toisi mitään lisäarvoa työhön.

5.2 Mobiiliohjelmisto

Mobiiliohjelmiston perusrakenne toteutettiin web-tekniikoilla, ja siitä paketoitiin myös natiivisovellus PhoneGapin avulla. Nykyaikaiset JavaScript-ohjelmistokehykset ja CSS-ulkoasukirjastot tarjoavat työkalut lähes natiivien mobiilisovellusten kaltaisten monialustasovellusten tekoon, jotka mukautuvat erilaisten päätelaitteiden mukaan.

Vaatimusmäärittely

Mobiiliohjelmiston tärkein ominaisuus on kyselyiden vastaanottaminen ja niihin vastaaminen, mikä vaatii luonnollisesti käyttäjän tunnistamisen ja täten myös tavan tunnistautua. Lopulta mobiilisovelluksen toiminnallisia vaatimuksia kertyi vaatimaton määrä, mutta ne kattoivat kaikki prototyypiltä halutut toiminnallisuudet.

Kuten taustajärjestelmänkin kanssa, vaatimukset kerättiin toimeksiantajalta haastattelemalla ja koottiin listaksi. Taustajärjestelmässä toteutustapa ei ollut tärkeä, mutta mobiilisovellukselle oli asetettu ei-toiminnalliseksi vaatimukseksi myös se, että sen tuli toimia kaikilla tulevien käyttäjien käyttämillä mobiilialustoilla. Mobiilisovelluksen tarkempi vaatimusmäärittely löytyy liitteestä kolme.

Käytettävät tekniikat

Mobiilisovelluksen arkkitehtuuri jakautuu kahtia taustalla olevaan PhoneGappiin, joka mahdollistaa sovelluksen paketoimisen natiiviksi mobiilisovellukseksi, sekä itse sovellukseen, joka rakennettiin AngularJS: n päälle.

Kaikista vaihtoehdoista PhoneGap valittiin sovelluksen pohjalle, koska se oli lopulta ainoa, joka oli sekä toiminnallisuudeltaan tarpeeksi edistynyt, edelleen aktiivisesti kehityksessä ja tuki kaikkia haluttuja alustoja. PhoneGapin pohjalla olevan Cordova-kirjaston – josta PhoneGap on siis vain yksittäinen Adoben tekemä jakeluversio – siirtyminen Apachen hallinnoimaksi projektiksi antoi sille uskottavuutta ja ainakin teoreettisen lupauksen tuen jatkumisesta.

JavaScript-ohjelmistokehyksistä valittiin projektiin AngularJS. Vaihtoehdot rajautuivat lopulta kahteen laajimpaan ohjelmistokehykseen, AngularJS:ään sekä Emberiin, joista molemmat tarjosivat sovelluksen vaatimat toiminnallisuudet yhdessä paketissa. AngularJS valittiin lopulta lähinnä sen takia, että Ember oli valintavaiheessa vielä beta-asteella ja sen toiminnallisuuksia ei ollut vielä lukittu ja ne olisivat saattaneet muuttua projektin kuluessa; Ember 1.0 julkaistiin vasta elokuun lopulla, mikä olisi ollut aivan liian myöhään (Ember 1.0 released 2013).

AngularJS:n päälle tarvittiin vielä ulkoasua varten CSS-kehys, joka tarjosi kaiken prototyyppiä varten tarvittavan yhdessä paketissa, jotta aikaa ei menisi turhaan visuaalisen puolen hiomiseen. Valinta tehtiin kahden käytetyimmän kehyksen, Twitter Bootstrapin ja ZURB Foundationin, välillä. ZURB tarjosi sen hetkisessä versiossaan jo mobile first -ominaisuuksia, ja jatkokehityksestä vastaava Silmu Software käytti Foundationia projekteissaan laajalti, mutta tätä valintaa lykättiin osittain, koska kyseessä ei ollut toiminnan kannalta kriittinen asia ja osittain, koska Bootstrapin uuden version oli tarkoitus olla kesällä valmis. Kun uudesta Bootstrapin versiosta tuli ensimmäinen julkinen beta-versio, todettiin tämän tarjoavan Foundationia vastaavat ominaisuudet ja samankaltaisen helposti muokattavan litteän tyylittelyn, joten projektiin valittiin käyttöön Bootstrap. Bootstrap on näistä kahdesta laajemmin käytetty ja sille on saataville enemmän ohjeistusta ja tukea, mikä myös teki siitä paremman valinnan.

