Olli Kasari

Käyttöliittymäkehitys kosketuskäyttöisille älypuhelimille

Tietotekniikan pro gradu -tutkielma

21. joulukuuta 2012

Jyväskylän yliopisto

Tietotekniikan laitos

Tekijä: Olli Kasari

Yhteystiedot: ollikasari@hotmail.com

Ohjaajat: Tommi Kärkkäinen ja Tommi Lahtonen

Työn nimi: Käyttöliittymäkehitys kosketuskäyttöisille älypuhelimille

Title in English: User Interface Development for Touchscreen Operated Smart Phones

Työ: Pro gradu -tutkielma

Suuntautumisvaihtoehto: Ohjelmisto- ja tietoliikennetekniikka

Sivumäärä: 92+53

Tiivistelmä: Kosketusohjauksen ja sovelluskauppojen läpimurron seurauksena älypuhelinvalmistajat ovat viime vuosina panostaneet entistä enemmän laitealustojensa käytettävyyteen sekä kolmansille osapuolille tarkoitettuihin kehitystyökaluihin. Samaan aikaan markkinoille on ilmestynyt lukuisia ratkaisuja sovellusten alustariippumattomaan toteutukseen. Tässä työssä tutustutaan sovelluskehittäjille tarjottuihin vaihtoehtoihin käyttöliittymäkehityksen perspektiivistä ja selvitetään ovatko yleistyvät web-pohjaiset alustariippumattomat ratkaisut varteenotettava vaihtoehto natiiveille kehitystyökaluille.

Avainsanat: Käyttöliittymä, käytettävyys, mobiili, älypuhelin, kosketusnäyttö, web-sovellus, alustariippumattomuus, sovelluskehys, käyttöliittymäkirjasto, sovelluskehitys

Abstract: Following the rise of app stores and touch user interface driven smartphones in the recent years, practically all of the major device vendors have started paying more attention to usability and third party development tools. At the same time, cross-platform tools led by applications utilizing web technologies are gaining ground from conventional native applications. This thesis compares the different approaches to UI implementation by means of case study.

Keywords: User interface, usability, mobile, smartphone, touchscreen, web application, cross-platform, application framework, widget toolkit, application development

Termiluettelo

ADT	Android Developer Tool Eclipse liitännäinen joka integroi sen käyttöliittymään kaikki Android-kehityksessä tarvitut työkalut. (Meier 2010, 19–21)
Ajax	Asynchronous JavaScript and XML Joukko web-sovelluskehityksen tekniikoita, joiden avulla web-sovelluksista voi tehdä vuorovaikutteisempia.
API	Application Programming Interface Ohjelmointirajapinta, eli määritelmä, jonka mukaan eri ohjelmat voivat keskustella keskenään.
APK	Application package file JAR-paketointiin perustuva Android-sovellusten paketointimuoto.
C	1970-luvun alussa UNIX-käyttöjärjestelmälle kehitetty ohjelmointikieli.
CLR	Common Language Runtime Microsoftin toteuttama ajoympäristö, joka kääntää ajettavan tavukoodin laitealustakohtaiseen binäärimuotoon.
CSS	Cascading Style Sheets Rakenteellisten dokumenttien ulkoasun kuvailuun käytetty tyyliohjekieli.
C++	C-kieleen pohjautuva ohjelmointikieli joka tukee mm. olio-ohjelmointia.
DOM	Document Object Model Ohjelmointirajapinta, joka mahdollistaa HTML-dokumentin dynaamisen muokkauksen

GUI	Graphical User Interface Käyttöliittymä, jonka interaktio perustuu graafisiin elementteihin.
HTML	Hypertext Markup Language Hypertekstidokumenteissa käytetty rakenteellinen kuvauskieli.
IANA	Internet Assigned Numbers Authority Muun muassa IP-osoitteiden jakoa, Internetin juuripalvelimia ja MIME-tyyppejä hallinnoiva voittoa tavoittelematon organisaatio.
JAR	Java Archive Java-sovellusten käyttämä paketointimuoto, joka perustuu Zip-pakkausmenetelmään.
JSON	JavaScript Object Notation JavaScript-ohjelmointikieleen perustuva tekstipohjainen datan esitysmuoto.
JSONP	JSON with Padding Selainten lähdetietoturvan kiertävä Ajax-tekniikka, jossa tieto haetaan palvelimelta määritettyä takaisinkutsufunktiota kutsuvan suoritettavan koodin muodossa.
MIME	Multipurpose Internet Mail Extensions Alun perin sähköpostijärjestelmiin kehitetty datan tiedostomuodon kertova kaksiosainen määritys, jota käytetään Internetissä nykyään yleisesti.
MVC	Model-View-Controller Ohjelmistoarkkitehtuurissa yleisesti käytetty arkkitehtuurimalli, jossa käyttöliittymäkontrollin data, käsittelyrajapinta ja graafinen ulkoasu on erotettu toisistaan.

NDK	Native Development Kit Androidin tarjoama erillinen SDK, jonka avulla sille voi luoda C/C++-kielisiä sovelluksia. (Meier 2010, 15)
NPAPI	Netscape Plugin Application Programming Interface Useiden nettiselainten tukema liitännäisrajapinta.
QML	Qt Meta Language / Qt Modelling Language JavaScript-pohjainen kuvauskieli käyttöliittymäkehitykseen.
Qt	Alustariippumaton C++-kielinen kehitysympäristö. (Blanchette ja Summerfield 2006)
Qt Quick	Qt User Interface Creation Kit Käyttöliittymäohjelmointia helpottavat Qt työkalut, sisältäen QML kuvauskielen.
SDK	Software Development Kit Tietyn kehitysympäristön ja siihen liittyvät varusohjelmat sisältävä ohjelmistopaketti.
SIS	Software Installation Script Symbian-laitealustan käyttämä natiivi paketointimuoto.
UI	User Interface Laitteen käyttöliittymä. sisältäen näyttö- sekä syöttölaitteet.
UX	User Experience Loppukäyttäjän kokonaiskokemus laitteen käytöstä, joka sisältää myös käyttöliittymän.
W3C	World Wide Web Consortium Kansainvälisten yritysten ja yhteisöjen yhteenliittymä, joka ylläpitää ja kehittää WWW:n standardeja. (W3C 2012)
WAC	Wholesale Application Community Älypuhelinten web-sovellusten yhteensopivuutta ja saatavuutta suuremmalle kohdeyleisölle ajava organisaatio, joka

	ylläpitää samannimistä laitealustan palveluja tarjoavaa raja-pintakirjastoa.
WIMP	Windows, Icons, Menus and Pointing device Työpöydällisten graafisten käyttöliittymien joukko
WWW	World Wide Web Internetissä toimiva hajautettu hypertekstijärjestelmä.
XAML	Extensible Application Markup Language Microsoftin kehittämä XML-pohjainen tapa käyttöliittymän määrittämiseen.
XML	Extensible Markup Language W3C:n ylläpitämä tekstimuotoinen merkintäkieli.

Kuviot

Taulukot

Sisältö

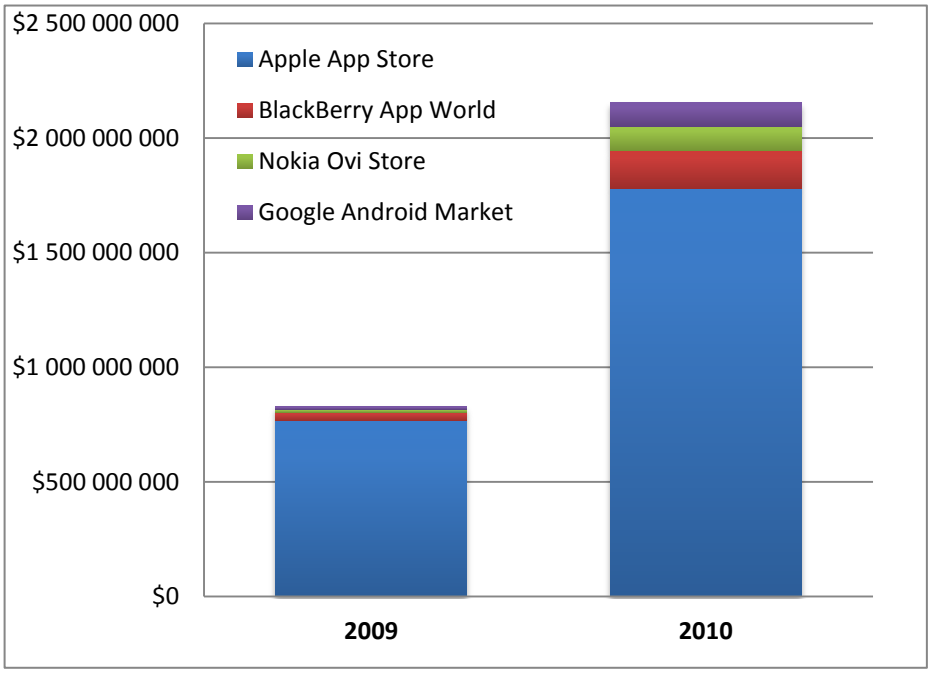
1 Johdanto

Älypuhelinmarkkinoilla on viime vuosina tapahtunut selvä painopisteen muutos pelkkien teknisten ominaisuuksien painottamisesta käyttöliittymävetoiseen käyttäjäkokemukseen. Tämän uuden trendin voi katsoa alkaneeksi vuonna 2007, jolloin Apple esitteli vallankumouksellisen iPhone-älypuhelimensa (Kuvio 1), jossa kosketuksella ohjattava käyttöliittymä korvasi perinteisen numero- ja suuntanäppäimiin perustuvan ohjausmallin. Sittemmin käytännössä katsoen kaikki älypuhelinvalmistajat ovat tuoneet markkinoille omia näkemyksiään kosketusohjattavista älypuhelimista, joka on johtanut ennennäkemättömän suureen markkinoiden pirstaloitumiseen erilaisten laitealustojen välillä.

Kuvio 1. Neljännen sukupolven iPhone 4S älypuhelin. (Apple 2012b)

Toinen merkittävä kehitysaskel viime vuosina on ollut älypuhelinten ohjelmistotarjonnan räjähdysmäinen kasvu, jonka on mahdollistanut kolmansien osapuolien yhä kasvava osuus

ohjelmistokehityksestä ja -tarjonnasta. Puhelinvalmistajat ovatkin aloittaneet taistelun paitsi loppukäyttäjien, myös kolmansien osapuolien ohjelmistokehittäjien sieluista. Tässä sodassa tärkeimmät aseet ovat ohjelmistojen levityskanavat sekä kehittyneet, modernit kehitysympäristöt. Pelkästään Applen App Store -sovelluskaupasta on ladattu vuoden 2008 avautumisen jälkeen yli 15 miljardia sovellusta kolmen vuoden sisällä avautumisestaan (Apple 2011a), joten kyseessä on valtava markkinapotentiaali. Sovelluskaupat nauttivatkin tällä hetkellä jopa yli sadan prosentin vuotuisista kasvuluvuista myynnissään (Kuvio 2) ja mukaan onkin pyrkimässä lukuisia yrittäjiä myös älypuhelinvalmistajien ulkopuolelta.



Kuvio 2. Sovelluskauppojen kokonaismyynti Yhdysvaltain dollareissa vuosina 2009 ja 2010. (IHS Screen Digest 2011)

Johtuen kolmansien osapuolten sovelluskehityksen kasvusta sekä painopisteen siirtymisestä kosketusohjattaviin käyttöliittymiin, markkinoille on ilmestynyt lukuisia uusia kehitysympäristöjä, joiden keskeisessä asemassa on juuri käyttöliittymäkehitystä tukeva käyttöliittymäkirjasto. Myös vanhoja sovelluskehyksiä on päivitetty tukemaan kosketusohjausta, mutta erityisen mielenkiintoisia ohjelmistokehittäjien näkökulmasta ovat uudet alusta alkaen kosketusohjausta silmällä pitäen suunnitellut käyttöliittymäkirjastot.

Tässä työssä on tarkoitus tarkastella älypuhelinkentällä tarjolla olevia moderneja käyttöliittymäkirjastoja sekä tutustua pienempään alijoukkoon tapaustutkimuksen avulla. Koska alan kehitys on poikkeuksellisen nopeaa, aihetta tarkastellaan paikoitellen työn aloittamisvuoden (2011) tilanteen näkökulmasta. Kyseinen vuosi on erityisen kiinnostava suomalaisesta näkökulmasta Nokian yrittäessä mukautua muuttuneeseen markkinatilanteeseen tuomalla markkinoille MeeGo- ja Windows Phone 7 -älypuhelinalustoihin perustuvia laitteita, jotka ovat molemmat erittäin käyttöliittymäpainotteisia ja alusta alkaen suunniteltu kosketusohjattaviksi.

Perinteisten käyttöliittymäkirjastojen lisäksi työssä tutustutaan web-sovelluksien älypuhelinsovituksiin ja niissä käytettäviin kosketusohjatuille mobiililaitteille optimoituihin käyttöliittymäkirjastoihin. Web-sovellukset ovat tällä hetkellä älypuhelinten sovellussuunnittelun nopeimmin suosiota kasvattava sovellusalustatyyppi (Vision Mobile 2011). Näyttääkin siltä, että web-sovellukset ovat tulossa älypuhelimiin isolla voimalla ja kasvua tuskin ainakaan hidastaa alan toimijoiden yhteishankkeina kehittämät laitealustakirjastot ja paketointistandardit. Työn näkökulma web-sovelluksiin on verrata niiden ja natiivisovellusten ominaisuuksia erityisesti käyttöliittymäsuunnittelun näkökulmasta.

Työ sisältää kolmeen päälukuun jaetun teoriataustan, jossa käydään läpi graafisten käyttöliittymien perusteet (Luku 2), sovelluskehitys merkittävimmille älypuhelinlaitteiden ekosysteemeille (Luku 3) sekä alustariippumattomat ratkaisut älypuhelinsovellusten tuottamiseksi (Luku 4). Lisäksi työ sisältää empiirisen osuuden raportointiin keskittyvän pääluvun (Luku 5) ja sen tuloksien pohjalta kirjoitetun yhteenvedon (Luku 6). Empiirinen osuus suoritettiin toteuttamalla ominaisuuksiltaan samankaltainen älypuhelinsovellus pienellä joukolla lähempään tarkasteluun valittuja käyttöliittymäkirjastoja. Työn liitteet sisältävät empiiristä osuutta varten tuotettujen sovellusten lähdekoodilistaukset.

2 Graafinen käyttöliittymä

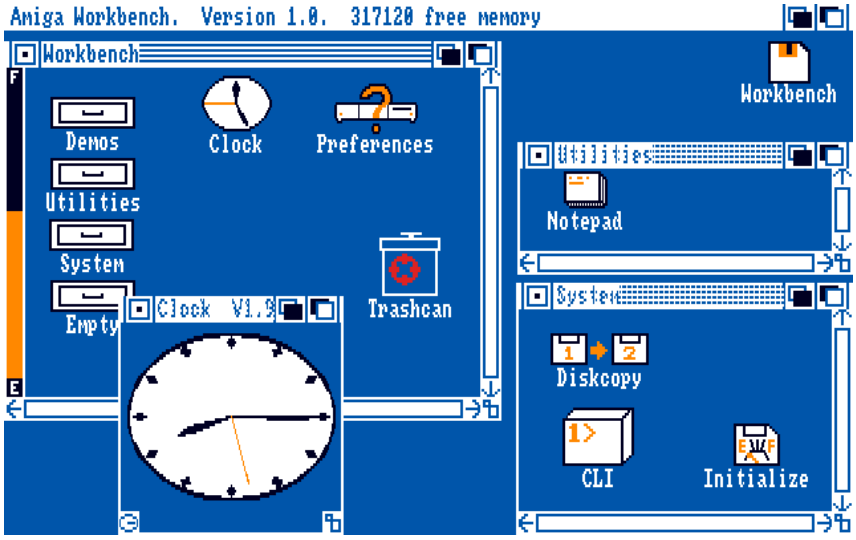
2.1 Perusteet

Termillä käyttöliittymä (engl. User Interface, UI) tarkoitetaan tapaa, jolla käyttäjä on vuorovaikutuksessa jonkin asian kanssa, kuten tietotekniikasta puhuttaessa tietokoneen tai vastaavan laitteen tai sen ohjelmiston kanssa. Perinteisesti tietokoneen tai ohjelmiston käyttöliittymäksi lasketaan sen näytölle piirtämä sisältö, sekä sisällön manipulointiin käytettävät ohjainlaitteet, kuten esimerkiksi näppäimistö, hiiri tai kosketusohjaus. Nykyisin laajassa käytössä olevilla graafisilla käyttöliittymillä tarkoitetaan sellaisia käyttöliittymiä, joissa informaatio esitetään tekstin lisäksi graafisten elementtien avulla. (Galitz 2007, 16–17)

Graafiset käyttöliittymät perustuvat näytöllä näkyvien objektien välittömän manipuloinnin (engl. direct manipulation) ideaan. Käyttäjä voi ohjainlaitetta käyttäen aktivoida tai siirrellä näytöllä näkyviä graafisia objekteja, jotka reagoivat visuaalisesti käyttäjän toimenpiteisiin välittömästi. Tämän mahdollistaa osoitinlaite, kuten hiiri tai kosketusnäyttö. (Cooper, Reimann, ja Cronin 2007, 375–385)

2.1.1 WIMP- eli työpöytäkäyttöliittymät

Perinteisen osoitinlaitteella käytettäviin ikkunoituihin sovelluksiin perustuvan graafisen käyttöliittymän kehitti alun perin Xerox PARC vuonna 1973 julkaisemaansa kokeelliseen Xerox Alto -tietokonejärjestelmään. Toimistolaitteita ja -tarvikkeita ydinliiketoimintanaan valmistava Xerox ei kuitenkaan kaupallistanut keksintöään kovin tehokkaasti. Varsinainen läpimurto tapahtui vasta Applen kopioitua idean, kun se vuonna 1984 julkaisi ensimmäisen Macintosh-tietokoneensa. Jo seuraavana vuonna vastaavan graafisen käyttöliittymän esitteli Microsoft Windows 1.0 -tuotteellaan sekä Commorode ensimmäisellä Amiga-tietokoneellaan (Kuvio 3). (Cooper, Reimann, ja Cronin 2007, 423–427; Galitz 2007, 7–8; Reimer 2005)



Kuvio 3. Ensimmäisen Amiga-tietokoneen Workbench-työpöytäympäristö. (Reimer 2005)

Tällainen graafinen käyttöliittymä perustuu vertauskuvallisesti työpöytään, jolla on hajallaan erilaisia muistiinpanoja, papereita ja muita esineitä. Toisinaan tällaisten työpöytäjärjestelmien käyttöliittymiä kutsutaan nimellä WIMP-käyttöliittymät (engl. Windows, Icons, Menus and Pointing device). Nimensä mukaisesti tällaisissa järjestelmissä tärkeässä osassa ovat sovellusikkunat, graafisten kuvakkeiden runsas käyttö, valikkoihin kätketyt monipuoliset toiminnot ja osoitinlaitteeseen perustuva ohjausmalli. (Galitz 2007, 16–17)

Työpöytäympäristössä yleisin osoitinlaite on hiiri, jonka avulla käyttäjä liikuttaa näytölle piirrettyä kursorisymbolia ja aktivoi kursorin alla olevan objektin käyttämällä hiiren ensisijaista painiketta. Yleensä hiirissä on useampia painikkeita ja vieritysrulla, joita voidaan yksinkertaisen valinnan lisäksi käyttää esimerkiksi kontekstivalikon avaamiseen tai näytön sisällön vierittämiseen. Hiiri painikkeineen mahdollistaa monipuolisia, yleisessä käytössä olevia ohjauseleitä, kuten tuplaklikkauksen ja objektin liikuttamisen pitämällä valintapainiketta pohjassa. (Cooper, Reimann, ja Cronin 2007, 375–385)

2.1.2 Web-käyttöliittymät

Toinen yleisesti käytetty graafisen käyttöliittymän muoto on Internetin web-sivuissa käytetty hypertekstiin perustuva web-käyttöliittymä. Se sai alkunsa vuonna 1989 Tim Berners-

Leen alkaessa kehittää HTML-kuvauskieltä (engl. HyperText Markup Language) ja sen siirtämisessä käytettävää HTTP-sovellusprotokollaa (engl. HyperText Transfer Protocol). Kokonaisuutena tätä hypertekstijärjestelmää alettiin kutsua nimellä WWW (World Wide Web). Vuonna 1994 perustettiin World Wide Web Consortium (W3C) -organisaatio kehittämään webin standardeja. (Galitz 2007, 8–9)

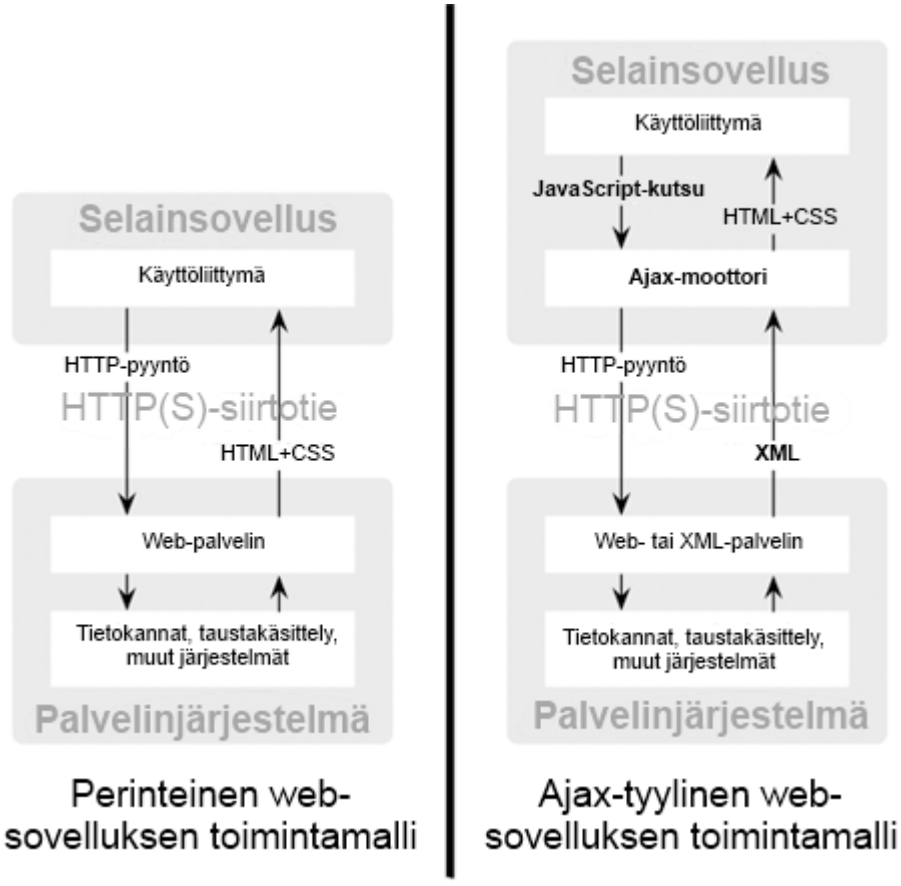
Web-käyttöliittymissä käytettävää HTML-kieltä ei alun perin suunniteltu valtaväestön käyttöön. Se oli rajoittunut tarjoamassaan interaktiossa ja käyttöliittymäelementtiensä kattavuudessa. Verrattuna aikansa työpöytäsovelluksiin se tuntui jopa askeleelta taaksepäin graafisen käyttöliittymän kehityksessä. Web-käyttöliittymä ei esimerkiksi näytä yhtenäiseltä käytetyn käyttöjärjestelmän käyttöliittymän kanssa, koska web-sisällön piirtämisestä vastaa selaimen käyttämä *selainmoottori*. Toisaalta web-käyttöliittymissä on juuri tästä syystä suurempi vapaus kehittää yksilöllisiä käyttöliittymiä. (Galitz 2007, 28–36)

HTML-kuvauskieli oli alusta lähtien tarkoitettu kuvailemaan dokumentin rakennetta, ei niinkään sitä miltä sisällön pitäisi näyttää. Ensimmäisissä web-sivuissa suunnittelijalla ei ollut esimerkiksi ollenkaan mahdollisuutta vaikuttaa tekstissä käytettävään fonttiin. WWW:n yleistyessä tämä aiheutti tietysti isoja ongelmia ja selainten valmistajat alkoivat itse lisätä selainkohtaisia esittämistapaa koskevia laajennuksia HTML-kieleen. Vuonna 1994 W3C alkoi kehittää esittämistavan määrittelyyn tarkoitettua CSS-tyylikieltä (engl. Cascading Style Sheet) ja sen ensimmäinen versio julkaistiin vuonna 1996. (Lie ja Boss 1999)

Dynaaminen sisältö. Alun perin WWW-sivut olivat sisällöltään staattisia ja uusi sisältö ladattiin näkymä kerrallaan eri hyperlinkkejä seuraamalla. Sittemmin sivujen sisältö on kehittynyt dynaamisemmaksi lukuisten asiakas- ja palvelinpuolen laajennusten avulla. Asiakaspuolelle on kehitetty muun muassa JavaScript-kieli, Java Appletit ja Flash-animaatiot, kun taas palvelinpäähän on kehitetty joukko sivun dynaamisesti generoivia ja sivupyyntöjen välityksellä saatua tietoa käsitteleviä tekniikoita kuten CGI, ASP, JSP ja PHP. (Asleson ja Schutta 2006, 1–10)

Dynaamisen sisällön läpimurto tapahtui kuitenkin vasta vuonna 2005 Ajax-konseptin (Asynchronous JavaScript And XML) syntymisen myötä. Ajax sai alkunsa Jesse James

Garretin kirjoittamasta artikkelista *Ajax: A New Approach to Web Applications* (2005), jossa hän kuvaili uudenlaisten palveluiden, erityisesti Google Maps ja Google Suggest, käyttämiä tekniikoita sekä niiden mahdollistamaa käyttäjäkokemusta, jossa WWW-sivuja ei tarvinnut enää ladata uudelleen jokaisen käyttäjän toiminnon jälkeen vaan tarvittu uusi data haettiin dynaamisesti palvelimelta XML-muodossa ja päivitettiin käyttäjän näkemään sivuun dynaamisesti (Kuvio 4). Näissä palveluissa sovelluslogiikka oli osittain tuotu asiakaspäähän ja tiedonsiirto palvelimelta asiakkaalle tapahtui XML-muodossa. Jo seuraavana vuonna W3C standardisoi Ajax-tekniikoille keskeisen *XMLHttpRequest*-tiedonsiirtokanavan palvelinten ja asiakassovellusten välille (W3C 2010). Nykyisin Ajax-termillä viitataan sen käyttämään toimintamalliin, eikä se ole varsinaisesti sidottu tiettyihin toteutustekniikkoihin kuten JavaScript ja XML.



Kuvio 4. Ajax-tekniikan toimintaperiaate verrattuna perinteisiin web-sovelluksiin. (Garret 2005)

Web-sovellukset. WWW:n yleistyessä yhä useampia aikaisemmin työpöytäjärjestelmiin kehitettyjä sovelluksia ollaan viemässä webiin. Siinä missä sisältöön ja informaatioon keskittyviä staattisia hypertekstidokumentteja kutsutaan yleisesti web-sivuiksi, kutsutaan sisällöltään dynaamisia kokonaisuuksia web-sovelluksiksi. Jako ei ole ihan yksikäsitteinen, mutta yleensä web-sivu on tuotettu pääasiassa antamaan jotain informaatiota, kun taas web-sovelluksella voi tehdä jotain tuottavaa. (Galitz 2007, 28–30)

2.1.3 Älypuhelinten kosketusohjatut käyttöliittymät

Andries van Dam esitti 1997 käsitteen post-WIMP, jolla tarkoitetaan kursoriohjattujen työpöytäympäristön jälkeisiä käyttöliittymiä, joissa interaktio tapahtuu luonnollisemmin, käyttäen hyväksi esimerkiksi puhetta, kuuloa tai kosketusta. Post-WIMP -käyttöliittymät saattavan kuitenkin käyttää joitain WIMP-käyttöliittymien elementtejä, kuten valikkoja, grafiikkaa tai ikkunoita. Esimerkiksi kosketusohjattujen mobiililaitteiden käyttöliittymät lasketaan tähän luokkaan. (van Dam 1997)

Aiemmin kosketusohjausta pidettiin lähinnä kursoriohjauksen erikoistapauksena. Kosketusohjaus on kuitenkin luonteeltaan epätarkempaa ja kömpelömpää kuin hiiren ja näppäimistön käyttäminen, joten kosketusohjauksen apuna jouduttiin käyttämään erillistä osoitinkynää (engl. stylus). Osoitinkynä on kuitenkin epäkäytännöllinen lisäväline, erityisesti käytettynä yhdessä fyysisen näppäimistön kanssa (Cooper, Reimann, ja Cronin 2007, 188). Osoitinkynällä toimivia kosketusnäyttöjä käytettiin lähinnä kämmentietokoneissa (engl. Personal Digital Assistant, PDA), joista osassa oli myös puhelinominaisuuksia.

Vuonna 2007 julkaistu Applen iPhone käynnisti kosketusnäytöllisten älypuhelinten vallankumouksen. Sen käyttöliittymä oli suunniteltu kosketusohjauksen ehdoilla eikä sen käytössä tarvittu lainkaan osoitinkynää. Siinä käytettiin myös innovatiivisella tavalla monipistekosketusta ja sen mahdollistamia *kosketuseleitä*. iPhone-älypuhelinta ja sitä seuranneita kilpailevia laitteita yhdistää joukko samankaltaisia kosketusohjauksen peruseleitä (Taulukko 1), joskin eleiden kutsumanimet eivät ole vakiintuneet vaan eri alustojen käytetään erilaisia ja jopa päällekkäisiä nimeämiskäytäntöjä.

Symboli	Kosketusele	Kuvaus ja yleinen toiminto
	Painallus (engl. tap)	Kontrollin valinta tai aktivointi. Analogia hiiren valintapainikkeeseen.
	Tuplapainallus (engl. double tap)	Zoomaustilan vaihto.
	Pitkä painallus (engl. touch and hold / long tap)	Avaa kontekstivalikon tai tekstin päällä aktivoi suurennus- tai valintatilan.
	Raahaus (engl. drag)	Elementtien siirtäminen tai näytön vieritys. Analogia hiirellä tehtävään raahaukseen.
	Vieritys (engl. scroll / pan)	Pysty- tai vaakasuunnassa tapahtuva raahaus, jota käytetään näytetyn sisällön vierittämiseen.
	Sipaisu (engl. flick)	Yksisuuntainen nopea vieritys, jota käytetään sisällön nopeaan vieritykseen tai viereiseen sisältöelementtiin siirtymiseen.
	Pyyhkäisy (engl. swipe)	Yksisuuntainen pitkä vieritys, jossa liike aloitetaan yleensä näytön tai jonkin kontrollin reunasta. Käytetään tuomaan esille ja piilottamaan jokin valikko, näkymä tai toiminto.
	Nipistys (engl. pinch) ja laajennus (engl. stretch / spread / pinch out)	Kahden sormen liikuttaminen lähemmäs tai kauemmas toisistaan zoomaa sisältöä kauemmaksi tai lähemmäksi.

Taulukko 1. Älypuhelimissa käytetyt yleiset kosketuseleet. (Android Developers 2012c; Apple 2012a; Bowman 2011; Ideum 2012; Microsoft 2010; Wroblewski 2010)

Älypuhelinten käyttöliittymissä on monia samankaltaisuuksia WIMP-käyttöliittymiin. Molemmissa käytetään runsaasti graafisia kuvakkeita sekä piilotetaan lisätoiminnot erillisten hierarkisten valikoiden alle. Molemmissa käytetään myös osoitinlaitetta, joskin kosketusohjatuissa älypuhelimissa ei tarvita enää erillistä apuvälinettä osoittamiseen. Älypuhelimissa sen sijaan ikkunan käsite ei ole yhtä keskeinen, koska normaalisti kerrallaan näytetään vain yksi koko ruudun täyttävä, reunaton sovellusikkuna. Sen sijaan sovellukset jaetaan yleensä useisiin *näkymiin*, jotka näytetään jaetussa sovellusikkunassa yksi kerrallaan. Älypuhelimissa ei myöskään ole rajoittuneen kokonsa vuoksi perinteistä työpöytää, vaan sen tilalla käytetään erilaisia kotinäyttöjä ja sovellusvalikoita.

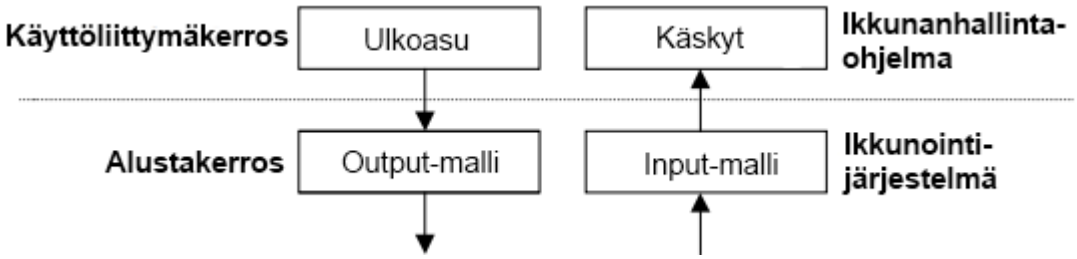
2.2 Käyttöliittymien sisäinen rakenne

2.2.1 Ikkunointijärjestelmät

Graafista käyttöliittymää pyörittää käyttöjärjestelmän päällä ikkunointijärjestelmänä tunnettu komponentti, jonka vastuulla on erillisten ohjelmistojen hallinnoimien *ikkunoiden* ylläpito ja piirtäminen käyttäjälle. Ikkunointijärjestelmä siis pitää yllä eri ikkunoiden geometriatietoja sekä tietoa siitä mikä on päällimmäinen, aktiivinen ikkuna. Rajapintatasolla ikkunointijärjestelmä tarjoaa sovelluksille mahdollisuuden piirtää tekstiä ja grafiikkaa tietyn kokoiselle alueelle. Ikkunointijärjestelmä on myös vastuussa eri ohjainlaitteiden kuuntelemisesta ja niiltä saatujen tapahtumien ohjaamisesta oikealle prosessille. Esimerkiksi kirjoitettu teksti ohjataan yleensä päällimmäiselle ikkunalle, kun taas hiiren napin painallus ohjautuu päällimmäisenä kursorin alle olevalle ikkunalle.

Ikkunointijärjestelmä voidaan teknisesti jakaa kahteen kerrokseen (Kuvio 5). Alempi kerros abstraktoi näyttölaitteen (output) tarvitsemat piirtoprimitiivit sekä luo käyttäjän syötteen (input) perusteella tapahtumat tarvittaville prosesseille. Ylempi kerros, eli *ikkunanhallintaohjelma*, puolestaan keskittyy käyttöliittymään ja tarjoaa työpöydän ikkunanhallinnan sekä ylläpitää tietoa aktiivisesta ikkunasta (engl. active window / input focus / listener). Microsoft Windows- ja Macintosh-käyttöjärjestelmissä ikkunanhallintaohjelma on kiinteä osa ikkunointijärjestelmää, mutta Unix-pohjaisten järjestelmien yleisessä käytössä olevassa

X-ikkunointijärjestelmässä on mahdollista käyttää useita eri ikkunanhallintaohjelmia. Harva sovellus käyttää ikkunointijärjestelmän tarjoamia palveluja suoraan, vaan yleensä sovelluksen ja ikkunointijärjestelmän välissä on erillinen *käyttöliittymäkirjasto*. (Myers 2004)



Kuvio 5. Ikkunointijärjestelmä voidaan jakaa alustakerrokseen (eli varsinaiseen ikkunointijärjestelmään) sekä käyttöliittymäkerrokseen (eli ikkunanhallintaohjelmaan), jotka molemmat voidaan edelleen jakaa pienempiin, input- ja output-tietoja hallitseviin osiin. (Myers 2004)

2.2.2 Käyttöliittymäkirjastot ja sovelluskehykset

Käyttöliittymäkirjasto (engl. widget toolkit) on ikkunointijärjestelmän kapseloiva ohjelmistokomponentti, joka tarjoaa sovelluskehittäjälle käytettäväksi joukon erilaisia valmiita käyttöliittymäkomponentteja eli *kontrolleja*. Tyypillisiä kontrolleja ovat esimerkiksi erilaiset tekstikentät, kuvakkeet, painonapit, paneelit ja valikot. Käyttöliittymäkirjaston käyttämisestä sovelluksessa on myös se etu, että sovellus tulee näyttämään yhdenmukaiselta muiden samalla käyttöliittymäkirjastolla toteutettujen sovellusten kanssa. (Myers 2004)

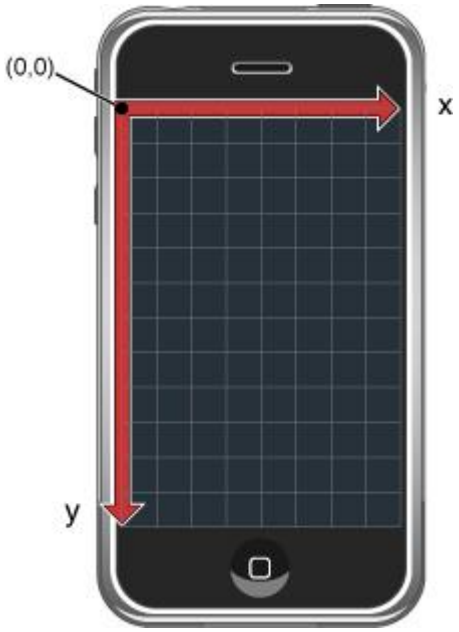
Käyttöliittymäkirjasto voi olla arkkitehtuurisesti täysin erillään ikkunointijärjestelmästä tai sitten osana sitä, jolloin ikkunointijärjestelmä voi myös itse käyttää käyttöliittymäkirjastonsa kontrolleja esimerkiksi ikkunoiden kehyksen rakennuspalikoina. Tämä ei kuitenkaan estä erillisten, alemman tason piirtorutiineja hyödyntävien käyttöliittymäkirjastojen käyttöä. Sama käyttöliittymäkirjasto voikin olla tarjolla usealle eri ikkunointijärjestelmälle, joka helpottaa alustariippumattomien sovellusten rakentamista. (Myers 2004)

Ikkunointijärjestelmän päällä toimiva käyttöliittymäkirjasto on yleensä vain pieni osa suurempaa *sovelluskehystä* (engl. application framework), jonka tarkoituksena on avustaa ja

ohjata sovelluskehittäjän työtä. Sovelluskehys sisältää yleensä joukon erilaisia sovelluskirjastoja, kääntäjiä ja muita kehitystyökaluja. Sovelluskehys tarjoaa sovellukselle valmiin arkkitehtuuripohjan ja laajennettavan oletustoiminnallisuuden, jonka päälle sovelluskohtainen logiikka rakennetaan. Sovelluskehyksen tehokas hyödyntäminen vaatii kuitenkin usein pitkän, 6–12 kuukauden kokemuksen sen käytöstä. (Fayad ja Schmidt 1997)

2.2.3 Sovellusikkuna

Yksittäinen käyttöjärjestelmän ajama graafinen sovellus koostuu yhdestä tai useammasta sovellusikkunasta, joiden sisällöstä se vastaa. Ikkunan sijainnin ja dimensiot näytöllä määrää ikkunointijärjestelmä, joskin sovelluskirjasto voi tarjota sovelluksille rajapintoja ikkunan hallintaan. Ikkuna tarjoaa sovellukselle kaksiulotteisen koordinaatiston, jonka origo on tyypillisesti ikkunan vasemmassa ylälaidassa (Kuvio 6). Koordinaatistossa perusyksikkö on yksi näytön pikseli eli väripiste, mutta useassa järjestelmässä on myös mahdollista käyttää millimetrejä tai muita yksikköjä kontrollien sijainnin ja koon määrittämiseen. Pikseleiden käyttö yksikkönä voi olla vaarallista, sillä jopa saman laitealustan eri toteutuksissa voidaan käyttää hyvinkin erikokoisia ja tarkkuuksisia näyttöjä. Esimerkiksi iPhone-älypuhelimen neljäs versio käyttää 3,5 tuuman näyttöelementtiä jonka resoluutio on 640x960 pikseliä, kun taas samoja sovelluksia pyörittävät edellisen sukupolven laitteet käyttävät samankokoista näyttöä, jonka resoluutio on 320x480 pikseliä. (Apple 2011c, 17–22)



Kuvio 6. Tyypillinen kokoruututilan sovellusikkunan koordinaatisto ja origon sijainti. (Apple 2011c)

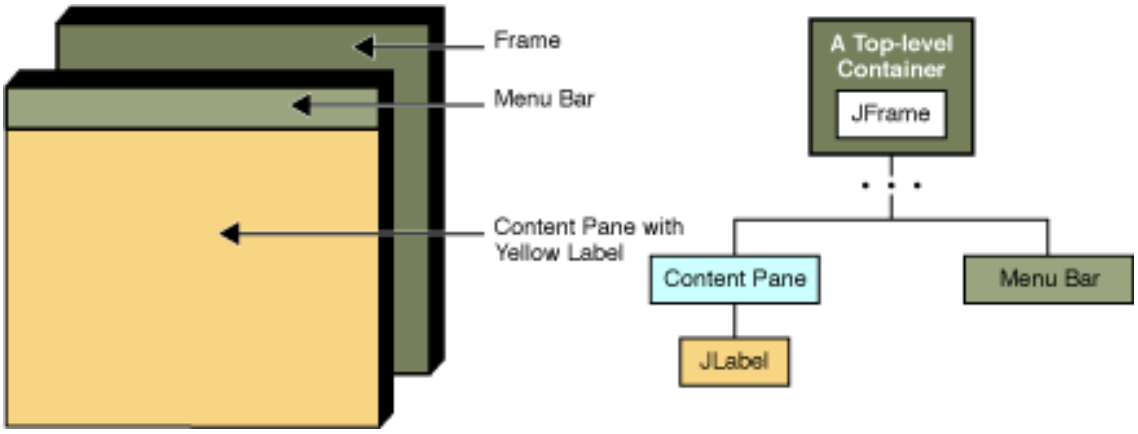
Sovellusikkunoiden yhteydessä puhutaan usein *kromista* (engl. chrome), jolla viitataan sovelluskäyttöliittymän toistuviin elementteihin, jotka vievät näytöltä tilaa varsinaiselta sisällöltä. Työpöytäjärjestelmissä kromiksi lasketaan esimerkiksi työpöytäjärjestelmän kiinteät osat, sovellusikkunan raamit sekä sovelluksen omat työkalupalkit. Älypuhelinjärjestelmissä näytetään yleensä vain yksi sovellusikkuna kerrallaan, mutta järjestelmä varaa usein itselleen pienen siivun näytön pinta-alasta näyttääkseen laitteen tilapalkin (engl. status bar), joka voi sisältää esimerkiksi graafisesti esitetyn tiedon akun lataustasosta. Jotkut sovelluskehykset mahdollistavat sovelluksen ajamisen *täyden ruudun tilassa*, jolloin koko näyttöpinta-ala on sovelluksen hallittavissa. Sovellukset voivat myös käyttää sovelluskehyskohtaisia vakiokäyttöliittymäelementtejä, jotka vievät osan käytettävissä olevasta näyttöpinta-alasta sovellusikkunan sisältä. Tällaisia ovat esimerkiksi erilaiset työkalupalkit (engl. toolbar / action bar). (Nielsen ja Budiu 2012, 55–59)

Sovellusikkuna voi koostua useasta eri *näkymästä*, joiden välillä käyttäjä voi navigoida. Navigointi voidaan vielä jakaa karkeasti hierarkiseen ja lateraaliseen malliin, joita voi myös yhdistellä monimuotoisempien rakenteiden saavuttamiseksi (Nokia 2012a; Android Developers 2012f). Hierarkisessa mallissa käyttäjä porautuu toiminnallaan alkunäkymästä

syvemmille tasoille näkymähierarkiassa tai palaa yhden tason ylöspäin kohti alkunäkymää. Ikkuna pitää yllä *näkymäpinoa*, jonka päällimmäinen näkymä vastaa ikkunassa sillä hetkellä aktiivista näkymää, jonka varaamat resurssit vapautetaan, kun käyttäjä navigoi pinon edelliseen näkymään. Lateraalisessa mallissa käyttäjälle tarjotaan joukko samantasoisia näkymiä esimerkiksi välilehtinä. Mikäli alkunäkymä ei kuulu välilehtiin, annetaan käyttäjälle myös mahdollisuus palata hierarkiassa ylöspäin, jolloin käyttäjä palautetaan suoraan näkymäpinossa samantasoisten lateraalinäkymien alla olevaan näkymään. Toinen tapa toteuttaa lateraalinavigaatio on toteuttaa saman näkymän sisään erilaisia dynaamisesti vaihtuvia sisältöjä, jolloin näkymäpinoa ei tarvitse manipuloida navigoidessa juurta kohti.

Älypuhelimet tukevat usein sovellusikkunan näyttämistä joko *pystytilassa* (engl. portrait) tai *vaakatilassa* (engl. landscape). Siirtymä vaaka- ja pystytilojen välillä voidaan pakottaa ohjelmallisesti tai se voi tapahtua automaattisesti laitteen sensoritietojen avulla, jolloin sovelluskehys tai itse sovellus reagoi orientaatiotapahtumaan. Sovellus voidaan myös lukita sopivaan orientaatiotilaan. Suositukset orientaatiotilojen käytöstä vaihtelevat eri alustojen välillä. Esimerkiksi iOS-sovellusten suositellaan tukemaan molempia orientaatiotiloja aina kun mahdollista (Apple 2012a, 57–58), kun taas MeeGo-järjestelmää käyttävällä N9-laitteella oletuksena sovellukset tukevat vain pystytilaa ja vaakatilaa tulisi käyttää vain erikoistapauksissa (Nokia 2012b).