Itse sovellus tehtiin alun perin normaalina web-sovelluksena, ja sitä pyöritettiin kevyen web-palvelimen päällä. Toimeksiantaja kuitenkin teki pian selväksi, että sovellus haluttiin ehdottomasti sovellusmarketteihin, joten sovellus siirrettiin PhoneGapin päälle. Käytännössä ainoat erot siirron jälkeen olivat osoite, jossa sovellus pyöri, ja testattavien selainten määrän väheneminen yhteen. Mobiilisovelluksen tarkempi arkkitehtuurikuvaus löytyy liitteestä neljä.

Tietoturva

Mobiilisovelluksen tietoturvassa keskityttiin lähinnä mobiilisovelluksen ja taustajärjestelmän välillä kulkevan datan pitämiseen salassa, koska itse sovelluksen koodin obfuskointi ja suojaaminen tehdään ennen lopullista julkaisua ja se on hyvin alustariippuvainen operaatio.

Kuten taustajärjestelmän rajapinta-alaluvussa todettiin, oli tiedonkulun suojauksen pohjana obfuskointi ja SSL-salaus, ja siihen ei ole tarpeen palata enää tässä. Kun käyttäjän tiedot on haettu rajapinnasta sovellukseen, ne tallennetaan lokaalisti PhoneGapin käyttämän selaimen localStorageen. Tiedot tallennetaan selkokielisinä, mikä oli tietoinen valinta prototyyppivaiheessa, kun oli tärkeää voida tarkistaa nopeasti rajapinnasta tulleen tiedon oikeellisuus, mutta tuotantoon mennessä nuo tiedot on vähintään obfuskoitava sekä salattava jotenkin ja mieluummin vielä kryptattava. Koska tarjolla on bcryptin implementaatio JavaScriptillä, olisi se looginen valinta sovelluksen localStoragen suojaamiseen. Jos sovellus olisi päätetty toteuttaa web-sovelluksena, ei olisi ollut tarvetta kommunikoida palvelimen ulkopuolisten tahojen kanssa, jolloin jo tuo SSL-salaus olisi ollut riittävä.

Testaus

Prototyypissä ei keskitytty erityisesti testeihin, mikä tietysti olisi ollut arvokas lisä, mutta ilman hyvää TDD:n osaamista olisi testaamiseen kulunut huomattavasti aikaa. Yksikkötestejä tehtiin jonkin verran, ja integraatiotestejä tehtiin erikseen cURLin avulla.

AngularJS tarjoaa erittäin hyvän jatkuvaan testaukseen tehdyn kirjaston nimeltä Karma, jonka tiimi on itse kehittänyt. Karma tarvitsee rinnalleen jonkin JavaScript-testikirjaston, joka tässä tapauksessa oli Jasmine; vaihtoehtoja olisivat olleet mm. QUnit ja Mocha. Karmalle kerrotaan, mistä testit löytyvät ja halutaanko ne ajaa automaattisesti tiedostojen muuttuessa, minkä jälkeen Karma päivittää automaattisesti testien tilan erilliselle lokaalille verkkosivulle. Projektin puitteissa tehtiin Karman ja Jasminen avulla vain yksinkertaisia yksikkötestejä, joilla varmistettiin muuttujien tyypityksiä; JavaScript dynaamisena ja heikosti tyypitettynä kielenä vaatii erityisen tarkkaa parametrien tarkistamista, koska niistä johtuvat virheet tulevat vastaan vasta ajon aikana.

Mobiilisovelluksen ja taustajärjestelmän rajapintaa testattiin cURLia apuna käyttäen ja myöhemmin sovelluksesta käsin samalla debuggerista sovelluksen suoritusta seuraten. cURLilla ajettiin mobiilisovelluksen lähettämien HTTP-pyyntöjen kanssa identtisiä pyyntöjä ja varmistettiin rajapinnan toimivuus erilaisilla syötteillä. Kun mobiilisovellus oli tarpeeksi pitkällä, asennettiin siitä aina uusin versio testipuhelimeen, joka oli kiinni tietokoneessa, ja testattiin toiminta käytännössä.

6 POHDINTA

6.1 Johtopäätökset

Sovellusta tehtäessä kävi selväksi, että web-tekniikoilla toteutettu kaverijohtajuus-sovellus toimisi sekä puhtaana web-sovelluksena että paketoituna PhoneGapilla natiivisovellukseksi. Työkalut ja tekniikat ovat kehittyneet nopeasti, mutta vertailuun päätyneet – ja monet vertailusta pois jääneetkin – olivat kaikki kuitenkin jo tarpeeksi vakaita tuotantokäyttöön. Suurimmat haasteet natiivisovelluspaketoinnissa ovat jokaisen mobiilialustan ulkoasulle asettamat uniikit vaatimukset ja käytettävyydelle asetetut odotukset, joista lipsuminen johtaa nopeasti muiden kuin pääasiallisen alustan käyttäjien käyttökokemuksen kärsimiseen.

Toimeksiantajan tahdosta päädyttiin lopulta tekemään prototyypistä natiivisti paketoitu sovellus. Päätöksen takana oli halu saada sovellus sovellusmarketteihin ladattavaksi, mutta mitään suoranaista teknistä syytä tälle päätökselle ei ollut. Sivustot, kuten Forecast.io, hämärtävät jo nyt sovelluksen ja verkkosivun rajaa, ja responsiiviset käyttöliittymät mahdollistavat erilaisille päätelaitteille sopivan käyttökokemuksen luomisen. Web-sovelluksen etuna luonnollisesti olisi se, että ylimääräisiä liikkuvia osia olisi mobiilisovelluksien verran vähemmän, mutta näkyvyys peruskäyttäjien suuntaan mahdollisesti kärsii sovellusmarkettiedustuksen puutteesta johtuen.

Natiivisovelluspaketoinnilla ei tässä tapauksessa kuitenkaan saavuteta mitään konkreettisia etuja, koska kyseessä ei ole julkiseen käyttöön tarkoitettu sovellus eikä siinä käytetä mitään vain natiivisovelluksille tarjolla olevia päätelaitteiden ominaisuuksia. Natiivipaketointi ei onneksi kuitenkaan tuottanut erityisesti lisätöitä, sillä tukea PhoneGapille on ilmestynyt mm. suosittuun Netbeans IDE:hen, ja sovelluksen

paketointi ja asentaminen testilaitteeseen tai emulaattoriin on parhaimmillaan yhden painikkeen takana.

6.2 Tulevaisuus

Monialustaisuus on pelikentän fragmentoituessa entistäkin tärkeämpää. Nykyisten kolmen ison alustan rinnalle pyrkii suomalainen Jolla uudella Sailfish-käyttöjärjestelmällään, Samsung saattaa hyvinkin alkaa tuomaan markkinoille enemmän Tizen-alustaisia puhelimia, ensimmäiset Firefox OS -puhelimet on myyty jo loppuun, ja Ubuntu Phone tähtää myös mullistamaan mobiilialustojen pelikenttää. Paria natiivia versiota on vielä suhteellisen helppoa ylläpitää, mutta seitsemän erilaista alustaa vaatisi jo sellaisen määrän dedikoituja työntekijöitä ja osaamista, että erillisten natiiviversioiden tukeminen tuskin olisi ajallisesti ja taloudellisesti kannattavaa. Tällä hetkellä Applen iOS on monelle pääasiallinen alusta sen maineen ja markkinaosuuden takia, mutta Android ei ole enää se vieroksuttu halpapuhelinten käyttöjärjestelmä, millaisena sitä pitkään tunnuttiin pidettävän, ja Microsoft yrittää ainakin kaikkensa, että Windows Phonesta tulisi kilpailija näille kahdelle.