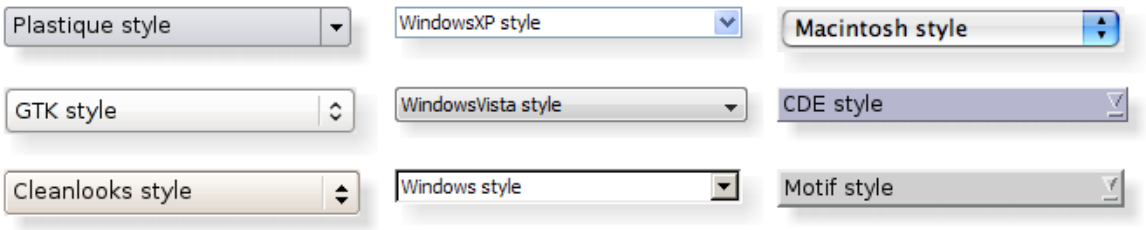
Käyttöliittymien kontrollit asemoidaan yleensä sovellusikkunaan tai näkymään hierarkisesti toisten kontrollien sisään, sovelluksen ikkunan tai näkymän toimiessa kyseisen puurakenteen juurena. Sama puurakenne ilmentää myös kontrollien piirtojärjestystä, joten kauempana juuresta oleva kontrolli piirretään aina taustalla olevan kontrollin päälle. Asemoinnissa käytetään joko *absoluuttista asemointia*, jossa jokaisen kontrollin sijainti asetetaan manuaalisesti suhteessa ylemmän tason kontrolliin, tai sitten erityisiä *asemointikontrolleja* (engl. layout manager), joiden lapsisolmujen kontrollit voidaan asemoida esimerkiksi allekkain, rinnakkain tai taulukkoon (Kuvio 7). Asemointikontrollien käyttö on erityisen hyödyllistä silloin kun kontrollien käytettävissä oleva tila ei ole vakio. (Android Developers 2012d)



Kuvio 7. Esimerkki näkymän hierarkisesta rakenteesta. (Oracle 2012)

2.2.4 Käyttöliittymästandardit

Käyttöliittymästandardit (engl. look and feel) määrittelevät sen miltä sovelluksen ikkuna, kontrollit ja teksti näyttävät (Kuvio 8). Käyttöliittymästandardeja ovat esimerkiksi Windows-käyttöjärjestelmän eri versioiden tyylit, Macintosh-tyyli sekä Unix-pohjaisten työpöytäympäristöjen Motif-käyttöliittymän esittelemä samanniminen tyyli. Yleensä käyttöjärjestelmä määrittää käytetyn käyttöliittymästandardin, mutta sovelluskehittäjällä voi olla mahdollisuus valita useamman käyttöliittymästandardin joukosta, riippuen käytetystä käyttöliittymäkirjastosta. (Myers 2004)



Kuvio 8. Sama kontrolli erilaisilla Qt-kehityskirjaston tukemilla käyttöliittymästandardeilla. (Qt Developer Network 2012a)

Eri käyttöliittymästandardeissa kontrollit ovat usein samankaltaisia (Kuvio 8). Siitä huolimatta sovelluksen käyttöliittymän porttaaminen yhteen alustaan sidotulta käyttöliittymäkirjastolta toiselle on erittäin työlästä. Useamman eri käyttöliittymästandardin kanssa yhteensopivia käyttöliittymäkirjastoja kutsutaan *virtuaalisiksi käyttöliittymäkirjastoiksi* (engl.

virtual toolkit / cross-platform development system). Yksinkertaisimmillaan tällainen on alustasidonnaisten käyttöliittymäkirjastojen päälle toteutettu ylimääräinen abstraktiokerros, joka todellisuudessa käyttää alustan omia kontrolleja. Huonona puolena tässä lähestymistavassa voidaan tarjota vain kontrollit, jotka löytyvät kaikista tuetuista natiivikirjastoista. Suositumpi ratkaisu on toteuttaa oma kontrollivalikoima usean eri käyttöliittymästandardin mukaisesti. (Myers 2004)

2.2.5 Käyttöliittymien oliomalli

Modernit käyttöliittymäkirjastot perustuvat vahvasti olio-ohjelmointiin, jossa kutakin sovelluksessa olevaa kontrollia kuvaa itsenäisesti sisäistä tilaansa ylläpitävä *olio* (engl. object). Kaikki kontrolliluokat puolestaan periytyvät samasta käyttöliittymäkontrollien kantaluokasta, jolloin kontrollien rajapinnoissa on paljon yhteisiä piirteitä. Kontrollit keskustelevat yleensä sovelluksen suuntaan erilaisten *takaisinkutsujärjestelmien* avulla, joissa kontrolli kutsuu sovelluksen rajapintaa esimerkiksi käyttäjän tehtyä valintansa valintalistaa esittävällä kontrollilla. (Myers 2004)

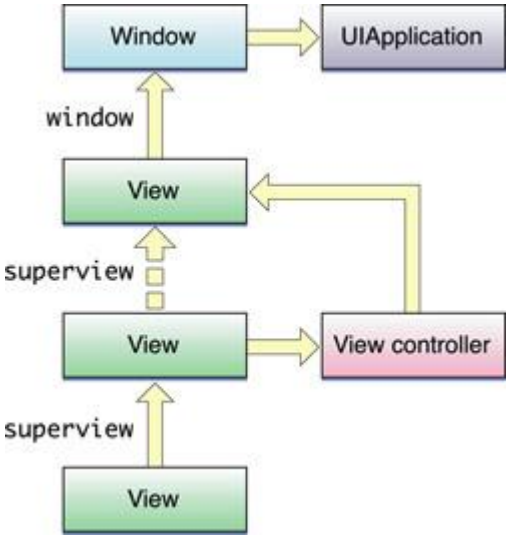
Oliopohjainen lähestymistapa sopii erityisen hyvin käyttöliittymäkehitykseen, sillä se on luonnollinen tapa ajatella käyttöliittymän elementtejä. Olioista on myös käytännön hyötyä, sillä niiden sisäiseen logiikkaan voidaan ulkoistaa toimintoja, jotka olisivat muuten sovelluksen vastuulla. Lisäksi sovelluskohtaisten kontrollien luominen on helpompaa kontrollien perinnän ja komposiittiluokkien ansiosta. (Myers 2004)

Oliopohjaisten käyttöliittymäkirjastojen kontrolleissa käytetään usein vuoden 1980 Smalltalk-ohjelmointikielessä ensimmäistä kertaa esiteltyä MVC-arkkitehtuurimallia (model-view-controller) tai jotain sen lukuisista johdannaisista. MVC-mallissa sovelluslogiikka ja käyttöliittymä on eristetty toisistaan jakamalla kontrollit kolmeen eri osaan. MVC-mallissa *näkymä* vastaa tiedon piirtämisestä, *malli* tiedon säilyttämisestä ja *käsittelijä* käyttäjäinteraktion hallinnasta. Näkymän ja mallin erottelu mahdollistaa saman mallin käytön usean eri näkymän välillä sekä helpottaa myös usean eri käyttöliittymästandardin tukemista. (Buschmann, ym. 1996)

2.2.6 Tapahtumankäsittely

Tapahtumapohjainen ohjelmointi (engl. event-driven programming) on käyttöliittymäohjelmoinnissa yleisesti käytetty ohjelmointiparadigma. Tapahtumapohjaisissa sovelluksissa ohjelman suoritus ei etene ennalta määrätyn sekvenssin mukaisesti eikä sillä ole selkeää etukäteen määrättyä loppua, vaan sovellus jää *tapahtumasilmukkaan* odottamaan käsiteltäviä tapahtumia. Tapahtumiksi voidaan laskea esimerkiksi näppäimistön painallus, ikkunaraamin pienentäminen tai nappikontrollin aktivoiminen. Tapahtuman laukaisun yhteydessä tapahtuma lisätään sovelluksen keskitettyyn *tapahtumajonoon*, josta tapahtumasilmukka asynkronisesti poimii tapahtumat yksitellen ja lähettää eteenpäin *tapahtumankäsittelijöille*, jotka ovat etukäteen tilanneet kyseisin tapahtumatyypin. Joissain tapauksissa tapahtumankäsittelijöitä voidaan myös kutsua suoraan, käyttämättä tapahtumajonoa. Tapahtumasilmukan toiminnallisuus on yleensä implementoitu osaksi käyttöliittymä- tai sovelluskirjastoa. Käyttöliittymäkirjastoissa sovelluksen tapahtumankäsittelyä käytetään ikkunointijärjestelmältä tulevien tapahtumien käsittelyn lisäksi yhtenä mahdollisena kontrollien takaisinkutsumenetelmänä. (Tucker ja Noonan 2004)

Mikäli yksittäisellä tapahtumalla on useita käsittelijöitä, on täysin käytetyn toteutuksen vastuulla päättää missä järjestyksessä tapahtumankäsittelijöitä kutsutaan. Käyttöliittymäkontrollien kohdalla järjestys on erityisen tärkeä, jotta esimerkiksi kontrollit piirretään oikeassa järjestyksessä tai hiiren painallus käsitellään päällimmäisessä kontrollissa eikä sen alla olevassa. Esimerkiksi web-käyttöliittymissä on jopa selainkohtaisia fundamentaalisia eroja siinä, käsitelläänkö tapahtuma ensin takimmaisessa juuritason kontrollissa (tapahtuman kaappaus, engl. event capturing), päällimmäisessä kontrollissa (tapahtuman kupliminen, engl. event bubbling), vai onko tapahtumilla sekä kaappaus- että kuplimisvaihe (Koch 2012). Joidenkin käyttöliittymätapahtumien, kuten ohjauslaitteen painalluksen kohdalla halutaan yleensä että vain yksi kontrolli reagoi siihen. Tällöin jokin tapahtumankäsittelijä *kuluttaa* tapahtuman, eikä sitä enää lähetetä jonossa seuraaville käsittelijöille. Tapahtumankulku voi olla toteutettuna käyttöliittymäkirjastossa myös siten, että tapahtuma ohjataan vain yhdelle kontrollille, jonka vastuulla on kutsua hierarkiassa juurta kohti seuraavaa kontrollia, mikäli se ei itse kuluta tapahtumaa (Kuvio 9).



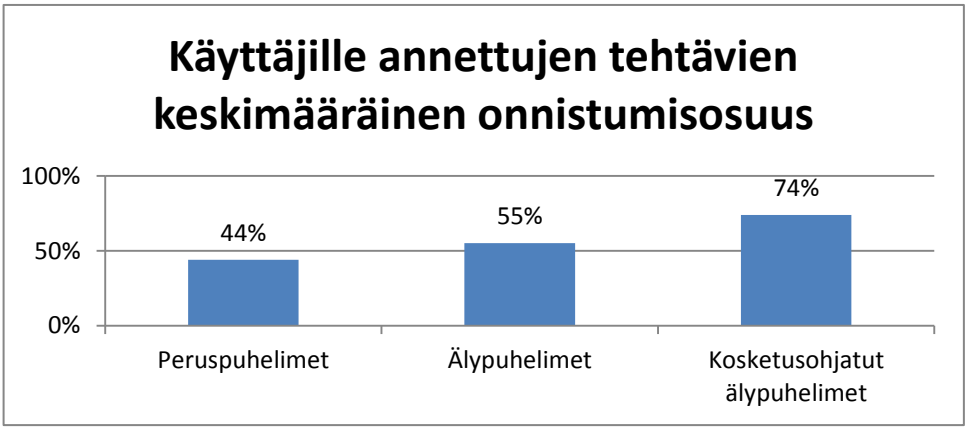
Kuvio 9. Tapahtumankulku iOS alustan UIKit-kirjastolla. (Apple 2011b)

2.3 Käytettävyys

Nielsen (2003) määrittää käytettävyyden kvalitatiiviseksi ominaisuudeksi, joka kuvaa käyttöliittymän helppokäyttöisyyttä. Hänen mukaansa käytettävyys koostuu viidestä laatukomponentista, joita ovat opittavuus, tehokkuus, muistettavuus, virheherkkyys ja käyttäjätyytyväisyys. Käytettävyys on tärkeä ominaisuus sovellukselle, sillä se saa loppukäyttäjät pysymään sovelluksen parissa pidempään ja useammin, joka näkyy sovelluksen toteuttajalle usein myös suurempana kassavirtana.

Käytettävyyttä voidaan arvioida erilaisten käyttäjätestien avulla. Kun käytettävyyttä halutaan verrata numeerisesti erilaisten ratkaisujen välillä, Nielsen (2001) suosittelee mittaamaan käytettävyyttä antamalla testikäyttäjille sarjan erilaisia käyttötehtäviä, joista lasketaan kullekin käyttöliittymäratkaisulle keskimääräinen onnistumisosuus. Yksittäisen käyttäjän onnistumisosuus voidaan laskea tehtäväjoukon pisteytyksien keskiarvona, kun yksittäinen tehtävä onnistuessaan täysin vastaa yhtä pistettä, onnistuessa osittain puolta pistettä ja epäonnistuessa nollaa pistettä. Kyseisellä mittaustavalla tulosten kerääminen on kohtuullisen vaivatonta ja tulokset kertovat kuitenkin käytettävyyden kannalta olennaisimman tiedon.

Nielsen ja Budiu (2012, 10–27) kertovat käytettävyyden merkityksen korostuvan entisestään mobiililaitteissa. Heidän tutkimustensa (50–51) mukaan neljä suurinta haastetta mobiililaitteiden käytettävyydelle ovat pieni näyttökoko, ohjausmenetelmät, latausajat sekä työpöytäjärjestelmille suunnitellut käyttöliittymät. Näistä erityisen merkittävä on näytön koko. He (15–16) luokittelevat erilaiset matkapuhelimet näyttökokonsa perusteella joko peruspuhelimiksi, älypuhelimiksi tai kosketusohjatuiksi älypuhelimiksi. Heidän suorittamat käytettävyysmittaukset (Kuvio 10) kertovat käytettävyyden nousevan dramaattisesti kosketusohjattujen älypuhelinten kohdalla. Uusissa älypuhelimissa kosketusohjaus alkaa olla normi, joten tässä työssä kaikilla viittauksilla älypuhelimiin tarkoitetaan nimenomaan kosketusohjattuja älypuhelimia.



Kuvio 10. Erityyppisten matkapuhelinten käyttäjille (n=124) annettujen arkipäiväisten tehtävien onnistumisosuudet. (Nielsen ja Budiu 2012, 15)

Sormenpäällä tapahtuva kosketusohjaus on luonteeltaan selvästi epätarkempaa kuin erillistä osoitinlaitetta käytettäessä. Mobiililaitteissa ei yleensä myöskään ole fyysistä näppäimistöä tekstinsyöttöä varten, vaan käytössä on erilaisia *virtuaalisia näppäimistöjä*, joilla kirjoittaminen on kuitenkin erittäin työlästä ja keskittymistä vaativaa. Nielsen ja Budiu suosittelevat ruudulla valittavissa olevien kohteiden vähimmäiskooksi yhtä senttimetriä tai 0,4 tuumaa. Usein myös älypuhelinvalmistajat antavat suosituksia interaktiivisten kontrollien vähimmäiskooksi omalla laitteellaan. Esimerkiksi Apple (2012a) suosittelee iPhone-älypuhelimelleen minimikooksi 44 pikseliä, joka vastaa noin 0,27 tuuman fyysistä kokoa. (Nielsen ja Budiu 2012, 76–78)

Mobiililaitteissa käytettävyyttä laskee myös erilaiset latausajat, jotka aiheutuvat laitteen suorituskyvystä sekä Internet-yhteyden hitaudesta. Käyttäjän toiminnoilla tulisi aina olla välitön visuaalinen tai äänellinen indikaatio, jotta käyttäjä tietää toiminnon menneen perille. Mikäli käyttäjä joutuu lisäksi odottamaan toiminnon valmistumista, tulisi käyttäjälle esittää tieto toiminnon edistymisestä alla olevan taulukon (Taulukko 2) mukaisesti. Toimintojen, jotka edellyttävät käyttäjän ajatteluprosessin jatkuvuutta ei kuitenkaan saisi aiheuttaa käyttöliittymässä yli kahden sekunnin viivettä. Toimintokontekstin vaihtuessa hyväksyttävä vasteaika on korkeintaan 15 sekunnin luokkaa, mutta mikäli toiminto asettaa kohtuullisia vaatimuksia käyttäjän lyhytkestoiselle muistille, tulisi viiveen olla alle neljä sekuntia. (Galitz 2007, 593–601)

Viive	**Suositeltu indikaatio**
Alle 10 sekuntia	Yksinkertainen signaali sovelluksen varatusta tilasta. Esimerkiksi tiimalasikursori tai latausteksti.
Yli 10 sekuntia, mutta alle minuutti	Suuri animoitu visuaalinen objekti tai edistymispalkki. Selkeä ilmoitus toiminnon valmistumisesta.
Yli minuutti	Arvio jäljellä olevasta ajasta sekä päivittyvä tieto toiminnon edistymisestä. Toiminnon valmistuminen ilmoitettava merkittävällä visuaalisella muutoksella sekä äänimerkillä.

Taulukko 2. Eripituisille viiveille suositellut käyttöliittymäindikaatiot.
(Galitz 2007, 593–601)

Käytettävyyden kannalta Nielsenin ja Budiun (2012, 17–27) mukaan on parasta aina tarjota käyttäjille erillinen mobiilioptimoitu käyttöliittymä. Mobiilikäyttöliittymässä tulee suunnitella käyttöliittymän ulkonäkö pienelle ruudulle välttäen näyttötilan tuhlausta esimerkiksi ylenpalttiseen määrään käyttöliittymäkromia (52–59). Myös toissijaisen ja haastavan tekstin määrää tulisi rajata, sillä luetunymmärtäminen mobiililaitteilla on merkittävästi suurinäyttöisempiä laitteita haastavampaa (102–110).

3 Käyttöliittymäkehitys ohjelmistoekosysteemeissä

Boschin (2009) mukaan *ohjelmistoekosysteemi* on ohjelmistoliiketoiminnallinen toimintamalli, jossa ohjelmistovalmistaja hakee tuotteelleen lisäarvoa kolmannen osapuolen toimijoiden kautta. Ekosysteemi koostuu joukosta ohjelmistoratkaisuja, jotka mahdollistavat, tukevat ja automatisoivat näiden toimijoiden aktiviteettejä. Keskeisessä roolissa on ekosysteemin *ohjelmistoalusta* (engl. platform), johon näiden toimijoiden kehittämät sovellukset kytkeytyvät. Ekosysteemien tehokkuus perustuu verkostovaikutukseen, jossa saatavilla olevat sovellukset houkuttelevat sovellusalustalle lisää loppukäyttäjiä, jotka puolestaan tuovat lisää sovelluskehittäjiä sovelluksineen (VisionMobile 2011b, 21). Bosch kategorisoi ohjelmistoekosysteemit vielä tarkemmin kolmeen kategoriaan (Taulukko 3), joista tämän työn kannalta erityisen olennaisia ovat käyttöjärjestelmäkeskeiset ohjelmistoekosysteemit, joita tässä luvussa tarkastellaan.

Kategoria	**Ominaispiirteet**	**Esimerkkejä**
Käyttöjärjestelmäkeskeinen	Kolmannet osapuolet tuottavat käyttöjärjestelmään sidottuja sovelluksia, joiden toimialue on vapaa.	Windows, Linux, Mac OS X, Symbian, iOS, Android
Sovelluskeskeinen	Kolmansien osapuolten laajennukset tuovat lisäarvoa ohjelmistoalustana toimivaan sovellukseen. Rajoittunut ohjelmistoalustan toimialueeseen.	Microsoft Office, Facebook, eBay
Loppukäyttäjäkeskeinen	Ohjelmistoalustat, joilla loppukäyttäjät tuottavat itse haluamansa sovelluksen.	Microsoft Excel, Lego Mindstorm

Taulukko 3. Erilaiset ohjelmistoekosysteemien kategoriat. (Bosch 2009)

Älypuhelinten käyttöjärjestelmien ohjelmistoalustoista käytetään usein nimitystä *laitealusta*, joka kuvaa sen tiukempaa sidonnaisuutta juuri tietynlaisiin laitteistoihin sekä sen tarjoamia laajempia mahdollisuuksia hyödyntää laitteiston tarjoamia toimintoja eri sovelluksissa. Kullakin laitealustalla on yleensä omat laitealustan tukemat lisäpalvelut, kehitystyöka-

lut, *sovelluskauppa* sekä tämän työn kannalta merkityksellisesti oma käyttöliittymästandardi ja sen toteuttava *natiivi sovelluskehys*. Laitealusta voi lisäksi tukea vaihtoehtoisia sovelluskehyksiä, jotka voivat myös olla kolmansien osapuolten tuottamia.

Kaikki markkinoilla olevat merkittävät älypuhelinalustat tukevat nykyään kosketusohjausta (Taulukko 4). Vanhemmissa laitealustoissa kosketusohjaus on esitelty uudempien versioiden myötä, mutta usein lopputulos ei ole yhtä saumaton kuin natiivilla kosketustuella saavutettu. Ohjelmistosuunnittelijoiden kannalta jälkikäteen lisätty tuki on myös haastava, koska tällöin osa käyttöliittymäkomponenteista saattaa olla optimoitu kosketustuelle ja osa jollekin muulle ohjausmuodolle, jota käyttävä laitekanta on otettava huomioon sovellusta kehittäessä.

Laitealusta (kehittäjä)	**Sovelluskaupppa**	**Natiivi sovelluskehys**	**Markkinaosuus 2010 Q3**	**Kosketustuki**
Symbian OS (Nokia)	Ovi Store	S60 UI	36,6%	Lisätty versiossa 5th Edition (2008)
Android (Google)	Android Market	Android Application Framework	25,5%	Natiivi (2008)
iOS (Apple)	App Store	Cocoa Touch	16,7%	Natiivi (2007)
BlackBerry OS (RIM)	Blackberry App World	Osana BlackBerry JDE kirjastoa	14,8%	Lisätty versiossa 5.0.0 (2008)
Windows Mobile (Microsoft)	Ei keskitettyä jakelua	.NET Compact Framework	2,8%	Lisätty versiossa 6 (2007)
Windows Phone 7 (Microsoft)	Windows Phone Marketplace	Silverlight	-	Natiivi (2010)

MeeGo (Nokia ja Intel)	Ovi Store	MeeGo Touch Framework	-	Natiivi (2011)
Bada (Samsung)	Samsung Apps	Bada UI Framework	-	Natiivi (2010)

Taulukko 4. Merkittäviä älypuhelinalustoja. (Gartner 2010; VisionMobile 2012a)

Globaalisti merkittävistä älypuhelinalustoista (Taulukko 4) tämän työn ulkopuolelle rajattiin BlackBerry OS ja Bada johtuen niiden markkinoiden keskittymisestä pääasiassa Suomen ja Euroopan ulkopuolelle. Lisäksi Windows Mobile ja Windows Phone 7 ovat teknisesti saman käyttöjärjestelmän eri versioita, joten työssä niitä käsitellään yhdessä ja keskitytään erityisesti uudempaan 7-versioon.

3.1 Sovellusten jakelu älypuhelinten ekosysteemeissä

Apple avasi vuonna 2008 ensimmäisenä suurena valmistajana keskitetyn sovelluskaupan iOS-laitealustalleen. Keskitetyssä jakelumallissa kaikki laitealustalle julkisesti saatavilla olevat sovellukset ostetaan ja ladataan saman, valmistajan kontrolloiman palvelun kautta. Useassa tapauksessa valmistaja pyrkii rajoittamaan tai jopa estämään kokonaan sovellusten asentamisen sovelluskaupan ulkopuolelta. Lisäksi tarjolla voi olla sovellusten lisäksi staattista sisältöä, kuten esimerkiksi teemoja tai taustakuvia. Valmistaja perii yleensä jokaisesta sovellusostoksesta itselleen kiinteän osuuden, esimerkiksi Applen, Googlen ja Nokian tapauksessa 30 % ostetun sovelluksen hinnasta. Loppukäyttäjien kannalta sovelluskaupan tuoma hyöty on sovellusten vaivattomassa löytämisessä sekä valmistajan kontrolloimassa, luotettavammassa maksuprosessissa Keskitetty jakelu estää myös tehokkaasta haittaohjelmien leviämistä, sillä valmistaja voi poistaa sovelluskaupasta haittaohjelmaksi osoittautuneita sovelluksia tai jopa testata sovellukset ennen kauppaan hyväksymistä, kuten esimerkiksi Apple ja Nokia tekevät. (Campbell ja Ahmed 2011)

Sovelluskaupoissa jaettavat sovellukset noudattavat jotain valmistajan määräämää *pakketointimuotoa*, jota sovelluskehittäjän täytyy noudattaa julkaistakseen sovelluksen. Käytetty

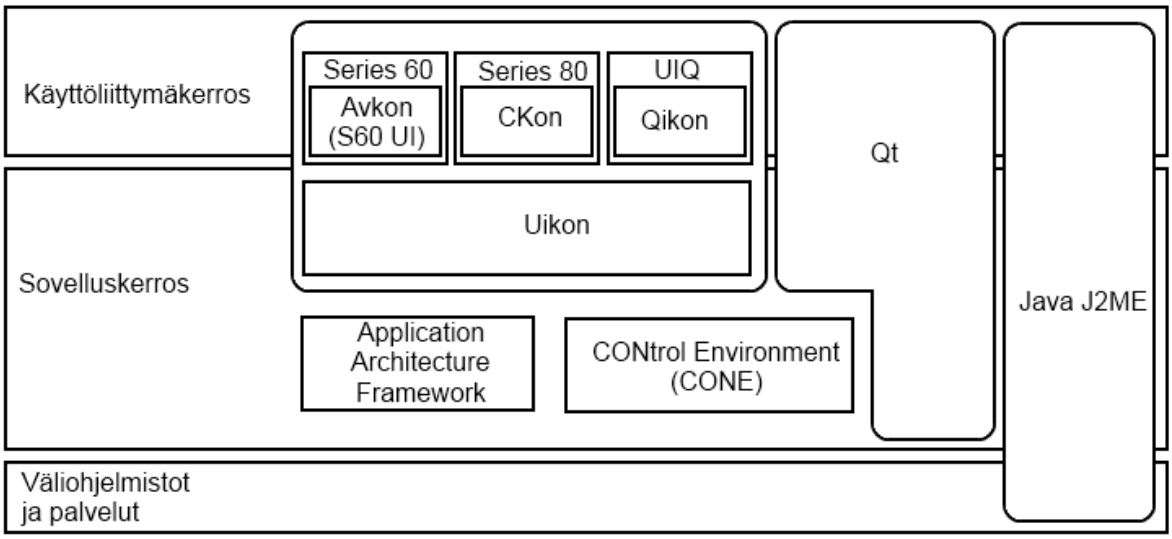
paketointi riippuu kohteena olevan laitealustan lisäksi käytetystä kehitystekniikasta. Sovelluskaupasta riippuen käytössä saattaa olla useita eri paketointimuotoja. Esimerkiksi Nokian sovelluskauppa tukee useita eri laitealustoja ja paketointimuotoja, mukaan lukien Debian-paketointia Maemo/MeeGo-laitealustoille, SIS-paketointia (Software Installation Script) Symbian-laitealustalle sekä useita alustariippumattomia sovellusmuotoja kuten JAR-paketointia (Java Archive) Java-sovelluksilla ja web-sovelluksia, joiden paketoinnista tarkemmin luvussa 4.1.3 (Nokia 2012c).

3.2 Keskeiset älypuhelinten ohjelmistoekosysteemit

3.2.1 Symbian

Symbian-käyttöjärjestelmällä on suurimmista älypuhelinalustoista pisin kehityskaari. Sen perusta on Psionin PDA-laitteisiin kehittämässä EPOC-käyttöjärjestelmässä, jonka 32-bittisen EPOC32-version kehitys aloitettiin jo vuonna 1994. Kesäkuussa 1998 perustettiin Symbian Ltd, yhteishanke, jonka tarkoituksena oli tuoda EPOC-käyttöjärjestelmä älypuhelimiin Symbian-nimellä. Hankkeeseen osallistuivat alun perin Psionin lisäksi Nokia, Ericsson ja Motorola. (Fitzek, Torp, ja Mikkonen 2010, 4–5; Orlowski 2010)

Symbian-käyttöjärjestelmä on käyttänyt modulaarista käyttöliittymäkirjastoa jo PDA-aikakautenaan. Tällöin käytössä oli Psionin kehittämä alkeellinen Eikon-käyttöliittymäkirjasto (Telcontar.net 2008), jonka Unicode-merkistöä tukeva versio tunnetaan nimellä Uikon. Uikon-käyttöliittymäkirjaston päälle tehtiin useita erilaisiin laiteympäristöihin tarkoitettuja sovituksia, joista merkittävimmät ovat osoitinkynällä ohjattuihin kosketusnäyttölaitteisiin kehitetty Qikon, muutamissa Nokian kommunikaattorimalleissa käytetty CKon sekä alun perin numeronäppämistöohjattuihin laitteisiin suunnattu Avkon (Kuvio 11). Qikon-kirjaston päälle on toteutettu nykyisin käytöstä poistunut Symbianin UIQ-ohjelmistoalusta, CKon-kirjaston päälle Series 80 ja vastaavasti Avkon-kirjaston päällä toimii Nokian edelleen käyttämä Series 60 -sovellusalusta, jonka uudemmat versiot kulkevat lyhennetyllä S60-nimellä. (Nokia Developer Wiki 2011b)



Kuvio 11. Symbian-käyttöjärjestelmän arkkitehtuuri kerroksittain. (Nokia Developer 2008a; Nokia Developer 2008b)

Symbian-käyttöjärjestelmän natiivi ohjelmointikieli on Symbian C++, joka perustuu suositun C++ -ohjelmointikielen varhaiseen kehitysversioon (Nokia 2004). Symbian C++ otettiin käyttöön jo ennen ANSI C++ -standardin valmistumista ja tästä syystä siihen on tehty hyvin ainutlaatuisia ratkaisuja muun muassa muistinhallinnan ja poikkeustenkäsittelyn osalta. Omaperäisen ohjelmointikielen ja historian painolastin vuoksi Symbian on oppimiskynnykseltään yksi vaikeimmista ohjelmistoalustoista (VisionMobile 2011a, 44).

Nokian S60-sovellusalusta sai tuen kosketusnäytöille vuonna 2008 julkaistussa 5.0 versiossa (Nokia 2008), joka tunnetaan myös nimillä S60 5th Edition sekä Symbian^1. Vuotta myöhemmin muodostettiin Symbian Foundation -säätiö, jonka oli tarkoitus muuttaa Symbian-alusta avoimen lähdekoodin projektiksi, johon olisi sisältynyt myös S60-sovelluskirjastot. Säätiön toiminta ilmoitettiin kuitenkin ajettavan alas jo vuonna 2010 ja nykyään se on enää vastuussa Symbian-alustan lisensoinnista (Symbian Foundation 2012).

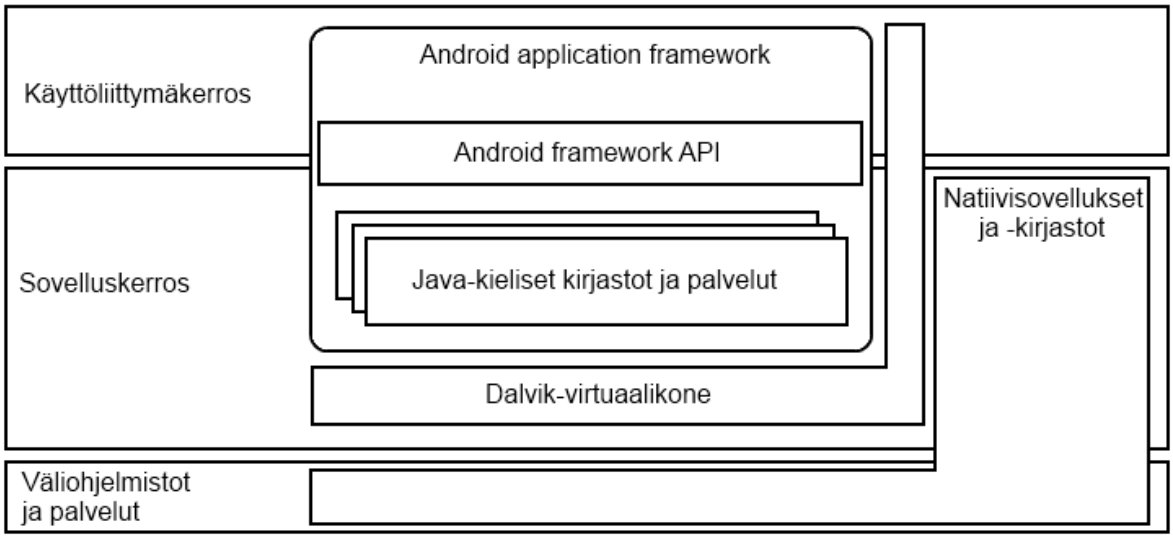
Symbian tukee nykyään Symbian C++ -kielisten S60-sovellusten lisäksi virallisesti myös Qt-sovelluksia. Tuki on saatavilla Symbian^1 -ohjelmistoalustaa käyttäviin älypuhelimiin erikseen asennettavan sovelluspaketin myötä, mutta Symbian^3 ja sitä uudemmat versiot sisältävät sille natiivin tuen. Nokia suosittelee Qt-sovelluskehitystä ensisijaisena sovelluskehitysmuotona Symbian^3 -alustalle. (Qt Developer Network 2012b)

3.2.2 Google Android

Android on Linux-käyttöjärjestelmän päälle rakennettu avoimen lähdekoodin mobiililaitealusta, jonka kehityksestä vastaa Google ja yli 50 teknologia-alan yrityksestä koostuva Open Handset Alliance -järjestö. Ensimmäinen Androidia käyttävä älypuhelin tuli markkinoille 2008 ja vuoden 2010 loppuun mennessä siitä on tullut eniten myyvä älypuhelinalusta. (Meier 2010, 1–9)

Google ei itse valmista Android-laitteita, vaan keskittyy pelkästään alustan kehittämiseen. Periaatteessa kuka tahansa valmistaja voi hyödyntää Androidia ilmaiseksi laitteissaan, mutta Android-tavaramerkin sekä joidenkin suljettujen palveluiden, kuten Android Market -sovelluskaupan, käyttäminen edellyttää kuitenkin Googlen sertifiointia (Android Open Source Project 2012). Androidia älypuhelimissaan käyttävät muun muassa Samsung, HTC, Huawei, LG, Motorola ja Sony Ericsson. Elokuussa 2011 Google osti itselleen Motorolan mobiilitoiminnot. Kaupan tärkeimmäksi syyksi on arvioitu Motorolan mittavaa patenttisalkkua, mutta kaupan myötä muiden puhelinvalmistajien asema saattaa tulevaisuudessa heikentyä (Laitila 2011).

Sovelluskehitys. Androidin virallinen ohjelmointikieli on Java. Sillä tehtyjä sovelluksia ei kuitenkaan ajeta perinteisessä Java-virtuaalikoneessa, vaan kutakin Androidin sovellusprosessia pyörittää oma instanssi Androidia varten kehitetyssä virtuaalikoneessa nimeltään Dalvik. Dalvikin ymmärtämä tavukoodi on erilaista kuin Java-virtuaalikoneen, eivätkä näiden sovellukset siten ole keskenään yhteensopivia. Android sisältää kuitenkin omien ohjelmistokirjastojensa lisäksi oman toteutuksensa myös Javan keskeisimmistä kirjastoista (Kuvio 12). Android-sovellukset jaellaan APK-paketointimuodossa. (Meier 2010, 13–15)



Kuvio 12. Android-käyttöjärjestelmän arkkitehtuuri kerroksittain. (Android Developers 2011a)

Android-käyttöjärjestelmän sovelluskehitysympäristö (SDK) on saatavilla Windows-, Mac- ja Linux-alustoille. Sen lisäksi sovelluskehitykseen tarvitaan Java Development Kit (JDK) versio 5 tai 6. Käytettäväksi kehitystyökaluksi suositellaan Eclipse-sovellusta ja Android tarjoaakin siihen oman liitännäisen nimeltään Android Developer Tool (ADT). (Meier 2010, 19–21)

Javan lisäksi Android-käyttöjärjestelmälle on mahdollista tehdä sovelluksia ja erityisesti sovelluskirjastoja myös C/C++-kielellä. Android tarjoaa tähän tarkoitukseen valmiin ohjelmistokehitysympäristön jota kutsutaan NDK:ksi (Native Development Kit). NDK:n käyttäminen on suositeltua lähinnä ajoituskriittisten toimintojen sekä 3D-grafiikan ohjaamiseen, eikä se esimerkiksi tarjoa ollenkaan valmista käyttöliittymäkirjastoa. (Android Developers 2011b)

Versiohistoria. Android-käyttöjärjestelmän kehitysnopeus on ollut poikkeuksellisen nopea. Google on julkistanut useita uusia versioita Androidista vuosittain. Usein laitteet on myös mahdollista päivittää käyttämään uudempaa alustaversiota (Talk Android 2011). Rajapintatasolla versiot ovat aina taaksepäin yhteensopivia, mutta käytännössä jokaisessa uudessa versiossa rajapintoihin on tullut lisäyksiä. Sovelluskehittäjät voivatkin määrittää pienimmän vaaditun käyttöjärjestelmäversion toteuttamalleen sovellukselle valmiiksi mää-

riteltyjen API-tasojen avulla (Android Developers 2012b). API-tasot ovat numeroitu yhdestä alkavalla kokonaislukusarjalla, jota nostetaan aina kun rajapinnassa tapahtuu muutoksia. Google (Android Developers 2012e) mukaan tämän kappaleen kirjoitushetkellä, 4.9.2012, yli 57,2 % käytössä olevasta laitekannasta käyttää API-tasoa 10 vastaavia Androidin versioita 2.3.3–2.3.7.

Käyttöliittymäkehitys. Androidin käyttöliittymällisten sovellusten kehitys perustuu aktiviteetteihin, jotka vastaavan jotakuinkin muiden alustojen ikkunoita tai näkymiä. Sovelluksella voi olla monta aktiviteettia ja Androidin sovelluskirjasto pitää huolen niiden pinoutumisesta. Kunkin aktiviteetin voi sallia kutsuttavaksi erikseen toisten prosessien toimesta. Aktiviteetti näyttää myös käyttöliittymän ja käsittelee interaktion käyttäjän kanssa. (Android Developers 2012a)

Käyttöliittymän sisältö voidaan ladata erillisestä XML-kielisestä tiedostosta. Käyttöliittymän kontrolleita Androidissa kutsutaan näkymiksi (engl. View). Näkymät puolestaan asemoidaan keskenään käyttäen näkymäryhmiä (engl. ViewGroup), kuten esimerkiksi vertikaalisen lineaarisen asemoinnin ryhmää. (Android Developers 2012g)

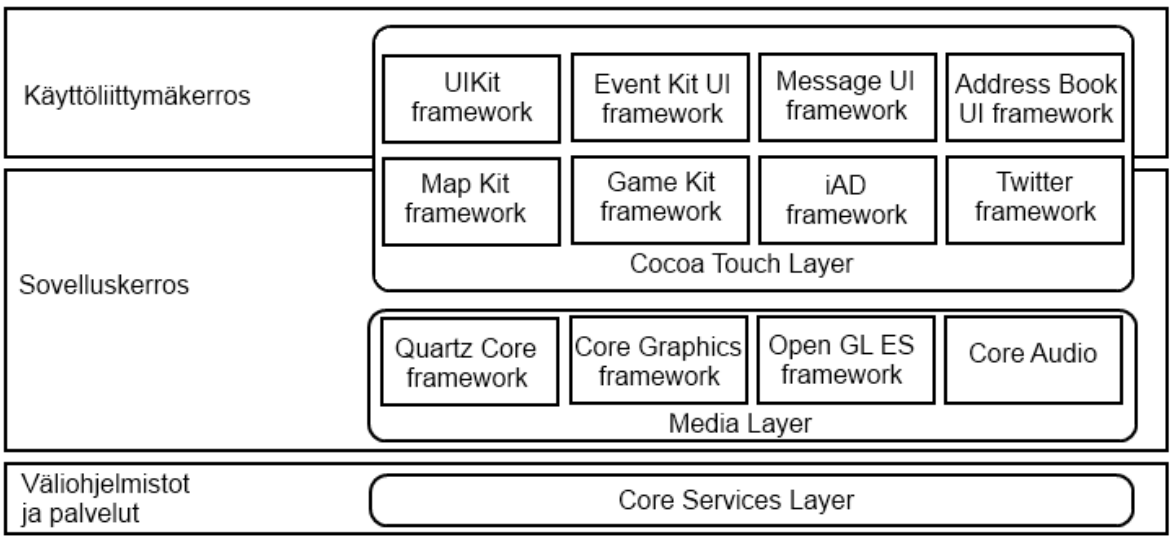
3.2.3 Apple iOS

Apple esitteli vallankumouksellisen iPhone-älypuhelimensa (Kuvio 1) ensimmäisen version vuoden 2007 tammikuussa MacWorld -tapahtumassa (VisionMobile 2011b). Siihen aikaan tyypilliset älypuhelimet ja mobiililaitteet käyttivät vielä osoitinkynällä ohjattua resistiivistä kosketusnäyttöä tai qwerty-/numeronäppäimistöä. Esitellyn iPhonen mullistavin ominaisuus oli minimalistinen käyttöliittymä ilman perinteistä näppäimiä tai hankalaksi koettua osoitinkynäohjausta, joiden sijaan sen ohjaus perustui kapasitiiviseen monikosketustekniikkaan ja innovatiivisiin kosketuseleisiin. Laitteen etupuolella oli suuren 3,5” näyttöelementin lisäksi poikkeuksellisesti vain yksi fyysinen nappi, jonka avulla käyttäjä pystyi palaamaan aktiivisesta sovelluksesta takaisin kotinäyttöön. Puhelimen ensimmäisessä versiossa oli myös paljon puuttuvia ominaisuuksia, kuten 3G-yhteydet ja MMS-viestit, mutta tästä huolimatta se oli menestys ja pakotti muut valmistajat miettimään uudelleen omaa älypuhelinmallistoaan.

Toinen iPhonen tuoma vallankumous liittyi tapaan jaella kolmansien osapuolten sovelluksia. Heinäkuussa 2008 Apple avasi toisen sukupolven iPhone 3G -laitteiden julkaisun yhteydessä App Store -sovelluskaupan, joka on keskitetty, Applen tiukasti kontrolloima sovellusten ja lisäsisällön lataus- ja kauppapalvelu. Sovelluskauppaa pystyi käyttämään myös ensimmäisen sukupolven iPhone-älypuhelimilla, kunhan siihen asensi maksullisen ohjelmistopäivityksen. Heinäkuussa 2011 Apple ilmoitti että App Store -sovelluskaupasta on tehty yhteensä 15 miljardia latausta ensimmäisen kolmen vuoden aikana (Apple 2011a).

iPhone käyttää alustanaan iOS-käyttöjärjestelmää, joka perustuu Applen Mac OS X -työpöytäkäyttöjärjestelmään. Applen iPhone-älypuhelinten lisäksi sitä käytetään myös iPad-tableteissa ja iPod Touch -musiikkisoittimissa. Alustan natiivi ohjelmistokehityskieli on Objective-C, joka perustuu C-kieleen ja tukee olio-ohjelmointia. (VisionMobile 2011b)

Cocoa Touch UIKit. Käyttöliittymäkehityksessä iOS-alustalle käytetään Mac OS X Cocoa API -sovelluskehityskirjastoon perustuvaa Cocoa Touch -sovelluskirjastoa ja erityisesti sen MVC-pohjaista UIKit-kontrollikirjastoa (Kuvio 13). Käyttöliittymän koordinaatistona käytetään pikseleiden sijaan pisteitä, joiden avulla saavutetaan sama ulkonäkö laitteilla joiden resoluutio eroaa toisistaan. Kaikissa iPhone- ja iPod Touch -laitteissa ruudun ulottuvuudet ovat 320x480 pistettä ja iPad taulutietokoneessa 768x1024 pistettä. (Apple 2011d)



Kuvio 13. iOS-käyttöjärjestelmän arkkitehtuuri kerroksittain. (Apple 2011c).