Kaikkia näitä alustoja kuitenkin yhdistää se, että niiden ympärillä on jo tällä hetkellä valmiiksi osaavia tekijöitä, ja niiden kehitystyökalut sekä ekosysteemit ovat varttuneet. Uudet tulokasalustat tuovat mukanaan uudet työkalut ja uudet käytettävät ohjelmointikieletkin: Jollan Sailfish käyttää C++:aa ja Qt-ohjelmistokehystä, Ubuntu Phone antaa mahdollisuuden käyttää HTML5:tä sekä Qt:n QML-malliinnuskieltä ja Firefox OS ja Tizen pohjaavat HTML5:een. Jos haluttaisiin tukea näitä kaikkia alustoja, olisi käytössä kuusi erilaista ohjelmointikieltä ja -ympäristöä erilaisine oikkuineen ja konventioineen. Olisi ilmiselvästi helpompaa kehittää yhtä sovellusta, joka vaatisi vain erillisen tyylittelyn per alusta.

LÄHTEET

Adobe. 2011. Adobe Announces Agreement to Acquire Nitobi, Creator of PhoneGap. Viitattu 9.3.2013. http://www.adobe.com/aboutadobe/pressroom/pressreleases/201110/AdobeAcquiresNitobi.html .

Apple, Inc. 2013. Apple Reports Record Results. Viitattu 9.3.2013. http://www.apple.com/pr/library/2013/01/23Apple-Reports-Record-Results.html .

appMobi. 2012. How to use the appMobi XDK with Visual Studio or other IDEs to develop cross platform mobile apps. Viitattu 21.4.2013. http://youtu.be/k04U-0p0WoA .

Appspresso. 2011. Twitter / appspresso: Appspresso 1.0 Final Released! Viitattu 14.4.2013. https://twitter.com/appspresso/status/143522046058561537 .

Armano, D. 2012. The Future Isn't About Mobile; It's About Mobility. Viitattu 10.3.2013. http://blogs.hbr.org/cs/2012/07/the_future_isnt_about_mobile_its.html .

Austin, S. 2013. The Surprising Numbers Behind Apps. Viitattu 9.3.2013. http://blogs.wsj.com/digits/2013/03/11/the-surprising-numbers-behind-apps/ .

BRAT Tech, LLC. 2009. Get Angular. Viitattu 7.4.2013. http://web.archive.org/web/20091002201638/http://www.getangular.com/ .

LeRoux, B. 2012. PhoneGap, Cordova, and what’s in a name? Viitattu 17.4.2013. http://phonegap.com/2012/03/19/phonegap-cordova-and-what%E2%80%99s-in-a-name/ .

Byford, S. 2013. Samsung finally folding Bada OS into Tizen. Viitattu 23.4.2013. http://www.theverge.com/2013/2/25/4026848/bada-and-tizen-to-merge .

Cantelon, M., Holowaychuk, TJ., Rajlich, N. 2013. Node.js in Action. Manning, Co.

Cashmore, P. 2012. Why 2013 Is the Year of Responsive Web Design. Viitattu 19.3.2013. http://mashable.com/2012/12/11/responsive-web-design/ .

Cederholm, D. 2010. CSS3 for Web Designers. New York, NY: A Book Apart.

comScore, Inc. 2012. EU5 Smartphone Penetration Reaches 55 Percent in October 2012. Viitattu 19.3.2013. http://www.comscore.com/Insights/Press_Releases/2012/12/EU5_Smartphone_Penetration_Reaches_55_Percent_in_October_2012 .

comScore, Inc. 2013. Smartphones Reach Majority in all EU5 Countries. Viitattu 19.3.2013. http://www.comscoredatamine.com/2013/03/smartphones-reach-majority-in-all-eu5-countries/ .

Crockford, D. 2008. JavaScript: The Good Parts. Sebastopol, CA: O'Reilly Media Inc.
de Icaza, M. 2009. MonoTouch 1.0 goes live. Viitattu 10.4.2013. http://tirania.org/blog/archive/2009/Sep-14.html .

Deveria, A. 2013. Viitattu 18.3.2013. http://caniuse.com/#feat=css-mediaqueries .

Deveria, A. 2013. Can I use... Support tables for HTML5, CSS3 etc. Viitattu 13.4.2013. http://caniuse.com/#compare=ie+9,ie+10,ios_saf+5.0-5.1,ios_saf+6,android+2.3,android+3,android+4,android+4.1,android+4.2 .