3.2.4 Windows Phone 7

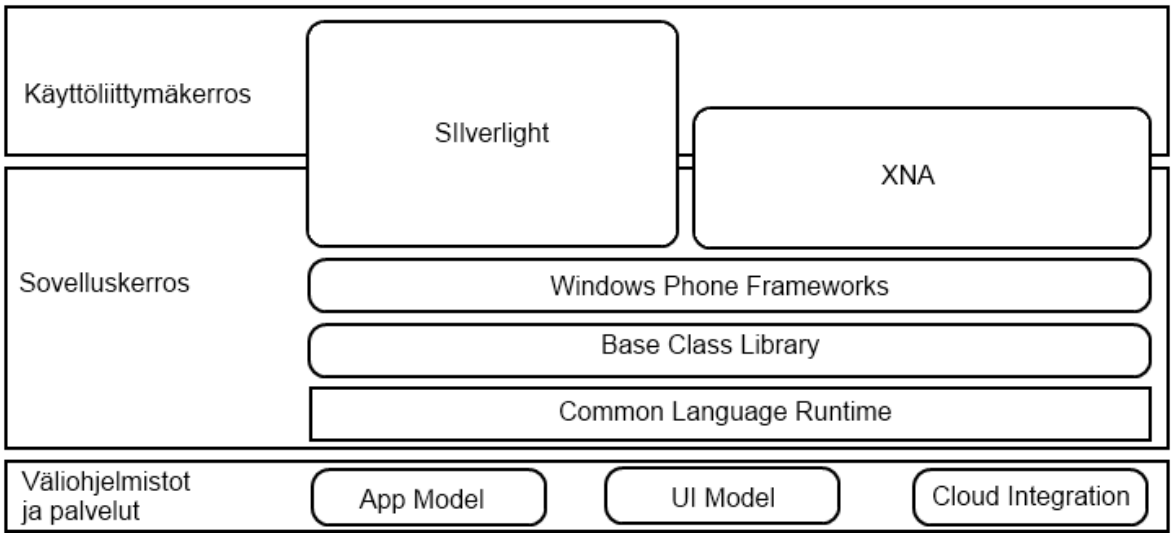
Microsoft on jo pitkään pyrkinyt laajentamaan toimintaansa erilaisiin mobiililaitteisiin. Ensimmäinen yritys tapahtui vuonna 1996, kun Microsoft julkaisi erilaisille sulautetuille järjestelmille suunnitellun Windows CE -käyttöjärjestelmän, joka myöhemmin nimettiin uudelleen Windows Embedded Compact -käyttöjärjestelmäksi. Kilpaillakseen silloisten kämmentietokoneiden markkinoilla, Microsoft kehitti Windows CE -järjestelmän pohjalta osoitinkynällä ohjatun Pocket PC -laitealustan, jonka ensimmäinen versio julkaistiin vuonna 2000. (Thurrott 2010, 65–69)

Vuonna 2003 Pocket PC -laitealusta lanseerattiin uudelleen Windows Mobile -nimellä. Windows Mobile -alustan kuudetta pääversiota edeltävät iteraatiot jakaantuivat erillisiin Pocket PC Phone- ja Windows Mobile Smartphone -tuoteperheisiin, joista ensin mainitut olivat monipuolisia, osoitinkynällä ohjattuja kämmentietokoneita. Älypuhelinten tuoteperhe puolestaan koostui ominaisuuksiltaan rajusti karsituista näppäinohjatuista laitteista. Käyttöliittymältään Windows Mobile -laitteet muistuttivat aikaisempia Pocket PC -laitteita. (Pocket PC Central 2011)

Kilpaillakseen modernien kosketusohjattujen älypuhelinten kanssa Microsoft uudisti radikaalisti älypuhelinalustansa vuoden 2010 lopulla julkaistun Windows Phone 7 -version myötä. Sitä varten kehitettiin käyttöliittymäkonsepti, joka tunnetaan nimellä Metro. Sen keskeinen ajatus on keskittyä näkymissä keskeiseen sisältöön ja minimoida kaikki muu. Tyylissä painottuu erityisesti typografinen sisältö erilaisten ikonien ja käyttöliittymäkromin sijaan. Ehkäpä Metron ainutlaatuisin ominaisuus on sen käyttämät panoraamanäkymät, joissa toisiinsa liittyvä tieto on järjestetty vaakatasossa pyyhkäisyn avulla navigoitaviin alinäkymiin (Kuvio 14). Alustan sovellukset jaellaan keskitetysti Windows Phone Store -sovelluskaupan kautta. (Kruzeniski 2011)

Kuvio 14. Metron panoraamanäkymän konseptikuva. (Microsoft 2010, 164)

Windows Phone 7 ei ole taaksepäin yhteensopiva vanhempien alustaversioiden sovellusten kanssa. Sovelluskehityksen natiivina ympäristönä siinä käytetään Microsoftin Silverlight-kehitysympäristön älypuhelimille sovitettua versiota. Vaihtoehtoisesti on mahdollista käyttää pelikehitykseen suunniteltua XNA-sovelluskehystä. Ohjelmointikielenä molemmissa lähestymistavoissa käytetään pääasiassa C#-kieltä, jonka lisäksi Silverlight-sovellusten käyttöliittymä voidaan määrittää XML-pohjaisella XAML-kuvauskielellä. Tuotettu koodi käännetään ensin alustariippumattomaan muotoon, joka käännetään arkkitehtuurikohtaiseen binäärimuotoon vasta kohdealustan CLR-ajoympäristössä (Kuvio 15). (Microsoft 2011)

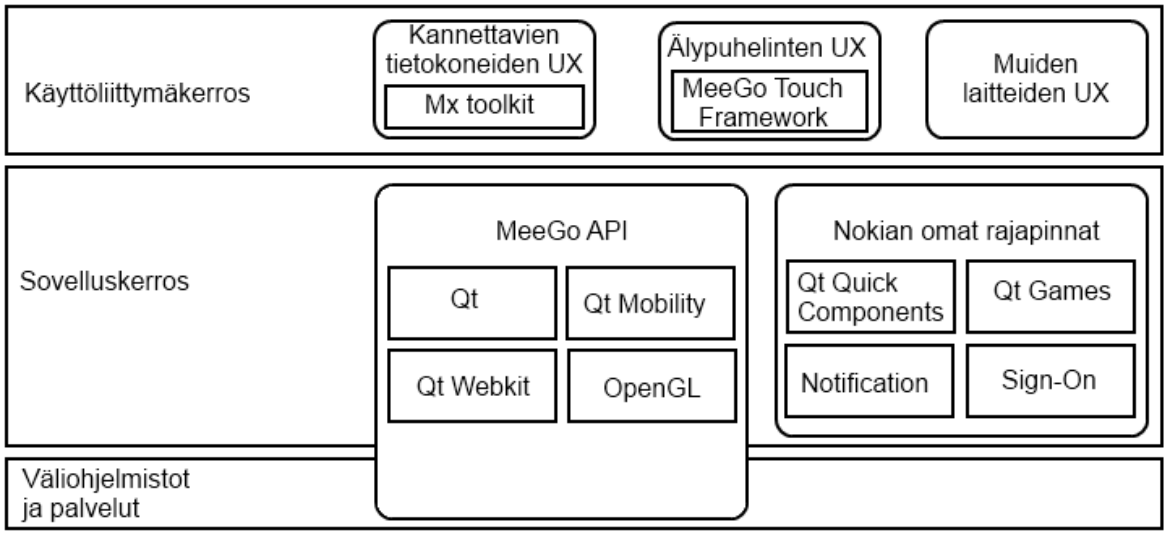


Kuvio 15. Windows Phone 7 kerrosarkkitehtuuri. (Microsoft 2011)

3.2.5 MeeGo

Linux-pohjainen MeeGo-käyttöjärjestelmä on tuore tulokas kasvaville älypuhelinmarkkinoille. Sen taustalla on Nokian kehittämä Maemo-alusta sekä Intelin kehittämä Moblin, joiden ilmoitettiin yhdistyvän helmikuussa 2010 Mobile World Congress -tapahtumassa. Älypuhelinten lisäksi MeeGo on suunnattu myös monille muille laitteille, kuten minikannettaville, tableteille ja älytelevisioille. (Sousou 2010)

Arkkitehtuurisesti MeeGo on moderni, selkeisiin kerroksiin jaettu laitealusta, jota kehitetään pääasiassa avoimen lähdekoodin kehitysmenetelmällä. Merkittävä osa sen lähdekoodeista on vapaasti saatavilla osoitteessa http://meego.gitorious.org/. Tämän tutkielman kannalta erityisen kiinnostavaksi MeeGon tekee sen modulaarinen rakenne (Kuvio 16), jonka ansiosta esimerkiksi älypuhelin- ja tablettisovituksissa voidaan käyttää eri sovelluskehyksiä. MeeGon sovelluskehykset ovatkin erittäin hyvin rajattuja kokonaisuuksia, jotka keskittyvät lähinnä käyttöliittymän kannalta oleellisiin asioihin.



Kuvio 16. MeeGo alustan arkkitehtuuri kerroksittain. (Nokia Developer Wiki 2012; Saxena 2010)

MeeGo Touch Framework. MeeGo-alustan ensisijainen älypuhelinsovituksen sovelluskehys on Nokian kehittämä Qt-pohjainen MeeGo Touch Framework. Toisin kuin pääasiassa hiiriohjattuun työpöytäympäristöön suunnattu Qt:n oma käyttöliittymäkirjasto Qt GUI (josta tarkemmin luvussa 4.2.1), MeeGo Touch on suunniteltu kosketuksella ohjattavaksi.

MeeGo Touch tarjoaa sovelluksille mm. kosketusohjaukselle optimoidut kontrollit, animoidun kontrollien asemoinnin, kosketuseleet ja tuen monikosketukselle. Siinä missä Qt:n listat, gridit ja muut monimuotoisemmat käyttöliittymäkontrollit perustuvat typistettyyn model/view-arkkitehtuuriin (Qt Reference Documentation 2012a), MeeGo Touch -kirjasto käyttää perinteistä model-view-controller-arkkitehtuuria kaikissa kontrollereissaan. MeeGo Touch -kirjaston kontrollit ovat yhteensopivia Qt:n kontrollien kanssa ja näitä voidaan käyttää sekaisin sovelluksessa tarpeen mukaan. (Nokia 2010c)

Sovelluskehyksille poikkeuksellisesti MeeGo Touch -kirjaston lähdekoodi on täysin avointa ja käännettävissä sellaisenaan ainakin Linux-, Windows ja Mac OS X - työpöytäkäyttöjärjestelmille. Lähdekoodien mukana toimitetaan myös kontrolleja esittelevä *galleria*-sovellus. (Nokia 2010a)

Laitteet ja tulevaisuus. Helmikuussa 2011, ennen kuin yhtään MeeGo-pohjaista älypuhelinta oli vielä kaupoissa, Nokia ilmoitti strategisesta yhteistyöstä Microsoftin kanssa. Yh-

teistyön myötä Nokian suunnitelmat käyttää MeeGoa ensisijaisena älypuhelinten käyttöjärjestelmänä peruuntuivat ja MeeGon sijaan Nokia päätti käyttää Windows Phone -käyttöjärjestelmää tulevissa lippulaivatuotteissaan. Samassa yhteydessä Nokia ilmoitti kuitenkin tuovansa ainakin yhden MeeGo-älypuhelimen markkinoille. (BBC News 2011)

Nokian ensimmäinen ja toistaiseksi ainoa MeeGo-älypuhelin, N9 (Kuvio 17), julkaistiin Nokia Connection tapahtumassa Singaporessa kesäkuussa 2011. Sen käyttämästä MeeGo-sovituksesta käytetään koodinimeä Harmattan. N9:n käyttöliittymätoteutus oli julkaisun aikaan uniikki lajissaan, sillä puhelimen etupuolella ei ole yhtään fyysistä nappia, vaan koko käyttökokemus perustuu kosketusohjaukseen. Käyttöliittymän siirtymät eri näkymien kotinäkymien ja avointen ohjelmien välillä on toteutettu perinteisen paluu- tai kotipainikkeen sijaan sormieleillä, joissa avoin ikkuna pyyhkäistään pois ruudulta. (Lemmetyinen 2011)

Kuvio 17. MeeGo-käyttöjärjestelmää käyttävä Nokia N9 älypuhelin.
(Lemmetyinen 2011)

Intelin ja Nokian yhteistyön päättymisen seurauksena syyskuussa 2011 Intel ja Samsung julkistivat uuden suunnitelman kehittää yhteistyössä MeeGolle seuraajan nimeltään Tizen (Linux Foundation 2011). Tizenin ensisijainen sovelluskehitysmuoto on web-sovellukset ja se sisältää oman Tizen Web UI -käyttöliittymäkirjaston, johon kuuluu osana myös muilla alustoilla käytettävissä oleva jQuery Mobile (Linux Foundation 2012), jota käsitellään tarkemmin luvussa 4.2.3.

4 Alustariippumattomat ratkaisut

Tässä luvussa tarkastellaan merkittävimpiä älypuhelimille suunnattuja alustariippumattomia tekniikoita. Sovelluskehittäjät luonnollisesti haluavat mahdollisimman laajan levikin tuotteelleen, joka edellyttää sovelluksen julkaisemista useammalle eri laitealustalle. Älypuhelinvalmistajat pyrkivät kuitenkin lukitsemaan sovelluskehittäjät omien kehitystyökalujensa käyttäjäksi, eivätkä esimerkiksi valmistajien omat sovelluskaupat yleensä kelpuuta muita kuin natiivisovelluksia tarjontaansa. Joissain tapauksissa valmistajan sovelluskauppa on myös loppukäyttäjän ainoa mahdollinen tapa asentaa uusia sovelluksia.

Erillisen tuotteen toteutus ja ylläpitäminen kullekin halutulle laitealustalle syö paljon resursseja ja rahaa, jonka vuoksi alustariippumattomien sovellusten tuottamiseen on kehitetty todella mittava joukko erilaisia tekniikoita ja työkaluja (VisionMobile 2012a). Tekniseltä lähestymistavaltaan eri työkalut voidaan karkeasti jakaa kahteen pääryhmään: erillistä sovelluskehystä käyttäviin sovelluksiin sekä virtuaalisovelluksiin. Näiden rinnalle voi myös nostaa kolmanneksi kategoriaksi web-sovellukset (Luku 4.1), jotka ovat teknisesti virtuaalisovellusten erikoistapauksia, mutta niissä käytettyjä tekniikoita voi hyödyntää myös natiivisovelluksissa.

Teknisesti lähimmäs natiivisovelluksia päästään toteuttamalla alustariippumaton sovellus käyttäen kohdealustan yksityiskohdat abstraktoivaa sovelluskehystä. Käyttöliittymän osalta tällöin käytetään virtuaalista käyttöliittymäkirjastoa, joka saattaa myös tukea natiivia käyttöliittymästandardia. Alustariippumatonta sovelluskehystä käyttävä sovellus käännetään ja paketoidaan erikseen kullekin kohdealustalle, jolloin saadaan kullekin alustalle natiivi asennuspaketti, jota voidaan jaella alustan sovelluskaupan kautta samaan tapaan kuin natiivisovelluksia. Tällaisia tekniikoita ovat mm. Qt (Luku 4.2.1) ja Marmalade (Luku 4.2.4).

Vaihtoehtoista lähestymistapaa edustavat erilaiset virtuaaliympäristöt, joille sovellus voidaan suunnitella. Tällöin sovellus suoritetaan virtuaalisessa ajoympäristössä, joka on usein suorituskyvyltään ja ominaisuuksiltaan rajoittuneempi kuin natiivi suoritusympäristö. Etuna on puolestaan se, että sovellussuunnittelijan ei yleensä tarvitse nähdä ylimääräistä vaivaa sovelluksen kääntämiseksi eri alustoille eikä edes asentaa kohdealustojen kehitystyö-

kaluja. Perinteisesti tätä lähestymistapaa ovat edustaneet erityisesti Java ME ja Adobe Flash Lite, jotka olivat vielä vuonna 2005 merkittävimpiä työkaluja alustariippumattomuuden toteuttamisessa (VisionMobile 2012a). Molemmat näistä työkaluista käyttävät oman ajoympäristönsä lisäksi omaa paketointimuotoaan. Sovelluskauppojen mukanaan tuoman suljetun sovellusjakelun myötä tuki näille tekniikoille on kuitenkin käytännössä kadonnut uusista laitteista, eikä niihin tutustuta tässä työssä tarkemmin.

4.1 Paikallisesti asennettavat web-sovellukset

Vaikka web-sovelluksilla luonnostaan on natiivisovelluksia huonompi suorituskyky ja sellaisenaan älypuhelinten pienille näytöille huonosti soveltuva käyttöliittymä (Morgan Stanley 2009, 162–163), web-sovelluksista on nopeasti tullut käytetyin tapa toteuttaa alustariippumattomia sovelluksia, koska web-sovelluksilla saavutetaan laajin potentiaalinen levikki. Käytännössä katsoen kaikissa markkinoilla olevissa mobiililaitealustoissa löytyy valmiina jokin selainmoottori, usein vieläpä samaan webkit-selainmoottoriin pohjautuva toteutus. Perinteisen selainkäytön lisäksi web-sovelluksen voi jaella loppukäyttäjille lokaaliasennuksena käyttäen kohdealustalla pyörivää web-sovellusalustaa tai paketoimalla se *hybridisovellukseksi* (Luku 4.2.2). (VisionMobile 2012a, 95–96)

4.1.1 Web-sovellusalustojen lyhyt historia

2000-luvulta alkaen on julkaistu joukko web-sovellusympäristöinä tunnettuja ohjelmistoratkaisuja, joilla web-sovelluksia on mahdollista ajaa paikallisena sovelluksena, myös ilman aktiivista verkkoyhteyttä. Tällöin web-sovellusta ajetaan normaalisti selainmoottorin päällä, mutta sen käyttöliittymä näytetään sille rajatulla itsenäisellä alueella ilman selaimesta tuttuja navigointitoimintoja. Tällainen web-sovellus toimitetaan asennuspaketissa, joka sisältää kaikki sovelluksen tärkeimmät resurssit.

Paikallisten web-sovellusten kehitys sai alkunsa työpöydälle integroituvista pienoissovelluksista. Ensimmäinen tällainen sovellusalusta oli 2003 Mac OS X -käyttöjärjestelmälle julkaistu Konfabulator, joka tunnetaan nykyään nimellä Yahoo! Widgets. Se kuitenkin alun perin erosi teknisesti nykyisistä web-sovellusalustoista käyttäen sovelluksissaan XML-

rakennetta HTML-kuvauskielen sijaan. HTML-tuki siihen lisättiin vuonna 2007. (Software Informer 2009)

Nykyisessä muodossaan web-sovellukset työpöydälle toi vuonna 2005 Mac OS X -käyttöjärjestelmän version 10.4 mukana tuleva Dashboard-näkymä (Kuvio 18). Vuonna 2006 Google Desktop -ohjelmistoon lisättiin vastaava tuki Google Gadget -web-sovelluksille ja vain vuotta myöhemmin julkaistu Windows Vista sisälsi sisäänrakennetun tuen sivupalkissa pyöriville web-sovelluksille. Loppuvuodesta 2012 ilmestynyt Windows 8 puolestaan tukee myös kokoruututilan web-sovelluksia ja tarjoaa sovelluskehittäjille pääsyn kattaviin Windows Runtime- ja Windows Library for JavaScript -rajapintakirjastoihin (Microsoft 2012).

Kuvio 18. Mac OS X ja Dashboard-näkymä. (Apple 2008)

4.1.2 Web-sovellusalustat älypuhelimissa

Älypuhelinten nopean kehityksen myötä paikallisesti asennettavat web-sovellukset ovat rantautuneet myös useille älypuhelinalustoille joko suoraan alustan tukemana tai erikseen asennettavan kolmannen osapuolen tarjoaman sovellusympäristön avulla. Erikseen asennettavissa ympäristöissä web-sovellukset ajetaan yleensä erillisen hallintasovelluksen kautta, kuten esimerkiksi S60- ja Windows Mobile -alustoille saatava Opera Widgets Manager -sovellus tekee (Tømmerholt 2010).

Ensimmäisenä älypuhelinvalmistajista sisäänrakennetun tuen paikallisesti asennettaville web-sovelluksille toi Nokia Symbian S60 -laitteilleen. Jo osa vuonna 2006 markkinoille tulleista S60 3rd Edition Feature Pack 1 -version sovellusalustaa käyttävistä älypuhelimista sisälsi Web Runtime 1.0 -teknologiana tunnetun tuen paikallisille, omassa sovellusikkunassaan pyöriville web-sovelluksille. Feature Pack 2 -version myötä tuki ulotettiin kaikkiin sitä käyttäviin malleihin. Asennetut web-sovellukset integroituvat suoraan laitteen sovellusvalikkoon, jonka lisäksi sovellus voidaan myös näyttää pienennetyssä tilassa laitteen kotinäytössä (Kuvio 19). Teknisesti toteutus perustuu Applen Dashboard-näkymään. (Nokia Developer 2011a)

Kuvio 19. S60 5th Edition kotinäyttötila 312 x 82 pikselin kokoon pienennettyjen web-sovellusten tuella. (Forum Nokia 2011)

4.1.3 Web-sovellusten paketointi

Teoriassa kerran kirjoitettu W3C:n web-sisällön standardeja noudattava web-sovellus toimii sellaisenaan kaikissa standardeja noudattavissa web-sovellusympäristöissä. Hieman yllättäen paikallisesti asennettavien web-sovellusten yhteensopivuuden suurimmaksi esteeksi on muodostunut niiden paketointi. Yhteistä kaikille merkittäville web-sovellusympäristöille (Taulukko 5) on sovelluksen pakkaaminen yksittäiseen tiedostoon käyttäen Zip-pakkausmuotoa. Tyypillisesti tällainen paketti sisältää konfiguraatiotiedoston, joka sisältää web-sovelluksen meta-datan sekä mm. määrittää HTML-dokumentin joka näytetään sovelluksen käynnistyksen yhteydessä. Valitettavasti konfiguraatiotiedostot eivät ole yhteensopivia keskenään ja vaikka paketti sisältäisi useamman konfiguraatiotiedoston, pakettien MIME-tyypit ja tiedostopäätteet ovat erilaisia. (W3C 2008)

Web-sovellusalusta	**Tiedosto-pääte**	**MIME-tyyppi**	**Konfiguraatio-tiedosto**
Konfabulator	.widget	application/vnd.yahoo.widget	widget.xml
Windows Sidebar	.gadget	application/x-windows-gadget	gadget.xml
Goodle Desktop	.gg	app/gg	gadget.gmanifest
Opera Widgets	.wgt	application/x-opera-widgets	config.xml
Apple Dashboard	.wdgt tai .zip	application/x-macbinary	Info.plist
Nokia Web Runtime	.wgz	application/x-nokia-widgets	Info.plist
W3C:n suositus	.wgt	application/widget	config.xml

Taulukko 5. Merkittävimpien web-sovellusalustojen paketointien eroavaisuudet. (W3C 2008; Tømmerholt 2010)

W3C on jo vuodesta 2006 alkaen valmistellut yhteistä paketointistandardia ratkaisemaan paikallisesti asennettavien web-sovellusten yhteensopivuusongelmat. Lähtökohtana on yhdistää olemassa olevat käytännöt mahdollisimman taaksepäin yhteensopivasti. Nähtä-

väksi jää millä aikataululla nykyiset sovellusalustojen ylläpitäjät ottavat käyttöön W3C:n määrittelemät yhteiset pelisäännöt. Opera tukee jo nykyään myös W3C:n standardin mukaan paketoituja web-sovelluksia, sillä sen oma paketointi on ollut hyvin läheistä sukua W3C:n formaatille.

4.1.4 Rajapintalaajennukset

Web-sovellusalustat tarjoavat yleensä perinteisten W3C:n web-standardien lisäksi omia JavaScript-rajapintoja, joiden avulla sovelluskehittäjät voivat tuottaa ominaisuuksiltaan paremmin natiivisovelluksia vastaavia web-sovelluksia. Esimerkiksi Nokian Symbian-laitealustassa käyttämä web-sovellusalusta tarjoaa tuen natiivien sovellusvalikoiden luomiselle, softkey-kustomoinnille, näytön rotaation hallinnalle, sisällön lokalisoinnille sekä mahdollisuuden tallentaa sovelluskohtaista dataa pysyvään muistiin (Nokia Developer 2011c).

W3C on aloittanut standardointityön myös rajapintalaajennusten suhteen. Pisimmälle edennyt *Widget Interface* (W3C 2012) -standardiluonnos kuvailee W3C-yhteensopivien web-sovellusympäristöjen tarjoaman rajapinnan, jonka kautta web-sovellukset pääsevät käsiksi sovelluksen paketointitietoihin ja web-näkymän dimensioihin, sekä voivat lukea ja kirjoittaa sovelluskohtaista dataa pysyvään muistiin.

Oma joukkonsa rajapintalaajennuksissa ovat erilaiset laitealustapalvelut (engl. Platform services / Device APIs), jotka keskittyvät mobiililaitteille tyypillisten palvelujen tarjoamiseen web-sovelluksille. Tällaisia palveluja ovat esimerkiksi kalenteritietojen hallinta, kameran käyttö, paikannuspalvelut sekä kiihtyvyysanturin seuranta. Eri web-sovellusympäristöjen laitealustapalvelujen laajasta kirjosta on kuitenkin seurannut merkittävää sovellusten yhteensopimattomuutta eri web-sovellusalustojen välillä (Taulukko 6).

Web-sovellusalusta	**Esittelyvuosi**	**Paketointi**	**Alustapalvelut**
Nokia Web Runtime	2006 (S60 3rd Edition FP1)	Oma	Oma (Platform Services)

BlackBerry WebWorks	2009 (BlackBerry OS 5.0)	Oma	Oma (BlackBerry WebWorks API)
HP webOS	2009	Oma	Oma
Windows Mobile (vain 6.5)	2009	W3C	Ei
LiMo (Linux Mobile)	2009 (R2)	W3C	BONDI (R2) tai WAC (2011 julkaistu R4)
Bada	2010	W3C	BONDI tai WAC (2011 julkaistu 2.0)
MeeGo Web Runtime	2010 (1.1 release)	W3C	NPAPI-liitännäiset
Nokia Series 40 Web Apps	2011	W3C	Ei
Tizen	2012 (alustava)	W3C	WAC

Taulukko 6. Älypuhelinalustojen integroidut web-sovellusalustat ja niiden käyttämät laitealustapalvelurajapinnat. (Idsinga 2010; Nokia Developer 2011b; LiMo Foundation 2011; Tizen 2011; BadaDev.com 2010; BadaDev.com 2011; Research in Motion 2010; PhoneGap Wiki 2011; Windows Mobile Team Blog 2009)

Vuonna 2010 suuri joukko alan operaattoreita, puhelinvalmistajia ja ohjelmistotaloja perustivat Wholesale Applications Community (WAC) -yhteishankkeen, jonka tavoitteena on edistää web-sovellusten yhteensopivuutta eri älypuhelinalustojen välillä (WAC 2011c). Käytännössä tavoitteeseen pyritään luomalla yhteinen rajapinta, jonka kautta web-sovellukset voivat käyttää älypuhelinten tarjoamia alustapalveluja. WAC-standardi edellyttää web-sovelluksilta myös yhteensopivuutta W3C:n paketointistandardin kanssa. WAC-hankkeen johtokuntaan kuuluu mm. GSM Association-järjestö, operaattorijätit AT&T, Vodafone, China Mobile, Telenor Group ja Orange, matkapuhelinvalmistajat Nokia, Samsung, Huawei ja Ericsson sekä piirivalmistajat Intel ja Qualcomm (WAC 2011b).

WAC-standardi korvaa aiemmin standardoidut OMTP:n BONDI-, GSMA:n OneAPI- ja Joint Innovation Lab (JIL) -rajapinnat, joiden takana olleet organisaatiot ovat kaikki liittyneet WAC-hankkeeseen. Ensimmäinen 1.0-versio WAC -standardista perustui suoraan JIL 1.2.2 määrittelyyn, kun taas seuraava WAC 2.0 ei ole enää täysin taaksepäin yhteensopiva sen kanssa. (WAC 2011a)

4.2 Alustariippumattomia sovelluskehitystyökaluja

4.2.1 Qt

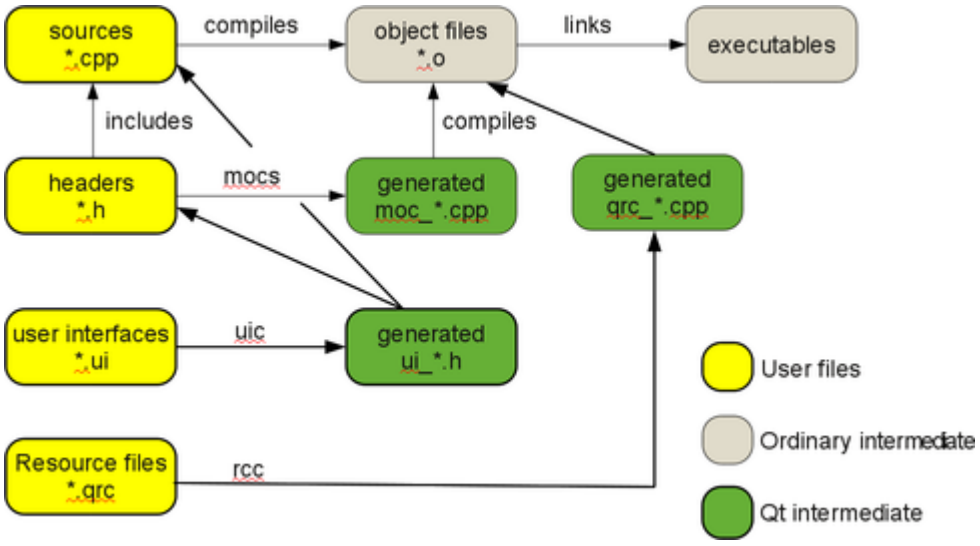
Qt on C++ -kielinen avoimen lähdekoodin alustariippumaton ohjelmointiympäristö ja luokkakirjasto. Sen kehityksin aloitti Norjalainen Trolltech vuonna 1991. Ensimmäiset versiot tukivat vain X-ikkunointijärjestelmää käyttäviä alustoja sekä Windows-käyttöjärjestelmiä. Vuonna 1997 Qt valittiin kehitteillä olevan Linuxin KDE-työpöytäympäristön toteutuskirjastoksi, jonka kautta se on sittemmin päässyt suuren yleisön tietoisuuteen. Tuki Mac OS X käyttöjärjestelmälle lisättiin 2001 julkaistun kolmannen pääversion myötä. Vuonna 2005 julkaistiin Qt:n neljäs pääversio, jota tässä työssä ensisijaisesti käsitellään. Qt:n eri versiot ovat saman pääversion sisällä rajapinnoiltaan taaksepäin yhteensopivia. (Blanchette ja Summerfield 2006; Qt Developer Network 2011a)

Viime vuosien aikana Qt on vaihtanut omistajaa kahdesti. Ensin matkapuhelinvalmistaja Nokia osti Trolltechin vuonna 2008, jonka seurauksena Qt-sovellusten tuki on tuotu Symbian-, Maemo- ja MeeGo-laitealustoille (Qt Reference Documentation 2012c). Nokian strategiamuutoksen myötä Qt kuitenkin myytiin vuonna 2012 suomalaiselle Digialle, joka aikoo ulottaa Qt:n alustatuen myös iOS ja Android-laitealustoille (Digia 2012).

Qt:n merkittävin lisäys C++ -kielen perusominaisuuksiin ovat sen meta-objektit, joihin tallennetaan ajonaikaisia tietoja Qt:n kantaluokkana toimivasta QObject-luokasta perityistä olioista. Meta-objektia käyttäen on muun muassa mahdollista luoda olioinstanssille dynaamisesti attribuutteja. Tärkein syy meta-objektien sisällyttämiseen on kuitenkin Qt:n esittelemä signaalien ja slottien suunnittelumalli, jota Qt-sovelluksissa käytetään olioiden takaisinkutsumekanismina. Mallissa signaalien kautta tapahtuvien funktiokutsujen vas-

taanottaja on abstraktoitu siten, että signaalit voidaan yhdistää vastaanottaviin funktioihin dynaamisesti. Tällöin signaalin lähettäjä ja vastaanottaja eivät myöskään ole teknisesti riippuvaisia toisistaan. (Blanchette ja Summerfield 2006, 20–22)

Qt:lla tehtyjen sovellusten käännösprosessi eroaa hieman normaalista C++ -käännöksestä. Qt sisältää joukon työkaluja lähdekoodien automaattiseen generointiin erilaisista resurssitiedostoista sekä moc-nimisen työkalun, joka generoi meta-objektiluokkien lähdekoodit ennen varsinaista käännöstä (Kuvio 20). Qt:ssa on myös mahdollista generoida Makefile-käännösohjetiedosto käyttäen qmake-työkalua, jolloin generointityökalujen käyttöä ei tarvitse säätää manuaalisesti. (Qt Reference Documentation 2011d)



Kuvio 20. Qt:n käännösjärjestelmä. (Thelin 2011b)

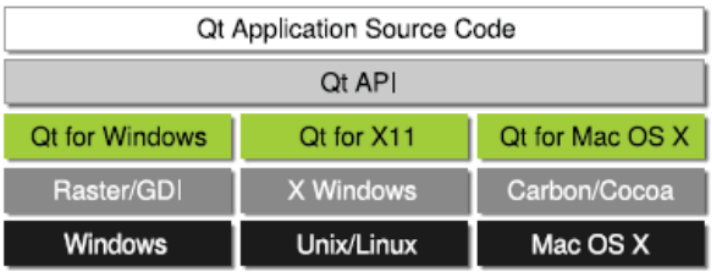
Moduulit. Qt:n luokkakirjasto koostuu useasta pienemmästä moduulista (Taulukko 7). Näistä QtCore tarvitaan jokaisessa Qt:ta käyttävässä sovelluksessa ja se sisältää Qt:n ydintoiminnallisuudet ja tietorakenteet. Oletuksena qmake-työkalua käytettäessä Qt-sovellukseen linkitetään QtCore- ja QtGui-moduulit. (Qt Reference Documentation 2011a)

Moduuli	**Sisältö**
QtCore	Ydintoiminnot käyttöliittymättömien sovellusten tekemiseen.
QtGui	Graafisten käyttöliittymien rakentamiseen tarvittavat toiminnot

QtMultimedia	Multimedia-toiminnot.
QtNetwork	Verkko-ohjelmointiin tarvittavat luokat.
QtOpenGL	OpenGL-grafiikan käyttöön tarvitut toiminnallisuudet.
QtSql	Tietokantakyselyissä tarvitut toiminnallisuudet.
QtSvg	Scalable Vector Graphics (SVG) kuvauskielen esittämiseen tarvitut toiminnallisuudet.
QtWebKit	Selainmoottori integroidun web-sisällön näyttämiseksi.
QtXml	Apuluokat XML-tietorakenteiden käsittelyyn.
Qt3Support	Tukimoduuli Qt:n kolmatta pääversiota käyttävien sovellusten porttaamisen helpottamiseksi.

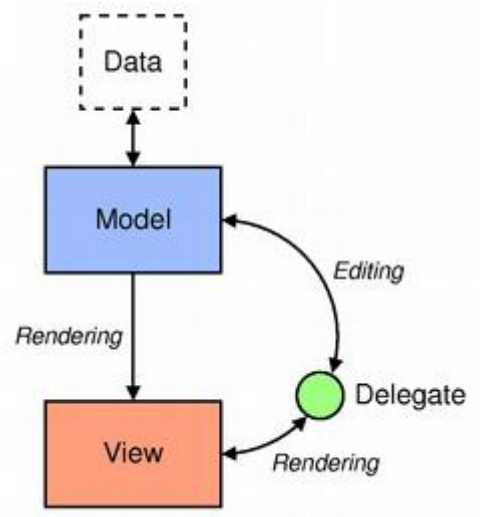
Taulukko 7. Joitain Qt:n neljännen version moduuleista. (Qt Reference Documentation 2011a)

QtGui. Graafisen käyttöliittymän tarjoava QtGui-moduuli sisältää tehokkaan 2D-grafiikkamoottorin ja tätä hyödyntävän kontrollivalikoiman kattavilla asemointimahdollisuuksilla. Käyttöliittymän voi luoda joko luoda dynaamisesti koodin avulla tai ladata XML-kielisestä kuvaustiedostosta (Qt Reference Documentation 2011c). Kontrollit on pyritty toteuttamaan siten, että ne piirretään käyttäen kunkin tuetun laitealustan omia natiiveja komponentteja (Kuvio 21), jolloin saavutetaan yhtenäinen tyyli alustan natiivien sovellusten kanssa. (Ezust ja Ezust, 2006, 237–240)



Kuvio 21. Qt:n alustariippumaton arkkitehtuuri. (Xingyan ja Wei 2010)

Qt:n kontrollit ovat itsenäisiä kokonaisuuksia jotka hoitavat sisäisen tilansa ylläpidon sekä näytölle piirtämisen saman komponentin sisällä. Monimutkaisemmat kontrollit, kuten erilaiset lista-, taulukko- ja puunäkymät sen sijaan käyttävät yleisen MVC-arkkitehtuurin Model/View-muunnelmaa, jossa näkymän ja käsittelijän toiminnot ovat yhdistettyinä, mutta datamalli on siirrettävissä eri näkymäkontrollien välillä (Kuvio 22). QtGui-moduuli tarjoaa tällaisiin kontrolleihin abstraktoidut kantaluokat sekä niistä perittyjä perustason malli- ja näkymäluokkia yleisimpiin käyttötapauksiin. Lisäksi tarjolla on kattavasti erilaisia delegaattiluokkia datan muokkaamiseen, valintamalliluokkia käyttäjän valitsemien datasolujen käsittelyyn sekä datan järjestyksestä ja suodattamisesta huolehtivia välittäjäluokkia. (Qt Reference Documentation 2011b)



Kuvio 22. Qt:n käyttämä model/view-arkkitehtuuri. (Qt Reference Documentation 2011b)

Qt mobiililaitteissa. Qt:n kehityksen painopiste on viime vuosina liikkunut kohti mobiililaitteita. Vuonna 2009 esiteltiin Qt Mobility projekti, joka kapseloi eri laitealustojen palvelut, kuten esimerkiksi paikannuksen ja kontaktienhallinnan, yhtenäisen rajapinnan alle (Qt Developer Network 2011b). Vuonna 2010 julkaistussa Qt:n 4.7-versiossa esiteltiin Qt Quick, joka sisältää tuen käyttöliittymän määrittämiseen deklaratiivisella QML-kielellä. QML-kieli perustuu JavaScript-kieleen ja sillä tehtyjen käyttöliittymien käyttäminen sovelluksissa edellyttää uuden QtDeclarative-moduulin käyttöä. (Thelin 2011a)

4.2.2 Hybridisovellustyökalut

Web-sovelluksia on mahdollista jaella eri laitealustojen sovelluskaupoissa paketoimalla ne sovelluskaupan kelpuuttaman natiivisovelluksen sisään. Tällöin web-sovelluksen lähdekoodeista tuotetaan yhden web-sovelluspaketin sijaan oma hybridisovellus kullekin tuetulle alustalle. Käytännössä tämä toteutetaan luomalla pieni natiivi sovellus, joka lataa ja pyörittää kapseloitua web-sovellusta sovelluksen ikkunassa. Kapseloivan sovelluksen voi koodata manuaalisesti, tai käyttää valmiita työkaluja (Taulukko 8) kuten PhoneGap, MoSync tai Sencha Touch 2 (josta tarkemmin luvussa 4.2.3), jotka sisältävät usein myös toiminnallisuutta laajentavia JavaScript-kirjastoja.

Työkalu	**Tuetut alustat**	**JavaScript-rajapinnat**	**Lisenssi**
PhoneGap	Android, Bada, BlackBerry, iOS, Symbian, WebOS ja Windows Phone 7	Laitealustapalvelut	Apache 2.0
MoSync	Android, iOS, Java ME, Moblin, Symbian ja Windows Phone 7	Käyttöliittymäkirjasto ja laitealustapalvelut	GNU GPL V2, suljetun lähdekoodin sovelluksessa vaatii tilauksen
Sencha Touch 2	Android ja iOS	Käyttöliittymäkirjasto	GNU GPL V3, suljetun lähdekoodin sovellukset sallittu, poislukien sulautetut järjestelmät

Taulukko 8. Hybridisovellustyökalut. (VisionMobile 2012a; Sencha 2012a)

PhoneGap on avoimen lähdekoodin projekti, jonka tarkoituksena on mahdollistaa web-sovellusten natiivi paketointi kaikille merkittäville laitealustoille. Se tarjoaa kaikille tukemilleen alustoille yhtenäinen JavaScript-rajapinnan, jonka kautta sovelluskehittäjät pääse-

vät käsiksi laitteiden alustapalveluihin. Web-sovelluksen jakelu hybridimuodossa on myös tehty erityisen helpoksi W3C:n paketointistandardin mukaisen sovelluspaketin natiivipaketoinniksi muuntavalla, osittain maksullisella PhoneGap Build -pilvipalvelulla. VisionMobilen (2012a, 34) tutkimuksen mukaan PhoneGap on sovelluskehittäjien käytetyin työkalu alustariippumattomuuden toteuttamiseen. Adoben ostettua PhoneGap-työkalua kehittävän Nitobi Software -nimisen yrityksen rajapintojen lähdekoodiprojekti nimettiin uudelleen Cordovaksi, kun taas kokonaisuutena työkalu kulkee yhä vanhalla nimellään. (VisionMobile 2012a, 51–53)

MoSync on alun perin C++/C-kieliä käyttävä sovelluskehys ja kehitysympäristö, joka tukee nykyään myös web-sovelluksia. MoSync sisältää JavaScript-kielisen Wormhole-käyttöliittymäkirjaston ja tarjoaa PhoneGap-yhteensopivan rajapintakokoelman. Sillä toteutettuihin web-sovelluksiin on myös mahdollista lisätä C++/C-kielisiä laajennuksia. Muista esitellyistä työkaluista poiketen sillä toteutetut sovellukset käännetään joko standardiksi Java ME -tavukoodiksi tai MoSync-tavukoodiksi, jota pyörittää paketointiin automaattisesti integroituva tulkkaava ajoympäristö. Työkalun kehitysympäristö osaa tuottaa asennuspaketit suurimpaan osaan tuettuja laitealustoja asentamatta kyseisen laitealustan omaa kehitysympäristöä (VisionMobile 2012a, 70–71)

4.2.3 Web-sovellusten käyttöliittymäkirjastot

Web-sovelluksille on saatavilla useita kosketusohjatuille mobiililaitteille suunnattuja JavaScript-kielisiä käyttöliittymäkirjastoja, joista tässä luvussa tutustutaan muutamaan (Taulukko 9). Kirjastot tarjoavat joukon omia kontrolleja, jotka joissain tapauksissa pyritään automaattisesti näyttämään käytetyn alustan käyttöliittymästandardia käyttäen. JavaScript-käyttöliittymissä erityisen tärkeää on mahdollisimman kattava laitetuki, jotta samaa käyttöliittymäkoodia voisi käyttää sellaisenaan eri konteksteissa web-sivuista lokaaliasennuksiin. JavaScript-kirjastot ovat pääsääntöisesti optimoitu ensisijaisesti web-selaimissa ja -sovellusympäristöissä usein käytetylle webkit-selainmoottorille, joten erityisesti omaa selainmoottoriaan käyttävällä Windows Phone -laitealustalla on ongelmia useiden kirjastojen kanssa. Alla olevan taulukon käyttöliittymäkirjastoista ainoastaan jQuery

Mobile tukee Windows Phone -laitealustaa (Anglin 2012; jQuery Foundation 2012a; Sencha 2012c).

Kirjasto ja versio	**Lähestymistapa**	**Käyttöliittymästandardit**
Sencha Touch 2.0	JavaScript	Yleinen
jQuery Mobile 1.2.0	HTML	Yleinen
Kendo UI Mobile 2012.2.710	HTML	iOS (oletus), Android ja BlackBerry

Taulukko 9. Tarkastellut JavaScript-käyttöliittymäkirjastot. (jQuery Foundation 2012c; Sencha 2012a; Telerik 2012a)

Web-sovellusten käyttöliittymäkirjastot voidaan lähestymistavaltaan jakaa karkeasti kahteen ryhmään. Käyttöliittymäkirjastot edellyttävät käyttöliittymän rakentamista joko käyttäen JavaScript-kieltä vastaavaan tapaan kuin käyttöliittymät luodaan perinteisillä ohjelmointikielillä, tai käyttöliittymä pyritään rakentamaan automaattisesti mobiilikontekstiin sopivaksi HTML-syntaksin pohjalta. Puhtaasti JavaScript-kieltä käyttävissä käyttöliittymäkirjastoissa etuna on monipuolisuus ja kustomoitavuus.

HTML-kieleen perustuvat käyttöliittymäkirjastot puolestaan tarjoavat paremman yhteensopivuuden, koska pohjalla oleva HTML-syntaksi voidaan aina näyttää myös sellaisenaan hienompia toimintoja tukemattomassa selainmoottorissa (jQuery Foundation 2012c). HTML-kieleen perustuva syntaksi on myös helpompi oppia, koska se vastaa HTML-kieltä muutamin lisäyksin. Esimerkiksi VisionMobilen tutkimuksessa (2012a, 50) jQuery Touch -käyttöliittymäkirjastolla oli paras oppimiskynnys vertailtavista alustariippumattomista työkaluista. Lähestymistapana HTML ei kuitenkaan poissulje sisällön dynaamista muodostusta JavaScript-kielen avulla käyttäen DOM-rajapintoja.

Siinä missä perinteisissä web-sivuissa kukin sivu, eli näkymä, koostuu yhdestä HTML-dokumentista, web-sovelluksissa usein kaikki näkymät sijoitetaan samaan HTML-dokumenttiin erillisten päätason sivua kuvaavien elementtien sisään ja kullakin hetkellä näkymättömät sivut piilotetaan CSS-tyylien avulla dynaamisesti. Mikäli näin ei meneteltäisi, JavaScript-suoritusympäristö ladattaisiin aina uudelleen näkymän vaihtuessa ja tämän

vuoksi kaikki tarvittavat muuttujat pitäisi kirjoittaa väliaikaisesti levylle talteen vaihdosta varten. Myös selainnavigaatio eteen- ja taaksepäin toimii samaan dokumenttiin sijoitettujen näkymien kanssa normaalisti, kunhan kullekin näkymälle määritetään sivun sisäinen yksilöllinen URL-osoite käyttäen hyväksi dokumentin sisäisiä hyperlinkkejä.

Sencha Touch. Sencha Touch on JavaScript-käyttöliittymäkirjasto, joka edustaa perinteistä käyttöliittymäkirjastojen lähestymistapaa. Se syntyi pyrkimyksestä yhdistää kolmen aikaisemman käyttöliittymäkirjaston (Ext JS, jQTouch ja Raphaël) parhaat puolet yhteen kirjastoon. Siinä olemassa olevien web-standardien päälle on rakennettu täysin oma ohjelmistokerros, jonka avulla käyttöliittymän kontrollit asetellaan ruudulle sen sijaan, että sisällön kuvaamiseen käytettäisiin HTML-kieltä. 2012 julkaistu 2.0-versio lisäsi Sencha Touch -kehitysympäristöön tuen hybridisovellusten luomiselle iOS- ja Android-alustoille (Sencha 2012a). Sencha Touch on käytettävissä pääosin ilmaiseksi, myös kaupallisten älypuhelinohjelmistojen tuotannossa (Sencha 2012b).

jQuery Mobile. jQuery Mobile perustuu erittäin suosittuun jQuery Core -JavaScript-kirjastoon, jota tutkimusten (Q-Success 2012) mukaan käyttää yli puolet kaikista websivustoista. Sekä jQuery että jQuery Mobile käyttävät MIT-lisenssiä, joka mahdollistaa kirjastojen vapaan kaupallisen käytön. Web-sovellusten käyttöliittymien tekeminen jQuery Mobilea käyttäen on tehty hyvin yksinkertaiseksi, eikä kirjoittajan tarvitse välttämättä käyttää ollenkaan JavaScript-koodia. Tämä saavutetaan käyttäen hyväksi HTML5-standardin esittelemää kustomoitavissa olevaa datakokoelma-attribuuttia (Ortiz 2011). jQuery Mobile tarjoaa virallisen tuen kaikille yleisimmille älypuhelinalustoille, mukaan lukien Windows Phone ja MeeGo (jQuery Foundation 2012a). Se käyttää progressiivista tekniikkaa web-sivun kontrollien rikastamisessa, jolloin myös selaimet jotka eivät esimerkiksi tue HTML5-kieltä, osaavat näyttää sisällön, joskin ilman paranneltua käyttöliittymää. jQuery Mobile osaa myös ladata näkymiä eri HTML-dokumenteista dynaamisesti käyttäen Ajax-tekniikkaa, jolloin suorituskonteksti ei katkea sivusiirtymän vuoksi. VisionMobilen kyselytutkimuksessa (2012a, 50) jQuery Mobile sai kaikista vertailluista työkaluista parhaat pisteet neljässä kategoriassa yhdeksästä, mukaan lukien hinnoittelu, alustatuki, oppimiskynnys ja julkaisumahdollisuudet. Mielenkiintoisen piirteenä jQuery Mobile -

käyttöliittymäkirjastoa on myös käytetty pohjana tulevan Tizen-laitealustan käyttöliittymäkirjastossa (Linux Foundation 2012).

Kendo UI Mobile. Kendo UI Mobile hyödyntää samaa teknistä lähestymistapaa käyttöliittymän rakentamisessa kuin jQuery Mobile. Se käyttää myös jQuery Core -kirjastoa. Merkittävänä erona Kendo UI Mobile osaa piirtää sillä toteutetun käyttöliittymän käyttäen kohdealustan käyttöliittymästandardia. Tuettuja kohdealustoja ovat Android, BlackBerry sekä oletuskäyttöliittymästandardina toimiva iOS (Telerik 2012a). Kendo UI Mobile käyttää GPLv3 lisenssiä, jonka lisäksi saatavilla on maksullinen lisenssi kaupallista kehitystä varten (Telerik 2012b).

4.2.4 Marmalade

Marmalade (aiemmin Airplay) on Ideaworks3D-yrityksen tuottama kehitysympäristö alustariippumattomien sovellusten suunnitteluun. Marmaladen toimii kääntämällä C++-koodin suoraan ARM-prosessoriarkkitehtuurin ymmärtämään binäärimuotoon. Koska Android- ja iOS-laitteet käyttävät samaa prosessoriarkkitehtuuria, käännetty binääritiedosto kelpaa sellaisenaan molemmille alustoille. Binääritiedostot käynnistetään erillisellä alustaspesifisellä lataajasovelluksella, joka tarjoaa myös sovellukselle alustariippumattoman rajapinnan Marmaladen palveluihin. (Ideaworks3D 2012d)

Marmalade suunniteltiin alun perin pelien kehittämiseen (VisionMobile 2012a, 24). Sen uudemmat versiot ovat laajentaneet työkalun ominaisuuksia mm. esittelemällä PhoneGap-yhteensopivan web-sovellusympäristön hybridisovellusten tuottamiseksi, tuen työpöytäympäristösovelluksille (Windows ja Mac OS X) ja käyttöliittymäkirjastot natiivin käyttöliittymästandardin tuottamiseksi Android- ja iOS-alustoilla sekä vaihtoehtoiset oman käyttöliittymästandardin mukaiset kontrollit muille tuetuille alustoille (Ideaworks3D 2012c).

Marmalade tukee mobiilialustoista iOS-, Android-, BlackBerry PlayBook OS- ja Bada-laitealustoja (Ideaworks3D 2012b). Marmelade-työkalusta on saatavilla ilmainen versio, mutta kaupallisten sovellusten tuottamiseen vaaditaan maksullinen lisenssi (Ideaworks3D 2012d).

5 Tapaustutkimus: sovelluskehitys eri tekniikoilla

Tässä pääluvussa perehdyn yksinkertaisen sovelluksen toteuttamiseen pienellä joukolla lähempään tarkasteluun valittuja sovelluskehyksiä. Ensimmäisissä kahdessa alaluvussa esitellen valitut sovelluskehykset ja määritän rakennettavan sovelluksen ominaisuudet. Kolmannessa alaluvussa käyn läpi implementoinnin aikana tehdyt toteutuskohtaiset havainnot ja viimeisessä alaluvussa vertaan eri toteutuksissa saavutettuja tuloksia keskenään.

5.1 Sovelluskehysten valinta

Työn laajuuden pitämiseksi kohtuullisena, päätin ottaa vertailuun vain pienen joukon sovelluskehyksiä. Sovelluskehysten valinnassa oleellisimpana kriteerinä käytin vertailuasetelman muodostamista natiivin sovelluskehyksen ja jonkin samalla laitealustalla käytettävissä olevan alustariippumattoman tekniikan välillä.

Alustariippumattomaksi tekniikaksi valitsin välittömästi web-sovellukset, sillä niiden avulla on mahdollista saavuttaa suurin mahdollinen alustariippumattomuus. Web-sovellusten valinta mahdollisti myös mielenkiintoisen vertailun selaimessa ajetun web-sovelluksen, hybridisovellukseen ja natiivisovelluksen välillä. Web-sovelluksille saatavasta runsaasta valikoimasta erilaisia sovelluskehyksiä päädyin jQuery Mobile -sovelluskehykseen lähinnä sen merkittävän markkina-aseman, HTML-keskeisen käyttöliittymämäärittelyn, kattavan selaintuen, avoimuuden ja käytön tulevassa Tizen-käyttöjärjestelmässä vuoksi.

Natiivin sovelluskehyksen valinta oli hieman haastavampaa. Pääluvussa 3 esitellyistä laitealustoista vanhemmat Windows Mobile- ja Symbian-laitealustat päätin sivuuttaa, sillä molemmat laitealustat ovat elinkaarensa loppupäässä ja niiden sovelluskehykset eivät ole alun perin suunniteltu monikosketusta ja sormiohjausta ylipäätään silmällä pitäen. Myös iOS-laitealustan päätin saavutuksistaan huolimatta sivuuttaa, sillä toteutus iOS-alustalle käyttäen virallista kehitysympäristöä edellyttäisi Mac-tietokoneen hankkimista sekä 99 dollarin vuodessa maksavaan kehittäjäohjelmaan liittymistä.

Jäljelle jäävistä kolmesta laitealustasta (Android, MeeGo ja Windows Phone) kaikissa on vahva fokus käyttöliittymään ja HTML5-yhteensopiva web-selainmoottori, jota voi hyö-

dyntää hybridisovelluksissa. Lopulta valitsin näistän MeeGon, koska se on näistä alustoista tuorein ja edustaa avointa ja modulaarista lähestymistapaa. Nokian tekemä MeeGon Harmattan-sovitus on myös ensimmäinen laitealusta, jonka käyttöliittymä ei tarvitse fyysisiä painonappeja laitteen etupuolella, jolloin laitteen pinta-ala saadaan hyödynnettyä mahdollisimman tehokkaasti käyttöliittymän tarpeisiin.

5.2 Sovelluksen rajaus

Suunnitellessani työssä toteutettavaa vertailusovellusta päätin, että sen tulisi sisältää ainakin seuraavia käyttöliittymille keskeisiä ominaisuuksia:

- Useita näkymiä, joiden välillä navigoidaan käyttäen sekä lateraalista että hierarkista navigointimallia
- Listanäkymä, jonka sisältö tulisi skaalautua satoihin kuvallisiin valintoihin ja tukea dynaamista suodatusta
- Lokalisointi vähintään suomeksi ja englanniksi
- Käyttöliittymän ohjaukseen liittymätön sovelluslogiikka pitäisi olla selkeästi erotettu, uudelleenkäytettävä komponentti sovelluksen sisällä
- Tietolähteen ja tietorakenneluokkien rajapintojen tulisi olla mahdollisimman samankaltaisia vertailtavissa sovelluskehyksissä
- Tekstinsyöttöä virtuaalinäppäimistön avulla

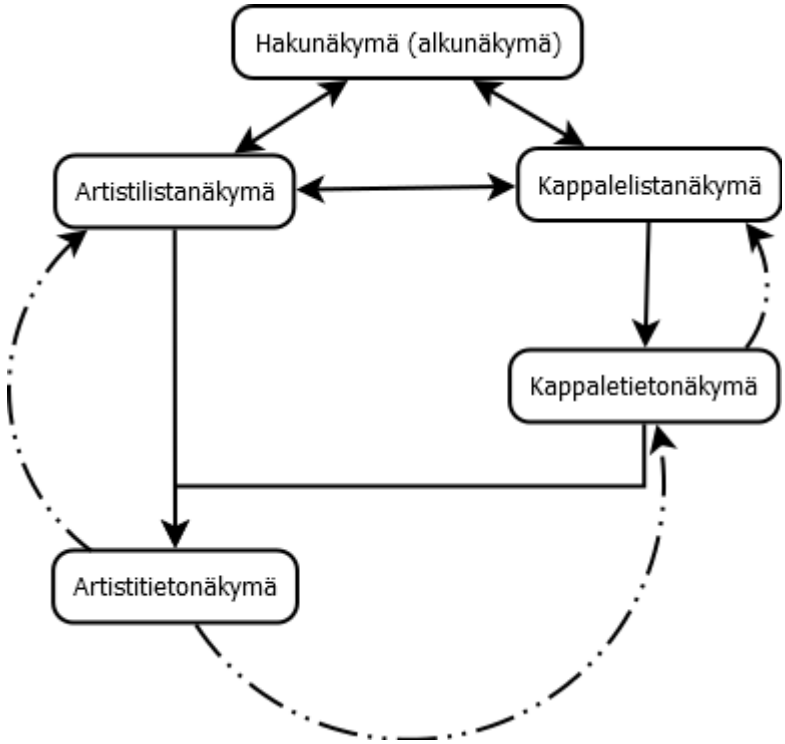
5.2.1 Tietolähteen valinta ja esittely

Tietolähteeksi valitsin Last.fm-musiikkisivuston tarjoaman rajapintapalvelu, jonka kautta on mahdollista päästä käsiksi muun muassa suosituimpien artistien ja musiikkikappaleiden listoihin, jotka sisältävät jopa tuhat kuvallista tietuetta. Tämä riittää mainiosti erilaisten listanäkymien populointiin ja suorituskykytestaukseen. Last.fm-rajapinta on tyypillinen REST-protokollaa käyttävä Ajax-toteutus, jossa asiakassovellus lähettää HTTP-pyyntöjä palvelimille, joka vastaa palauttamalla XML-muotoiltua dataa. Rajapinta tukee myös JSON-muotoisia vastauksia määrittämällä haluttu vastausmuoto kyselyjen *format*-parametriin. (Last.fm 2012)

Last.fm-rajapinta tarjoaa kuvamateriaalia eri artisteista ja albumeista valmiiksi skaalattuna useille eri tarkkuuksille, jotka ovat rajapinnassa yksilöity nimillä *small*, *medium*, *large*, *extralarge* ja *mega*. Kokoja ei määritellä tarkemmin rajapinnan dokumentaatiossa, mutta pääsääntöisesti luokittelu näyttäisi olevan tehty kuvan leveyden mukaan. Havaintojen perusteella eri kokoja vastaavat kuvaleveydet ovat artistikuville 34, 64, 126, 252 ja 500 pikseliä. Yksittäisille kappaleille ja albumeille tarjotut kuvat ovat kooltaan muuten samankokoisia, mutta extralarge-koon kuvat ovatkin 300 pikseliä leveitä ja mega-kokoa ei näytä olevan ollenkaan tarjolla. Tarjolla olevista kokoluokista valitsin medium-koon näytettäväksi listakonteksteissa ja extralarge-koon näytettävissä tietonäkymissä.

5.2.2 Sovelluskäyttöliittymä

Sovelluksen käyttöliittymä tulisi koostumaan viidestä näkymästä. Alkunäkymänä toimisi hakunäkymä, jonka avulla voi hakea artisteja ja musiikkikappaleita samassa näkymässä sijaitsevaan listakontrolliin. Hakunäkymän kanssa samalla lateraalinavigaation tasolla toimisi omissa välilehdissään suosituimpien artistien ja musiikkikappaleiden listat. Lisäksi sovellus tulisi sisältämään erilliset tietonäkymät käyttäjän listasta valitsemille artisteille ja musiikkikappaleille. Alla navigaatiomallia esittävä havainnekuva (Kuvio 23), jossa kaksinuoliset viivat kuvaavat lateraalinavigaatiota, yksinuoliset syvemmälle menemistä näkymähierarkiassa ja katkoviiva mahdollisia polkuja palata takaisin näkymähierarkiassa. Hakunäkymästä ei ole erikseen piirretty viivoja tietonäkymiin selkeyden vuoksi.



Kuvio 23. Vertailtavan sovelluksen näkymien välinen navigaatiomalli.

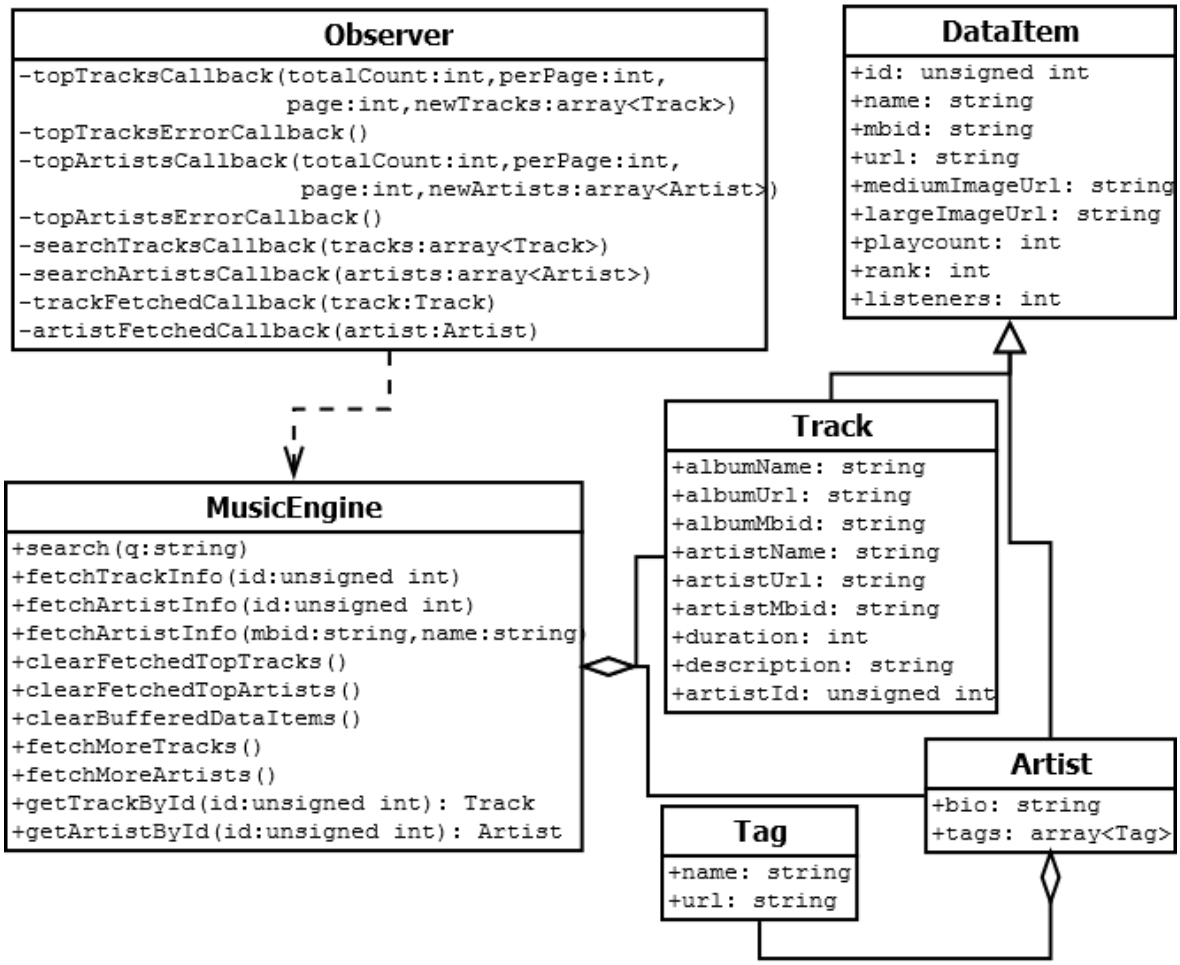
Listanäkymien listat populoidaan vasta kun näkymä avataan ensimmäisen kerran. Koska latausajat ovat yksi älypuhelinten käytettävyyden merkittävistä ongelmista (Nielsen ja Budiu 2012, 10), näkymiin haetaan aina kerralla vain 50 tietuetta ja käyttäjällä on mahdollisuus hakea listaan lisää tietueita näkymän alareunaan sijoitetun hakupainikkeen avulla. Hakunäkymän puolestaan tulee näyttää korkeintaan viisi parhaiten soveltuvaa hakutulosta sekä artistien että musiikkikappaleiden osalta.

Tietonäkymien tulee esittää näyttämänsä artistin tai kappaleen tärkeimmät tiedot. Lisäksi näkymistä tulee olla mahdollisuus siirtyä Last.fm-palvelun sivustolle tarkastelemaan lisätietoja. Kappaletietonäkymässä tulee myös olla erillinen toiminto, joka avaa kappaleen tekemän artistin tietonäkymän.

5.3 Toteutus

Sovellusten implementaatiossa pyrin käyttämään mahdollisimman paljon valmiita käyttöliittymäkontrolleja ilman kustomointia. Tästä huolimatta tavoitteenani oli tuottaa niin yhte-

näinen käyttökokemus kuin valmiilla komponenteilla on mahdollista. Myös käyttöliittymä- ja sovelluslogiikkakerrosten välisen rajapinnan pyrin määrittämään niin samankaltaiseksi kuin käytettyjen ohjelmointikielten käytännöt mahdollistavat. Last.fm-rajapintojen käytön yksityiskohdat päätin abstraktoida erillisen asynkronisesti toimivan *MusicEngine*-luokkarakenteen taakse (Kuvio 24).



Kuvio 24. UML-havainnekaavio *MusicEngine*-rajapinnasta, sen käyttämistä tietoyksikköluokista ja takaisinkutsuista.

Käytännössä *MusicEngine*-rajapintaluokkaa implementoidessa törmäsin muutamaan ongelmaan Last.fm-rajapintojen kanssa. Joissain rajapintakutsuissa oli esimerkiksi mahdollista määrittää palautetulle datalle haluttu kieli, mutta suomenkielellä ei ollut käytännössä saatavilla mitään tietoja, joten päätin hyväksyä englanninkieliset vastaukset myös silloin kun sovelluskäyttöliittymässä käytettiin suomenkieltä. Myös palvelun antamissa tiedoissa

oli epäyhteneväisyyksiä. Palvelun käyttämät mbid-tietuetunnisteet eivät olleet saatavilla kaikille tietueille ja joissain tapauksissa eri rajapintakutsut palauttivat myös eri tunnisteen samalla tietueelle. Erikoisena ongelmana palvelu myös palauttaa musiikkikappaleiden keston joillekin kappaleille sekunteina ja toisille millisekunteina ilmoittamatta käytettyä yksikköä. *MusicEngine*-luokkaan toteutettiin heuristiikka, jossa kappaleen kestot yli 5000 yksiköllä oletetaan millisekunneiksi, koska sekunteina kappaleen pituus olisi tuolloin yli 80 minuuttia, eli enemmän kuin yhdelle CD-levylle mahtuu.

5.3.1 Toteutus MeeGo Touch -kirjastolla

Alkuun pääseminen MeeGo Touch -sovelluskehyksen kanssa on melko työlästä. Vaihtoehtoina on käyttää Windows-, Linux- ja Mac OS X -käyttöjärjestelmille saatavaa geneeristä Qt SDK -kehitysympäristöä, tai vain Linux-käyttöjärjestelmän kanssa yhteensopivaa Platform SDK -kehitysympäristöä (Taulukko 10). Näistä ensin mainittu on tarkoitettu yksittäisten sovellusten vaivattomaan tuottamiseen, jossa kaikki sovelluksen kehitysvaiheet hoituu graafista Qt Creator -ohjelmointiympäristöä käyttäen. Jälkimmäinen puolestaan tarjoaa kehittäjälle monipuolisemmat työkalut alemman tason ohjelmistokomponenttien tuottamiseen, mutta edellyttää monimutkaisempien komentorivityökalujen käyttöä. Molemmat vaihtoehdot sisältävät samat Qt- ja MeeGo Touch -sovelluskirjastot, joten valinta ei vaikuta käytettyihin sovellustason implementaatioratkaisuihin.

	Qt SDK	**Platform SDK**
Ohjelmointiympäristö	Qt Creator-	Komentorivipohjaisia työkaluja, kuten Scratchbox ja GDB
Suoritinarkkitehtuuri	ARM	ARM ja x86
Testausympäristö	QEMU (ARM)	Xephyr (x86)

Taulukko 10. MeeGo-laitealustan vaihtoehtoiset kehitysympäristöt.

Asensin molemmat kehitysympäristöt samaan Linux-käyttöjärjestelmään, mutta hetken testailun jälkeen siirryin käyttämään lähes yksinomaan Platform SDK -kehitysympäristöä.

Qt SDK -kehitysympäristön käyttämä QEMU-emulaattori oli sovelluksen kehitysvaiheessa lähes käyttökelvottoman hidas, sillä se tulkkaa kohdelaitteen ARM-arkkitehruurille käännettyä binäärikoodia. Platform SDK -kehitysympäristön sisältämä Scratchbox-työkalu puolestaan mahdollistaa sovelluksen kääntämisen ja testauksen kehitysalustan natiivia x86-suoritinarkkitehtuuria käyttäen. Tällöin laitealustan käyttöliittymä pyörii Xephyr-työkalun tarjoamassa erillisessä X-palvelimessa. Ainakin omalla tietokoneellani tämä tarjosi selvästi nopeamman testausympäristön kuin QEMU-emulaattori. Xephyr-työkalun käyttö ei kuitenkaan poissulje Qt Creator -ohjelmointiympäristön käyttämistä sovelluskoodin tuottamiseen, kunhan molemmat kehitysympäristöt on asennettu samaan käyttöjärjestelmään ja tuotetun koodin kääntää ja ajaa Xephyr-työkalua käyttäen.

Platform SDK -kehitysympäristössä uusi projekti voidaan luoda ohjatusti käyttäen *project-templates* paketin mukana asentuvaa *create-project* -komentorivityökalua. Työkalu kysyy joitain yksinkertaisia tietoja aiotusta sovelluksesta ja luo valmiiksi kansiorakenteen, josta löytyy tarvittavat projekti- ja Debian-paketointitiedostot sekä *src*-niminen projektialikansio yksinkertaisella sovellusrungolla. Sovellusrunko sisältää tyhjän sovellusnäkymän ja sovelluksen lisäämiseksi laitteen sovellusvalikkoon tarvittavan .desktop-tunnisteisen määritystiedoston. Hieman oudosti työkalu antaa sovellukselle oletuksena version ”0.1”, joka ei kuitenkaan noudata Nokian sovelluskaupan edellyttämää kolmitasoista versionumerointia (Nokia 2012c). Työkalun luoma projektitiedosto sisällyttää projektiin vain Qt:n QtCore- ja QtGui-moduulit, joten verkkoyhteydet sisältävä QtNetwork-moduuli täytyi lisätä manuaalisesti.

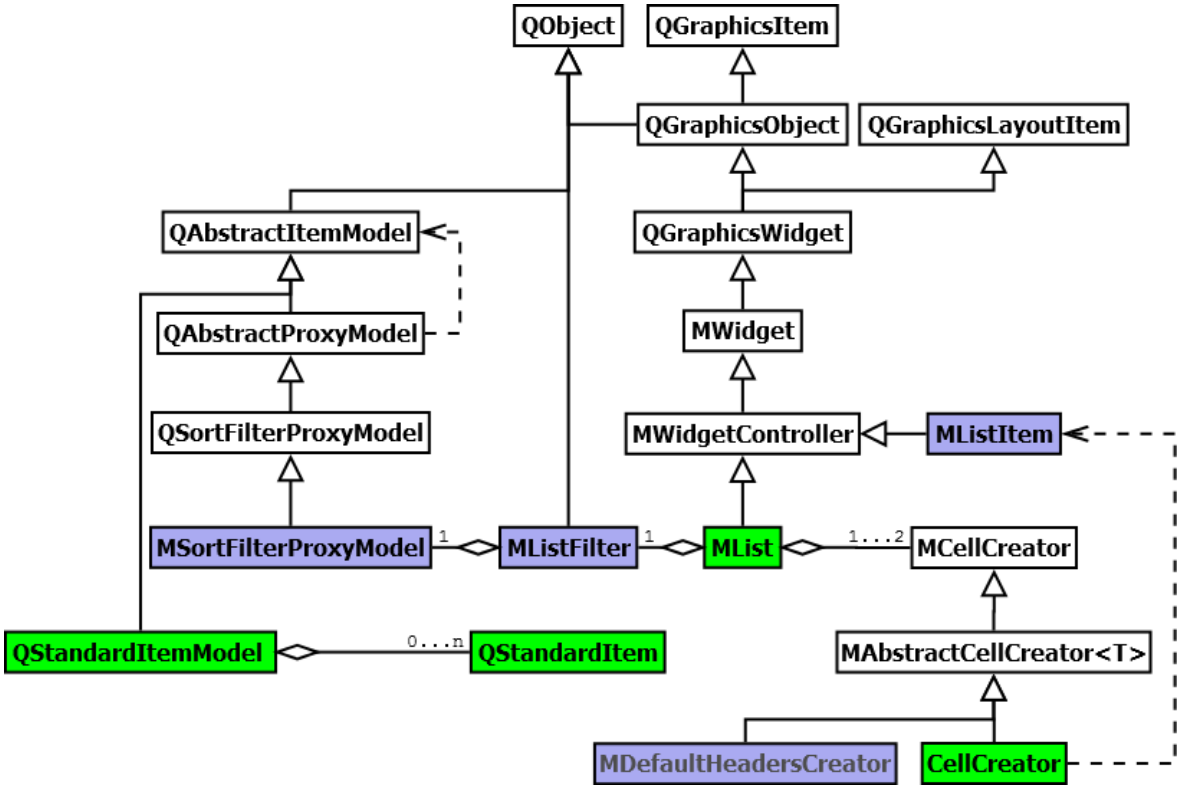
Toteutus. Rakensin sovelluksen (Liite A) pääosin olio-ohjelmoinnille tyypillisellä tavalla, jossa kullekin luokalle on omistettu erillinen otsikkotiedosto (.h) ja varsinainen lähdekooditiedosto (.cpp). Pienet, *MusicEngine*-rajapintaan tiiviisti kuuluvat tietorakenneluokat, kuten yksittäistä artisti- ja kappaletietuetta kuvaavat luokat päätin kuitenkin jättää samaan tiedostoon itse rajapintaluokan kanssa. Sovelluksen takaisinkutsumallina käytettiin Qt:n mahdollistamia signaaleja ja slotteja.

MeeGo Touch -sovelluskehyksessä näkymiä kutsutaan sovellussivuiksi ja ne peritään *MApplicationPage*-luokasta. Toteutin käyttöliittymän luomalla omat sivut päänäkymän

välilehtikokonaisuudelle sekä molemmille tietonäkymille. Tietonäkymille luin lisäksi yhteinen kantaluokan, jonne sijoitin näille yhteiset toiminnallisuudet. Päänäkymän toteutin yhtenä sivuna, jossa eri sisältöjen näyttämisessä hyödynnetään MeeGo Touch -kirjaston tukemaa vaihtoehtoisten asemointikaavojen (engl. layout policy) toimintamallia.

Sovelluskehys tukee myös web-tekniikkoihin kuuluvia CSS-tyyliohjeita, joilla asetetaan kontrolleille mm. niiden dimensiot, fontti ja marginaalit. Kontrolleja voi käyttää joko niiden oletustyylillä, jollain teemaan valmiiksi määritetyllä CSS-tyylillä tai luomalla kontrollille uudet CSS-määritykset sovelluksen omaan CSS-tyylisivutiedoston (MeeGo Wiki 2010). Toteutetussa sovelluksessa omaa tyylisivutiedostoa ei tarvittu, koska valmiit tyylit kattoivat tarvitut käyttötarkoitukset riittävän hyvin. Sovelluskehyksen sisältämä teema sisältää myös laajan, yli 1600 ikonin kokoelman, josta löytyy myös erilaisten tunnettujen palveluiden logoja, mukaan lukien Last.fm-palvelulle. Levynkäytön minimoimiseksi sovelluskehys lataa teemaan kuuluvat ikonit keskitetysti erillisen taustaprosessin kautta.

Ylivoimaisesti monimutkaisin kokonaisuus MeeGo Touch -kirjastolla luodussa sovelluksessa oli sen listakontrollin käyttäminen. Kirjasto tarjoaa todella monipuolisen listakontrollin, joka tukee mm. vapaamuotoisia lista-alkioita, alkioiden ryhmittelyä, järjestämistä, suodatusta ja näyttämistä useassa sarakkeessa. Kontrolli hyödyntää Qt:n valmista *QAbstractItemModel*-malliluokkarajapintaa tietolähteenään, mutta ulkoistaa varsinaisten lista-alkioiden sekä ryhmäotsikoiden muodostamisen erilliselle *MCellCreator*-luokasta peritylle toteutukselle (Kuvio 25). *MCellCreator*-luokan käyttö mahdollistaa skaalautuvuutta parantavan kontrollien kierrätyksen, jolloin varsinaisia lista-alkioita luodaan muistiin vain ruudulla enimmillään näkyvä määrä. Listakontrollia vierittäessä näkyviin tulevat kontrollit ovat itse asiassa aiemmin vastakkaisesta suunnasta näkymättömiin vieritettyjä kontrolleja, joille listakontrolli pyytää sisällön päivitystä niiden tullessa näkyviin. Valitettavasti kirjasto ei kuitenkaan anna minkäänlaista valmista toteutusta malliluokaksi tai alkioiden muodostukseen. Malliluokkana voi käyttää esimerkiksi Qt:n tarjoamaan *QStandardItemModel*-luokkaa, mutta erillinen automaattista ryhmittelyä tukeva malliluokka sekä sen kanssa toimiva lista-alkioita luova *MCellCreator*-toteutus olisi varmasti käyttökelpoinen monelle sovelluskehittäjälle.



Kuvio 25. Listakontrolli ja sen käytössä keskeiset luokat sekä niiden keskinäiset suhteet. Värittämättömät luokat kuvaavat perinnässä käytettyjä luokka ja vihreät sovelluslogiikan hallinnoimia luokkia.

Lokalisointi. MeeGo Touch käyttää Qt:n lokalisointiominaisuuksia, joissa käännetyt tekstit löytyvät erillisistä *katalogeista*, joita sovellus voi dynaamisesti ladata. Yksittäinen katalogi koostuu joukosta tietyllä tavalla nimettyjä binäärimuotoisia käännösdatatiedostoja (.qm), joista järjestelmä valitsee automaattisesti käytetylle kielelle sopivimman tai viimeisenä vaihtoehtona käyttää katalogin *oletuskäännöksiä*. MeeGo Touch -sovelluskehys yrittää automaattisesti ladata sovelluksen nimen mukaista katalogia. Käytetyistä katalogeista tekstejä voi ladata joko käyttäen oletuskäännöstä tai erillistä tunnisteena toimivaa merkkijonoa. Selkeyden vuoksi sovelluksessa päätettiin käyttää erillisiä tunnisteita. (Nokia 2010b)

Sovelluksen käännöksille luodaan oma katalogi lisäämällä sovellusprojektiin uusi aliprojekti, jonka projektitiedostoon määritetään tuettavat kielet. Aliprojekti avulla voidaan käydä läpi sovelluksen lähdekoodit ja koostaa näistä kääntäjille tarkoitetut XML-muotoiset

kielikohtaiset käännöstiedostot (.ts), sekä luoda näistä katalogin käännösdatatiedostot. Qt tarjoaa myös erillisen, kääntäjille suunnatun Qt Linguist -sovelluksen käännöstietojen käsittelyyn. (Nokia 2010b)

5.3.2 Toteutus web-sovelluksena jQuery Mobile -kirjastolla

Web-sovelluksia voi kehittää hyvinkin yksinkertaisilla työkaluilla. Lähdekoodi voidaan kirjoittaa millä tahansa tekstieditorilla, eikä sitä tarvitse missään vaiheessa kääntää tai muutoin prosessoida. Myös kehityksen aikainen testaus onnistuu ilman ylimääräisiä työkaluja, käyttäen mitä tahansa web-selainta sisällön testaukseen. Nykyaikaisista selaimista löytyy myös kattavat mahdollisuudet tutkia suorituksen etenemistä sekä pysäyttää sovellus mille tahansa koodiriville ja jatkaa suoritusta esimerkiksi rivi kerrallaan.

Web-sovelluskehykseksi valitsemani jQuery Mobile otetaan käyttöön yksinkertaisesti linkittämällä sen sisältämät resurssit halutusta HTML-dokumentista. Tarvittuihin resursseihin kuuluu yksi CSS- ja JavaScript-tiedosto sekä näiden vaatima jQuery-kirjasto. jQuery Foundation (2012a) suosittelee näiden kolmen resurssin linkittämistä tarjoamastaan sijainnista suoraan verkon yli, mutta lokaalin web-sovelluksen tapauksessa on parempi jaella tiedostot sovelluksen mukana, jolloin käyttöliittymä toimii myös ilman verkkoyhteyttä. Lokaalijakelussa sovelluksen paketointiin tulee lisätä myös joitain sovelluskehyksen käyttämiä grafiikkatiedostoja, jotka sisältävät pääasiassa sovelluskehyksen oletusikonit.

JavaScript-resursseja linkittäessä on tärkeää huolehtia tiedostojen linkitysjärjestyksestä, sillä tiedostot myös suoritetaan tässä järjestyksessä. jQuery-kirjasto on aina linkitettävä ennen jQuery Mobile -kirjastoa. Mikäli omassa koodissa käytetään jQuery-kirjaston toiminnallisuutta, tulisi myös se linkittää tämän jälkeen. Käytännössä havaitsin, että sovelluksen omat JavaScript-resurssit voidaan ladata joko ennen tai jälkeen jQuery Mobile -kirjastoa, mutta mikäli sovelluksessa halutaan käsitellä oletusasetusten säätöön tarkoitettu *mobileinit*-tapahtuma, täytyy sille rekisteröidä käsittelijäfunktio ennen kirjaston linkittämistä.

Toteutus. Toteutin web-sovelluksen (Liite B) jakamalla implementaation viiteen eri tiedostoon (Taulukko 11). Näistä musicengine.js toteuttaa tässä pääluvussa aiemmin määritel-

lyn *MusicEngine*-rajapintaluokan. JavaScript-toteutuksessa asynkronisten toimintojen takaisinkutsut toteutin kielelle ominaisella tavalla, eli antamalla tarvittavat takaisinkutsufunktiot parametreina niitä käyttäville funktioille. JavaScript-kielen erityispiirteisiin kuuluvien sulkeumien (engl. closures) ja paikallismuuttujina määriteltävien takaisinkutsufunktioiden ansiosta asynkronisen kutsun päätyttyä voidaan aina hyödyntää alkuperäistä suoritustaskontekstia, joka vähentää huomattavasti tietojen väliaikaiseen puskurointiin vaadittavien tietorakenteiden määrää. Itse asiassa myös JavaScriptin oliot voivat olla sulkeumia, joissa voidaan tallentaa esimerkiksi yksityisiä muuttujia ja apufunktioita paikallismuuttujina, joita voidaan myöhemmin hyödyntää.

Tiedosto	**Sisältö**
index.html	Rakenteellinen kuvaus käyttöliittymästä.
styles.css	CSS-tyylimääritykset käyttöliittymälle.
engine.js	Käyttöliittymää ohjaava sovelluslogiikka.
musicengine.js	Last.fm rajapinnan kapsuloiva *MusicEngine*-luokka.
localization.js	Käyttöliittymän lokalisaatiotiedosto, josta jokaiselle tuetulle kielelle oma versio sisältäen kyseisen kielen toiminnallisuuden.

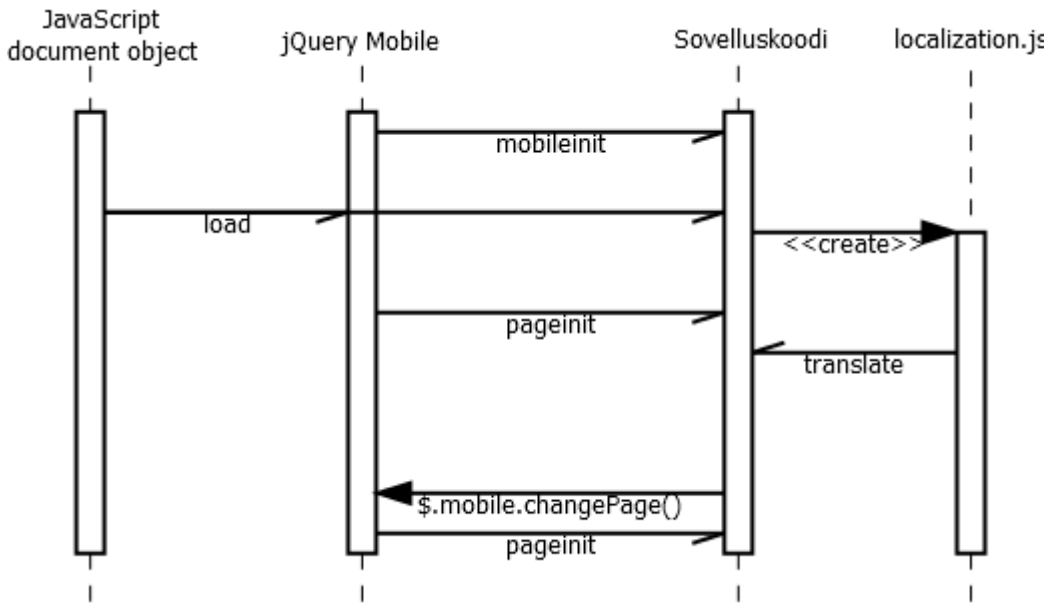
Taulukko 11. Web-sovellustoteutuksen tiedostojako.

Eräs tärkeä seikka web-sovellusta suunniteltaessa on tiedostaa selainten käyttämä tietoturvapolitiikka koskien Ajax-kutsuja. Selaimet nimittäin estävät oletuksena kaikki ajetun JavaScript-sovelluksen tekemät HTTP-kutsut URL-osoitteisiin, jotka viittaavat ajettavan web-sisällön URL-osoitteen lähteen (eng. origin) ulkopuolelle (Barth 2011). Ongelma ei kuitenkaan koske web-sovellusympäristöjä ja hybridisovelluksia, joissa on omat toteutuskohtaiset tietoturvasääntönsä. Mahdollisimman suuren yhteensopivuuden saavuttamiseksi päätin kuitenkin käyttää ongelman kiertävää JSONP-teknikkaa (JSON with padding), joka perustuu selainten tapaan olla soveltamatta lähdetietoturvaa HTML-elementtien pyytämiin resursseihin. Käytännössä tällöin sovellus suorittaa rajapintakutsut lisäämällä dokumenttiin

dynaamisesti uuden *script*-elementin, jonka sisältö haetaan rajapinnan URL-osoitteesta. Osoitteen osana toimitetaan parametrina takaisinkutsufunktion nimi, jota rajapinnan palauttama koodi kutsuu pyydetyillä tiedoilla. Tekniikkaa ei kuitenkaan tulisi käyttää kuin luotettujen lähteiden kanssa, sillä vastaus palautetaan aina koodina, joka suoritetaan sovelluksen kontekstissa. jQuery-kirjaston Ajax-toiminnallisuus kätkee käytetyn siirtotien yksityiskohdat, joten toteutuksessa JSONP-tekniikan hyödyntäminen ei lisää sovelluksen monimutkaisuutta.

Lokalisointi. Web-sovelluksille ei ole yhtä standardoitua tapaa suorittaa lokalisointi. Yleensä se hoidetaan palvelinpäässä, perustuen selaimen lähettämään tietoon käytetystä kielestä. Web-sovellusympäristöissä puolestaan W3C (2011) suosittelee käytettäväksi *kansiopohjaista lokalisointia*, jossa sovellusympäristö yrittää automaattisesti ladata sovelluksen käyttämiä paikallisia resursseja ensisijaisesti polun /locales/X alta, jossa X vastaa jotain käyttäjän ymmärtämää kieltä kuvaavaa IANA-järjestön ylläpitämän rekisterin kielikoodia.

Yhteensopivuuden ja yksinkertaisuuden nimissä päätin toteuttaa lokalisoinnin kansiopohjaisen lokalisoinnin kaltaisella tavalla, mutta vain yhdellä JavaScript-tiedostolla, joka ladataan automaattisesti käyttäjän kielen perusteella sopivasta kansiosta. Tuettuja kieliä ovat oletuksena käytettävä englanti, jota vastaavan localization.js-tiedoston sijoitin sovelluksen juurikansioon, sekä suomi, joka ladataan automaattisesti polusta locales/fi/localization.js. Tämä lähestymistapa osoittautui hieman ongelmalliseksi, sillä lisättäessä JavaScript-tiedosto dynaamisesti sitä ei suoriteta välittömästi ja niinpä ensimmäisenä ladattavan näkymän lokalisoidut tekstit täytyy asettaa erikseen vasta jQuery Mobile -sovelluskehyksen lähettämän, sivun progressiivisen parannuksen valmistumista merkitsevän *pageinit*-tapahtuman jälkeen. Ja koska jokaisella sivulla on oma URL, voi sovelluksen suoritus alkaa mistä tahansa sivusta. Ratkaisuksi (Kuvio 26) loin globaalin *translationsLoaded*-muuttujan, localization.js tiedostojen lopussa laukaistavan oman *translate*-tapahtuman sekä kunkin näkymän *pageinit*-tapahtumalle oman *translate()*-funktion sisältävän sulkeuman, joka käsittelee tuon tapahtuman ja asettaa näkymän staattisen sisällön sen mukaan. Myöhempien näkymämuutosten yhteydessä lokalisoitu sisältö on jo saatavilla ja sulkeuman *translate()*-funktiota voidaan kutsua suoraan tapahtumankäsittelijästä.



Kuvio 26. Web-sovelluksen tapahtumien järjestystä sovellusta ladattaessa sekä myöhemmin sivua vaihtaessa havainnoiva sekvenssikaavio.

Testaus. Vertailtavuuden vuoksi sovellusta testattiin ensisijaisesti Nokian N9-älypuhelimella. Aluksi testasin sovellusta lähiverkkoon sijoitetun HTTP-palvelimen avulla puhelimen oletusselaimessa, joka jQuery Foundation (2012a) -järjestön kolmiportaisen yhteensopivuusluokittelun mukaan kuuluu yhteensopivimpaan A-luokkaan. Lokaaliasennusta varten sovelluksesta tehtiin myös hybridiversio (Luku 5.3.3). Hybridisovelluksen, web-sovelluksen ja natiivisovelluksen vertailusta enemmän luvussa 5.4.

Sovellusta kokeiltiin myös Symbian-pohjaisilla N97- ja E51-älypuhelimilla, jotka edustavat yhteensopivuusluokittelun B- ja C-luokkia. Yllättäen sovellus ei toiminut käytettävästi kummankaan laitteen oletusselaimella, vaikka B-luokkaan kuuluvalla N97-älypuhelimella toimivuuden olisi pitänyt olla melko hyvällä tasolla. Erikseen asennetulla Opera-selaimella sovellus kuitenkin toimi täysin. Tätä ei kuitenkaan jääty tutkimaan tarkemmin, koska nämä vanhentuneet älypuhelimet eivät varsinaisesti kuulu tämän työn rajaukseen. Sovellus todettiin toimivan virheettä Google Chrome (versio 22) ja Internet Explorer (versio 9) -työpöytäselaimilla.

Havaintoja toteutuksesta. jQuery Mobile tukee monipuolisia listakontrolleja (2012d), joiden lista-alkioissa voidaan käyttää joitain sovelluskehyksen tukemia valmiita tyylejä tai

luoda kokonaan omia tyylejä. Lista-alkioita kuvaavan HTML-elementin sisään ensimmäiseksi elementiksi asetettu kuva aiheuttaa lista-alkion muotoilun sisältämään 80x80 pikselin kokoisen kuvakkeen alkion vasemmassa laidassa. Vaihtoehtoisesti lista-alkion kuvalle voi määrittää *ui-li-icon*-luokan, jolloin kuva asetetaan 16x16 pikselin kokoiseen tilaan. Toteutettavan sovelluksen 64x64 kokoisia kuvia ei siis tueta suoraan, mutta käyttäen 80x80 kokoa pohjana myös haluttu koko onnistuu kohtuullisella määrällä CSS-säätöjä.

Listakontrollit tukevat myös listojen dynaamista suodatusta, joka kytketään päälle yksinkertaisesti asettamalla lista-elementille attribuutti *data-filter* arvolla *true*. Tällöin listan yläpuolelle ilmestyy automaattisesti tekstinsyöttökenttä, johon kirjoitettu teksti suodattaa näytettäviä lista-alkioita niiden sisältämän tekstin perusteella. Ominaisuutta ei kuitenkaan ole mietitty ihan loppuun asti, sillä listaan lisätyt alkiot eivät suodatu automaattisesti, edes sen jälkeen kun pyytää listaa päivittämään sisältönsä. Tästä syystä suodatusteksti joudutaan sovelluksessa tyhjäämään aina kun listaan haetaan lisää alkioita.