Gartner, Inc. 2013. Gartner Says Worldwide Mobile Phone Sales Declined 1.7 Percent in 2012. Viitattu 9.4.2013. http://www.gartner.com/newsroom/id/2335616 .

GitHub. 2010. angular/angular.js at v0.9.0. Viitattu 7.4.2013. https://github.com/angular/angular.js/tree/v0.9.0 .

GitHub. 2013. Popular Starred Repositories. Viitattu 19.3.2013. https://github.com/popular/starred .

Google. 2013. Dashboards | Android Developers. Viitattu 13.4.2013. http://developer.android.com/about/dashboards/index.html .

Google. 2013. Developer Tools | Android Developers. Viitattu 23.4.2013. http://developer.android.com/tools/index.html .

Hazaël-Massieux, D. 2009. Return of the Mobile Stylesheet. Viitattu 17.3.2013. http://alistapart.com/article/return-of-the-mobile-stylesheet .

Heitkötter, H., Hanschke, S. & Majchrzak, Tim A. 2012. Comparing cross-platform development approaches for mobile applications. WEBIST 2012 - 8th International Conference on Web Information Systems and Technologies, 2012.

Husson, T & Reitsma, R. 2013. The European Tablet Landscape. Viitattu 19.3.2013. http://www.forrester.com/The+European+Tablet+Landscape/fulltext/-/E-RES91561?objectid=RES91561 .

IDC. 2012. Android Marks Fourth Anniversary Since Launch with 75.0% Market Share in Third Quarter, According to IDC. Viitattu 23.4.2013. http://www.idc.com/getdoc.jsp?containerId=prUS23771812 .

Information Solutions Group. 2012. 2012 PopCap Games Mobile Gaming Research. Viitattu 9.3.2013. http://www.infosolutionsgroup.com/popcapmobile2012.pdf .

Intel. 2013. App Framework. Viitattu 21.4.2013. http://app-framework-software.intel.com/documentation.php .

Intel. 2013. The XDK turbocharges PhoneGap. Viitattu 21.4.2013. http://developer.html5dev-software.intel.com/?q=node/153 .

Ionescu, D. 2010. Angry Birds Devs Angry At Android Fragmentation. Viitattu 9.3.2013. http://www.pcworld.com/article/211152/angry_birds_devs_angry_at_android_fragmentation.html .

Keith, J. 2010. HTML5 for Web Designers. New York, NY: A Book Apart.

Klug, B. 2012a. Nokia Lumia 800 Review - Nokia's Brave New Foray into WP7. Viitattu 13.4.2013. http://www.anandtech.com/show/5266/nokia-lumia-800-review-nokias-brave-new-foray-into-wp7/ .

Klug, B. 2012b. Nokia Lumia 900 Review - Windows Phone with LTE. Viitattu 13.4.2013. http://www.anandtech.com/show/5724/nokia-lumia-900-review-supersized-with-lte/ .

Klug, B. 2012c. Windows Phone 8 and Windows Phone 8X by HTC Preview. Viitattu 13.4.2013. http://www.anandtech.com/show/6415/windows-phone-8-and-htc-8x-preview/ .

KnockoutJS. 2013. Knockout: Downloads. 2013. Viitattu 8.3.2013. http://knockoutjs.com/downloads/index.html .

Krill, P. 2009. Appcelerator enables iPhone, Android app dev. Viitattu 10.4.2013. http://www.infoworld.com/d/developer-world/appcelerator-enables-iphone-android-app-dev-655 .

Lal Shimpi, A., Klug, B. & Gowri, V. 2012. The iPhone 5 Review. Viitattu 13.4.2013. http://www.anandtech.com/show/6330/the-iphone-5-review/ .

Loosemoore, T. 2013. We're not 'appy. Not 'appy at all. Viitattu 13.3.2013. http://digital.cabinetoffice.gov.uk/2013/03/12/were-not-appy-not-appy-at-all/ .