Erityisen paljon pieniä ongelmia aiheutti valittu tapa lokalisoida sovellus. Sovelluskehyksen tapa muokata HTML-rakennetta dynaamisesti halutun ulkonäön saavuttamiseksi estää useassa tapauksessa alkuperäiseen HTML-rakenteeseen pohjautuvan dynaamisen sisällön muokkaamisen. Esimerkiksi lisättäessä lista-alkioita lista täytyy erikseen käskeä päivittämään sisältönsä, jotta lisätty sisältö näkyisi oikein. Sama päivitystarve koskee myös useita yksinkertaisempia toimenpiteitä, kuten painonapissa käytetyn tekstin muuttamista. Päivitystarve ei sinänsä ole ongelma, kunhan sen tiedostaa, mutta joissain kontrolleissa mahdollisuutta näytettävien tekstien päivitykseen ei yksinkertaisesti ole tarjolla. Joidenkin sovelluskehyksen käyttämien oletustekstien kohdalla puolestaan jQuery Foundation (2012d) ohjeistaa tekstien asetuksen tapahtuvan *mobileinit*-tapahtuman käsittelijässä, jolloin sovelluksessa käytetyt lokalisoidut tekstit eivät ole vielä käytettävissä (Kuvio 26). Näissä tapauksissa ainoaksi vaihtoehdoksi jää manuaalisesti muokata sovelluskehyksen generoimaa sisältöä, joka on kuitenkin huono lähestymistapa, sillä tällaiset sisäiset rakenteet voivat muuttua myöhemmissä versioissa. Tästä ongelmasta kärsivää tekstisisältöä havaitsin esimerkiksi listakontrollin dynaamisen suodatuksen tekstikentän ohjetekstistä ja tyhjennyspainikkeen ohjetekstistä, sekä otsikko- ja työkalupalkkeihin sijoitetuista painonapeista ja välilehtivalitsimista.

Suurimmaksi yksittäiseksi ongelmaksi jQuery Mobile -sovelluskehyksen käytössä osoittautui yllättäen työkalupalkin kiinnittäminen näytön alalaitaan, johon olisin halunnut asettaa käyttöliittymän navigointivaihtoehdot. jQuery Foundation (2012b) myös itse dokumentoi ongelmista joillain mobiiliselaimilla, mutta toteaa tällaisilla selaimilla työkalupalkin automaattisesti putoavan sivun sisällön alareunaan. Valitettavasti tämä on toteutettu kirjaston sisällä selaintunnistuksella selaimen User-Agent-tunnisteen perusteella, eikä näin ole kovin luotettava. Esimerkiksi Nokian MeeGo-alustalla sovelluskirjasto taantuu käyttämään sisällön alalaitaan kiinnittyvää työkalupalkkia, mutta Qt-sovelluskehyksellä toteutetuissa hybridisovelluksissa oletuksena käytettävällä User-Agent-tunnisteella työkalupalkki piirtyy aluksi oikein ruudun alalaitaan, mutta sijainti ei päivity vierittäessä ruudun sisältöä, vaan jää häiritsevästi muun sisällön päälle. Tällaisenaan ominaisuus ei siis ole kovin käyttökelpoinen, joten navigoinnin osalta päätin käyttää aina sisällön alalaitaan kiinnittyvää työkalupalkkia. Tämän lisäksi näkymissä, joissa tarvitaan paluutoimintoa, lisättiin se myös näkymän otsikkopalkkiin, jolloin navigointi onnistuu sekä sisällön ylä- että alapäästä. Vastaavasti välilehdistä koostuvissa näkymissä sijoitetin välittömästi otsikkopalkin alapuolelle kopion sisällön lopussa olevasta, välilehtivalinnat sisältävästä työkalupalkista. Hybridisovelluksissa ongelma olisi tietysti mahdollista kiertää myös käyttämällä natiivin sovelluskehyksen työkalupalkkia sovelluksen sisäiseen navigointiin, mutta tämä ominaisuus pitäisi toteuttaa kaikille kohdealustoille erikseen.

Eräs mielenkiintoinen havainto sovellusta testatessa oli huomata sovelluksen kontrollien näyttävän yllättävän suurilta kohdealustan selainta käyttäessä. Hybridisovelluksessa moista ongelmaa ei ollut, joten asia vaati vähän tutkimista. Olin määrittänyt HTML-dokumenttiin mobiilisivuja varten kehitetyn *viewport*-meta-elementin, jossa ohjeistetaan selainta näyttämään dokumentti ikkunassa, jonka leveys vastaa laitteen näytön leveyttä. Kävi kuitenkin ilmi (Nokia Developer 2012), että laitteen oletusselain näyttää tällöin sisällön iPhone-yhteensopivuuden vuoksi 1,5-kertaisella skaalauksella. Ongelma korjaantui lisäämällä meta-elementtiin määritys *target-densitydpi* arvolla *device-dpi*.

5.3.3 Toteutus hybridisovelluksena

Vertailun vuoksi päätin toteuttaa web-sovelluksesta myös hybridiversion. Vaikka Cordova-työkalu ei virallisesti tue MeeGoa, sen 2.1.0-version lähdekoodipaketin Qt-projekti sisältää yllättäen myös MeeGon Harmattan-toteutuken projektitiedostot. Kyseessä on QML-kielellä käyttöliittymän toteutettu sovellus, joka on tarkoitettu sellaisenaan käännettäväksi sekä Symbian- että Harmattan-alustoille. Päällisin puolin Harmattan-tuki ei vaikuttanut millään tavalla keskeneräisemmältä verrattuna virallisesti tuettuun Symbianiin, joten päätin luoda hybridisovelluksen sitä käyttäen. PhoneGap Build -pilvipalvelu ei kuitenkaan tue MeeGoa, joten levityspaketin joutuu luomaan manuaalisesti käyttäen MeeGon kehitysympäristöä käyttäen.

Hybridisovelluksen rakentaminen oli yllättävän helppoa Qt SDK -kehitysympäristöä käyttäen. Cordova-työkalun Qt-projektista voi tehdä Qt Creator -ohjelmointiympäristölle sovelluspohjan kopioimalla työkalun tiedostot sen hakemistorakenteen sisäisen polun *share/qtcreator/templates/wizards* alle. Tämän jälkeen kehitysympäristössä aloitetaan uusi projekti käyttäen Cordova-työkalun sovelluspohjaa ja lisätään luotuun projektiin oman web-sovelluksen koodit projektin sisältämän *www*-hakemiston alle. Qt Creator osaa sovelluspohjan avulla kääntää ja paketoida sovelluksen jakeluvalmiiksi ilman, että käyttäjän tarvitsisi käyttää mitään komentorivityökaluja.

Ainoa sovelluspohjan puute on, että vaikka Cordovan lähdekoodipaketissa on mukana laitteen sovellusvalikkoon tulevan pikakuvakkeen määritystiedosto (.desktop), sitä ei jostain syystä luoda automaattisesti sovelluspohjasta luoduille sovellusprojekteille. Tämä siis piti säätää projektiin käsin, jotta sovelluksen voisi laitteella käynnistää muutoin kuin pelkästään komentokehotteen kautta. Tämä on kuitenkin tehty sikäli helpoksi, että lisätty määritystiedosto lisätään automaattisesti paketointiin mukaan mikäli sen tiedostonimi vastaa muotoa X_harmattan.desktop, jossa X on projektin nimi..

Cordovan rajanpintalaajennukset on toteutettu erillisten liitännäisten kautta. Rajapinnan toiminnallisuus on jaettu eri liitännäisiin aihekokonaisuuksittain. Oletuksena sovellus käyttää kaikkia rajapintakirjaston moduuleja. Sovelluksen käyttämät liitännäiset voi valita muokkaamalla *xml/plugins.xml*-tiedostoa. Mikäli liitännäisiä käyttää, tulee omaan web-

sovellukseen lisäksi linkittää *www/js*-hakemistosta cordova.js- ja cordova.qt.js-tiedostot sekä haluttujen rajapintaliitännäisten JavaScript-rajapintatiedostot. Toteutetusta sovelluksesta poistettiin käytöstä kaikki työkalun mukana tulevat liitännäiset.

Cordova-työkalulla toteutetussa hybridisovelluksessa ainoa merkittävä ongelma liittyi hyperlinkkien käsittelyyn. Oletuksena kaikki sovelluksen hyperlinkit avataan automaattisesti hybridisovelluksen omaan web-näkymään. Tämä on kuitenkin ulkoisten hyperlinkkien kohdalla ongelmallista, sillä käyttäjälle ei tarjota lainkaan web-selaimen navigointipainikkeita, joilla hän voisi palata takaisin web-sovelluksen kontekstiin. Ongelma ratkaistiin toteuttamalla yksinkertainen *MusicApp*-liitännäinen (Liite C), joka tarjoaa web-sovellukselle funktion, jonka kautta haluttu hyperlinkki avataan järjestelmän oletusselaimessa. Liitännäisen integroitiin web-sovellukseen lisäämällä *engine.js*-tiedoston loppuun *MusicApp*-luokka, liitännäisen rekisteröintiin tarvittava koodi sekä liitännäistä kutsuva tapahtumankäsittelijä hyperlinkeille.

5.4 Toteutusten vertailu

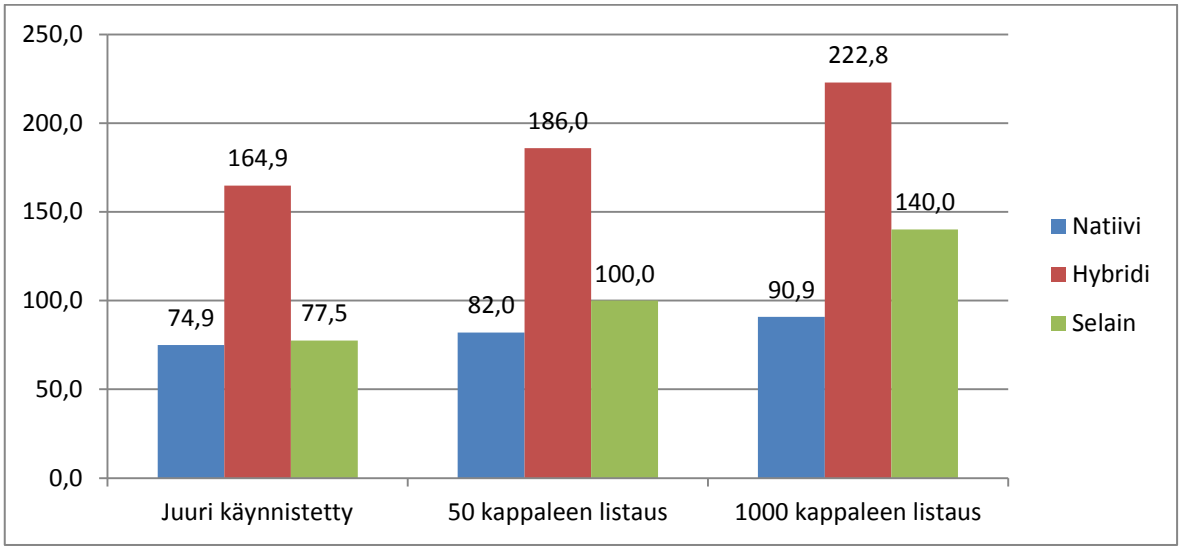
5.4.1 Kvantitatiiviset ominaisuudet

Lähdekoodia kirjoittaessa pyrin säilyttämään samankaltainen muotoilutyylin eri toteutuksissa. En myöskään kirjoittanut lähdekoodiin kommentteja niiden aiheuttaman koodirivimäärän vääristymän välttämiseksi. Rivimääräisesti tarkasteltuna natiivisovellus vaati 61,5 prosenttia järeämmän toteutuksen kuin web-sovellus (Taulukko 12). Pelkän käyttöliittymän osalta ero kaventuu 39,2 prosenttiin, sillä merkittävin ero oli *MusicEngine*-rajapintaluokan toteutuksessa (113 %). Lähdekoodiriveihin laskettiin mukaan kaikki suoritettava koodi, jonka vuoksi web-sovelluksessa mukaan laskettiin käännöstiedostot (yhteensä 90 riviä), mutta natiivitoteutuksessa vain lähdekoodin seassa olevat oletuskäännökset. Hybridisovelluksen osalta eritellyt arvot on taulukossa ilmoitettu web-sovelluksen koodien lisäksi tarvittavina koodiriveinä. Lähdekoodien lisäksi natiivi- ja hybridisovellukseen kuuluu joukko projekti-, käännös- ja paketointitiedostoja.

	Natiivisovellus	**Web-sovellus**	**Hybridi**
Käyttöliittymä (deklaratiivinen)	-	HTML: 286 CSS: 55	HTML: +2
Käyttöliittymä (logiikka)	1194	Logiikka: 427 Käännökset: 90	Liitännäinen: +43
MusicEngine toteutus	789	370	-
Yhteensä	**1983**	**1228**	**1228 + 45**

Taulukko 12. Sovelluksen eri osa-alueiden pituus koodiriveinä.

Muistivaraus. Mittasin sovellusten varaaman muistin kokonaismäärän lukemalla tiedon suoraan sovellusprosessin status-tiedostosta (*/proc/[pid]/status*) kolmessa eri testaustilanteessa (Kuvio 27). Toistin mittaukset kahdesti virhemittausten eliminoimiseksi. Selaimen kohdalla mitattu arvo on kuitenkin täysin viitteellinen, koska se on suljettu sovellus, jonka sisäisestä rakenteesta ei ole saatavilla tarkkaa tietoa. Selain kuitenkin näyttäisi käyttävän ainakin kahta pääprosessia, joiden binääritiedostojen nimet ovat *grob* ja *QtWebProcess*. Ensin mainitun prosessin muistijälki pysyi kaikkien mittausten aikana melko tasaisesti keskimäärin 174 megatavussa alle yhden megatavun keskihajonnalla, mutta *QtWebProcess*-binäärin muistinkulutus puolestaan reagoi mittaustilanteisiin odotetulla tavalla, joten oheisessa kuviossa viitataan tämän prosessin muistijälkeen. Yhteenlaskettuna web-selaimen muistijälki olisi kuitenkin kaikissa testitilanteissa suurin. Keskihajonta oli kaikissa mittauksissa maltillinen ja kohosi yli megatavun vain hybridisovelluksen 50 alkion (1,5 MB) ja 1000 alkion (2,0 MB) testeissä.



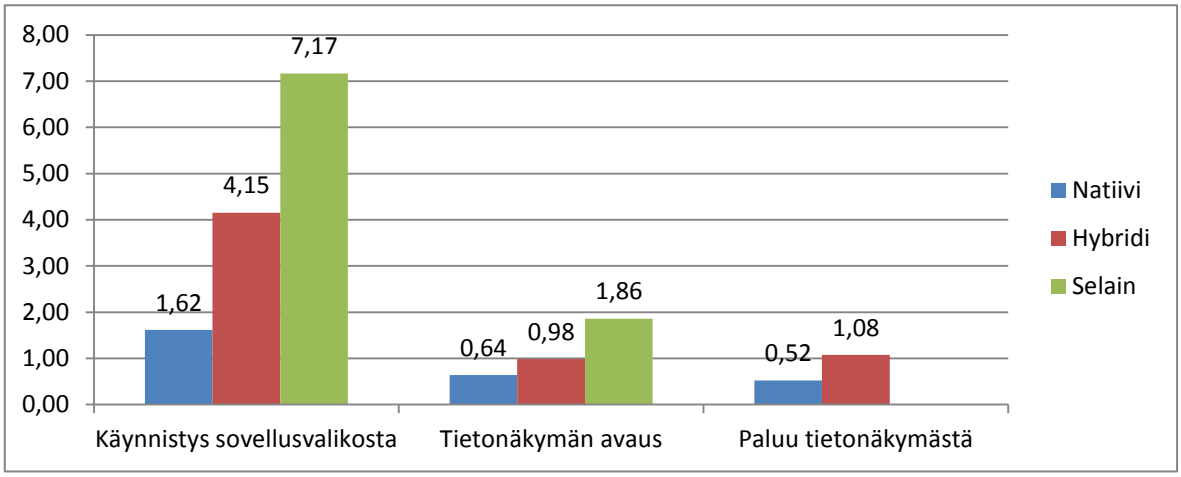
Kuvio 27. Mitatut muistijäljet megatavuissa.

Lista-alkioissa näytetyt kuvat olivat kymmenen kuvan satunnaisotannalla tarkasteltuna kooltaan keskimäärin 8,1 kB (1,9 kB keskihajonnalla). Tarkasteltaessa muutosta 50:n alkion listauksesta 1000:n alkion listaukseen sovellus siis lataa verkon yli yhteensä noin 7,5 MB pelkästään kuvadataa. Muistinvarauksen suhteen selvästi parhaiten tällöin skaalautui natiivisovellus, jonka muistinvaraus kasvoi 8,9 MB (11 %), joka on vain 1,4 MB enemmän kuin laskennallinen kuvadatan määärä. Hybridisovelluksella muistinvaraus kasvoi vastaavassa tilanteessa 36,9 MB (20 %) ja selaimen 40,1 MB (40 %).

Suorituskyky. Sovelluksen suorituskykyä mittasin *xresponse*-työkalun avulla (Taulukko 13 ja Kuvio 28). Työkalu seuraa näytöllä tapahtuvia muutoksia ja mahdollistaa viiveen mittaamisen käyttäjän toiminnosta laitteen näytöllä tapahtuvaan viimeiseen muutokseen. Mittaukset suoritettiin sovelluksen käynnistymisnopeudesta, tietonäkymän avauksesta ja paluusta tietosivulta takaisin edelliseen hakunäkymään. Mittauksissa seurattiin koko ruudulla tapahtuvia muutoksia, joten mittauksista jouduttiin suorittamaan uudelleen muutamia mittauksia, jotka menivät pieleen esimerkiksi tilapalkin kellonajan muuttuessa kesken mittausjakson. Stabiilien tulosten varmistamiseksi kustakin mittaussarjasta eliminoitiin myös ensimmäinen mittaus, jolloin esimerkiksi tietonäkymän avauksessa tapahtuva tietojen lataus Last.fm-palvelulta saatiin pois mittauksista.

	Natiivi	Hybridi	Selain
Käynnistymisviive	1 616 (32)	4 155 (38)	7 166 (589)
Tietonäkymän avaukseen kuluva viive	640 (17)	984 (57)	1 862 (26)
Tietonäkymältä paluun viive	524 (6)	1 080 (21)	-

Taulukko 13. Mitattujen vasteaikojen (ms) mediaanitulokset kymmenen mittauksen otannasta (suluissa keskihajonta).



Kuvio 28. Mitatut vasteaikojen (s) mediaanitulokset graafisessa muodossa.

Mittausten valinnan kannalta suurin ongelma oli natiivisovelluksessa ja selainympäristössä sisällön reunassa automaattisesti esitettävä vierityspalkki, joka haihtuu näytöltä hitaan animaation kera. Vierityspalkki kertoo käyttäjälle uuden näkymän olevan vieritettävissä ja se näkyy näkymän ensimmäisen piirron lisäksi vierittämisen aikana. Palkkia ei kuitenkaan näytetä silloin kun sisältöä ei voi vierittää. Niinpä mittaukset valittiin suoritettavaksi hakunäkymässä sellaisella haulla, jonka tuloksia ei voinut vierittää, käyttäen sellaista tietonäkymää, jonka sisältöä ei myöskään voinut vierittää.

Selaimella suoritetut mittaukset vaativat hieman täsmennystä. Web-sovellus ladattiin paikallisverkossa sijaitsevalta web-palvelimelta osoitteesta, jolle oli luotu testilaitteen sovel-

lusvalikkoon pikakuvake. Käynnistysnopeus ei siis ole täysin vertailukelpoinen, koska nopeuteen vaikuttaa myös verkon toimintanopeus. Toisaalta, ajettaessa web-sovellusta selaimella tämä lisäviive on aina mukana loppukäyttäjillä ja lähiverkossa sijaitseva palvelin edustaa optimaalista tilannetta verrattuna ulkoverkosta ladattaviin web-sovelluksiin.

Selaimella ajetuista testeistä käynnistys- ja paluumittauksien kohdalla selain esitti jostain syystä aina vierityspalkin, mutta käynnistysnopeuden mittauksissa silmämääräisesti arvioituna hakukentän suurennuslasi-ikoni piirretään vasta palkkien haihtumisen jälkeen, joten käynnistysnopeuden mittauksiin tällä ei ole merkitystä. Tietonäkymältä palattaessa tämä kuitenkin sotki mittausasetelman ja tulokseksi saatiin vertailukelvoton 3 277 ms (72 ms keskihajonnalla).

5.4.2 Erot toteutuksissa

Vaikka pyrin implementoimaan vertailtavista toteutuksista mahdollisimman samankaltaiset, jouduin web-sovellusten ja natiivisovellusten ominaispiirteiden vuoksi tekemään joitain kompromisseja. Kuten edellisissä luvuissa mainittiin, toteutuksissa käytettiin käytetylle ohjelmointikielelle tyypillistä takaisinkutsujärjestelmää. Myös Last.fm-rajapinnan kutsuissa pyydettiin data eri muodossa, koska JSON-muoto on JavaScript-kielelle ominainen muoto ja vastaavasti XML-muodolle löytyy Qt-sovelluskehyksestä valmis tuki, mutta JSON-muodolle ei.

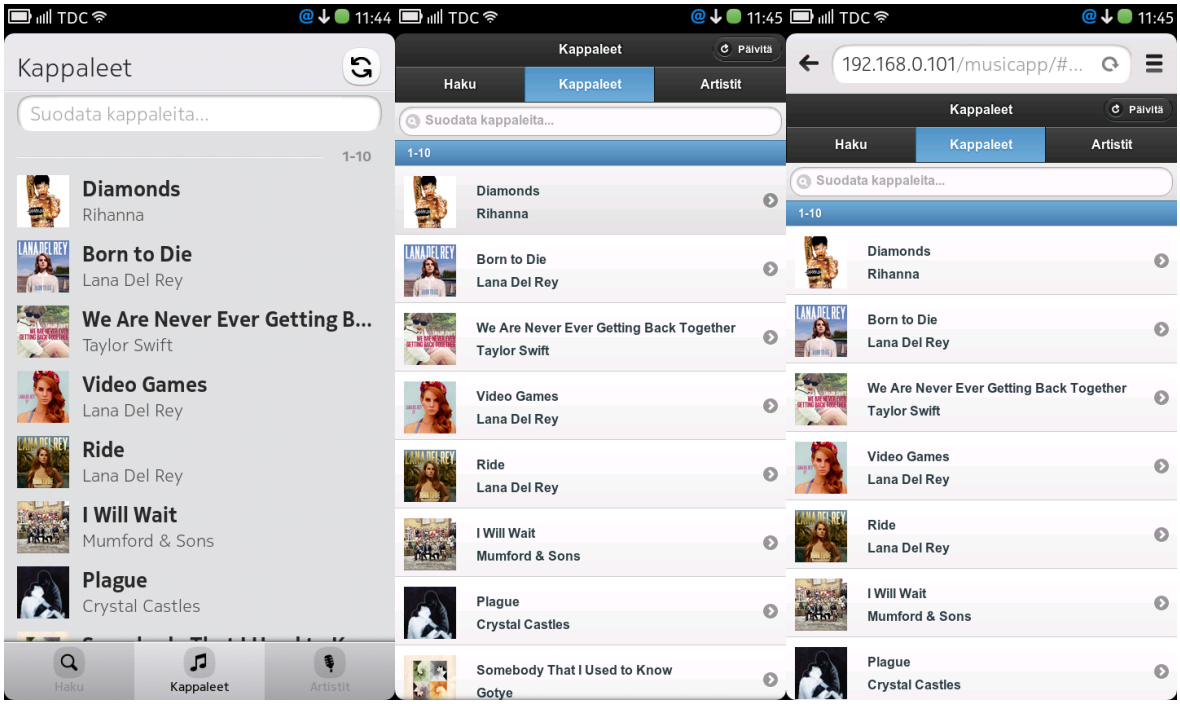
Käyttöliittymätoteutuksissa tehtiin joitain ratkaisuja käytetylle ohjelmointikielelle ominaisten piirteiden pohjalta. Web-sovelluksissa esimerkiksi kuvia ei tarvitse ladata manuaalisesti, vaan riittää, että kuvaelementille määrittää kuvan URL-osoitteen kuvalähteeksi ja selainmoottori hoitaa kuvan latauksen sovelluksen näkökulmasta automaattisesti. Natiivisovelluksessa kuvat joudutaan lataamaan manuaalisesti, mutta toisaalta sovelluslogiikalla on täydellinen kontrolli prosessista. MeeGo Touch -toteutuksessa lista-alkioissa näytetään ensin väliaikainen teemaan kuuluva ikoni, joka korvataan kuvan latauduttua oikealla kuvalla. Kuvien lataus tapahtuu yksitellen, ensisijaisesti ladaten puuttuvia kuvia näkyville lista-alkioille. Suorituskyvyn optimoimiseksi uusia kuvia ei kuitenkaan ladata silloin kun käyttäjä vierittää listaa. Kuvien latauksessa käytetyt tekniset ratkaisut näkyvät käytettävyy-

dessä siten, että web-sovelluksissa kuvat latautuvat silminnähtävästi nopeammin, mutta vastaavasti sovellus tahmaa hieman mikäli käyttäjä pyrkii vierittämään listaa ennen kuin kaikki kuvat on ladattu. MeeGo Touch -sovelluksessa kuvat ilmestyvät verkkaisesti, mutta sisällön vieritys toimii ripeästi alusta alkaen ja kuvat ilmestyvät ensisijaisesti juuri niille lista-alkioille jotka käyttäjä näkee.

Last.fm-rajapinnat palauttavat vain suppean määrän tietoa kustakin tietueesta pyytäessä pitkiä tietuelistauksia. Tietonäkymiin siirryttäessä tietueesta joudutaan hakemaan erillisellä pyynnöllä tietueen tarkemmat tiedot, jotka kuitenkin puskuroidaan siten, että myöhemmillä katselukerroilla tiedot ovat jo valmiiksi saatavilla. Natiivisovelluksessa tämä toteutettiin käynnistämällä tarkempien tietojen haku käyttäjän valitessa lista-alkion ja automaattisesti täydentämällä käyttöliittymän puuttuvat kentät haun valmistuessa. Web-sovelluksessa puolestaan käytetään jQuery Mobile -kirjaston tarjoamaa kätevää sivunlatausanimaatiota ja sivu näytetään vasta kun tieto on haettu. Erikoistapauksen tietojen haun käsittelyssä muodostaa vielä kappaletietonäkymältä artistitietonäkymälle siirtyminen natiivisovelluksessa, jossa artistin tiedot haetaan automaattisesti kappaletietosivulle saavuttaessa ja artistitietonäkymälle vievä nappi aktivoituu vasta tietojen ollessa saatavilla.

5.4.3 Käyttöliittymä ja käytettävyys

Toteutetun sovelluksen eri versioiden (Kuvio 29) toiminnassa silmiinpistävin ero on niiden käyttöliittymästandardeissa. MeeGo Touch -sovelluskehyksellä toteutettu sovellus näyttää ja tuntuu samalta kuin muut laitteen natiivit sovellukset. jQuery Mobile -sovelluskehys puolestaan pyrkii käyttämään sovelluksissa erilaista, iOS-laitealustan käyttöliittymästandardia jäljittelevää ulkonäköä, jossa esimerkiksi käytetään enemmän kirkkaita värejä ja erilaisia väriliukumia.



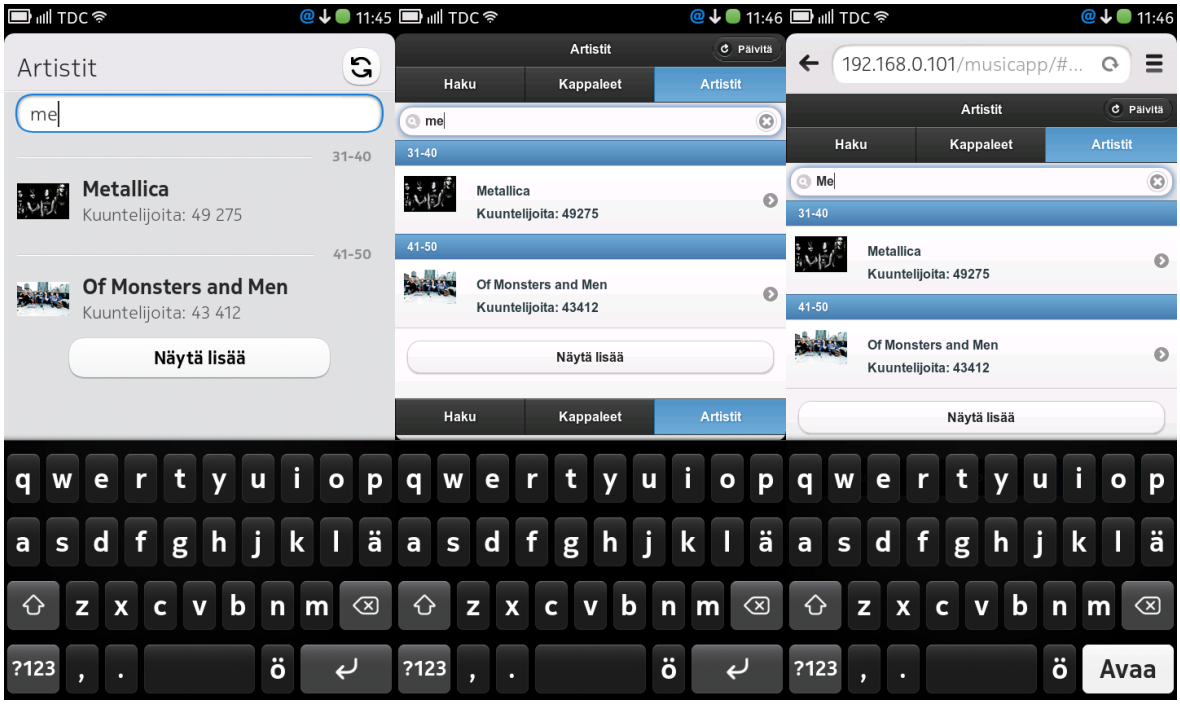
Kuvio 29. Kappalelistanäkymä natiivi-sovelluksella, hybridisovelluksella ja oletusselaimessa ajetulla web-sovelluksella.

Toteutukset eroavat merkittävästi erityisesti käyttöliittymäkrominsa määrän ja sijainnin osalta. Kaikissa kolmessa toteutuksessa sisällön ylälaitaan on kiinnittynyt otsikkopalkki. Selaimessa sen ja järjestelmän tilapalkin välissä on vielä ruudun ylälaitaan kiinnittyvä selaimen oma työkalupalkki. Sovelluksen navigointipalkki on natiivisovelluksessa kiinnitetty ruudun alalaitaan, kun taas web-sovellustoteutuksissa se on kiinnitetty sisällön alalaitaan sekä lisäksi otsikkopalkin alapuolelle listanäkymissä. Hybridisovelluksessa oli selvästi eniten tilaa sisällölle, kun taas natiivisovelluksessa navigointivalinnat olivat kätevästi aina saatavilla.

jQuery Mobile tuntuu kontrolliensa fyysisten dimensioiden perusteella olevan suunniteltu 320 vaakapikselin iPhone-laitteille, sillä käytetyn N9-älypuhelimen 480 vaakapikselin näytössä useat kontrollit jäivät turhan pieniksi. Vaihtoehtoisesti web-sovelluksen olisi voinut toteuttaa 1,5-kertaisella skaalauksella, mutta tällöin kuvien skaalaus olisi aiheuttanut epäterävyyttä ja lista-alkioista olisi tullut turhan suuria. Erityisen hankalakäyttöiseksi osoittautuivat otsikkopalkkiin sijoitetut painonapit, joiden noin 30 pikselin korkeus ei yllä edes iPhone-alustan 44 pikselin minimisuositukseen (Apple 2012a). Ongelma korostui hybridi-

sovelluksessa käytetyllä N9-älypuhelimella entisestään, koska heti sovelluksen oman työkalupalkin yläpuolella sijaitsee laitealustan tilapalkki, jonka valinta tuo esiin järjestelmän tilavalikon.

Tekstinsyötön osalta kaikki kolme toteutusta käyttävät samaa, laitealustan tarjoamaa virtuaalinäppäimistöä. Selaimen kohdalla tosin rivinvaihtopainikkeen tilalla käytetään ”Avaa” tekstiä. (Kuvio 30). Näppäimistön käyttö listanäkymien dynaamisessa suodatuksessa kuitenkin erosi huomattavasti. jQuery Mobile -sovelluskehyksen tapauksessa suodatus tuli voimaan vasta painaessa rivinvaihtopainiketta tai fokuksen poistuessa syötekentästä, kun taas natiivisovelluksessa suodatus tapahtui välittömästä tekstiä syöttäessä. Hybriditoteutus vaati lisäksi syötekentän aktivoimisen aina kahdesti ennen kuin näppäimistö tuli esiin.



Kuvio 30. Sisällön dynaaminen suodatus ja virtuaalinäppäimistö.

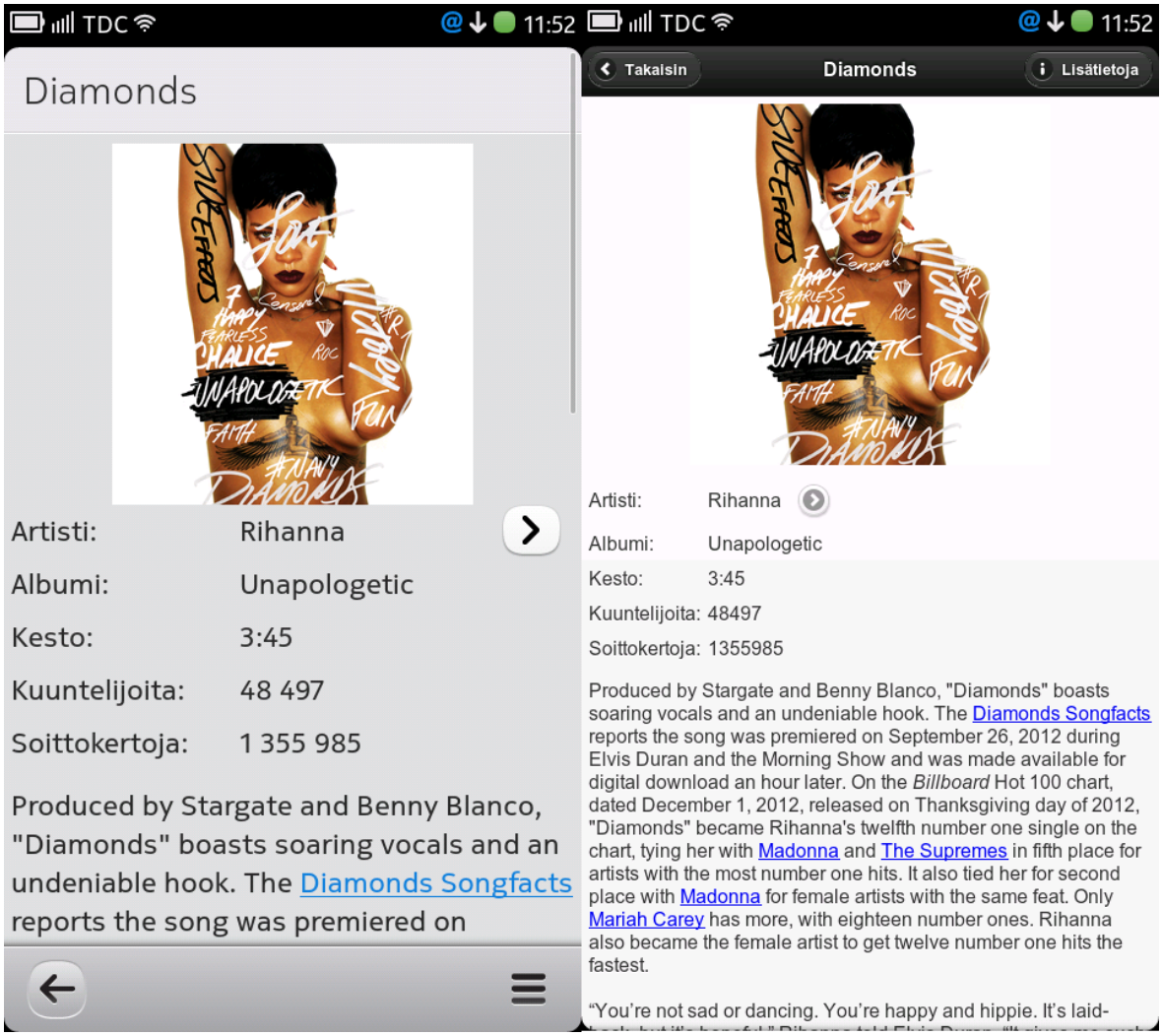
5.4.4 Sovelluskehysten monipuolisuus

Molemmat vertaillut sovelluskehykset tarjoavat yleisimmät käyttötapaukset kattavan joukon erilaisia käyttöliittymäkontrolleja. Kattavuudeltaan MeeGo Touch -sovelluskehys on kuitenkin jonkin verran edellä, sillä jQuery Mobile kärsii pienistä puutteista esimerkiksi

kontrolliensa lokalisointimahdollisuuksiensa osalta. jQuery Mobile -kirjaston rajapintojen toivoisi laajenevan kattamaan kaikki tilanteet siten, ettei sovelluskehittäjän tarvitsi tietää millaisia sisäisiä rakenteita sovelluskehys luo.

Selkeästi pidemmän korren MeeGo Touch -sovelluskehys vie myös erilaisten oheisominaisuuksien kattavuudella. Erityisesti kehyksen mukana tuleva ikonivalikoima on paljon käytännöllisempi kuin jQuery Mobile -kehykseen kuuluva 18 perusikonin valikoima. Lisäksi MeeGo Touch tukee esimerkiksi prosessien välistä kommunikaatiota, joka ei web-sovelluksessa ole mahdollista.

Natiivisovellus tarjoaa myös etuja erilaisten tietojen esitysmuodoissa. MeeGo Touch esimerkiksi tukee numeroiden esittämistä lokalisoidussa muodossa (Kuvio 31). Lisäksi listojen suodatuksessa voidaan käyttää mitä tahansa tietoa näkyvän tekstin sijaan. Lista-alkiot olisi myös voitu automaattisesti järjestää käyttäen mitä tahansa, myös näkymätöntä, tietoa. Järjestämisessä MeeGo Touch menee monipuolisuudessaan pidemmälle kuin Qt tukemalla käytetyn kielen huomioivaa järjestysalgoritmia, jossa esimerkiksi tšekin kielen ”ch”-alkuiset sanat katsotaan tulevan aakkosjärjestyksessä oikeaoppisesti ”h”-kirjaimen jälkeen. Lista-alkioille on myös mahdollista luoda pitkän painalluksen avaama kontekstivalikko.



Kuvio 31. Kappaletietonäkymän natiivi- ja hybridisovelluksessa.

Last.fm-palvelulta haettujen tietueiden joukossa on pidempiä kuvaustekstejä, jotka sisältävät yksinkertaista HTML-merkkausta esimerkiksi hyperlinkkien muodostamiseen (Kuvio 31). Web-sovelluksessa tällaisen tekstin käyttö on luonnollista ja upotetut hyperlinkit toimivat automaattisesti. MeeGo Touch -sovelluskehyksen tekstikontrollit puolestaan tukevat Qt:n tekstikontrollien tavoin tiettyä HTML-kielen alijoukkoa (Qt Reference Documentation 2012b). Mikäli käyttäjä aktivoi tekstin seassa olevan linkin, tekstikontrolli lähettää signaalin, jonka käsittely jää sovelluksen vastuulle. Qt tarjoaa *QDesktopServices*-luokan, jonka avulla aktivoitu URL on mahdollista avata järjestelmän oletusselaimessa. Web-sovelluksessa tällaiset linkit implementoitiin aukeamaan aina uuteen selainikkunaan. Hyb-

ridisovelluksessa puolestaan jouduttiin implementoitiin erillinen liitännäinen (Liite C) selainikkunan avaamiseksi.

Käytetyn Last.fm-palvelun tarjoama kuvamateriaali on pääosin neliön muotoista. Joissain yksittäisissä kuvissa kuvasuhde kuitenkin poikkeaa neliöstä. MeeGo Touch -sovellus näyttää tällaisen kuvan oletuksena keskitettynä kuvalle varattuun tilaan, käyttäen oikeaa kuvasuhdetta. jQuery Mobile puolestaan oletuksena venyttää lista-alkioiden kuvat neliön muotoon, eikä tarjoa valmista ratkaisua kuvasuhteen säilyttämiseksi. Toteutetussa sovelluksessa ongelma kierretään asettamalla kuvaelementti neliön muotoisen elementin sisään siten, että kuvalle on määritetty kiinteä leveys ja kuvasuhteen oikeana pitävä automaattinen korkeus. Tällöin kuva näytetään aina oikean levyisenä, mutta kuvasuhteeltaan leveämpi kuva näkyy varatun tilan ylälaidassa ja vastaavasti kuvasuhteeltaan korkeammasta kuvasta leikataan alapäästä osa pois.

jQuery Mobile tarjoaa laajan valikoiman erilaisia näkymänvaihtoanimaatioita. Toteutuksessa päädyttiin käyttämään MeeGo Touch -kirjaston käyttämää mallia, jossa uusi näkymä liikkuu vanhan päälle ruudun oikeasta laidasta ja palatessa edelliseen näkymään vasemmasta laidasta. Käyttöliittymän animaation suorituskyvyssä natiivisovellus vie kuitenkin selvästi pidemmän korren. Siinä missä natiivisovelluksen sivunvaihtoanimaatiot pyörivät pehmeästi, oletusselaimessa web-sovelluksen sivunvaihtoanimaatiot eivät toimineet ollenkaan ja hybridisovelluksessakin melko tahmeasti. Oletusselaimella myös listanäkymien vierityksessä sisällön piirtäminen tapahtui välillä viiveellä.

6 Yhteenveto

Aloitin tämän tutkimuksen käymällä läpi käyttöliittymien ja käyttöliittymäkehityksen keskeiset käsitteet ja toimintaperiaatteet. Tämän jälkeen tarkastelin markkinoilla olevia älypuhelinekosysteemejä sekä erilaisia työkaluja alustariippumattomien älypuhelinsovellusten rakentamiseen. Tutkimuksen empiirisessä osassa toteutin samankaltaisen sovelluksen vertailuun valitsemalleni MeeGo-laitealustalle käyttäen älypuhelinalustan natiivia sovelluskehystä sekä web-sovelluksiin suunnattua jQuery Mobile -sovelluskehystä. Suoritin sovellusten vertailun Nokian N9-älypuhelimella, jossa käytin web-sovellusta laitteen oletusselaimen lisäksi myös avoimen lähdekoodin Cordova-työkalulla toteutettuna hybridisovelluksena.

Tutkimuksen perusteella natiivisovelluksilla on yhä käyttöliittymäkehityksessä merkittävä etumatka web-sovelluksiin useilla eri osa-alueilla. Verrattuna jQuery Mobile -sovelluskehykseen, natiivi MeeGo Touch tarjoaa sovelluskehittäjille muun muassa kattavammat kontrollirajapinnat, yhtenäisen käyttöliittymästandardin, sisäänrakennetun lokalisointituen, monipuoliset kehitystyökalut, korkeamman suorituskyvyn, integroidun IPC-mekanismin ja kattavan, keskitetysti ladatun ikonivalikoiman. Myös Nielsen ja Budiu (2012, 41) suosittelevat natiivisovellusten käyttöä parhaan käyttökokemuksen saavuttamiseksi. Lisäksi natiivisovellus voi olla usein ainoa käytettävissä oleva lähestymistapa rakentaessa monimutkaisempia ohjelmistokokonaisuuksia, jotka vaativat esimerkiksi useaan prosessiin jaettua arkkitehtuuria, 3D-kiihdytystä tai erikoisempia verkkoprotokollia.

Web-sovellukset eivät kuitenkaan välttämättä kilpaile suoraan natiivisovellusten kanssa, vaan toimivat enemmänkin täydentävänä teknologiana, kuten myös VisionMobile tutkimuksessaan (2012b, 29) esittää. Web-sovelluksille ominaisia käyttökohteita ovat erityisesti yksinkertaisemmat ja Internet-keskeiset sovellukset, joissa nopea kehitys usealle kohdealustalle on tärkeässä roolissa. Tällöin asiakkaan laitteella ajetaan lähinnä kevyt sovelluskäyttöliittymä ja alemman tason toiminnot suoritetaan pääosin erillisellä web-palvelimella. Juuri tämän kaltaisille sovelluksille jQuery Mobile -sovelluskehys tuntuu tutkimuksen perusteella olevan suunniteltu. Vastoin yleistä käsitystä, web-sovellukset eivät kuitenkaan ole erityisen mutkaton tai helppo kehitystekniikka oppimiskynnykseltään, vaan

sovelluskehittäjän täytyy selvitä useiden eri ohjelmointi- ja kuvauskielten, asiakas-palvelinympäristön, erilaisten sovelluskehysten sekä selainyhteensopivuuden tuomien haasteiden kanssa (VisionMobile 2011a, 44).

Johtuen sovelluskauppojen tiukasta kontrollista, tällä hetkellä web-sovelluksia on mahdollista asentaa paikallisesti lähinnä hybridimuodossa. Toteutustekniikkana web-sovellukset kuitenkin kasvattavat suosiotaan nopeasti sovelluskehittäjien keskuudessa (VisionMobile 2011a) ja on vaikea sanoa varmuudella, mihin suuntaan tekniikka kehittyy. W3C yrittää standardoida web-sovellusten käyttämiä rajapintoja ja paketointia alustariippumattomiksi ja aika tulee näyttämään, yleistyvätkö sen standardoimat ratkaisut. Esimerkiksi tuleva Tizen-laitealusta hyödyntää W3C:n tämänhetkisiä standardeja, jolloin sama sovellus voidaan teoriassa asentaa mihin tahansa samoja standardeja noudattavaan web-sovellusympäristöön. Kuten useat tahot arvelevat (mm. Nielsen ja Budiu 2012, 34–43; Sarrafi 2012; VisionMobile 2011a, 57), on todennäköistä, että lähivuosina web-sovellukset tulevat olemaan merkittävä trendi mobiilialalla.

Vaikka suoritin tämän tutkimuksen 2011 ja 2012 aikana, tarkastelin älypuhelinten laitealustoja vuoden 2011 tilanteen näkökulmasta. Ala on kuitenkin jatkuvassa liikkeessä ja työn toteutuksen aikana sekä ilmestyi uusia että lopetettiin olemassa olevia laitealustoja ja alustariippumattomia tekniikoita. Työn toteutuksen kannalta merkittävin muutos oli tietysti Nokian luopuminen MeeGo-alustasta, jonka olin tuolloin jo valinnut mukaan alustojen väliseen vertailuun. Vuoden 2012 lähestyessä loppuaan alan kilpailu on kärjistynyt lähinnä Applen iOS- ja Googlen Andoid-laitealustojen väliseksi kaksintaisteluksi, jossa web-sovellukset ovat lujittaneet rooliaan alustojen välisenä sillanrakentajana. Markkinoille on kuitenkin tulossa uusien kilpailijoiden sukupolvi, johon kuuluu mm. Microsoftin Windows Phone 8, Samsungin vetämä Tizen ja suomalaisen Jollan MeeGo-alustan päälle rakentama Sailfish.

Lähteet

Android Open Source Project. “Frequently Asked Questions.” 2012. http://source.android.com/faqs.html#compatibility (haettu 6.9.2012).

Android Developers. “Activities.” 13.6.2012a. http://source.android.com/faqs.html#compatibility.

—. ”Android API Levels.” 14.2.2012b: http://developer.android.com/guide/appendix/api-levels.html.

—. ”Gestures.” 2012c. http://developer.android.com/design/patterns/gestures.html (haettu 21.8.2012).

—. ”Layouts.” 16.11.2012d. http://developer.android.com/guide/topics/ui/declaring-layout.html.

—. ”Platform Versions.” 4.9.2012e. http://developer.android.com/about/dashboards/index.html.

—. “Providing Descendant and Lateral Navigation.” 2012f. http://developer.android.com/training/design-navigation/descendant-lateral.html (haettu 13.9.2012).

—. ”User Interface.” 13.6.2012g. http://developer.android.com/guide/topics/ui/index.html.

—. ”What is the Android?” 2.9.2011a. http://developer.android.com/guide/basics/what-is-android.html.

—. ”What is the NDK?” 2011b. http://developer.android.com/sdk/ndk/overview.html (haettu 11.9.2011b).

Anglin, Todd. “Kendo UI on Windows Phone.” *Kendo UI Team Blog*. 6.2.2012. http://www.kendoui.com/blogs/teamblog/posts/12-02-06/kendo_ui_on_windows_phone.aspx.

Apple. “Apple’s App Store Downloads Top 15 Billion.” 7.7.2011a. http://www.apple.com/pr/library/2011/07/07Apples-App-Store-Downloads-Top-15-Billion.html.

—. “Event Handling Guide for iOS.” 10.3.2011b. http://developer.apple.com/library/ios/DOCUMENTATION/EventHandling/Conceptual/EventHandlingiPhoneOS/EventHandlingiPhoneOS.pdf>, 10.3.2011b.

—. “iOS Human Interface Guide.” 14.8.2012a. http://developer.apple.com/library/ios/DOCUMENTATION/UserExperience/Conceptual/MobileHIG/MobileHIG.pdf.

—. “iOS Technology Overview.” 12.10.2011c. http://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechOverview.pdf.

—. “Mac 101: Dashboard.” 13.11.2008. http://support.apple.com/kb/HT2492?viewlocale=fi_FI&locale=fi_FI.

—. “Product Images & Info.” 2012b. http://www.apple.com/pr/products/iphone/iphone.html (haettu 22.1.2012b).

—. “View Programming Guide for iOS.” 8.3.2011d. http://developer.apple.com/library/ios/documentation/windowsviews/conceptual/viewpg_iphoneos/ViewPG_iPhoneOS.pdf.

—. “What is Cocoa?” 13.12.2010. https://developer.apple.com/library/ios/#documentation/Cocoa/Conceptual/CocoaFundamentals/WhatIsCocoa/WhatIsCocoa.html.

Asleson, Ryan, ja Nathaniel T. Schutta. *Foundations of Ajax.* New York: Apress, 2006.

BadaDev.com. “BadaDev Will Cover Cross-Platform Technologies in 2011.” 18.2.2011. http://www.badadev.com/badadev-will-cover-cross-platform-technologies-in-2011/.

—. “Developing Widgets for Bada Devices.” 25.3.2010. http://www.badadev.com/developing-widgets-for-bada-devices/.

Barth, A. “The Web Origin Concept.” *Internet Engineering Task Force (IETF)*. 12 2011. http://tools.ietf.org/html/rfc6454 2011.

BBC News. “Nokia and Microsoft form Partnership.” 11.2.2011. http://www.bbc.co.uk/news/business-12427680.

Blanchette, Jasmin, ja Mark Summerfield. *C++ GUI Programming with Qt 4.* Ensimmäinen painos. Stoughton, Massachusetts: Prentice Hall, 2006. http://www.qtrac.eu/marksummerfield.html.

Bosch, Jan. ”From Software Product Lines to Software Ecosystems.” *International Software Product Line Conference.* San Francisco, 2009. http://janbosch.com/Jan_Bosch/Publications_files/SPLC09-SoftwareEcosystems-Accepted.pdf

Bowman, Shai. “MeeGo Touch UI Component Guidelines.” *Intel.* 2011. https://meego.com/sites/all/files/users/admin/meego_touch_ui_v1.2.pdf.

Buschmann, Frank, Regine Meunier, Hans Rohnert, Peter Sommerlad, ja Michael Stal. *Pattern-Oriented Software Architecture Volume 1: A System of Patterns.* Chichester: John Wiley & Sons, 1996.

Campbell, Piers R. J., ja Faheem Ahmed. ”An Assessment of Mobile OS-Centric Ecosystems.” *Journal of Theoretical and Applied Electronic Commerce Research* (Universidad de Talca - Chile) 6, nro 2 (Elokuu 2011): 50 - 62.

Cooper, Alan, Robert Reimann, ja Dave Cronin. *About Face 3: The Essentials of Interaction Design*. Kolmas painos. Indianapolis: Wiley Publishing, 2007.

Digia. “Digia to acquire Qt from Nokia.” 9.8.2012. http://www.digia.com/en/Home/Company/Press/2012/Digia-to-acquire-Qt-from-Nokia/.

Ezust, Alan, ja Paul Ezust. *An Introduction to Design Patterns in C++ with Qt 4*. Stoughton, Massachusetts: Prentice Hall, 2006. http://www.informit.com/store/product.aspx?isbn=0131879057.

Fayad, Mohamed, ja Douglas C. Schmidth. ”Object-oriented application frameworks.” Leikannut Diane Crawford. *Communications of the ACM* 40, nro 10 (Lokakuu 1997): 32 - 38.

Fitzek, Frank H.P., Tony Torp, ja Tommi Mikkonen. *Qt for Symbian.* Chichester: John Wiley & Sons, 2012.

Forum Nokia. “Web Runtime Widgets.” 2011. http://library.forum.nokia.com/topic/Web_Developers_Library/GUID-57DAF2D2-28DF-45F4-B901-05F4B2CFF109.html (haettu 11.62011).

Galitz, Wilbert O. *The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques*. Kolmas painos. Indianapolis: Wiley Publishing, 2007.

Garrett, Jesse James. “Ajax: A New Approach to Web Applications.” *Adaptive Path*. 18.2.2005. http://adaptivepath.com/ideas/ajax-new-approach-web-applications.

Gartner. ”Gartner Says Worldwide Mobile Phone Sales Grew 35 Percent in Third Quarter 2010; Smartphone Sales Increased 96 Percent.” 10.11.2010. http://www.gartner.com/it/page.jsp?id=1466313.

Ideaworks3D. “Buy Marmalade SDK.” 2012a. http://www.madewithmarmalade.com/buy (haettu 14.9.2012).

—. “Supported Platforms.” 2012b. http://www.madewithmarmalade.com/marmaladesdk/supported-platforms (haettu 14.9.2012).

—. “UI frameworks.” 2012c. http://www.madewithmarmalade.com/marmaladesdk/features/ui-frameworks (haettu 14.9.2012).

—. “You’ve got Marmalade. What will you do with it? - Loader/Binary System.” 2012d. https://www.madewithmarmalade.com/devnet/documentation#/main/native/overview/concepts/loaderbinarysystem.html (haettu 14.9.2012d).

Ideum. “Open Source Multitouch Gesture Library and Illustrations.” *GestureWorks*. 2012. http://gestureworks.com/features/open-source-gestures/ (haettu 21.8.2012).

Idsinga, Andy. “Introduction to Web Runtime on MeeGo.” *Intel AppUp developer program*. 6.11.2010. http://appdeveloper.intel.com/en-us/article/introduction-web-runtime-meego.

IHS Screen Digest. “Apple Maintains Dominance of Mobile Application Store Market in 2010.” 15.2.2011. http://press.ihs.com/press-release/product-design-supply-chain/apple-maintains-dominance-mobile-application-store-market-.

jQuery Foundation. “Announcing jQuery Mobile 1.2.0 Final.” 2.10.2012a. http://jquerymobile.com/blog/2012/10/02/announcing-jquery-mobile-1-2-0-final/.

—. “Fixed toolbars.” 2012b. http://jquerymobile.com/demos/1.2.0/docs/toolbars/bars-fixed.html (haettu 5.10.2012).

—. “jQuery Mobile Overview.” 2012c. http://jquerymobile.com/demos/1.2.0/docs/about/intro.html (haettu 5.10.2012).

—. “List basics & API.” 2012d. http://jquerymobile.com/demos/1.2.0/docs/lists/docs-lists.html (haettu 5.10.2012).

Koch, Peter-Paul. ”Event order.” *QuirksMode*. 2012. http://www.quirksmode.org/js/events_order.html (haettu 27.8.2012).

Kruzeniski, Mike. ”From Transportation to Pixels.” *The Windows Phone Developer Blog*. 17.2.2011. http://windowsteamblog.com/windows_phone/b/wpdev/archive/2011/02/16/from-transportation-to-pixels.aspx.

Laitila, Teemu. ”Google ostaa Motorolan puhelinliiketoiminnan.” *AfterDawn*. 15.8.2011. http://www.puhelinvertailu.com/uutiset.cfm/2011/08/15/google_ostaa_motorolan_puhelinliiketoiminnan.

Last.fm. “REST Requests.” *Last.fm Web Services*. 4.7.2012. http://www.last.fm/api/rest.

Lemmetyinen, Heidi. ”Introducing the Nokia N9: all it takes is a swipe!” *Nokia*. 21.6.2011. http://conversations.nokia.com/2011/06/21/introducing-the-nokia-n9-all-it-takes-is-a-swipe/.

Lie, Håkon Wium, ja Bert Boss. ”The CSS saga.” Teoksessa *Cascading Style Sheets, designing for the Web*. Toinen painos. Addison Wesley, 1999. http://www.w3.org/Style/LieBos2e/history/.

LiMo Foundation. “History.” 2011. http://www.limofoundation.org/en/history.html (haettu 30.10.2011).

Linux Foundation. “Welcome to Tizen!” 27.9.2011. https://www.tizen.org/blogs/dawnfoster/2011/welcome-tizen.

—. “Tizen Web UI Guide – Introduction.” 2012. https://developer.tizen.org/help/index.jsp?topic=%2Forg.tizen.help.web.ui.guide%2FIntroduction.html (haettu 6.5.2012).

MeeGo Wiki. “Theming in MeeGo Touch.” 21.12.2010. http://wiki.meego.com/Theming_in_MeeGo_Touch.

Meier, Reto. *Android 2 Application Development*. Indianapolis: Wiley Publishing, 2010.

Microsoft. “Qt to WP7 - Chapter 1: Introducing Windows Phone Platform to Symbian Qt Application Developers.” *Microsoft Developer Network*. 19.9.2011. http://windowsphone.interoperabilitybridges.com/articles/qt-to-wp7-chapter-1-introducing-windows-phone-platform-to-symbian-qt-application-developers.

Microsoft. “UI Design and Interaction Guide for Window Phone 7.” Versio 2.0. 2010. http://go.microsoft.com/fwlink/?LinkID=183218.

Microsoft. “What’s a Windows Store app?” *Microsoft Developer Network*. 2012. http://msdn.microsoft.com/en-US/library/windows/apps/hh974576 (haettu 10.9.2012).

Myers, Brad A. ”Graphical User Interface Programming.” Teoksessa *Computer Science Handbook*, leikannut Allen B. Tucker. Toinen painos. Chapman & Hall/CRC, 2004.

Nielsen, Jakob, ja Raluca Budiu. *Mobile Usability*. Berkeley: New Riders, 2012.

Nielsen, Jakob. ”Usability 101: Introduction to Usability.” *Jakob Nielsen’s Alertbox*. 25.8.2003. http://www.useit.com/alertbox/20030825.html.

—. ”Success Rate: The Simplest Usability Metric.” *Jakob Nielsen’s Alertbox*. 18.2.2001. http://www.useit.com/alertbox/20010218.html.

Nokia. “Getting Started with Series 40 Web Apps.” Versio 1.0. 7.4.2011. http://tools.nokia.com/wt/doc/s40/Series_40_Web_App_Getting_Started_Guide_Doc.pdf

—. “MeeGo Touch Reference Documentation - Installing MeeGo Touch.” 2010a. http://harmattan-dev.nokia.com/docs/platform-api-reference/xml/daily-docs/libmeegotouch/installation.html.

—. “MeeGo Touch Reference Documentation – Internationalisation.” 2010b http://harmattan-dev.nokia.com/docs/platform-api-reference/xml/daily-docs/libmeegotouch/i18n.html.

—. “MeeGo Touch Reference Documentation - Introduction to MeeGo Touch.” 2010c http://harmattan-dev.nokia.com/docs/platform-api-reference/xml/daily-docs/libmeegotouch/introduction.html.

—. “Nokia N9 UX Guidelines - Navigation Structures.” 2012a http://harmattan-dev.nokia.com/docs/ux/pages/Navigation_Structures.html (haettu 13.9.2012).

—. “Nokia N9 UX Guidelines - Portrait & Landscape.” 2012b http://harmattan-dev.nokia.com/docs/ux/pages/Portrait_vs_Landscape.html (haettu 14.9.2012).

—. “Nokia Store Content Guidelines.” Versio 1.4.1. 2.5.2012c. https://admin.support.publish.nokia.com/wp-content/uploads/2012/05/Nokia-Store-Content-Guidelines-1.4.1.a.pdf.

—. “S60 5th Edition brings new, advanced multimedia and Internet experiences to devices with touch user interfaces and sensor technologies.” 2.10.2008. http://press.nokia.com/2008/10/02/s60-5th-edition-brings-new-advanced-multimedia-and-internet-experiences-to-devices-with-touch-user-interfaces-and-sensor-technologies/.

—. “Symbian OS: From ANSI C/C++ To Symbian C++.” Versio 1.0. 22.3.2004. http://www.zdnetasia.com/whitepapers/symbian-os-from-ansi-cc-to-symbian-c-45234897/2/.

Nokia Developer. “Application development tutorial - The basics.” 2008a. http://library.developer.nokia.com/index.jsp?topic=/S60_5th_Edition_Cpp_Developers_Library/GUID-35228542-8C95-4849-A73F-2B4F082F0C44/sdk/doc_source/NewStarter/1-basics.html.

—. “Symbian Web Runtime versions and device support.” 21.6.2011a. http://www.developer.nokia.com/Resources/Library/Web/web-apps/symbian-web-runtime/symbian-web-runtime-versions-and-device-support.html.

—. “Using Uikon - Uikon Overview.” 2008b. http://library.developer.nokia.com/index.jsp?topic=/S60_5th_Edition_Cpp_Developers_Library/GUID-35228542-8C95-4849-A73F-2B4F082F0C44/sdk/doc_source/guide/Application-Framework-subsystem-guide/uikon/UikonOverview.guide.html.

—. “Using Platform Services.” 18.10.2011b. http://www.developer.nokia.com/Resources/Library/Web/web-apps/symbian-web-runtime/using-platform-services.html.

—. “Viewport support.” 3.2.2012 http://www.developer.nokia.com/Resources/Library/Web/#!nokia-browsers/nokia-browser-85-for-meego-12-harmattan/viewport-support.html.

—. “Widget features.” 11.5.2011c. http://www.developer.nokia.com/Resources/Library/Web/web-apps/symbian-web-runtime/web-runtime-widgets/widget-features.html.

Nokia Developer Wiki. ”GUI Framework.” 24.3.2011a. http://www.developer.nokia.com/Community/Wiki/GUI_Framework.

—. ”Uikon-Eikon-Avkon-Qikon.” 24.3.2011b. http://www.developer.nokia.com/Community/Wiki/Uikon-Eikon-Avkon-Qikon.

—. ”MeeGo 1.2 Harmattan Architecture Layers.” 13.6.2012. http://www.developer.nokia.com/Community/Wiki/MeeGo_1.2_Harmattan_Architecture_Layers.

Oracle. “Using Top-Level Containers.” *The Java Tutorials*. 2012. http://docs.oracle.com/javase/tutorial/uiswing/components/toplevel.html (haettu 28.11.2012).

Orlowski, Andrew. “Symbian, The Secret History: Dark Star – How it almost never set sail.” *The Register*. 23.11.2010. http://www.theregister.co.uk/2010/11/23/symbian_history_part_one_dark_star/.

Ortiz, C. Enrique. “Introduction to jQuery Mobile.” *IBM DeveloperWorks*. 1.2.2011. http://www.ibm.com/developerworks/web/library/wa-jqmobile/.

PhoneGap Wiki. “Getting Started with PhoneGap BlackBerry WebWorks.” 3.10.2011. http://wiki.phonegap.com/w/page/31930982/Getting%20Started%20with%20PhoneGap%20BlackBerry%20WebWorks.

Pocket PC Central. ”The Basics: Pocket PC Phone v. Windows Mobile Smartphone.” 2011. http://pocketpccentral.net/smartphone/help/general/smartphone_ppcp.htm (haettu 31.7.2011).

Q-Success. ”Usage of JavaScript libraries for websites.” *W3Techs*. 2012. http://w3techs.com/technologies/overview/javascript_library/all (haettu 12.12.2012).

Qt Developer Network. ”Is Qt binary compatible?” 2011a. http://developer.qt.nokia.com/faq/answer/is_qt_binary_compatible (haettu 18.9.2011).

—. ”QStyle Class Reference.” 2012a. http://qt-project.org/doc/qt-4.8/qstyle.html (haettu 20.8.2012).

—. ”Qt Mobility Project.” 2011b. http://developer.qt.nokia.com/wiki/Category:Developing_with_Qt::QtMobility (haettu 2.10.2011).

—. ”Support for Symbian.” 29.8.2012b. http://qt-project.org/wiki/Support-for-Symbian.

Qt Reference Documentation. ”All Modules.” 2011a. http://doc.qt.nokia.com/latest/modules.html (haettu 18.9.2011).

—. ”The Interview Framework.” 2011b. http://doc.qt.nokia.com/4.7-snapshot/qt4-interview.html (haettu 24.9.2011).

—. ”Model/View Programming.” 2012a. http://doc-snapshot.qt-project.org/4.8/model-view-programming.html (haettu 19.6.2012).

—. ”Supported HTML Subset.” 2012b. http://doc.qt.digia.com/qt/richtext-html-subset.html (haettu 26.11.2012).

—. ”Supported Platforms.” 2012c. http://doc.qt.digia.com/qt/supported-platforms.html (haettu 13.12.2012).

—. ”Using a Designer UI File in Your Application.” 2011c. http://doc.qt.nokia.com/latest/designer-using-a-ui-file.html (haettu 24.9.2011).

—. ”Using the Meta-Object Compiler (moc).” 2011d. http://doc.qt.nokia.com/latest/moc.html (haettu 18.9.2011).

Reimer, Jeremy. ”A History of the GUI.” *Ars Technica*. 5.5.2005. http://arstechnica.com/features/2005/05/gui/.

Research in Motion. “Why Are BlackBerry Widgets Repackaged?” *Inside BlackBerry Developer Blog*. 20.4.2010. http://devblog.blackberry.com/2010/04/why-are-blackberry-widgets-repackaged/.

Sarrafi, Ali. “HTML5 and Mobile Apps - the road ahead.” *Straightforward*. 13.3.2012. http://www.straightforward.se/storyserver/html5-javascript-mobile-apps-road-ahead.

Saxena, Sunil. “MeeGo Architecture Layer View.” *MeeGo*. 22.10.2010. https://meego.com/developers/meego-architecture/meego-architecture-layer-view.

Sencha. “Built on HTML5 - Features of Sencha Touch 2.” 2012a. http://www.sencha.com/products/touch/features/ (haettu 12.9.2012).

—. “Sencha Touch – Licensing.” 2012b. http://www.sencha.com/products/touch/license/ (haettu 14.9.2012).

—. “Sencha Touch - Road Map.” 2012c. http://www.sencha.com/products/touch/road-map/ (haettu 12.9.2012).

Sousou, Imad. “Welcome to MeeGo.” *MeeGo*. 15.10.2010. https://meego.com/community/blogs/imad/2010/welcome-meego.

Software Informer. ”Yahoo! Widgets Wiki.” 2009. http://yahoo-widgets.software.informer.com/wiki/ (haettu 14.4.2009).

Symbian Foundation. ” Symbian Foundation has transitioned to a licensing body.” 2012. http://licensing.symbian.org/ (haettu 9.12.2012).

Talk Android. ”How To Check For Android Updates On Your Android Phone.” 2011. http://www.talkandroid.com/guides/check-for-android-updates/ (haettu 11.9.2011).

Telcontar.net. ”The EIKON GUI.” 23.1.2008. http://telcontar.net/Misc/GUI/EIKON/.

Telerik. “How-To: Build Apps With Kendo UI Mobile.” *Kendo UI Docs*. 2012a. http://docs.kendoui.com/howto/build-apps-with-kendo-ui-mobile (haettu 12.12.2012).

Telerik. “Licensing.” *Kendo UI*. 2012b. http://www.kendoui.com/faq/licensing.aspx (haettu 12.12.2012).

Thelin, Johan. ”Quick User Interfaces with Qt.” *Linux Journal*, nro 204 (Huhtikuu 2011). 30.6.2011a. http://www.linuxjournal.com/article/10919

—. ”Using CMake to Build Qt Projects.” *Qt Developer Network*. http://developer.qt.nokia.com/quarterly/view/using_cmake_to_build_qt_projects (haettu 18.9.2011b).

Thurrott, Paul. *Windows Phone 7 Secrets*. Indianapolis: Wiley Publishing, 2011.

Tizen. “Developers.” 2011. https://www.tizen.org/developers (haettu 30.10.2011).

Tucker, Allen B., ja Robert E. Noonan. ”Event-Driven Programming.” Teoksessa *Computer Science Handbook*, leikannut Allen B. Tucker. Toinen painos. Chapman & Hall/CRC, 2004.

Tømmerholt, Hans S. “The Opera Widgets Manager application.” *Opera Software*. 9.2.2010. http://dev.opera.com/articles/view/the-opera-widgets-manager-application/

van Dam, Andries. “Post-WIMP user interface.” *Communications of the ACM* 40, nro 2 (Helmikuu 1997): 63 - 67.

VisionMobile. “Developer Economics 2011 - How developers and brands are making money in the mobile app economy.” 2011a.

—. “Mobile Platforms: The Clash of Ecosystems - A critical analysis of mobile platforms and the battle for dominance.” 2011b.

—. “Cross-Platform Development Tools 2012 - Bridging the worlds of mobile apps and the web.” 2012a.

—. “Developer Economics 2012 - The new mobile app economy.” 2012b.

W3C. “About W3C.” 2012. http://www.w3.org/Consortium/ (haettu 5.2.2012).

W3C. ”Widgets 1.0: The Widget Landscape (Q1 2008).” Leikannut Marcos Caceres. 14.4.2008. http://www.w3.org/TR/widgets-land/.

—. “Widget Interface.” Leikannut Marcos Caceres. 22.5.2012. http://www.w3.org/TR/2012/PR-widgets-apis-20120522/.

—. “Widget Packaging and XML Configuration.” Leikannut Marcos Caceres. 27.9.2011. http://www.w3.org/TR/widgets/.

—. “XMLHttpRequest.” 3.8.2011. http://www.w3.org/TR/XMLHttpRequest/.

WAC. “Developer FAQs”. 2011a. http://www.wacapps.net/faqs1 (haettu 30.10.2011).

—. “Our Members.” 2011b. http://www.wacapps.net/our-members (haettu 30.10.2011).

—. “What is WAC?” 2011c. http://www.wacapps.net/what-is-wac (haettu 30.10.2011).

Windows Mobile Team Blog. ”Windows Mobile 6.5 - What’s in for developers?” 18.3.2009. http://blogs.msdn.com/b/windowsmobile/archive/2009/03/18/windows-mobile-6-5-what-s-in-for-developers.aspx.

Wroblewski, Luke. ”Touch Gesture Reference Sheet.” 20.4.2010. http://www.lukew.com/touch/.

Xinguan, Shin, ja Zhang Wei. “Qt-based Mobile Application GUI Style for Smart Phone Operating System.” *IEEE*. 2010.

Liitteet

A Toteutus natiivisovelluksena

main.cpp

```
#include <MApplication>
#include <MApplicationWindow>
#include "mainwindow.h"

int main(int argc, char *argv[])
{
    MApplication app(argc, argv);
    MApplicationWindow w;
    w.show();
    MainWindow p;
    p.appear(&w);
    return app.exec();
}
```

mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QModelIndex>
#include <MApplicationPage>

class QStandardItemModel;
class QStandardItem;
class QSortFilterProxyModel;
class MLinearLayoutPolicy;
class MList;
class MLabel;
class MTextEdit;
class MAction;
class MButton;
class ListCellCreator;
class Track;
class Artist;
class MusicEngine;
class DataItem;

class MainWindow : public MApplicationPage{
    Q_OBJECT
public:
    MainWindow();
    virtual void createContent();
    QStandardItem* itemWithValue( QStandardItem* item, int role,
        const QVariant& value);
    QStandardItem* itemWithValue( QStandardItemModel& itemModel, int role,
        const QVariant& value );
    QStandardItem* itemWithValue( int role, const QVariant& value );

private slots:
    void onListItemSelected(const QModelIndex &index);
```

```
    void onFetchMoreTracksClicked();
    void onFetchMoreArtistsClicked();
    void onMediumImageFetched(DataItem& item);
    void panningStarted();
    void panningStopped();
    void refreshTabContent();
    void search();
    void showSearchTab();
    void showTracksTab();
    void showArtistsTab();
    void onTracksSearched( QList<Track*> list );
    void onArtistsSearched( QList<Artist*> list );
    void onTracksFetched( int total, int perPage, int page, QList<Track*> list );
    void onArtistsFetched( int total, int perPage, int page, QList<Artist*> list );
    void onTopTracksErrorCallback();
    void onTopArtistsErrorCallback();
    void fetchMissingData();

    void enableToolbarTabMode();

private:
    QList<QModelIndex> mapToSource( QList<QModelIndex>& proxyIndexes,
        QSortFilterProxyModel& proxyModel );
    QList<QStandardItem*> getListItems( QList<QModelIndex>& indexes,
        QStandardItemModel& itemModel );
    bool getLeafNodeIndexes( QList<QModelIndex>& results,
        QAbstractItemModel& itemModel, QModelIndex index=QModelIndex(),
        QModelIndex stopAtIndex=QModelIndex() );
    bool fetchMissingData(QList<QStandardItem*> list);
    MList* createList( ListCellCreator* cellCreator, QStandardItemModel* itemModel );

private:
    bool m_panningList;
    bool m_tracksFetched;
    bool m_artistsFetched;
    bool m_fetchingMissingData;
    MLinearLayoutPolicy* m_tracksLayoutPolicy;
    MLinearLayoutPolicy* m_artistsLayoutPolicy;
    MLinearLayoutPolicy* m_searchLayoutPolicy;
    MLinearLayoutPolicy* m_emptyLayoutPolicy;
    MLinearLayoutPolicy* m_headerLayoutPolicy;
    QStandardItemModel* m_searchResultsItemModel;
    QStandardItemModel* m_tracksItemModel;
    QStandardItemModel* m_artistsItemModel;
    MList* m_searchList;
    MList* m_tracksList;
    MList* m_artistsList;
    MButton* m_refreshButton;
    MButton* m_tracksMoreButton;
    MButton* m_artistsMoreButton;
    MTextEdit* m_searchInput;
    MLabel* m_emptyLabel;
    MLabel* m_headerLabel;
    MAction* m_searchTabButton;
    MAction* m_artistsTabButton;
    MAction* m_tracksTabButton;
    MusicEngine* m_musicEngine;
};

#endif // MAINWINDOW_H
```

mainwindow.cpp

```
#include <QStandardItemModel>
#include <QAbstractItemModel>
#include <QStringListModel>
#include <QActionGroup>
#include <QGraphicsLinearLayout>
#include <QTimer>
#include <MList>
#include <MLayout>
#include <MLinearLayoutPolicy>
#include <MSortFilterProxyModel>
#include <MLabel>
#include <MApplicationPage>
#include <MListFilter>
#include <MAction>
#include <MApplicationWindow>
#include <MApplication>
#include <MToolBar>
#include <MTextEdit>
#include <MButton>
#include <MLocale>
#include "listcellcreator.h"
#include "mainwindow.h"
#include "musicengine.h"
#include "trackdetailspage.h"
#include "artistdetailspage.h"

#define ITEMS_PER_LIST_GROUP 10

MainWindow::MainWindow()
    : MApplicationPage(),
    m_panningList(false),
    m_tracksFetched(false),
    m_artistsFetched(false),
    m_fetchingMissingData(false),
    m_tracksLayoutPolicy(0),
    m_artistsLayoutPolicy(0),
    m_emptyLayoutPolicy(0),
    m_headerLayoutPolicy(0),
    m_searchResultsItemModel(0),
    m_tracksItemModel(0),
    m_artistsItemModel(0),
    m_searchList(0),
    m_tracksList(0),
    m_artistsList(0),
    m_refreshButton(0),
    m_tracksMoreButton(0),
    m_artistsMoreButton(0),
    m_searchInput(0),
    m_emptyLabel(0),
    m_headerLabel(0),
    m_searchTabButton(0),
    m_artistsTabButton(0),
    m_tracksTabButton(0),
    m_musicEngine( new MusicEngine("en",this) ){
}

void MainWindow::createContent(){
    QGraphicsWidget* pageWidget = centralWidget();
    MLayout* pageLayout = new MLayout( pageWidget );
    m_tracksLayoutPolicy = new MLinearLayoutPolicy( pageLayout, Qt::Vertical );
    m_tracksLayoutPolicy->setContentsMargins( 0, 0, 0, 0 );
    m_tracksLayoutPolicy->setSpacing( 0 );
    m_artistsLayoutPolicy = new MLinearLayoutPolicy( pageLayout, Qt::Vertical );
    m_artistsLayoutPolicy->setContentsMargins( 0, 0, 0, 0 );
    m_artistsLayoutPolicy->setSpacing( 0 );
    m_searchLayoutPolicy = new MLinearLayoutPolicy( pageLayout, Qt::Vertical );
```

```
m_searchLayoutPolicy->setContentsMargins( 0, 0, 0, 0 );
m_searchLayoutPolicy->setSpacing( 0 );
m_emptyLayoutPolicy = new MLinearLayoutPolicy( pageLayout, Qt::Vertical );
m_emptyLayoutPolicy->setContentsMargins( 0, 0, 0, 0 );
m_emptyLayoutPolicy->setSpacing( 0 );

MWidgetController* headerWidget = new MWidgetController();
headerWidget->setStyleName("CommonHeaderPanel");
MLayout* headerLayout = new MLayout( headerWidget );
m_headerLayoutPolicy = new MLinearLayoutPolicy(headerLayout,Qt::Horizontal);
m_headerLayoutPolicy->setContentsMargins(0,0,0,0);
m_headerLayoutPolicy->setSpacing(0);
m_headerLabel = new MLabel();
m_headerLabel->setStyleName("CommonHeader");
m_headerLabel->setWordWrap(false);
m_headerLabel->setTextElide(true);
m_headerLayoutPolicy->addItem(m_headerLabel, Qt::AlignTop);
m_refreshButton = new MButton();
m_refreshButton->setIconID("icon-s-refresh");
m_refreshButton->setTextVisible(false);
m_refreshButton->setStyleName("CommonFramePanelRightIcon");
m_headerLayoutPolicy->addStretch();
m_headerLayoutPolicy->addItem( m_refreshButton, Qt::AlignVCenter );
QObject::connect(m_refreshButton,SIGNAL(clicked()),
    this, SLOT(refreshTabContent()));
m_tracksLayoutPolicy->addItem( headerWidget );
m_artistsLayoutPolicy->addItem( headerWidget );
m_searchLayoutPolicy->addItem( headerWidget );
m_emptyLayoutPolicy->addItem( headerWidget );

m_emptyLabel = new MLabel();
m_emptyLabel->setWordWrap( true );
m_emptyLabel->setAlignment( Qt::AlignCenter );
m_emptyLabel->setSizePolicy( QSizePolicy::Expanding, QSizePolicy::Expanding );
m_emptyLabel->setStyleName( "CommonEmptyStateTitle" );
m_emptyLayoutPolicy->addItem( m_emptyLabel );

enableToolbarTabMode();
QActionGroup* tabs = new QActionGroup(this);
tabs->setExclusive( true );
//% "Search"
m_searchTabButton = new MAction("icon-m-content-search",
    qtTrId("qtn_tabs_search"),this);
m_searchTabButton->setLocation(MAction::ToolBarLocation);
m_searchTabButton->setCheckable(true);
m_searchTabButton->setChecked(true);
m_searchTabButton->setEnabled(true);
m_searchTabButton->setActionGroup(tabs);
QObject::connect( m_searchTabButton, SIGNAL(triggered()),
    this, SLOT(showSearchTab()));
addAction(m_searchTabButton);
//% "Tracks"
m_tracksTabButton = new MAction("icon-m-content-audio",
    qtTrId("qtn_tabs_tracks"), this);
m_tracksTabButton->setLocation(MAction::ToolBarLocation);
m_tracksTabButton->setCheckable(true);
m_tracksTabButton->setChecked(false);
m_tracksTabButton->setEnabled(true);
m_tracksTabButton->setActionGroup(tabs);
QObject::connect( m_tracksTabButton, SIGNAL(triggered()),
    this, SLOT(showTracksTab()) );
addAction(m_tracksTabButton);
//% "Artists"
m_artistsTabButton = new MAction("icon-m-content-artist",
    qtTrId("qtn_tabs_artists"), this);
m_artistsTabButton->setLocation(MAction::ToolBarLocation);
```

```
    m_artistsTabButton->setCheckable(true);
    m_artistsTabButton->setChecked(false);
    m_artistsTabButton->setEnabled(true);
    m_artistsTabButton->setActionGroup(tabs);
    QObject::connect( m_artistsTabButton, SIGNAL(triggered()),
        this, SLOT(showArtistsTab()) );
    addAction(m_artistsTabButton);

    QObject::connect( m_musicEngine, SIGNAL(searchTracksCallback(QList<Track*>)),
        this, SLOT(onTracksSearched(QList<Track*>)) );
    QObject::connect( m_musicEngine, SIGNAL(searchArtistsCallback(QList<Artist*>)),
        this, SLOT(onArtistsSearched(QList<Artist*>)) );
    QObject::connect( m_musicEngine,
        SIGNAL(topTracksCallback(int,int,int,QList<Track*>)),
        this, SLOT(onTracksFetched(int,int,int,QList<Track*>)) );
    QObject::connect( m_musicEngine,
        SIGNAL(topArtistsCallback(int,int,int,QList<Artist*>)),
        this, SLOT(onArtistsFetched(int,int,int,QList<Artist*>)) );
    QObject::connect( m_musicEngine, SIGNAL(topTracksErrorCallback()),
        this, SLOT(onTopTracksErrorCallback()) );
    QObject::connect( m_musicEngine, SIGNAL(topArtistsErrorCallback()),
        this, SLOT(onTopArtistsErrorCallback()) );
    QObject::connect( this, SIGNAL(appearing()),
        this, SLOT(enableToolbarTabMode()) );

    showSearchTab();
}

QStandardItem* MainWindow::itemWithValue(QStandardItem* item, int role,
    const QVariant& value){
    if( item->data(role) == value ) return item;
    else if( item->hasChildren() ){
        for( int i = 0; i < item->rowCount(); i++ ){
            QStandardItem* correctItem =
                itemWithValue( item->child(i,0), role, value );
            if( correctItem ) return correctItem;
        }
    }
    return 0;
}

QStandardItem* MainWindow::itemWithValue( QStandardItemModel& itemModel, int role,
    const QVariant& value ){
    for( int i = 0; i < itemModel.rowCount(); i++ ){
        QStandardItem* item = itemModel.item(i,0);
        QStandardItem* correctItem = itemWithValue( item, role, value );
        if( correctItem ) return correctItem;
    }
    return 0;
}

QStandardItem* MainWindow::itemWithValue( int role, const QVariant& value ){
    QStandardItem* match = 0;
    if( m_searchResultsItemModel )
        match = itemWithValue( *m_searchResultsItemModel, role, value );
    if( !match && m_tracksItemModel)
        match = itemWithValue( *m_tracksItemModel, role, value );
    if( !match && m_artistsItemModel )
        match = itemWithValue( *m_artistsItemModel, role, value );
    return match;
}

void MainWindow::onListItemSelected(const QModelIndex &index){
    unsigned int id = index.data( ListCellCreator::IdRole ).toInt();
    QString type = index.data( ListCellCreator::TypeRole ).toString();
    if( type == "track" ){
```

```
        Track* track = m_musicEngine->getTrackById( id );
        TrackDetailsPage* page = new TrackDetailsPage(*m_musicEngine,*track,this);
        page->appear(MSceneWindow::DestroyWhenDismissed);
    }
    else if( type == "artist" ){
        Artist* artist = m_musicEngine->getArtistById( id );
        ArtistDetailsPage* page = new ArtistDetailsPage(*m_musicEngine,*artist,this);
        page->appear(MSceneWindow::DestroyWhenDismissed);
    }
}

void MainWindow::onFetchMoreTracksClicked(){
    m_tracksMoreButton->setEnabled(false);
    m_musicEngine->fetchMoreTracks();
}

void MainWindow::onFetchMoreArtistsClicked(){
    m_artistsMoreButton->setEnabled(false);
    m_musicEngine->fetchMoreArtists();
}

void MainWindow::onMediumImageFetched(DataItem& item){
    QStandardItem* listItem = itemWithValue( ListCellCreator::IdRole, item.id );
    if( listItem && !item.mediumImage.isNull() )
        listItem->setData( item.mediumImage, ListCellCreator::PixmapRole );
    m_fetchingMissingData = false;
    fetchMissingData();
}

void MainWindow::panningStarted(){
    m_panningList = true;
}

void MainWindow::panningStopped(){
    m_panningList = false;
    fetchMissingData();
}

void MainWindow::refreshTabContent(){
    if( m_tracksTabButton->isChecked() ){
        if(m_tracksItemModel) m_tracksItemModel->clear();
        m_musicEngine->clearFetchedTopTracks();
        m_tracksFetched = false;
        showTracksTab();
        m_fetchingMissingData = false;
    }
    else if( m_artistsTabButton->isChecked() ){
        if(m_artistsItemModel) m_artistsItemModel->clear();
        m_musicEngine->clearFetchedTopArtists();
        m_artistsFetched = false;
        showArtistsTab();
        m_fetchingMissingData = false;
    }
}

void MainWindow::search(){
    if( !m_searchInput->text().isEmpty() ){
        m_searchResultsItemModel->clear();
        m_searchInput->clearFocus();
        m_musicEngine->clearBufferedDataItems();
        m_musicEngine->search( m_searchInput->text() );
    }
}

void MainWindow::showSearchTab(){
    //% "Search"
```

```
    m_headerLabel->setText(qtTrId("qtn_search_header"));
    if( !m_searchInput ){
        m_searchInput = new MTextEdit();
        m_searchInput->setStyleName("CommonTopInputField");
        m_searchLayoutPolicy->addItem( m_searchInput );
        QObject::connect( m_searchInput, SIGNAL(returnPressed()),
            this, SLOT(search()) );
        ListCellCreator* cellCreator = new ListCellCreator();
        m_searchResultsItemModel = new QStandardItemModel();
        m_searchList = createList( cellCreator, m_searchResultsItemModel );
        MSortFilterProxyModel* sortFilterProxyModel =
            m_searchList->filtering()->proxy();
        sortFilterProxyModel->setSortRole( ListCellCreator::SortFilterRole );
        sortFilterProxyModel->sort( 0, Qt::DescendingOrder );
        sortFilterProxyModel->setDynamicSortFilter(true);
        QObject::connect( m_searchList, SIGNAL(itemClicked(QModelIndex)),
            this, SLOT(onListItemSelected(QModelIndex)) );
        QObject::connect( m_searchList, SIGNAL(panningStarted()),
            this, SLOT(panningStarted()));
        QObject::connect( m_searchList, SIGNAL(panningStopped()),
            this, SLOT(panningStopped()));
        m_searchLayoutPolicy->addItem( m_searchList );
        m_searchLayoutPolicy->addStretch();
    }
    m_refreshButton->setVisible(false);
    m_searchLayoutPolicy->activate();
}

void MainWindow::showTracksTab(){
    //% "Tracks"
    m_headerLabel->setText(qtTrId("qtn_tracks_header"));
    m_refreshButton->setVisible(true);
    if( !m_tracksFetched ){
        //% "Loading tracks..."
        m_emptyLabel->setText(qtTrId("qtn_tracks_loading"));
        m_emptyLayoutPolicy->activate();
        m_musicEngine->fetchMoreTracks();
    }
    else if( m_tracksItemModel && m_tracksItemModel->rowCount() > 0 ){
        m_tracksLayoutPolicy->activate();
    }
    else{
        //% "Failed to load tracks"
        m_emptyLabel->setText(qtTrId("qtn_tracks_failed"));
        m_emptyLayoutPolicy->activate();
    }
    fetchMissingData();
}

void MainWindow::showArtistsTab(){
    //% "Artists"
    m_headerLabel->setText(qtTrId("qtn_artists_header"));
    m_refreshButton->setVisible(true);
    if( !m_artistsFetched ){
        //% "Loading artists..."
        m_emptyLabel->setText(qtTrId("qtn_artists_loading"));
        m_emptyLayoutPolicy->activate();
        m_musicEngine->fetchMoreArtists();
    }
    else if( m_artistsItemModel && m_artistsItemModel->rowCount() > 0 ){
        m_artistsLayoutPolicy->activate();
    }
    else{
        //% "Failed to load artists"
        m_emptyLabel->setText(qtTrId("qtn_artists_failed"));
        m_emptyLayoutPolicy->activate();
```

```
    }
    fetchMissingData();
}

void MainWindow::onTracksSearched( QList<Track*> list )
{
    //% "Tracks"
    QStandardItem* groupHeader = new QStandardItem(qtTrId("qtn_search_tracks"));
    groupHeader->setData( "track", ListCellCreator::TypeRole );
    groupHeader->setData( "track", ListCellCreator::SortFilterRole );
    m_searchResultsItemModel->appendRow( groupHeader );

    MLocale locale;
    foreach( Track* track, list ){
        QStandardItem* item = new QStandardItem( track->name );
        //% "Listeners: %1"
        QString listeners = QString(qtTrId("qtn_search_listeners")).arg(
            locale.formatNumber(track->listeners) );
        item->setData( listeners, ListCellCreator::SubtitleRole );
        item->setData( "icon-m-content-audio", ListCellCreator::DefaultIconRole );
        item->setData( "track", ListCellCreator::TypeRole );
        item->setData( track->id, ListCellCreator::IdRole );
        QString str = QString("%1").arg(track->listeners,10,10,QLatin1Char('0') );
        item->setData( str, ListCellCreator::SortFilterRole );
        groupHeader->appendRow( item );

        QObject::connect( track, SIGNAL(mediumImageFetched(DataItem&)),
            this, SLOT(onMediumImageFetched(DataItem&)), Qt::UniqueConnection );
    }
    QTimer::singleShot(10,this,SLOT(fetchMissingData()));
}

void MainWindow::onArtistsSearched( QList<Artist*> list )
{
    //% "Artists"
    QStandardItem* groupHeader = new QStandardItem(qtTrId("qtn_search_artists"));
    groupHeader->setData( "artist", ListCellCreator::TypeRole );
    groupHeader->setData( "artist", ListCellCreator::SortFilterRole );
    m_searchResultsItemModel->appendRow( groupHeader );

    MLocale locale;
    foreach( Artist* artist, list ){
        QStandardItem* item = new QStandardItem( artist->name );
        //% "Listeners: %1"
        QString listeners = QString(qtTrId("qtn_search_listeners")).arg(
            locale.formatNumber(artist->listeners) );
        item->setData( listeners, ListCellCreator::SubtitleRole );
        item->setData( "icon-m-content-artist", ListCellCreator::DefaultIconRole );
        item->setData( artist->id, ListCellCreator::IdRole );
        item->setData( "artist", ListCellCreator::TypeRole );
        QString str = QString("%1").arg(artist->listeners,10,10,QLatin1Char('0') );
        item->setData( str, ListCellCreator::SortFilterRole );
        groupHeader->appendRow( item );
        QObject::connect( artist, SIGNAL(mediumImageFetched(DataItem&)),
            this, SLOT(onMediumImageFetched(DataItem&)), Qt::UniqueConnection );
    }
    QTimer::singleShot(10,this,SLOT(fetchMissingData()));
}

void MainWindow::onTracksFetched( int total, int perPage, int page,
    QList<Track*> list )
{
    m_tracksFetched = true;
    if( m_tracksMoreButton && ( page < (total / perPage) ) )
        m_tracksMoreButton->setEnabled(true);
```