Lunden, I. 2013. In A Play For More HTML5 Muscle, Intel Buys appMobi’s HTML5 Developer Tools, Leaving appMobi To Focus On The Cloud. Viitattu 18.4.2013. http://techcrunch.com/2013/02/21/in-a-play-for-more-html5-muscle-intel-buys-appmobis-html5-developer-tools-leaving-appmobi-to-focus-on-the-cloud/ .

Marcotte, E. 2010. Responsive Web Design. Viitattu 18.3.2013. http://alistapart.com/article/responsive-web-design .

Marcotte, E. 2011. Responsive Web Design. New York, NY: A Book Apart.

Microsoft. 2011. Microsoft Shows New Features and Future Direction as Momentum Builds for Windows Phone 7. Viitattu 13.4.2013. http://www.microsoft.com/en-us/news/features/2011/feb11/02-14mwc.aspx .

Morris, C. 2012. Internet Explorer 10 brings HTML5 to Windows Phone 8 in a big way. Viitattu 13.4.2013. http://blogs.windows.com/windows_phone/b/wpdev/archive/2012/11/08/internet-explorer-10-brings-html5-to-windows-phone-8-in-a-big-way.aspx .

MoSync. 2010. What's New in MoSync SDK 2.4. Viitattu 16.4.2013. http://www.mosync.com/content/whats-new-mosync-24 .

MoSync. 2012. Feature/Platform Support. Viitattu 17.4.2013. http://www.mosync.com/widepage/feature-platform-support .

MoSync. 2012. What's New in MoSync SDK 3.0. Viitattu 16.4.2013. http://www.mosync.com/documentation/release-notes/whats-new-mosync-30 .

Mozilla. 2013. Introduction to Firefox OS. Viitattu 23.4.2013. https://developer.mozilla.org/en-US/docs/Mozilla/Firefox_OS/Introduction .

Otto, M. 2012. Say hello to Bootstrap 2.0. Viitattu 20.3.2013. https://dev.twitter.com/blog/say-hello-to-bootstrap-2 .

Page, L. 2013. Google Official Blog: Update from the CEO. Viitattu 22.3.2013. http://googleblog.blogspot.co.uk/2013/03/update-from-ceo.html .

Sanderson, S. 2012. Rich JavaScript Applications – the Seven Frameworks (Throne of JS, 2012). Viitattu 27.4.2013. http://blog.stevensanderson.com/2012/08/01/rich-javascript-applications-the-seven-frameworks-throne-of-js-2012/ .

Sights. 2013. The HTML5 test - How well does your browser support HTML5? Viitattu 13.4.2013. http://html5test.com .

Sällylä, A. & Ylikojola, P. 2012. Kokonaisvaltaisen johtajuuden ulottuvuudet: syväjohtaminen, psykologiset pääomat ja dialogi. Miten Y-sukupolven osuuskuntia johdetaan menestykseen? Tampere: Tampereen ammattikorkeakoulu.

Taft, Darryl K. 2009. PhoneGap Simplifies iPhone, Android, BlackBerry Development. Viitattu 10.4.2013. http://www.eweek.com/c/a/Application-Development/PhoneGap-Simplifies-iPhone-Android-BlackBerry-Development-788189/ .

Tilde Inc. 2013. Ember 1.0 released. Viitattu 29.9.2013. http://emberjs.com/blog/2013/08/31/ember-1-0-released.html .

Tizen Project. 2013. About | Tizen. Viitattu 23.4.2013. https://www.tizen.org/about .

Toivanen, H. 2013. ”Kokonaisvaltaisen johtajuuden ulottuvuudet: syväjohtaminen, psykologiset pääomat ja dialogi: osuuskuntien johtamisen mittaustyökalu, sovellus (APS) älypuhelimiin.”.

W3C. 2012. Media Queries. Viitattu 15.3.2013. http://www.w3.org/TR/2012/REC-css3-mediaqueries-20120619/ .

Wagenet, P. 2012. html5 - Ember.js browser support? Viitattu 27.4.2013. http://stackoverflow.com/a/9874257 .

Wauters, R. 2013. Geeksphone's Firefox OS smartphones go on sale tomorrow, but will there be buyers? Viitattu 23.4.2013. http://thenextweb.com/mobile/2013/04/22/geeksphone-firefox-os-keon-peak-launch/ .