```
    if( !m_tracksItemModel ){
        ListCellCreator* cellCreator = new ListCellCreator();
        m_tracksItemModel = new QStandardItemModel();
        m_tracksList = createList( cellCreator, m_tracksItemModel );
        QObject::connect( m_tracksList, SIGNAL(itemClicked(QModelIndex)),
            this, SLOT(onListItemSelected(QModelIndex)) );
        QObject::connect( m_tracksList, SIGNAL(panningStarted()),
            this, SLOT(panningStarted()));
        QObject::connect( m_tracksList, SIGNAL(panningStopped()),
            this, SLOT(panningStopped()));
        MTextEdit* entry = m_tracksList->filtering()->editor();
        entry->setStyleName("CommonTopInputField");
        //% "Filter..."
        entry->setPrompt(qtTrId("qtn_tracks_filter"));
        entry->setVisible( true );
        //% "More"
        m_tracksMoreButton = new MButton( qtTrId("qtn_tracks_more") );
        QObject::connect( m_tracksMoreButton, SIGNAL(clicked()),
            this, SLOT(onFetchMoreTracksClicked()) );
        m_tracksMoreButton->setStyleName("CommonHorizontalCenterButton");
        m_tracksLayoutPolicy->addItem( entry );
        m_tracksLayoutPolicy->addItem( m_tracksList );
        m_tracksLayoutPolicy->addItem( m_tracksMoreButton, Qt::AlignHCenter );
        m_tracksLayoutPolicy->addStretch();
    }
    if( m_tracksTabButton->isChecked() ) m_tracksLayoutPolicy->activate();

    int startIndex = (page-1)*perPage;
    int headerIndex = (startIndex) / ITEMS_PER_LIST_GROUP;
    QStandardItem* groupHeader = 0;
    if( headerIndex < m_tracksItemModel->rowCount() )
        groupHeader = m_tracksItemModel->item( headerIndex );

    for( int i = 0; i < list.count(); i++ ){
        headerIndex = (startIndex+i) / ITEMS_PER_LIST_GROUP;
        int itemIndex = (startIndex+i) % ITEMS_PER_LIST_GROUP;
        if( itemIndex == 0 || !groupHeader )
        {
            QString title = QString("%1-%2").arg( (startIndex+i)+1 ).arg(
                startIndex+i+ITEMS_PER_LIST_GROUP);
            groupHeader = new QStandardItem( title );
            m_tracksItemModel->appendRow( groupHeader );
        }
        QStandardItem* item = new QStandardItem( list[i]->name );
        item->setData( list[i]->artistName, ListCellCreator::SubtitleRole );
        item->setData( "icon-m-content-audio", ListCellCreator::DefaultIconRole );
        item->setData( "track", ListCellCreator::TypeRole );
        item->setData( list[i]->id, ListCellCreator::IdRole );
        item->setData( list[i]->name + " " + list[i]->artistName,
            ListCellCreator::SortFilterRole );
        groupHeader->appendRow( item );

        QObject::connect( list[i], SIGNAL(mediumImageFetched(DataItem&)),
            this, SLOT(onMediumImageFetched(DataItem&)), Qt::UniqueConnection );
    }
    QTimer::singleShot(10,this,SLOT(fetchMissingData()));
}

void MainWindow::onArtistsFetched( int total, int perPage, int page,
    QList<Artist*> list ){
    m_artistsFetched = true;
    if( m_artistsMoreButton && ( page < (total / perPage) ) )
        m_artistsMoreButton->setEnabled(true);
    if( !m_artistsItemModel ){
        ListCellCreator* cellCreator = new ListCellCreator();
        m_artistsItemModel = new QStandardItemModel();
```

```
        m_artistsList = createList( cellCreator, m_artistsItemModel );
        QObject::connect( m_artistsList, SIGNAL(itemClicked(QModelIndex)),
            this, SLOT(onListItemSelected(QModelIndex)) );
        QObject::connect( m_artistsList, SIGNAL(panningStarted()),
            this, SLOT(panningStarted()));
        QObject::connect( m_artistsList, SIGNAL(panningStopped()),
            this, SLOT(panningStopped()));
        MTextEdit* entry = m_artistsList->filtering()->editor();
        entry->setStyleName("CommonTopInputField");
        //% "Filter..."
        entry->setPrompt(qtTrId("qtn_artists_filter"));
        entry->setVisible( true );
        //% "More"
        m_artistsMoreButton = new MButton( qtTrId("qtn_artists_more") );
        QObject::connect( m_artistsMoreButton, SIGNAL(clicked()),
            this, SLOT(onFetchMoreArtistsClicked()) );
        m_artistsMoreButton->setStyleName("CommonHorizontalCenterButton");
        m_artistsLayoutPolicy->addItem( entry );
        m_artistsLayoutPolicy->addItem( m_artistsList );
        m_artistsLayoutPolicy->addItem( m_artistsMoreButton, Qt::AlignHCenter );
        m_artistsLayoutPolicy->addStretch();
    }
    if( m_artistsTabButton->isChecked() ) m_artistsLayoutPolicy->activate();

    int startIndex = (page-1)*perPage;
    int headerIndex = (startIndex) / ITEMS_PER_LIST_GROUP;
    QStandardItem* groupHeader = 0;
    if( headerIndex < m_artistsItemModel->rowCount() )
        groupHeader = m_artistsItemModel->item( headerIndex );

    MLocale locale;
    for( int i = 0; i < list.count(); i++ ){
        headerIndex = (startIndex+i) / ITEMS_PER_LIST_GROUP;
        int itemIndex = (startIndex+i) % ITEMS_PER_LIST_GROUP;
        if( itemIndex == 0 || !groupHeader ){
            QString title = QString("%1-%2").arg( (startIndex+i)+1 ).arg(
                startIndex+i+ITEMS_PER_LIST_GROUP);
            groupHeader = new QStandardItem( title );
            m_artistsItemModel->appendRow( groupHeader );
        }

        QStandardItem* item = new QStandardItem( list[i]->name );
        //% "Listeners: %1"
        QString listeners = QString(qtTrId("qtn_artists_listeners")).arg(
            locale.formatNumber(list[i]->listeners) );
        item->setData( listeners, ListCellCreator::SubtitleRole );
        item->setData( "icon-m-content-artist", ListCellCreator::DefaultIconRole );
        item->setData( list[i]->id, ListCellCreator::IdRole );
        item->setData( "artist", ListCellCreator::TypeRole );
        item->setData( list[i]->name, ListCellCreator::SortFilterRole );
        groupHeader->appendRow( item );
        QObject::connect( list[i], SIGNAL(mediumImageFetched(DataItem&)),
            this, SLOT(onMediumImageFetched(DataItem&)), Qt::UniqueConnection );
    }
    QTimer::singleShot(10,this,SLOT(fetchMissingData()));
}

void MainWindow::onTopTracksErrorCallback(){
    m_tracksFetched = true;
    showTracksTab();
}

void MainWindow::onTopArtistsErrorCallback(){
    m_artistsFetched = true;
    showArtistsTab();
}
```

```
void MainWindow::fetchMissingData(){
    if( !m_fetchingMissingData && !m_panningList ){
        m_fetchingMissingData = true;
        MList* list = 0;
        QStandardItemModel* itemModel = 0;
        if( m_tracksTabButton->isChecked() && m_tracksList ){
            list = m_tracksList;
            itemModel = m_tracksItemModel;
        }
        else if( m_artistsTabButton->isChecked() && m_artistsList ){
            list = m_artistsList;
            itemModel = m_artistsItemModel;
        }
        else if( m_searchTabButton->isChecked() && m_searchResultsItemModel ){
            list = m_searchList;
            itemModel = m_searchResultsItemModel;
        }

        if( list && itemModel && itemModel->rowCount() > 0 ){
            MSortFilterProxyModel* itemModelProxy = list->filtering()->proxy();
            QModelIndex firstVisible;
            QModelIndex lastVisible;
            firstVisible = list->firstVisibleItem();
            lastVisible =  list->lastVisibleItem();

            QList<QModelIndex> leafNodeIndexes;
            if( firstVisible.isValid() && lastVisible.isValid() )
            {
                QModelIndex index = firstVisible;
                while( index.isValid() ){
                    if( !getLeafNodeIndexes( leafNodeIndexes, *itemModelProxy,
                        index, lastVisible ) )
                        break;
                    if( index.row() + 1 < itemModelProxy->rowCount(index.parent()) )
                        index = index.sibling( index.row()+1, 0 );
                    else
                        index = index.parent().sibling( index.parent().row()+1, 0 );
                }
                QList<QModelIndex> sourceIndexes = mapToSource(
                    leafNodeIndexes, *itemModelProxy );
                QList<QStandardItem*> listItems = getListItems(
                    sourceIndexes, *itemModel );
                if( fetchMissingData( listItems ) ) return;
            }
            leafNodeIndexes.clear();
            getLeafNodeIndexes( leafNodeIndexes, *itemModel );
            QList<QStandardItem*> listItems = getListItems(
                leafNodeIndexes, *itemModel );
            if( fetchMissingData( listItems ) ) return;
        }
        m_fetchingMissingData = false;
    }
}

void MainWindow::enableToolbarTabMode(){
    MApplicationWindow* w = MApplication::activeApplicationWindow();
    w->setToolbarViewType(MToolBar::tabType);
}

QList<QModelIndex> MainWindow::mapToSource( QList<QModelIndex>& proxyIndexes,
    QSortFilterProxyModel& proxyModel ){
    QList<QModelIndex> sourceIndexes;
    foreach( QModelIndex proxyIndex, proxyIndexes ){
        QModelIndex sourceIndex = proxyModel.mapToSource( proxyIndex );
        sourceIndexes.append( sourceIndex );
```

```
    }
    return sourceIndexes;
}

QList<QStandardItem*> MainWindow::getListItems( QList<QModelIndex>& indexes,
    QStandardItemModel& itemModel ){
    QList<QStandardItem*> listItems;
    foreach( QModelIndex index, indexes ){
        listItems.append( itemModel.itemFromIndex(index) );
    }
    return listItems;
}

bool MainWindow::getLeafNodeIndexes( QList<QModelIndex>& results,
    QAbstractItemModel& itemModel, QModelIndex index, QModelIndex stopAtIndex ){
    if( itemModel.rowCount(index) == 0 ){
        results.append( index );
        if( stopAtIndex.isValid() && index == stopAtIndex ) return false;
    }
    else if( stopAtIndex.isValid() && index == stopAtIndex ) return false;
    else{
        for( int i = 0; i < itemModel.rowCount(index); i++ ){
            if( !getLeafNodeIndexes( results, itemModel, itemModel.index(i,0,index),
                stopAtIndex ) ){
                return false;
            }
        }
    }
    return true;
}

bool MainWindow::fetchMissingData(QList<QStandardItem*> list){
    foreach( QStandardItem* item, list ){
        QString itemType = item->data(ListCellCreator::TypeRole).toString();
        unsigned int id = item->data(ListCellCreator::IdRole).toUInt();
        DataItem* dataItem = 0;
        if( itemType == "track" ){
            dataItem = m_musicEngine->getTrackById( id );
        }
        else if( itemType == "artist" ){
            dataItem = m_musicEngine->getArtistById( id );
        }
        if( dataItem && !dataItem->mediumImageRequested ){
            m_musicEngine->fetchMediumImage( *dataItem );
            return true;
        }
    }
    return false;
}

MList* MainWindow::createList( ListCellCreator* cellCreator,
    QStandardItemModel* itemModel ){
    MList* list = new MList();
    list->setCellCreator( cellCreator );
    list->setShowGroups( true );
    list->setSelectionMode( MList::NoSelection );
    list->setIndexDisplayMode( MList::Hide );
    list->filtering()->setFilterMode(MListFilter::FilterAsSubstring);
    list->filtering()->setFilterRole( ListCellCreator::SortFilterRole );
    list->filtering()->setEnabled(true);
    list->setItemModel( itemModel );
    return list;
}
```

detailspagebase.h

```
#ifndef DETAILSPAGEBASE_H
#define DETAILSPAGEBASE_H

#include <MApplicationPage>
#include "musicengine.h"

class QGraphicsItem;
class MLabel;
class MImageWidget;
class MGridLayoutPolicy;

class DetailsPageBase : public MApplicationPage{
    Q_OBJECT
public:
    DetailsPageBase(MusicEngine& musicEngine, QGraphicsItem* parent = 0);
    virtual void createContent();
    MLabel* createLabel( const QString& text = QString::null );
    QString textToHtml( const QString text );

protected slots:
    void onLinkClicked( const QString& uriStr );

protected:
    MusicEngine& m_musicEngine;
    MImageWidget* m_image;
    MGridLayoutPolicy* m_detailsGrid;
    MLabel* m_header;
    MLabel* m_description;
};

#endif // DETAILSPAGEBASE_H
```

detailspagebase.cpp

```
#include <QDesktopServices>
#include <QUrl>
#include <MLayout>
#include <MLabel>
#include <MLinearLayoutPolicy>
#include <MGridLayoutPolicy>
#include <MImageWidget>
#include <MApplication>
#include <MApplicationWindow>
#include <MButton>
#include "musicengine.h"
#include "detailspagebase.h"

DetailsPageBase::DetailsPageBase(MusicEngine& musicEngine, QGraphicsItem* parent) :
    MApplicationPage( parent ), m_musicEngine(musicEngine), m_image(0),
    m_detailsGrid(0), m_header(0), m_description(0){
}

void DetailsPageBase::createContent(){
    MApplicationWindow* w = MApplication::activeApplicationWindow();
    w->setToolbarViewType("");

    QGraphicsWidget* pageWidget = centralWidget();
    MLayout* pageLayout = new MLayout( pageWidget );
    MLinearLayoutPolicy* layoutPolicy = new MLinearLayoutPolicy(
```

```
        pageLayout, Qt::Vertical );
    layoutPolicy->setContentsMargins( 0, 0, 0, 0 );
    layoutPolicy->setSpacing( 0 );

    MWidgetController* headerWidget = new MWidgetController();
    headerWidget->setStyleName("CommonHeaderPanel");
    MLayout* headerLayout = new MLayout( headerWidget );
    MLinearLayoutPolicy* headerLayoutPolicy = new MLinearLayoutPolicy(
        headerLayout, Qt::Horizontal );
    headerLayoutPolicy->setContentsMargins(0,0,0,0);
    headerLayoutPolicy->setSpacing(0);
    m_header = new MLabel();
    m_header->setStyleName("CommonHeader");
    m_header->setWordWrap(false);
    m_header->setTextElide(true);
    headerLayoutPolicy->addItem(m_header, Qt::AlignTop);
    layoutPolicy->addItem(headerWidget);

    MLabel* spacer = new MLabel();
    spacer->setStyleName("CommonSpacer");
    layoutPolicy->addItem(spacer);

    m_image = new MImageWidget();
    layoutPolicy->addItem(m_image);

    MWidget* detailsWidget = new MWidget();
    MLayout* detailsLayout = new MLayout( detailsWidget );
    m_detailsGrid = new MGridLayoutPolicy( detailsLayout );
    m_detailsGrid->setContentsMargins(0,0,0,0);
    m_detailsGrid->setSpacing(0);
    layoutPolicy->addItem( detailsWidget );

    MLabel* spacer2 = new MLabel();
    spacer2->setStyleName("CommonLargeSpacer");
    layoutPolicy->addItem(spacer2);

    m_description = createLabel();
    layoutPolicy->addItem( m_description );

    layoutPolicy->addStretch();
}

MLabel* DetailsPageBase::createLabel(const QString& text){
    MLabel* label = new MLabel();
    label->setWordWrap(true);
    label->setTextFormat(Qt::RichText);
    label->setAlignment(Qt::AlignTop);
    label->setText(text);
    QObject::connect( label, SIGNAL(linkActivated(QString)),
        this, SLOT(onLinkClicked(QString)) );
    return label;
}

QString DetailsPageBase::textToHtml( const QString text ){
    QString out = text;
    return out.replace('\n', "<br />");
}

void DetailsPageBase::onLinkClicked( const QString& uriStr ){
    QUrl url(uriStr);
    QDesktopServices::openUrl( url );
}
```

artistdetailspage.h

```
#ifndef ARTISTDETAILSPAGE_H
#define ARTISTDETAILSPAGE_H

#include "musicengine.h"
#include "detailspagebase.h"

class Artist;
class QGraphicsItem;
class MLabel;

class ArtistDetailsPage : public DetailsPageBase{
    Q_OBJECT
public:
    ArtistDetailsPage(MusicEngine& musicEngine, Artist& artist, QGraphicsItem* parent = 0);
    virtual void createContent();

private slots:
    void refresh();
    void onOpenClicked();

private:
    Artist& m_artist;
    MLabel* m_tags;
    MLabel* m_playcount;
};

#endif // ARTISTDETAILSPAGE_H
```

artistdetailspage.cpp

```
#include <MLocale>
#include <MAction>
#include <MButton>
#include <MLabel>
#include <MGridLayoutPolicy>
#include <MImageWidget>
#include "artistdetailspage.h"

ArtistDetailsPage::ArtistDetailsPage( MusicEngine& musicEngine, Artist& artist,
    QGraphicsItem* parent ) : DetailsPageBase(musicEngine,parent),
    m_artist(artist), m_tags(0), m_playcount(0){
    if( !m_artist.largeImageRequested )
        m_musicEngine.fetchLargeImage( m_artist );
    if( m_artist.bio.isEmpty() )
        m_musicEngine.fetchArtistInfo( m_artist.id );
}

void ArtistDetailsPage::createContent(){
    DetailsPageBase::createContent();

    //% "Open in browser"
    MAction* openButton = new MAction( qtTrId("qtn_artist_detail_browser"), this);
    openButton->setLocation(MAction::ApplicationMenuLocation);
    QObject::connect( openButton, SIGNAL(triggered()), this, SLOT(onOpenClicked()) );
    if( m_artist.url.isEmpty() ) openButton->setEnabled(false);
    addAction(openButton);

    m_header->setText(m_artist.name);
    MLocale locale;
```

```
    QString listeners = locale.formatNumber(m_artist.listeners);
    //% "Tags:"
    m_detailsGrid->addItem( createLabel(qtTrId("qtn_artist_detail_tags")), 0, 0 );
    //% "Listeners:"
    m_detailsGrid->addItem( createLabel(qtTrId("qtn_artist_detail_listeners")), 1, 0 );
    //% "Playcount:"
    m_detailsGrid->addItem( createLabel(qtTrId("qtn_artist_detail_playcount")), 2, 0 );
    m_tags = createLabel();
    m_playcount = createLabel();
    m_detailsGrid->addItem( m_tags, 0, 1 );
    m_detailsGrid->addItem( createLabel(listeners), 1, 1 );
    m_detailsGrid->addItem( m_playcount, 2, 1 );
    m_detailsGrid->setColumnMinimumWidth(0,170);
    m_detailsGrid->setColumnMinimumWidth(1,200);

    QObject::connect( &m_artist, SIGNAL(largeImageFetched(DataItem&)),
        this, SLOT(refresh()), Qt::UniqueConnection );
    QObject::connect( &m_artist, SIGNAL(fullInfoFetched(DataItem&)),
        this, SLOT(refresh()), Qt::UniqueConnection );
    refresh();
}

void ArtistDetailsPage::refresh(){
    MLocale locale;
    QString playcount = locale.formatNumber(m_artist.playcount);
    m_playcount->setText( playcount );
    QString tags;
    foreach( Tag* tag, m_artist.tags ){
        if( !tags.isEmpty() ) tags.append( ", " );
        QString link = QString("<a href=\"%1\">%2</a>").arg(tag->url).arg(tag->name);
        tags.append( link );
    }
    m_tags->setText( tags );
    QString bio;
    //% "Description not available."
    if( m_artist.bio.isEmpty() ) bio = qtTrId("qtn_artist_detail_no_description");
    else bio = textToHtml(m_artist.bio);
    m_description->setText( bio );
    if( !m_artist.largeImage.isNull() )
        m_image->setPixmap( m_artist.largeImage );
}

void ArtistDetailsPage::onOpenClicked(){
    onLinkClicked( m_artist.url );
}
```

trackdetailspage.h

```
#ifndef TRACKDETAILSPAGE_H
#define TRACKDETAILSPAGE_H

#include <MApplicationPage>
#include "musicengine.h"
#include "detailspagebase.h"

class QGraphicsItem;
class MLabel;
class MButton;

class TrackDetailsPage : public DetailsPageBase{
    Q_OBJECT
public:
```

```
    TrackDetailsPage(MusicEngine& musicEngine, Track& track, QGraphicsItem* parent = 0);
    virtual void createContent();

private slots:
    void refresh();
    void onOpenClicked();
    void onArtistButtonClicked();

private:
    Track& m_track;
    MLabel* m_album;
    MLabel* m_duration;
    MLabel* m_playcount;
    MButton* m_artistButton;
};

#endif // TRACKDETAILSPAGE_H
```

trackdetailspage.cpp

```
#include <MLocale>
#include <MLabel>
#include <MGridLayoutPolicy>
#include <MImageWidget>
#include <MAction>
#include <MButton>
#include "musicengine.h"
#include "artistdetailspage.h"
#include "trackdetailspage.h"

TrackDetailsPage::TrackDetailsPage( MusicEngine& musicEngine, Track& track,
    QGraphicsItem* parent ) : DetailsPageBase(musicEngine, parent),
    m_track(track), m_album(0), m_duration(0), m_playcount(0), m_artistButton(0){
    if( !m_track.largeImageRequested )
        m_musicEngine.fetchLargeImage( m_track );
    if( m_track.description.isEmpty() )
        m_musicEngine.fetchTrackInfo( m_track.id );
}

void TrackDetailsPage::createContent(){
    DetailsPageBase::createContent();

    //% "Open in browser"
    MAction* openButton = new MAction( qtTrId("qtn_track_detail_browser"), this);
    openButton->setLocation(MAction::ApplicationMenuLocation);
    QObject::connect( openButton, SIGNAL(triggered()),
        this, SLOT(onOpenClicked()) );
    if( m_track.url.isEmpty() ) openButton->setEnabled(false);
    addAction(openButton);

    m_header->setText(m_track.name);
    MLocale locale;
    QString listeners = locale.formatNumber(m_track.listeners);
    //% "Artist:"
    m_detailsGrid->addItem(createLabel(qtTrId("qtn_track_detail_artist")),0,0);
    //% "Album:"
    m_detailsGrid->addItem(createLabel(qtTrId("qtn_track_detail_album")),1,0);
    //% "Duration:"
    m_detailsGrid->addItem(createLabel(qtTrId("qtn_track_detail_duration")),2,0);
    //% "Listeners:"
    m_detailsGrid->addItem(createLabel(qtTrId("qtn_track_detail_listeners")),3,0);
    //% "Playcount:"
```

```
    m_detailsGrid->addItem(createLabel(qtTrId("qtn_track_detail_playcount")),4,0);
    m_detailsGrid->addItem(createLabel(m_track.artistName), 0, 1 );
    m_album = createLabel();
    m_duration = createLabel();
    m_playcount = createLabel();
    m_detailsGrid->addItem( m_album, 1, 1, 1, 2 );
    m_detailsGrid->addItem( m_duration, 2, 1, 1, 2 );
    m_detailsGrid->addItem( createLabel(listeners), 3, 1, 1, 2 );
    m_detailsGrid->addItem( m_playcount, 4, 1, 1, 2 );
    m_detailsGrid->setColumnMinimumWidth(0,170);
    m_detailsGrid->setColumnMinimumWidth(1,200);

    m_artistButton = new MButton();
    m_artistButton->setIconID("icon-s-common-next");
    m_artistButton->setStyleName("CommonSmallIconButton");
    QObject::connect( m_artistButton, SIGNAL(clicked()),
        this, SLOT(onArtistButtonClicked()) );
    if( !m_track.artistId ){
        m_artistButton->setEnabled( false );
        m_musicEngine.fetchArtistInfo( m_track.artistMbid, m_track.artistName );
    }
    m_detailsGrid->addItem( m_artistButton, 0, 2, Qt::AlignVCenter );

    QObject::connect( &m_track, SIGNAL(largeImageFetched(DataItem&)),
        this, SLOT(refresh()), Qt::UniqueConnection );
    QObject::connect( &m_track, SIGNAL(fullInfoFetched(DataItem&)),
        this, SLOT(refresh()), Qt::UniqueConnection );
    QObject::connect( &m_musicEngine, SIGNAL(artistFetchedCallback(Artist*)),
        this, SLOT(refresh()) );
    refresh();
}

void TrackDetailsPage::refresh(){
    MLocale locale;
    QString playcount = locale.formatNumber(m_track.playcount);
    int seconds = m_track.duration % 60;
    int minutes = m_track.duration / 60;
    QString duration = QString("%1:%2").arg(minutes).arg(seconds,2,10,QChar('0'));
    m_playcount->setText( playcount );
    m_duration->setText( duration );
    if( !m_track.largeImage.isNull() )
        m_image->setPixmap( m_track.largeImage );
    m_album->setText( m_track.albumName );
    QString desc;
    //% "Description not available."
    if(m_track.description.isEmpty())
        desc = qtTrId("qtn_track_detail_no_description");
    else desc = textToHtml(m_track.description);
    m_description->setText( desc );
    if( m_track.artistId )
        m_artistButton->setEnabled(true);
}

void TrackDetailsPage::onOpenClicked(){
    onLinkClicked( m_track.url );
}

void TrackDetailsPage::onArtistButtonClicked(){
    Artist* artist = m_musicEngine.getArtistById( m_track.artistId );
    if( artist ){
        ArtistDetailsPage* page = new ArtistDetailsPage(m_musicEngine,*artist,this);
        page->appear(MSceneWindow::DestroyWhenDismissed);
    }
}
```

listcellcreator.h

```
#ifndef LISTCELLCREATOR_H
#define LISTCELLCREATOR_H

#include <QObject>
#include <MAbstractCellCreator>
#include <MBasicListItem>

class ListCellCreator : public MAbstractCellCreator<MBasicListItem>{
public:
    explicit ListCellCreator(QObject *parent = 0);
    virtual void updateCell(const QModelIndex& index, MWidget* cell) const;
    virtual QSizeF cellSize() const;

    enum CustomDataRole{
        SortFilterRole = Qt::UserRole,
        SubtitleRole,
        DefaultIconRole,
        PixmapRole,
        IdRole,
        TypeRole
    };
private:
    mutable QSizeF m_itemSize;
};

#endif // LISTCELLCREATOR_H
```

listcellcreator.cpp

```
#include <QDebug>
#include <MImageWidget>
#include "listcellcreator.h"

ListCellCreator::ListCellCreator(QObject *parent)
    : MAbstractCellCreator<MBasicListItem>(){
    MBasicListItem * item = new MBasicListItem();
    item->setLayoutPosition(M::DefaultPosition);
    m_itemSize = item->effectiveSizeHint(Qt::PreferredSize);
    qDebug() << Q_FUNC_INFO << m_itemSize;
    delete item;
}


void ListCellCreator::updateCell(const QModelIndex& index, MWidget* cell) const{
    MBasicListItem * listItem = qobject_cast<MBasicListItem *>(cell);
    if( !listItem ) return ;
    listItem->setItemStyle( MBasicListItem::IconWithTitleAndSubtitle );
    listItem->setTitle( index.data( Qt::DisplayRole ).toString() );
    listItem->setSubtitle( index.data( SubtitleRole ).toString() );
    QVariant iconVariant = index.data( PixmapRole );
    if( iconVariant.isValid() ){
        QPixmap pixmap = iconVariant.value<QPixmap>();
        listItem->imageWidget()->setPixmap( pixmap );
    }
    else{
        QVariant variant = index.data( DefaultIconRole );
        QString defaultIconId = variant.toString();
        listItem->imageWidget()->setImage( defaultIconId );
    }
```

```
    m_itemSize = listItem->effectiveSizeHint(Qt::PreferredSize);
}

QSizeF ListCellCreator::cellSize() const{
    return m_itemSize;
}
```

musicengine.h

```
#ifndef MUSICENGINE_H
#define MUSICENGINE_H

#include <QObject>
#include <QPixmap>

class QNetworkAccessManager;
class QXmlStreamReader;

class Tag{
public:
    QString name;
    QString url;
};

class DataItem : public QObject{
    Q_OBJECT
public:
    DataItem( unsigned int identifier );

public slots:
    void onMediumImageFetched();
    void onLargeImageFetched();
    void emitFullInfoFetched(){ emit fullInfoFetched(*this); }

signals:
    void mediumImageFetched(DataItem& dataItem);
    void largeImageFetched(DataItem& dataItem);
    void fullInfoFetched( DataItem& dataItem );

public:
    unsigned int id;
    bool mediumImageRequested;
    bool largeImageRequested;
    int playcount;
    int listeners;
    int rank;
    QString mbid;
    QString name;
    QString url;
    QString mediumImageUrl;
    QString largeImageUrl;
    QPixmap mediumImage;
    QPixmap largeImage;
};

class Track : public DataItem{
    Q_OBJECT
public:
    Track( unsigned int identifier );
public:
    int duration;
    unsigned int artistId;
```

```
    QString albumName;
    QString albumUrl;
    QString albumMbid;
    QString artistName;
    QString artistUrl;
    QString artistMbid;
    QString description;
};

class Artist : public DataItem{
    Q_OBJECT
public:
    Artist( unsigned int identifier ) : DataItem(identifier){}
    virtual ~Artist(){ qDeleteAll(tags); }
    QList<Tag*> tags;
    QString bio;
};

class MusicEngine : public QObject
{
    Q_OBJECT
public:
    explicit MusicEngine( const QString& lang, QObject *parent = 0);
    virtual ~MusicEngine();
    Track* getTrackById( unsigned int id );
    Artist* getArtistById( unsigned int id );

signals:
    void topTracksCallback( int totalCount, int perPage, int page,
        QList<Track*> newTracks );
    void topTracksErrorCallback();
    void topArtistsCallback( int totalCount, int perPage, int page,
        QList<Artist*> newArtists );
    void topArtistsErrorCallback();
    void searchTracksCallback( QList<Track*> artists );
    void searchArtistsCallback( QList<Artist*> artists );
    void trackFetchedCallback( Track* track );
    void artistFetchedCallback( Artist* track );

public slots:
    void search( const QString& query );
    void fetchTrackInfo( unsigned int id );
    void fetchArtistInfo( unsigned int id );
    void fetchArtistInfo( const QString& mbid, const QString& name );
    void clearFetchedTopTracks();
    void clearFetchedTopArtists();
    void clearBufferedDataItems();
    void fetchMoreTracks();
    void fetchMoreArtists();
    void fetchMediumImage( DataItem& dataItem );
    void fetchLargeImage( DataItem& dataItem );

private slots:
    void associateArtistWithTracks( Artist* artist );
    void associateTrackWithArtist( Track* track );
    void onTracksSearchReply();
    void onArtistsSearchReply();
    void onTracksFetched();
    void onArtistsFetched();
    void onTrackInfoFetched();
    void onArtistInfoFetched();
    void parseTrack( QXmlStreamReader& reader, Track& track );
    void parseArtistForTrack( QXmlStreamReader& reader, Track& track );
    void parseAlbumForTrack( QXmlStreamReader& reader, Track& track );
    void parseArtist( QXmlStreamReader& reader, Artist& artist );
    void parseTag( QXmlStreamReader& reader, Tag& tag );
```

```
    void skipElement( QXmlStreamReader& reader );

private:
    QString extractCharactersWithinElement( QXmlStreamReader& reader,
        const QString& element );

private:
    QNetworkAccessManager* m_networkAccessManager;
    int m_tracksPagesFetched;
    int m_artistsPagesFetched;
    int m_tracksTotalPages;
    int m_artistsTotalPages;
    unsigned int m_idCounter;
    QString m_lang;
    QList<Track*> m_tracks;
    QList<Artist*> m_artists;
};

#endif // MUSICENGINE_H
```

musicengine.cpp

```
#include <QNetworkAccessManager>
#include <QNetworkRequest>
#include <QNetworkReply>
#include <QString>
#include <QList>
#include <QUrl>
#include <QXmlStreamReader>
#include "musicengine.h"

#define API_KEY "5eec32a3108c7b1c644d10784aa39f15"
#define API_ROOT "http://ws.audioscrobbler.com/2.0/"
#define METHOD_GETTOPTRACKS "chart.gettoptracks"
#define METHOD_GETTOPARTISTS "chart.gettopartists"
#define METHOD_GETTRACKINFO "track.getInfo"
#define METHOD_GETARTISTINFO "artist.getInfo"
#define METHOD_ARTISTSEARCH "artist.search"
#define METHOD_TRACKSEARCH "track.search"
#define ENTRIES_PER_PAGE "50"
#define SEARCH_ENTRIES_LIMIT "5"

DataItem::DataItem( unsigned int identifier ) : QObject(), id(identifier),
    mediumImageRequested(false), largeImageRequested(false), playcount(0),
    listeners(0), rank(0){
}

void DataItem::onMediumImageFetched(){
    QObject* obj = sender();
    if( obj ){
        QIODevice* input = qobject_cast<QIODevice*>(obj);
        if( input ){
            QPixmap pixmap;
            bool success = pixmap.loadFromData( input->readAll() );
            if( success ) this->mediumImage = pixmap;
        }
    }
    emit mediumImageFetched(*this);
}

void DataItem::onLargeImageFetched(){
    QObject* obj = sender();
```

```
    if( obj ){
        QIODevice* input = qobject_cast<QIODevice*>(obj);
        if( input ){
            QPixmap pixmap;
            bool success = pixmap.loadFromData( input->readAll() );
            if( success ) this->largeImage = pixmap;
        }
    }
    emit largeImageFetched(*this);
}

Track::Track(unsigned int identifier) : DataItem(identifier),
    duration(0), artistId(0){
}

MusicEngine::MusicEngine(const QString& lang, QObject *parent) :
    QObject(parent), m_networkAccessManager( new QNetworkAccessManager(this) ),
    m_lang(lang), m_tracksPagesFetched(0), m_artistsPagesFetched(0),
    m_tracksTotalPages(1), m_artistsTotalPages(1), m_idCounter(1){
}

MusicEngine::~MusicEngine(){
    qDeleteAll( m_tracks );
    qDeleteAll( m_artists );
}

Track* MusicEngine::getTrackById( unsigned int id ){
    foreach( Track* track, m_tracks ) if( track->id == id ) return track;
    return 0;
}

Artist* MusicEngine::getArtistById( unsigned int id ){
    foreach( Artist* artist, m_artists ) if( artist->id == id ) return artist;
    return 0;
}

void MusicEngine::search( const QString& query ){
    QUrl url = QUrl( API_ROOT );
    url.addQueryItem( "method", METHOD_TRACKSEARCH );
    url.addQueryItem( "limit", SEARCH_ENTRIES_LIMIT );
    url.addQueryItem( "api_key", API_KEY );
    url.addQueryItem( "track", query );
    QNetworkReply* reply = m_networkAccessManager->get( QNetworkRequest( url ) );
    connect( reply, SIGNAL(finished()),
        this, SLOT(onTracksSearchReply()), Qt::UniqueConnection );

    QUrl artistUrl = QUrl( API_ROOT );
    artistUrl.addQueryItem( "method", METHOD_ARTISTSEARCH );
    artistUrl.addQueryItem( "limit", SEARCH_ENTRIES_LIMIT );
    artistUrl.addQueryItem( "api_key", API_KEY );
    artistUrl.addQueryItem( "artist", query );
    QNetworkReply* artistsReply = m_networkAccessManager->get( QNetworkRequest( artistUrl ) );
    connect( artistsReply, SIGNAL(finished()),
        this, SLOT(onArtistsSearchReply()), Qt::UniqueConnection );
}

void MusicEngine::fetchTrackInfo( unsigned int id ){
    Track* track = this->getTrackById( id );
    if( track ){
        QUrl url = QUrl( API_ROOT );
        url.addQueryItem( "method", METHOD_GETTRACKINFO );
        if( !track->mbid.isEmpty() ) url.addQueryItem( "mbid", track->mbid );
        else{
            url.addQueryItem( "track", track->name );
            url.addQueryItem( "artist", track->artistName );
        }
```

```
        url.addQueryItem( "api_key", API_KEY );
        QNetworkReply* reply = m_networkAccessManager->get(QNetworkRequest(url));
        reply->setProperty("id", id);
        connect( reply, SIGNAL(finished()),
            this, SLOT(onTrackInfoFetched()), Qt::UniqueConnection );
    }
}

void MusicEngine::fetchArtistInfo( unsigned int id ){
    Artist* artist = this->getArtistById( id );
    if( artist ){
        QUrl url = QUrl( API_ROOT );
        url.addQueryItem( "method", METHOD_GETARTISTINFO );
        if( !artist->mbid.isEmpty() ) url.addQueryItem( "mbid", artist->mbid );
        else url.addQueryItem( "artist", artist->name );
        url.addQueryItem( "api_key", API_KEY );
        QNetworkReply* reply = m_networkAccessManager->get(QNetworkRequest(url));
        reply->setProperty("id", id);
        connect( reply, SIGNAL(finished()),
            this, SLOT(onArtistInfoFetched()), Qt::UniqueConnection );
    }
}

void MusicEngine::fetchArtistInfo( const QString& mbid, const QString& name ){
    QUrl url = QUrl( API_ROOT );
    url.addQueryItem( "method", METHOD_GETARTISTINFO );
    if( !mbid.isEmpty() ) url.addQueryItem( "mbid", mbid );
    else url.addQueryItem( "artist", name );
    url.addQueryItem( "api_key", API_KEY );
    QNetworkReply* reply = m_networkAccessManager->get( QNetworkRequest(url) );
    connect( reply, SIGNAL(finished()),
        this, SLOT(onArtistInfoFetched()), Qt::UniqueConnection );
}

void MusicEngine::clearFetchedTopTracks(){
    int i = 0;
    while(i < m_tracks.count()){
        if(m_tracks[i]->rank > 0 ) delete m_tracks.takeAt(i);
        else i++;
    }
    m_tracksPagesFetched = 0;
    m_tracksTotalPages = 1;
}

void MusicEngine::clearFetchedTopArtists(){
    int i = 0;
    while(i < m_artists.count()){
        if(m_artists[i]->rank > 0 ){
            foreach( Track* track, m_tracks )
                if( track->artistId == m_artists[i]->id )
                    track->artistId = 0;
            delete m_artists.takeAt(i);
        }
        else i++;
    }
    m_artistsPagesFetched = 0;
    m_artistsTotalPages = 1;
}

void MusicEngine::clearBufferedDataItems(){
    int i = 0;
    while(i < m_tracks.count()){
        if(m_tracks[i]->rank <= 0 ) delete m_tracks.takeAt(i);
        else i++;
    }
    i = 0;
```

```
    while(i < m_artists.count()){
        if(m_artists[i]->rank <= 0 ){
            foreach( Track* track, m_tracks )
                if( track->artistId == m_artists[i]->id )
                    track->artistId = 0;
            delete m_artists.takeAt(i);
        }
        else i++;
    }
}

void MusicEngine::fetchMoreTracks(){
    QUrl url = QUrl( API_ROOT );
    url.addQueryItem( "method", METHOD_GETTOPTRACKS );
    url.addQueryItem( "page", QString("%1").arg(m_tracksPagesFetched+1) );
    url.addQueryItem( "limit", ENTRIES_PER_PAGE  );
    url.addQueryItem( "api_key", API_KEY );
    QNetworkReply* reply = m_networkAccessManager->get( QNetworkRequest( url ) );
    connect( reply, SIGNAL(finished()),
        this, SLOT(onTracksFetched()), Qt::UniqueConnection );
}

void MusicEngine::fetchMoreArtists(){
    QUrl url = QUrl( API_ROOT );
    url.addQueryItem( "method", METHOD_GETTOPARTISTS );
    url.addQueryItem( "page", QString("%1").arg(m_artistsPagesFetched+1) );
    url.addQueryItem( "limit", ENTRIES_PER_PAGE  );
    url.addQueryItem( "api_key", API_KEY );
    QNetworkReply* reply = m_networkAccessManager->get( QNetworkRequest(url) );
    connect( reply, SIGNAL(finished()),
        this, SLOT(onArtistsFetched()), Qt::UniqueConnection );
}

void MusicEngine::fetchMediumImage( DataItem& dataItem ){
    dataItem.mediumImageRequested = true;
    if( !dataItem.mediumImageUrl.isEmpty() ){
        QUrl url( dataItem.mediumImageUrl );
        QNetworkReply* reply = m_networkAccessManager->get(QNetworkRequest(url));
        connect( reply, SIGNAL(finished()),
            &dataItem, SLOT(onMediumImageFetched()), Qt::UniqueConnection );
    }
    else dataItem.onMediumImageFetched();
}

void MusicEngine::fetchLargeImage( DataItem& dataItem ){
    dataItem.largeImageRequested = true;
    if( !dataItem.largeImageUrl.isEmpty() ){
        QUrl url( dataItem.largeImageUrl );
        QNetworkReply* reply = m_networkAccessManager->get( QNetworkRequest( url ) );
        connect( reply, SIGNAL(finished()),
            &dataItem, SLOT(onLargeImageFetched()), Qt::UniqueConnection );
    }
    else dataItem.onLargeImageFetched();
}

void MusicEngine::associateArtistWithTracks( Artist* artist ){
    foreach( Track* track, m_tracks ){
        if( !track->artistId &&
            ( !track->artistMbid.isEmpty() && track->artistMbid == artist->mbid
            || !track->artistUrl.isEmpty() && track->artistUrl == artist->url
            || !track->artistName.isEmpty()
                && track->artistName == artist->name ) ){
            track->artistId = artist->id;
        }
    }
}
```

```
void MusicEngine::associateTrackWithArtist( Track* track ){
    if( !track->artistId ){
        foreach( Artist* artist, m_artists ){
            if( !track->artistMbid.isEmpty() && track->artistMbid == artist->mbid
                || !track->artistUrl.isEmpty() && track->artistUrl == artist->url
                || !track->artistName.isEmpty()
                    && track->artistName == artist->name ){
                track->artistId = artist->id;
                break;
            }
        }
    }
}

void MusicEngine::onTracksSearchReply(){
    QObject* obj = sender();
    QNetworkReply* reply = qobject_cast<QNetworkReply*>(obj);
    if( reply ){
        QList<Track*> tracks;
        if( !reply->error() ){
            QByteArray data = reply->readAll();
            QXmlStreamReader reader( data );
            while(!reader.atEnd() && reader.error() == QXmlStreamReader::NoError ){
                reader.readNext();
                if( reader.tokenType() == QXmlStreamReader::StartElement ){
                    QXmlStreamAttributes attributes = reader.attributes();
                    if( reader.name() == "lfm" ){
                        if( attributes.value("status").toString() == "failed" ){
                            reader.raiseError("Error");
                        }
                    }
                    else if( reader.name() == "track" ){
                        Track* track = new Track( m_idCounter++ );
                        parseTrack( reader, *track );
                        m_tracks.append( track );
                        associateTrackWithArtist( track );
                        tracks.append( track );
                    }
                }
            }
        }
        emit searchTracksCallback(tracks);
    }
}

void MusicEngine::onArtistsSearchReply(){
    QObject* obj = sender();
    QNetworkReply* reply = qobject_cast<QNetworkReply*>(obj);
    if( reply ){
        QList<Artist*> artists;
        if( !reply->error() ){
            QByteArray data = reply->readAll();
            QXmlStreamReader reader( data );
            while(!reader.atEnd() && reader.error()==QXmlStreamReader::NoError){
                reader.readNext();
                if( reader.tokenType() == QXmlStreamReader::StartElement ){
                    QXmlStreamAttributes attributes = reader.attributes();
                    if( reader.name() == "lfm" ){
                        if(attributes.value("status").toString() == "failed"){
                            reader.raiseError("Error");
                        }
                    }
                    else if( reader.name() == "artist" ){
                        Artist* artist = new Artist( m_idCounter++ );
                        parseArtist( reader, *artist );
```

```
                    m_artists.append( artist );
                    associateArtistWithTracks( artist );
                    artists.append( artist );
                }
            }
        }
    }
    emit searchArtistsCallback(artists);
  }
}

void MusicEngine::onTrackInfoFetched(){
  QObject* obj = sender();
  QNetworkReply* reply = qobject_cast<QNetworkReply*>(obj);
  if( reply ){
    unsigned int id = reply->property("id").toUInt();
    Track* track = getTrackById( id );
    if( track ){
      if( !reply->error() ){
        QByteArray data = reply->readAll();
        QXmlStreamReader reader( data );
        while(!reader.atEnd() && reader.error()==QXmlStreamReader::NoError){
          reader.readNext();
          if( reader.tokenType() == QXmlStreamReader::StartElement ){
            QXmlStreamAttributes attributes = reader.attributes();
            if( reader.name() == "lfm" ){
              if( attributes.value("status").toString() == "failed" ){
                reader.raiseError("Error");
              }
            }
            else if( reader.name() == "track" ){
              parseTrack( reader, *track );
            }
          }
        }
      }
      if( !m_tracks.contains(track) ) m_tracks.append( track );
      track->emitFullInfoFetched();
      emit trackFetchedCallback(track);
    }
  }
}

void MusicEngine::onArtistInfoFetched(){
  QObject* obj = sender();
  QNetworkReply* reply = qobject_cast<QNetworkReply*>(obj);
  if( reply ){
    bool ok = false;
    unsigned int id = reply->property("id").toUInt( &ok );
    Artist* artist = ok ? getArtistById( id ) : 0;
    if( !artist ) artist = new Artist( m_idCounter++ );
    if( artist ){
      if( !reply->error() ){
        QByteArray data = reply->readAll();
        QXmlStreamReader reader( data );
        while(!reader.atEnd() && reader.error()==QXmlStreamReader::NoError){
          reader.readNext();
          if( reader.tokenType() == QXmlStreamReader::StartElement ){
            QXmlStreamAttributes attributes = reader.attributes();
            if( reader.name() == "lfm" ){
              if( attributes.value("status").toString() == "failed" ){
                reader.raiseError("Error");
              }
            }
            else if( reader.name() == "artist" ){
              parseArtist( reader, *artist );
```