Wroblewski, L. 2011. Mobile First. New York, NY: A Book Apart.

LIITTEET

Liite 1. Taustajärjestelmän toiminnalliset vaatimukset

Pakolliset

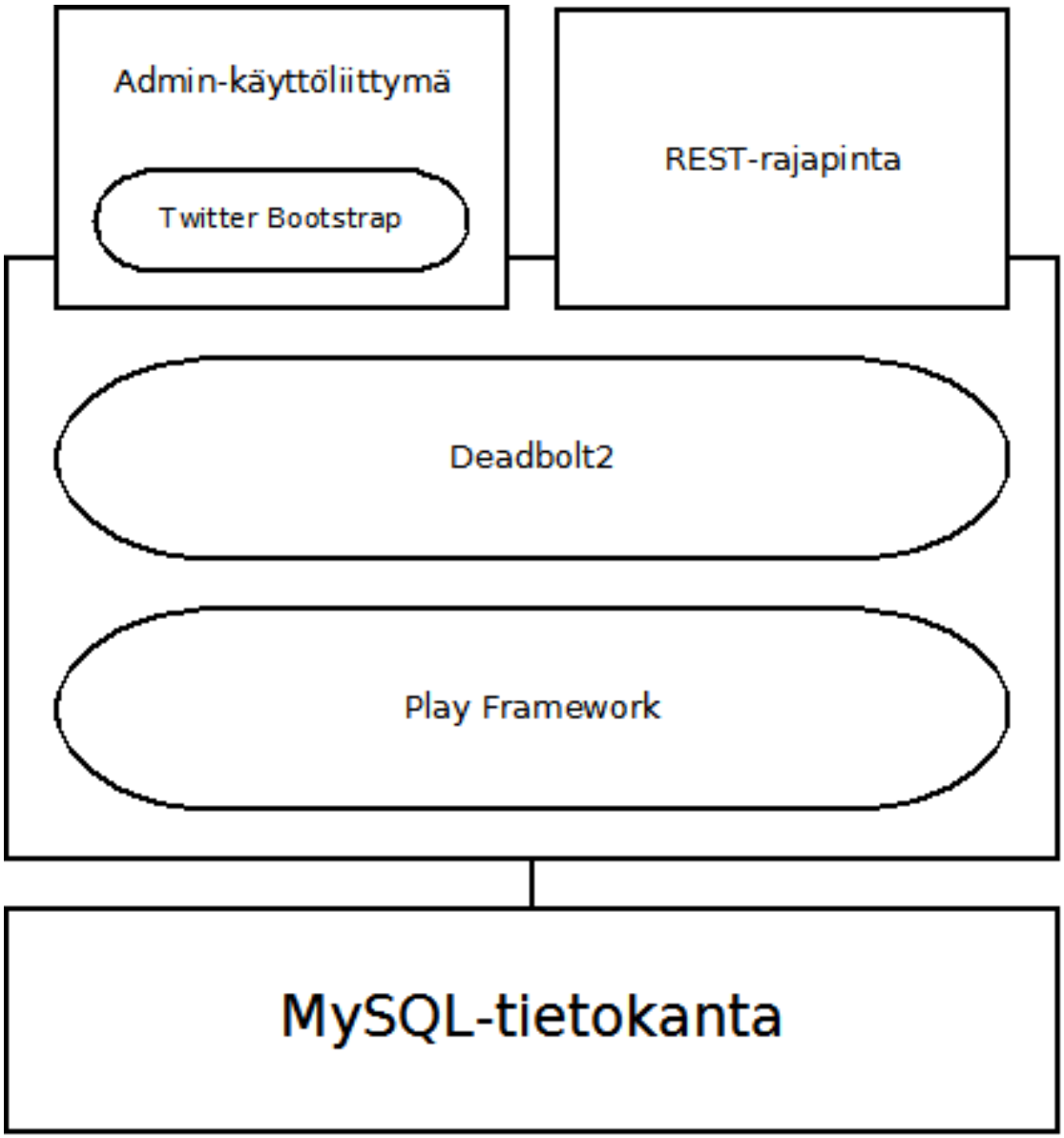
1. Pääkäyttäjän tulee pystyä lisäämään kysymyksiä
2. Pääkäyttäjän tulee pystyä muokkaamaan kysymyksiä
3. Pääkäyttäjän tulee pystyä poistamaan kysymyksiä
4. Pääkäyttäjän tulee pystyä luomaan kysymyksille alikysymyksiä
5. Pääkäyttäjän tulee pystyä muokkaamaan kysymysten alikysymyksiä
6. Pääkäyttäjän tulee pystyä poistamaan kysymysten alikysymyksiä
7. Pääkäyttäjän tulee pystyä luomaan käyttäjiä
8. Pääkäyttäjän tulee pystyä muokkaamaan käyttäjien tietoja
9. Pääkäyttäjän tulee pystyä poistamaan käyttäjiä
10. Pääkäyttäjän tulee pystyä deaktivoimaan käyttäjiä
11. Pääkäyttäjän tulee pystyä luomaan kyselyitä
12. Pääkäyttäjän tulee pystyä muokkaamaan kyselyitä
13. Pääkäyttäjän tulee pystyä julkaisemaan kyselyitä
14. Pääkäyttäjän tulee pystyä vanhentamaan kyselyitä
15. Pääkäyttäjän tulee pystyä poistamaan kyselyitä
16. Pääkäyttäjän tulee pystyä tarkastelemaan kyselyiden vastauksia
17. Pääkäyttäjän tulee pystyä rajaamaan kysely tietylle ryhmälle
18. Pääkäyttäjän tulee pystyä luomaan ryhmiä
19. Pääkäyttäjän tulee pystyä muokkaamaan ryhmien tietoja
20. Pääkäyttäjän tulee pystyä poistamaan ryhmiä

21. Pääkäyttäjän tulee pystyä lisäämään käyttäjiä ryhmiin
22. Pääkäyttäjän tulee pystyä poistamaan käyttäjiä ryhmistä
23. Ryhmänjohtajan tulee pystyä vastaamaan hänen ryhmälleen esitetyn kyselyn vastausten perusteella valittuihin alikysymyksiin
24. Ryhmänjohtajan tulee pystyä tarkastelemaan oman ryhmänsä kyselyiden tuloksia
25. Käyttäjän tulee pystyä tarkastelemaan oman ryhmänsä kyselyiden tuloksia
26. Käyttäjän tulee pystyä muokkaamaan omia tietojaan
27. Pääkäyttäjän, ryhmänjohtajan sekä käyttäjän tulee pystyä kirjautumaan järjestelmään
28. Pääkäyttäjän, ryhmänjohtajan sekä käyttäjän tulee pystyä kirjautumaan ulos järjestelmästä

Valinnaiset

1. Pääkäyttäjän tulee pystyä viemään järjestelmästä ulos arkistoitava versio kyselyiden tuloksista
2. Ryhmänjohtajan tulee pystyä viemään järjestelmästä ulos arkistoitava versio hänen ryhmänsä kyselyiden tuloksista
3. Käyttäjän tulee pystyä vastaamaan hänen ryhmälleen esitettyihin kyselyihin

Liite 2. Taustajärjestelmän arkkitehtuurikuvaus



Liite 3. Mobiilisovelluksen vaatimusmäärittely

Pakolliset

1. Käyttäjän tulee pystyä kirjautumaan sovellukseen
2. Käyttäjän tulee pystyä kirjautumaan ulos sovelluksesta
3. Käyttäjän tulee pystyä listaamaan avoinna olevat kyselyt
4. Käyttäjän tulee pystyä päivittämään kyselylistaus taustajärjestelmästä
5. Käyttäjän tulee pystyä vastaamaan kyselyihin

Valinnaiset

1. Käyttäjän tulee pystyä selaamaan menneitä kyselyitä
2. Käyttäjän tulee pystyä tarkastelemaan menneiden kyselyiden tuloksia
3. Ryhmänjohtajan tulee pystyä vastaamaan kyselyn tulosten perusteella valittuihin alikysymyksiin

Ei-toiminnalliset

1. Sovelluksen tulee toimia kolmella suurimmalla mobiilialustalla

Liite 4. Mobiilisovelluksen arkkitehtuurikuvaus