```
                }
            }
        }
    }
    if( !m_artists.contains(artist) ) m_artists.append( artist );
    associateArtistWithTracks( artist );
    artist->emitFullInfoFetched();
    emit artistFetchedCallback( artist );
  }
 }
}

void MusicEngine::onTracksFetched(){
  QObject* obj = sender();
  QNetworkReply* reply = qobject_cast<QNetworkReply*>(obj);
  if( reply ){
    if( reply->error() ){
      emit topTracksErrorCallback();
    }
    else{
      QByteArray data = reply->readAll();
      QXmlStreamReader reader( data );
      int perPage = 0;
      int page = 0;
      int totalPages = 0;
      int total = 0;
      QList<Track*> newTracks;
      Track* currentTrack=0;
      int rank = 1;
      while(!reader.atEnd() && reader.error()==QXmlStreamReader::NoError){
        reader.readNext();
        if( reader.tokenType() == QXmlStreamReader::StartElement ){
          QXmlStreamAttributes attributes = reader.attributes();
          if( reader.name() == "lfm" ){
            if(attributes.value("status").toString() == "failed"){
              reader.raiseError("Error");
            }
          }
          else if( reader.name() == "tracks" ){
            perPage=attributes.value("perPage").toString().toInt();
            page=attributes.value("page").toString().toInt();
            totalPages=attributes.value("totalPages").toString().toInt();
            total=attributes.value("total").toString().toInt();
            rank=perPage*(page-1)+1;
          }
          else if( reader.name() == "track" ){
            currentTrack = new Track(m_idCounter++);
            currentTrack->rank = rank;
            rank++;
            parseTrack( reader, *currentTrack );
            newTracks.append( currentTrack );
          }
        }
      }
      m_tracks.append( newTracks );
      m_tracksTotalPages = totalPages;
      m_tracksPagesFetched = page;
      if( reader.error() == QXmlStreamReader::NoError )
        emit topTracksCallback( total, perPage, page, newTracks );
      else emit topTracksErrorCallback();
    }
    reply->deleteLater();
  }
}

void MusicEngine::onArtistsFetched(){
```

```
    QObject* obj = sender();
    QNetworkReply* reply = qobject_cast<QNetworkReply*>(obj);
    if( reply ){
        if( reply->error() ){
            emit topArtistsErrorCallback();
        }
        else{
            QXmlStreamReader reader( reply );
            int perPage = 0;
            int page = 0;
            int totalPages = 0;
            int total = 0;
            QList<Artist*> newArtists;
            Artist* currentArtist=0;
            int rank = 1;
            while(!reader.atEnd() && reader.error() == QXmlStreamReader::NoError){
                reader.readNext();
                if( reader.tokenType() == QXmlStreamReader::StartElement ){
                    QXmlStreamAttributes attributes = reader.attributes();
                    if( reader.name() == "lfm" ){
                        if( attributes.value("status").toString() == "failed" ){
                            reader.raiseError("Error");
                        }
                    }
                    else if( reader.name() == "artists" ){
                        perPage=attributes.value("perPage").toString().toInt();
                        page=attributes.value("page").toString().toInt();
                        totalPages=attributes.value("totalPages").toString().toInt();
                        total=attributes.value("total").toString().toInt();
                        rank=perPage*(page-1)+1;
                    }
                    else if( reader.name() == "artist" ){
                        currentArtist = new Artist(m_idCounter++);
                        currentArtist->rank = rank;
                        rank++;
                        parseArtist( reader, *currentArtist );
                        newArtists.append( currentArtist );
                    }
                }
            }
            m_artists.append( newArtists );
            m_artistsTotalPages = totalPages;
            m_artistsPagesFetched = page;
            if( reader.error() == QXmlStreamReader::NoError )
                emit topArtistsCallback( total, perPage, page, newArtists );
            else
                emit topArtistsErrorCallback();
        }
        reply->deleteLater();
    }
}

void MusicEngine::parseTrack( QXmlStreamReader& reader, Track& track ){
    QString name = reader.name().toString();
    while(!reader.atEnd()){
        reader.readNext();
        if( reader.tokenType() == QXmlStreamReader::StartElement ){
            QXmlStreamAttributes attributes = reader.attributes();
            if( reader.name() == "name" )
                track.name = "<p>" + reader.readElementText() + "</p>";
            else if( reader.name() == "duration" ){
                int duration = reader.readElementText().toInt();
                track.duration = duration > 5000 ? duration/1000 : duration;
            }
            else if( reader.name() == "playcount" )
                track.playcount = reader.readElementText().toInt();
```

```
            else if( reader.name() == "listeners" )
                track.listeners = reader.readElementText().toInt();
            else if( reader.name() == "mbid" )
                track.mbid = reader.readElementText();
            else if( reader.name() == "url" )
                track.url = reader.readElementText();
            else if( reader.name() == "image"
                && attributes.value("size") == "medium" )
                track.mediumImageUrl = reader.readElementText();
            else if( reader.name() == "image"
                && attributes.value("size") == "extralarge" )
                track.largeImageUrl = reader.readElementText();
            else if( reader.name() == "album" )
                parseAlbumForTrack( reader, track );
            else if( reader.name() == "artist" )
                parseArtistForTrack( reader, track );
            else if( reader.name() == "toptags" )
                skipElement( reader );
            else if( reader.name() == "wiki" )
                track.description = extractCharactersWithinElement(reader,"content");
            else skipElement( reader );
        }
        else if( reader.tokenType() == QXmlStreamReader::EndElement
            && reader.name() == name ) break;
    }
}

void MusicEngine::parseArtistForTrack( QXmlStreamReader& reader, Track& track ){
    QString name = reader.name().toString();
    QString curName = name;
    while(!reader.atEnd()){
        reader.readNext();
        if( reader.isCharacters() && !reader.isWhitespace() && curName == name ) {
            track.artistName = reader.text().toString();
        }
        else if( reader.tokenType() == QXmlStreamReader::StartElement ){
            curName = reader.name().toString();
            if( reader.name() == "name" )
                track.artistName = reader.readElementText();
            else if( reader.name() == "mbid" )
                track.artistMbid = reader.readElementText();
            else if( reader.name() == "url" )
                track.artistUrl = reader.readElementText();
            else skipElement( reader );
        }
        else if(reader.tokenType() == QXmlStreamReader::EndElement
            && reader.name() == name) break;
    }
}

void MusicEngine::parseAlbumForTrack( QXmlStreamReader& reader, Track& track ){
    QString name = reader.name().toString();
    while(!reader.atEnd()){
        reader.readNext();
        if( reader.tokenType() == QXmlStreamReader::StartElement ){
            QXmlStreamAttributes attributes = reader.attributes();
            if( reader.name() == "title" )
                track.albumName = reader.readElementText();
            else if( reader.name() == "mbid" )
                track.mbid = reader.readElementText();
            else if( reader.name() == "url" )
                track.url = reader.readElementText();
            else if( reader.name() == "image"
                && attributes.value("size") == "medium" )
                track.mediumImageUrl = reader.readElementText();
            else if( reader.name() == "image"
```

```
            && attributes.value("size") == "extralarge" )
            track.largeImageUrl = reader.readElementText();
        else skipElement( reader );
    }
    else if(reader.tokenType() == QXmlStreamReader::EndElement
        && reader.name() == name) break;
  }
}

void MusicEngine::parseArtist( QXmlStreamReader& reader, Artist& artist ){
  QString name = reader.name().toString();
  while(!reader.atEnd()){
    reader.readNext();
    if( reader.tokenType() == QXmlStreamReader::StartElement ){
      QXmlStreamAttributes attributes = reader.attributes();
      if( reader.name() == "name" ) artist.name = reader.readElementText();
      else if( reader.name() == "mbid" ) artist.mbid = reader.readElementText();
      else if( reader.name() == "url" ) artist.url = reader.readElementText();
      else if( reader.name() == "playcount" )
        artist.playcount = reader.readElementText().toInt();
      else if( reader.name() == "listeners" )
        artist.listeners = reader.readElementText().toInt();
      else if( reader.name() == "image"
        && attributes.value("size") == "medium" )
        artist.mediumImageUrl = reader.readElementText();
      else if( reader.name() == "image"
        && attributes.value("size") == "extralarge" )
        artist.largeImageUrl = reader.readElementText();
      else if( reader.name() == "bio" )
        artist.bio = extractCharactersWithinElement( reader, "content" );
      else if( reader.name() == "tag" ){
        Tag* tag = new Tag();
        parseTag( reader, *tag );
        artist.tags.append( tag );
      }
      else if( reader.name() == "tags" ) continue;
      else if( reader.name() == "stats" ) continue;
      else skipElement( reader );
    }
    else if(reader.tokenType() == QXmlStreamReader::EndElement
      && reader.name() == name) break;
  }
}

void MusicEngine::parseTag( QXmlStreamReader& reader, Tag& tag ){
  QString name = reader.name().toString();
  while(!reader.atEnd()){
    reader.readNext();
    if( reader.tokenType() == QXmlStreamReader::StartElement ){
      if( reader.name() == "name" ) tag.name = reader.readElementText();
      else if( reader.name() == "url" ) tag.url = reader.readElementText();
      else skipElement( reader );
    }
    else if(reader.tokenType() == QXmlStreamReader::EndElement
      && reader.name() == name) break;
  }
}

void MusicEngine::skipElement( QXmlStreamReader& reader ){
  QString name = reader.name().toString();
  while(!reader.atEnd()){
    reader.readNext();
    if(reader.tokenType() == QXmlStreamReader::EndElement
      && reader.name() == name) break;
  }
}
```

```
QString MusicEngine::extractCharactersWithinElement( QXmlStreamReader& reader,
    const QString& element ){
    QString name = reader.name().toString();
    while(!reader.atEnd()){
        reader.readNext();
        if( reader.tokenType() == QXmlStreamReader::StartElement
            && reader.name() == element ){
            return reader.readElementText();
        }
        else if(reader.tokenType() == QXmlStreamReader::EndElement
            && reader.name() == name) break;
    }
    return QString::null;
}
```

B Toteutus web-sovelluksena

index.html

```
<!DOCTYPE html>
<html>
    <head>
     <title>MusicApp</title>
     <meta name="viewport" content="width=device-width,
           target-densitydpi=device-dpi, initial-scale=1, maximum-scale=1" />
        <script src="jQuery/jquery-1.8.1.min.js"></script>
        <script src="musicengine.js"></script>
        <script src="engine.js"></script>
        <link rel="stylesheet" href="jQuery/jquery.mobile-1.2.0.min.css" />
        <script src="jQuery/jquery.mobile-1.2.0.min.js"></script>
        <link rel="stylesheet" href="styles.css" />
    </head>
    <body style="margin: 0px; background-color:white;">

        <div data-role="page" id="SearchPage">
            <div data-role="header" data-position="inline">
                <h1 id="SearchPageHeader"></h1>
                <div data-role="navbar">
                    <ul class="SearchPageTabs">
                        <li>
                            <a href="#SearchPage" data-mini="false"
                                    class="ui-btn-active ui-state-persist">
                            </a>
                            </li>
                        <li>
                            <a href="#TracksPage" data-mini="false"
                                    data-transition="none">
                            </a>
                        </li>
                        <li>
                            <a href="#ArtistsPage" data-mini="false"
                                    data-transition="none">
                            </a>
                        </li>
                    </ul>
                </div>
            </div>
            <div data-role="content">
                <form id="SearchPageForm">
                    <input type="search" name="search" id="SearchPageInput"
                            value="" data-mini="false" />
                    <input type="submit" value="" id="SearchPageSubmit"
                            data-mini="false" />
                </form>
                <ul data-role="listview" id="SearchList">
                </ul>
            </div>
            <div data-role="footer" data-position="inline"
                    data-tap-toggle="false">
                <div data-role="navbar">
                    <ul class="SearchPageTabs">
                        <li>
                            <a href="#SearchPage" data-mini="false"
                                    class="ui-btn-active ui-state-persist">
                            </a>
                        </li>
                        <li>
                            <a href="#TracksPage" data-mini="false"
```

```
                                        data-transition="none">
                                    </a>
                                </li>
                                <li>
                                    <a href="#ArtistsPage" data-mini="false"
                                        data-transition="none">
                                    </a>
                                </li>
                            </ul>
                        </div>
                    </div>
                </div>

                <div data-role="page" id="TracksPage">
                    <div data-role="header" data-position="inline">
                        <h1 id="TracksPageHeader"></h1>
                        <a href="#" data-icon="refresh" class="ui-btn-right"
                            id="TracksPageRefresh">
                        </a>
                        <div data-role="navbar">
                            <ul class="TracksPageTabs">
                                <li>
                                    <a href="#SearchPage" data-mini="false"
                                        data-transition="none">
                                    </a>
                                </li>
                                <li>
                                    <a href="#TracksPage" data-mini="false"
                                        class="ui-btn-active ui-state-persist">
                                    </a>
                                </li>
                                <li>
                                    <a href="#ArtistsPage" data-mini="false"
                                        data-transition="none">
                                    </a>
                                </li>
                            </ul>
                        </div>
                    </div>
                    <div data-role="content">
                        <ul data-role="listview" id="TracksList" data-filter="true">
                        </ul>
                        <p class="SingleButtonContainer">
                            <button type="button" id="TracksPageMoreButton"
                                disabled="disabled" data-mini="false">
                            </button>
                        </p>
                    </div>
                    <div data-role="footer" data-position="inline" data-tap-toggle="false">
                        <div data-role="navbar">
                            <ul class="TracksPageTabs">
                                <li>
                                    <a href="#SearchPage" data-mini="false"
                                        data-transition="none">
                                    </a>
                                </li>
                                <li>
                                    <a href="#TracksPage" data-mini="false"
                                        class="ui-btn-active ui-state-persist">
                                    </a>
                                </li>
                                <li>
                                    <a href="#ArtistsPage" data-mini="false"
                                        data-transition="none">
                                    </a>
                                </li>
```

```
            </ul>
        </div>
    </div>
</div>

<div data-role="page" id="ArtistsPage">
    <div data-role="header" data-position="inline">
        <h1 id="ArtistsPageHeader"></h1>
        <a href="#" data-icon="refresh" class="ui-btn-right"
                id="ArtistsPageRefresh">
        </a>
        <div data-role="navbar">
            <ul class="ArtistsPageTabs">
                <li>
                    <a href="#SearchPage" data-mini="false"
                            data-transition="none">
                    </a>
                </li>
                <li>
                    <a href="#TracksPage" data-mini="false"
                            data-transition="none">
                    </a>
                </li>
                <li>
                    <a href="#ArtistsPage" data-mini="false"
                            class="ui-btn-active ui-state-persist">
                    </a>
                </li>
            </ul>
        </div>
    </div>
    <div data-role="content">
        <ul data-role="listview" id="ArtistsList"
                data-filter="true">
        </ul>
        <p class="SingleButtonContainer">
            <button type="button" id="ArtistsPageMoreButton"
                    disabled="disabled" data-mini="false">
            </button>
        </p>
    </div>
    <div data-role="footer" data-position="inline" data-tap-toggle="false">
        <div data-role="navbar">
            <ul class="ArtistsPageTabs">
                <li>
                    <a href="#SearchPage" data-mini="false"
                            data-transition="none">
                    </a>
                </li>
                <li>
                    <a href="#TracksPage" data-mini="false"
                            data-transition="none">
                    </a>
                </li>
                <li>
                    <a href="#ArtistsPage" data-mini="false"
                            class="ui-btn-active ui-state-persist">
                    </a>
                </li>
            </ul>
        </div>
    </div>
</div>

<div data-role="page" id="TrackPage">
    <div data-role="header" data-position="inline">
```

```
            <a data-role="button" data-transition="fade" href="#"
                        data-icon="arrow-l" data-iconpos="left"
                        class="ui-btn-left TrackPageBack" data-rel="back">
            </a>
            <h1 id="TrackPageHeader"></h1>
            <a href="#" data-icon="info" class="ui-btn-right"
                        id="TrackPageBrowser">
            </a>
        </div>
        <div data-role="content" class="DetailedContent">
            <div class='ImageFrameLarge'>
                <img src="" alt="" class="DetailsImage" id="TrackPageImage" />
            </div>
            <table class="Details">
                <tr>
                    <td id="TrackPageArtistTitle"></td>
                    <td>
                        <span id="TrackPageArtist"></span>
                        <a href="index.html" data-role="button"
                                    data-icon="arrow-r" data-inline="true"
                                    data-iconpos="notext" id="TrackPageArtistButton">
                        </a>
                    </td>
                </tr>
                <tr>
                    <td id="TrackPageAlbumTitle"></td>
                    <td id="TrackPageAlbum"></td>
                </tr>
                <tr>
                    <td id="TrackPageDurationTitle"></td>
                    <td id="TrackPageDuration"></td>
                </tr>
                <tr>
                    <td id="TrackPageListenersTitle"></td>
                    <td id="TrackPageListeners"></td>
                </tr>
                <tr>
                    <td id="TrackPagePlaycountTitle"></td>
                    <td id="TrackPagePlaycount"></td>
                </tr>
            </table>
            <p id="TrackPageDescription"></p>
        </div>
        <div data-role="footer" data-position="inline" data-tap-toggle="false">
            <h3></h3>
            <a data-role="button" data-transition="fade" href="#"
                        data-icon="arrow-l" data-iconpos="left"
                        class="ui-btn-left TrackPageBack" data-rel="back">
            </a>
        </div>
    </div>

    <div data-role="page" id="ArtistPage">
        <div data-role="header" data-position="inline">
            <a data-role="button" data-transition="fade" href="#"
                        data-icon="arrow-l" data-iconpos="left"
                        class="ui-btn-left ArtistPageBack" data-rel="back">
            </a>
            <h1 id="ArtistPageHeader"></h1>
            <a href="#" data-icon="info" class="ui-btn-right"
                        id="ArtistPageBrowser">
            </a>
        </div>
        <div data-role="content" class="DetailedContent">
            <div class='ImageFrameLarge'>
                <img src="" alt="" class="DetailsImage" id="ArtistPageImage" />
```

```
                </div>
                <table class="Details">
                    <tr>
                        <td id="ArtistPageTagsTitle"></td>
                        <td id="ArtistPageTags"></td>
                    </tr>
                    <tr>
                        <td id="ArtistPageListenersTitle"></td>
                        <td id="ArtistPageListeners"></td>
                    </tr>
                    <tr>
                        <td id="ArtistPagePlaycountTitle"></td>
                        <td id="ArtistPagePlaycount"></td>
                    </tr>
                </table>
                <p id="ArtistPageBio" lang="en"></p>
            </div>
            <div data-role="footer" data-position="inline" data-tap-toggle="false">
                <h3></h3>
                <a data-role="button" data-transition="fade" href="#"
                        data-icon="arrow-l"     data-iconpos="left"
                        class="ui-btn-left ArtistPageBack" data-rel="back">
                </a>
            </div>
        </div>

    </body>
</html>
```

styles.html

```
th, td {
    max-width:240px;
}
table th, td {
    text-align:left;
    vertical-align:text-top;
    padding:0px;
    word-wrap:break-word;
}
.SingleButtonContainer{
    padding-top: 6px;
}
.ListIcon{
    width: 64px;
}
.ImageFrameMedium{
    width: 64px;
    height: 64px;
    overflow: hidden;
    vertical-align: middle;
    top: auto;
    bottom: auto;
    margin-left: 10px;
    margin-bottom: auto;
    margin-top: auto;
}
.ImageFrameLarge{
    width: 300px;
    height: 300px;
    overflow: hidden;
    margin-left: auto;
```

```
        margin-right: auto;
        margin-top: 6px;
        margin-bottom: 6px;
}
.DetailedContent{
        padding: 0px;
}
.DetailsImage{
        width: 300px;
}
table{
        border-collapse: separate;
        border-spacing: 6px 10px;
}
#ArtistPageBio, #TrackPageDescription{
        margin: 6px;
}
#SearchPageForm{
        margin-bottom: 24px;
}
#TrackPageArtistButton{
        margin: 0px 0px 4px 10px;
}
```

localization.js

```
var QtnFailedToFetchTracks = "Failed to fetch more track data";
var QtnFailedToFetchArtists = "Failed to fetch more artist data";

var QtnTabSearch = "Search";
var QtnTabTopTracks = "Top Tracks";
var QtnTabTopArtists = "Top Artists";
var QtnToolbarRefresh = "Refresh";
var QtnToolbarBack = "Back";

var QtnSearchHeader = "Search";
var QtnSearchSubmit = "Search";
var QtnSearchClear = "Clear";
var QtnSearchArtists = "Artists";
var QtnSearchTracks = "Tracks";
var QtnSearchListeners = "Listeners: %1";

var QtnTopTracksHeader = "Top Tracks";
var QtnTopTracksMore = "More";
var QtnTopTracksFilter = "Filter items...";
var QtnTopTracksFilterClear = "Clear";

var QtnTopArtistsHeader = "Top Artists";
var QtnTopArtistsMore = "More";
var QtnTopArtistsListeners = "Listeners: %1";
var QtnTopArtistsFilter = "Filter items...";
var QtnTopArtistsFilterClear = "Clear";

var QtnTrackDetailArtist = "Artist:";
var QtnTrackDetailAlbum = "Album:";
var QtnTrackDetailDuration = "Duration:";
var QtnTrackDetailListeners = "Listeners:";
var QtnTrackDetailPlaycount = "Playcount:";
var QtnTrackDetailNoImage = "No cover art available";
var QtnTrackDetailNoDescription = "Description not available.";
var QtnTrackDetailMoreInfo = "More info";
```

```
var QtnArtistDetailTags = "Tags:";
var QtnArtistDetailListeners = "Listeners:";
var QtnArtistDetailPlaycount = "Playcount:";
var QtnArtistDetailNoImage = "No cover art available";
var QtnArtistDetailNoBio = "Description not available.";
var QtnArtistDetailMoreInfo = "More info";

$(window).trigger("translate");
```

locales/fi/localization.js

```
var QtnFailedToFetchTracks = "Kappalelistan haku epäonnistui";
var QtnFailedToFetchArtists = "Artistilistan haku epäonnistui";

var QtnTabSearch = "Haku";
var QtnTabTopTracks = "Kappaleet";
var QtnTabTopArtists = "Artistit";
var QtnToolbarRefresh = "Päivitä";
var QtnToolbarBack = "Takaisin";

var QtnSearchHeader = "Haku";
var QtnSearchSubmit = "Etsi";
var QtnSearchClear = "Tyhjennä";
var QtnSearchArtists = "Artisteja";
var QtnSearchTracks = "Kappaleita";
var QtnSearchListeners = "Kuuntelijoita: %1";

var QtnTopTracksHeader = "Kappaleet";
var QtnTopTracksMore = "Näytä lisää";
var QtnTopTracksFilter = "Suodata kappaleita...";
var QtnTopTracksFilterClear = "Tyhjennä";

var QtnTopArtistsHeader = "Artistit";
var QtnTopArtistsMore = "Näytä lisää";
var QtnTopArtistsListeners = "Kuuntelijoita: %1";
var QtnTopArtistsFilter = "Suodata artisteja...";
var QtnTopArtistsFilterClear = "Tyhjennä";

var QtnTrackDetailArtist = "Artisti:";
var QtnTrackDetailAlbum = "Albumi:";
var QtnTrackDetailDuration = "Kesto:";
var QtnTrackDetailListeners = "Kuuntelijoita:";
var QtnTrackDetailPlaycount = "Soittokertoja:";
var QtnTrackDetailNoImage = "Ei kuvaa saatavilla";
var QtnTrackDetailNoDescription = "Kuvausta ei saatavilla.";
var QtnTrackDetailMoreInfo = "Lisätietoja";

var QtnArtistDetailTags = "Tunnisteet:";
var QtnArtistDetailListeners = "Kuuntelijoita:";
var QtnArtistDetailPlaycount = "Soittokertoja:";
var QtnArtistDetailNoImage = "Ei kuvaa saatavilla";
var QtnArtistDetailNoBio = "Kuvausta ei saatavilla.";
var QtnArtistDetailMoreInfo = "Lisätietoja";

$(window).trigger("translate");
```

engine.js

```
var supportedLocales = ["fi"];
```

```
var translationsLoaded = false;
var musicEngine;
var currentArtist;
var currentTrack;

$(document).ready(function(){
	if( document.URL.indexOf("#") > 0 )
		window.location.assign("index.html");
	else{
		var language = window.navigator.userLanguage || window.navigator.language;
		language = language.toLowerCase();
		var parts = language.split("-");
		if( parts.length > 0 ) language = parts[0];
		musicEngine = new MusicEngine( "en" );
		var head = document.getElementsByTagName('head')[0];
		var script = document.createElement('script');
		script.type = "text/javascript";
		if( !window.widget && supportedLocales.indexOf( language ) >= 0 ){
			script.src = "locales/" + language + "/localization.js";
		}
		else{
			script.src = "localization.js";
		}
		head.appendChild(script);

		$(window).on("translate", function(e){
			translationsLoaded=true;
		});
	}
});

function onTrackItemClicked(e){
	if(e) e.preventDefault();
	var id = this.getAttribute("data-track-id" );
	showTrackDetailsPage( id );
	return false;
}

function onArtistItemClicked(e){
	if(e) e.preventDefault();
	var id = this.getAttribute("data-artist-id" );
	showArtistDetailsPage( id );
	return false;
}

function textToHtml( text ){
	var out = text;
	if( out ){
		var index = -1
		do{
			index =  out.indexOf( '\n', index+1 );
			if( index >= 0 ){
				out = out.slice(0,index)+"<br />"+out.slice(index+1,out.length);
				index = index + 6;
			}
		} while( index >= 0 );
	}
	return out;
}

function showTrackDetailsPage( id ){
	var track  = musicEngine.getItemById( id );
	var show = function(){
		currentTrack = track;
		$('#TrackPageHeader').html( track.name || "" );
		$('#TrackPageArtist').html( track.artistName || "" );
```

```
            $('#TrackPageAlbum').html( track.albumName || "-" );
            var seconds = track.duration % 60;
            var duration = Math.floor( track.duration / 60 ) + ":"
                  + (seconds < 10 ? "0" : "") + seconds;
            $('#TrackPageDuration').html( duration );
            $('#TrackPageDescription').html( textToHtml(track.description)
                  || QtnTrackDetailNoDescription );
            $('#TrackPageListeners').html( track.listeners || "" );
            $('#TrackPagePlaycount').html( track.playcount || "" );
            $('#TrackPageImage').attr( "src", track.largeImageUrl || "" );
            $('#TrackPageBrowser').attr( "href", track.url );
            $.mobile.changePage("#TrackPage", {
                  transition : "slide"
            });
      }
      if( !track || !track.albumName ){
            $.mobile.showPageLoadingMsg();
            musicEngine.fetchTrackInfo( id, function( trackObj ){
                  $.mobile.hidePageLoadingMsg();
                  if( trackObj ){
                        track = trackObj;
                  }
                  if( track ) show();
            });
      }
      else show();
}

function showArtistDetailsPage( id ){
      var artist = musicEngine.getItemById( id );
      if( !artist ) return;
      var show = function(){
            currentArtist = artist;
            $('#ArtistPageHeader').html( artist.name || "" );
            var tagsStr;
            if( artist.tags ){
                  for( var i = 0; i < artist.tags.length; i++ ){
                        var link = "<a href='" + artist.tags[i].url
                              + "'>" + artist.tags[i].name + "</a>";
                        if( tagsStr ) tagsStr += ", " + link;
                        else tagsStr = link;
                  }
            }
            $('#ArtistPageTags').html( tagsStr || "-" );
            $('#ArtistPageListeners').html( artist.listeners || "" );
            $('#ArtistPagePlaycount').html( artist.playcount || "" );
            $('#ArtistPageImage').attr( "src", artist.largeImageUrl || "" );
            $('#ArtistPageBio').html( textToHtml(artist.bio)||QtnArtistDetailNoBio );
            $('#ArtistPageBrowser').attr( "href", artist.url );
            $.mobile.changePage("#ArtistPage", {
                  transition : "slide"
            });
      }
      if( !artist.tags ){
            $.mobile.showPageLoadingMsg();
            musicEngine.fetchArtistInfo(artist.mbid,artist.name,function(artistObj){
                  $.mobile.hidePageLoadingMsg();
                  if( artistObj ){
                        artist = artistObj;
                  }
                  if( artist ) show();
            });
      }
      else show();
}
```

```
function createListItem( url, imgUrl, title, subtitle ){
	var listItem = document.createElement('li');
	listItem.setAttribute('class','listitem ui-li-has-thumb');
	listItem.innerHTML = "<a href='" + url
		+"' data-transition='slide'><div class='ImageFrameMedium ui-li-thumb'>"
		+ "<img src='" + imgUrl + "' class='ListIcon' /></div><h3>" + title
		+ "</h3><div>" + subtitle + "</div></a>";
	return listItem;
}

$('#SearchPage').live( 'pageinit',function(event){
	var translate = function(){
		$('#SearchPageHeader').html( QtnSearchHeader );
		$('#SearchPageSubmit').val( QtnSearchSubmit ).button('refresh');
		$('.SearchPageTabs a[href="#SearchPage"] .ui-btn-text').text(
			QtnTabSearch);
		$('.SearchPageTabs a[href="#TracksPage"] .ui-btn-text').text(
			QtnTabTopTracks);
		$('.SearchPageTabs a[href="#ArtistsPage"] .ui-btn-text').text(
			QtnTabTopArtists);
		$('#SearchPageForm a.ui-input-clear').attr("title", QtnSearchClear);
	}
	$(window).on("translate", translate );
	if( translationsLoaded ) translate();

	$('#SearchPageForm').on('submit', function(e){
		e.preventDefault();
		var parent = document.getElementById('SearchList');
		while(parent.hasChildNodes()) parent.removeChild(parent.firstChild);
		$('#list').listview('refresh');
		$("#SearchPageSubmit").button('disable');
		var q = $('#SearchPageInput').val();
		var tracksFetched = false;
		var artistsFetched = false;
		var done = function(){
			if( tracksFetched && artistsFetched ){
				$('#SearchList').listview('refresh');
				$("#SearchPageSubmit").button('enable');
			}
		}
		musicEngine.clearBufferedDataItems();
		musicEngine.search(q, function(artists){
			if( artists && artists.length > 0 )	{
				var groupHeader = document.createElement('li');
				groupHeader.setAttribute( 'data-role', "list-divider" );
				groupHeader.innerHTML = QtnSearchArtists;
				parent.appendChild( groupHeader );

				for( var i = 0; i < artists.length; i++ ){
					var artistObj = artists[i];
					var listItem = createListItem( '#ArtistPage',
						artistObj.mediumImageUrl, artistObj.name,
						QtnSearchListeners.replace("%1", artistObj.listeners ) );
					listItem.setAttribute('data-artist-id', artistObj.id);
					$(listItem).on('click', onArtistItemClicked );
					parent.appendChild( listItem );
				}
			}
			artistsFetched = true;
			done();
		}, function(tracks){
			if( tracks && tracks.length > 0 ){
				var groupHeader = document.createElement('li');
				groupHeader.setAttribute( 'data-role', "list-divider" );
				groupHeader.innerHTML = QtnSearchTracks;
				parent.appendChild( groupHeader );
```

```
                    for( var i = 0; i < tracks.length; i++ ){
                        var trackObj = tracks[i];
                        var listItem = createListItem( '#ArtistPage',
                            trackObj.mediumImageUrl, trackObj.name,
                            QtnSearchListeners.replace("%1", trackObj.listeners ) );
                        listItem.setAttribute('data-track-id', trackObj.id);
                        $(listItem).on('click', onTrackItemClicked );
                        parent.appendChild( listItem );
                    }
                }
                tracksFetched = true;
                done();
            });
            return false;
        });
});

$('#TracksPage').live( 'pageinit',function(event){
    var translate = function(){
        $("#TracksPageHeader").html(QtnTopTracksHeader);
        $("#TracksPageMoreButton").html(QtnTopTracksMore).button('refresh');
        $('.TracksPageTabs a[href="#SearchPage"] .ui-btn-text').text(
            QtnTabSearch);
        $('.TracksPageTabs a[href="#TracksPage"] .ui-btn-text').text(
            QtnTabTopTracks);
        $('.TracksPageTabs a[href="#ArtistsPage"] .ui-btn-text').text(
            QtnTabTopArtists);
        $("#TracksPage .ui-input-text").attr("placeholder", QtnTopTracksFilter);
        $("#TracksPageRefresh  .ui-btn-text").text(QtnToolbarRefresh);
        $('#TracksPage a.ui-input-clear').attr("title", QtnTopTracksFilterClear);
    }
    $(window).on("translate", translate );
    if( translationsLoaded ) translate();

    var fetchError = function(){
        $("#TracksPageMoreButton").button('enable');
        alert( QtnFailedToFetchTracks );
    }

    $('#TracksPageMoreButton').live('click', function(event){
        $("#TracksPageMoreButton").button('disable');
        $('#TracksPage input[data-type="search"]').val("").trigger("change");
        musicEngine.fetchMoreTracks( topTracksCallback, fetchError );
    });

    $('#TracksPageRefresh').click(function(event){
        $("#TracksPageMoreButton").button('enable');
        musicEngine.clearFetchedTopTracks();
        var parent = document.getElementById('TracksList');
        while(parent.hasChildNodes()) parent.removeChild(parent.firstChild);
        $('#TracksList').listview('refresh');
        $('#TracksPageMoreButton').trigger('click');
    });


    musicEngine.fetchMoreTracks( topTracksCallback, fetchError );
});

$('#ArtistsPage').live( 'pageinit',function(event){
    var translate = function(){
        $("#ArtistsPageHeader").html(QtnTopArtistsHeader);
        $("#ArtistsPageMoreButton").html(QtnTopArtistsMore).button('refresh');
        $('.ArtistsPageTabs a[href="#SearchPage"] .ui-btn-text').text(
            QtnTabSearch);
        $('.ArtistsPageTabs a[href="#TracksPage"] .ui-btn-text').text(
```

```
                QtnTabTopTracks);
        $('.ArtistsPageTabs a[href="#ArtistsPage"] .ui-btn-text').text(
                QtnTabTopArtists);
        $("#ArtistsPage .ui-input-text").attr("placeholder", QtnTopArtistsFilter);
        $("#ArtistsPageRefresh  .ui-btn-text").text(QtnToolbarRefresh);
        $('#ArtistsPage a.ui-input-clear').attr("title", QtnTopArtistsFilterClear);
    }
    $(window).on("translate", translate );
    if( translationsLoaded ) translate();

    var fetchError = function(){
        $("#ArtistsPageMoreButton").button('enable');
        alert( QtnFailedToFetchArtists );
    }

    $('#ArtistsPageMoreButton').live('click', function(event){
        $("#ArtistsPageMoreButton").button('disable');
        $('#ArtistsPage input[data-type="search"]').val("").trigger("change");
        musicEngine.fetchMoreArtists( topArtistsCallback, fetchError );
    });

    $('#ArtistsPageRefresh').click(function(event){
        $("#ArtistsPageMoreButton").button('enable');
        musicEngine.clearFetchedTopArtists();
        var parent = document.getElementById('ArtistsList');
        while(parent.hasChildNodes()) parent.removeChild(parent.firstChild);
        $('#ArtistsList').listview('refresh');
        $('#ArtistsPageMoreButton').trigger('click');
    });

    musicEngine.fetchMoreArtists( topArtistsCallback, fetchError );
});

$('#TrackPage').live( 'pageinit',function(event){
    var translate = function(){
        $('#TrackPageArtistTitle').html( QtnTrackDetailArtist );
        $('#TrackPageAlbumTitle').html( QtnTrackDetailAlbum );
        $('#TrackPageDurationTitle').html( QtnTrackDetailDuration );
        $('#TrackPageListenersTitle').html( QtnTrackDetailListeners );
        $('#TrackPagePlaycountTitle').html( QtnTrackDetailPlaycount );
        $('#TrackPageImage').attr( "alt", QtnTrackDetailNoImage );
        $(".TrackPageBack .ui-btn-text").text(QtnToolbarBack);
        $('#TrackPageBrowser .ui-btn-text').text(QtnTrackDetailMoreInfo);
    }
    $(window).on("translate", translate );
    if( translationsLoaded ) translate();

    $('#TrackPageArtistButton').on('click', function(e){
        if(e) e.preventDefault();
        if( currentTrack ){
            var artist;
            if( currentTrack.artistId )
                artist = musicEngine.getItemById( currentTrack.artistId );
            if(artist && artist.tags) showArtistDetailsPage(currentTrack.artistId);
            else{
                $.mobile.showPageLoadingMsg();
                musicEngine.fetchArtistInfo( currentTrack.artistMbid,
                    currentTrack.artistName, function( artistObj ){
                    $.mobile.hidePageLoadingMsg();
                    if( artistObj ){
                        artist = artistObj;
                    }
                    showArtistDetailsPage( artistObj.id );
                });
            }
        }
```

```
            return false;
        });
});

$('#ArtistPage').live( 'pageinit',function(event){
        var translate = function(){
                $('#ArtistPageTagsTitle').html( QtnArtistDetailTags );
                $('#ArtistPageListenersTitle').html( QtnArtistDetailListeners );
                $('#ArtistPagePlaycountTitle').html( QtnArtistDetailPlaycount );
                $('#ArtistPageImage').attr( "alt", QtnArtistDetailNoImage );
                $(".ArtistPageBack  .ui-btn-text").text(QtnToolbarBack);
                $('#ArtistPageBrowser .ui-btn-text').text(QtnArtistDetailMoreInfo);
        }
        $(window).on("translate", translate );
        if( translationsLoaded ) translate();
});

function topTracksCallback( totalCount, tracksPerPage, page, results ){
        var parent = document.getElementById('TracksList');
        for( var i = 0; i < results.length; i++ ){
                var trackObj = results[i];
                var rank = trackObj.rank;
                if( (rank-1) % 10 == 0 ){
                        var groupHeader = document.createElement('li');
                        groupHeader.setAttribute( 'data-role', "list-divider" );
                        var from = rank;
                        var to = from + 9;
                        groupHeader.innerHTML = " " + from + "-" + to;
                        parent.appendChild( groupHeader );
                }

                var listItem = createListItem( '#TrackPage', trackObj.mediumImageUrl,
                        trackObj.name, trackObj.artistName );
                listItem.setAttribute('data-track-id', trackObj.id);
                $(listItem).on('click', onTrackItemClicked);
                parent.appendChild( listItem );
        }

        $('#TracksList').listview('refresh');
        if( page < (totalCount / tracksPerPage) )
                $("#TracksPageMoreButton").button('enable');
}

function topArtistsCallback( totalCount, albumsPerPage, page, results ){
        var parent = document.getElementById('ArtistsList');
        for( var i = 0; i < results.length; i++ )
        {
                var artistObj = results[i];
                var rank = artistObj.rank;
                if( (rank-1) % 10 == 0 )
                {
                        var groupHeader = document.createElement('li');
                        groupHeader.setAttribute( 'data-role', "list-divider" );
                        var from = rank;
                        var to = from + 9;
                        groupHeader.innerHTML = " " + from + "-" + to;
                        parent.appendChild( groupHeader );
                }

                var listItem = createListItem('#ArtistPage', artistObj.mediumImageUrl,
                        artistObj.name,
                        QtnTopArtistsListeners.replace("%1",artistObj.listeners));
                listItem.setAttribute('data-artist-id', artistObj.id);
                $(listItem).on('click', onArtistItemClicked);
                parent.appendChild( listItem );
        }
```

```
        $('#ArtistsList').listview('refresh');
        if( page < (totalCount / albumsPerPage) )
                $("#ArtistsPageMoreButton").button('enable');
}

function MusicApp(){
    this.openUrl = function( url ){
        var cb = function(){};
        Cordova.exec( cb, cb, "com.cordova.MusicApp", "openUrl", [url] );
    }
    return this;
}

if( window.Cordova ){
    Cordova.addConstructor( "com.cordova.MusicApp", function () {
        navigator.MusicApp = new MusicApp();
    });
}

$('a').live('click', function(e){
        var url = this.href;
        if( url && this.host != window.location.host ){
                if( navigator.MusicApp ) navigator.MusicApp.openUrl( url );
                else window.open(url);
                return false;
        }
        else return true;
});
```

musicengine.js

```
var API_KEY = "5eec32a3108c7b1c644d10784aa39f15";
var API_ROOT = "http://ws.audioscrobbler.com/2.0/";
var METHOD_GETTOPTRACKS = "chart.gettoptracks";
var METHOD_GETTOPARTISTS = "chart.gettopartists";
var METHOD_GETTRACKINFO = "track.getInfo";
var METHOD_GETARTISTINFO = "artist.getInfo";
var METHOD_ARTISTSEARCH = "artist.search";
var METHOD_TRACKSEARCH = "track.search";
var ENTRIES_PER_PAGE = 50;
var SEARCH_ENTRIES_LIMIT = 5;
var TIMEOUT = 8000;

function MusicEngine( lang ){
        var m_lang = lang;
        var m_tracks = new Array();
        var m_tracksPagesFetched = 0;
        var m_tracksTotalPages = 1;
        var m_artists = new Array();
        var m_artistsPagesFetched = 0;
        var m_artistsTotalPages = 1;
        var self = this;
        var m_idCounter = 1;

        var associateArtistWithTracks = function( artist ){
                for( var i = 0; i < m_tracks.length; i++ ){
                        if( !m_tracks[i].artistId && (
                                artist.mbid && m_tracks[i].artistMbid == artist.mbid
                                || artist.url && m_tracks[i].artistUrl == artist.url
                                || artist.name && m_tracks[i].artistName == artist.name ) ){
                                m_tracks[i].artistId = artist.id;
```

```
            }
        }
}

var associateTrackWithArtist = function( track ){
        if( track && !track.artistId )
        {
                for( var i = 0; i < m_artists.length; i++ ){
                        if(track.artistMbid && m_artists[i].mbid == track.artistMbid
                                || track.artistUrl && m_artists[i].url == track.artistUrl
                                || track.artistName && m_artists[i].name == track.artistName){
                                track.artistId = m_artists[i].id;
                                break;
                        }
                }
        }
}

var clearArtistAssociation = function( id ){
        for( var i = 0; i < m_tracks.length; i++ ){
                if( m_tracks[i].artistId == id )
                        m_tracks[i].artistId = null;
        }
}

var extractImage = function( target, source ){
        if( source.image ){
                for( var j = 0; j < source.image.length; j++ ){
                        if( source.image[j].size == "medium" )
                                target.mediumImageUrl = source.image[j]["#text"];
                        else if( source.image[j].size == "extralarge" )
                                target.largeImageUrl = source.image[j]["#text"];
                }
        }
}

var extractCommonDataFields = function( target, source ){
        target.name = source.name;
        target.listeners = source.listeners;
        target.playcount = source.playcount;
        target.mbid = source.mbid;
        target.url = source.url;
        if( !target.id ) target.id = m_idCounter++;
        extractImage( target, source );
}

var lfmAjaxCall = function( method, data, successCallback, errorCallback ){
        var uri = API_ROOT + "?method=" + method;
        data.api_key = API_KEY;
        data.format = 'json';
        $.ajax({
                url: uri,
                dataType: 'jsonp',
                timeout: TIMEOUT,
                processData: true,
                data: data,
                error: function(response) {
                        errorCallback(response);
                },
                success: function(response) {
                        successCallback(response);
                }
        });
}

this.search = function( q, artistsCallback, tracksCallback ){
```

```
        var artistSearchData = {
                artist : q,
                limit: SEARCH_ENTRIES_LIMIT
        }

        lfmAjaxCall( METHOD_ARTISTSEARCH, artistSearchData, function(response){
                var artistResults = new Array();
                if( response.results && response.results.artistmatches.artist ){
                        var artists = response.results.artistmatches.artist;
                        for( var i = 0; i < artists.length; i++ ){
                                var fetchedArtistData = artists[i];
                                var artist = new Object();
                                extractCommonDataFields( artist, fetchedArtistData );
                                associateArtistWithTracks( artist );
                                artistResults.push( artist );
                                m_artists.push( artist );
                        }
                }
                if( typeof artistsCallback === "function" )
                        artistsCallback( artistResults );
        }, function(response){
                if( typeof artistsCallback === "function" )
                        artistsCallback();
        });

        var trackSearchData = {
                track : q,
                limit: SEARCH_ENTRIES_LIMIT
        }
        lfmAjaxCall( METHOD_TRACKSEARCH, trackSearchData, function(response){
                var trackResults = new Array();
                if( response.results && response.results.trackmatches.track ){
                        var tracks = response.results.trackmatches.track;
                        for( var i = 0; i < tracks.length; i++ ){
                                var fetchedTrackData = tracks[i];
                                var track = new Object();
                                extractCommonDataFields( track, fetchedTrackData );
                                track.artistName = fetchedTrackData.artist;
                                associateTrackWithArtist( track );
                                trackResults.push( track );
                                m_tracks.push( track );
                        }
                }
                if( typeof tracksCallback === "function" )
                        tracksCallback( trackResults );
        }, function(response){
                if( typeof tracksCallback === "function" ) tracksCallback();
        });
}

this.fetchTrackInfo = function( id, callback ){
        var track = this.getItemById( id );
        var data = {
                autocorrect: 0
        };
        if( track.mbid ) data.mbid = track.mbid;
        else{
                data.track = track.name;
                data.artist = track.artistName;
        }

        lfmAjaxCall( METHOD_GETTRACKINFO, data, function(response){
                if( response.track ){
                        if( !track ){
                                track = new Object();
                                extractCommonDataFields( track, response.track );
```

```
                        m_tracks.push( track );
                    }
                    track.duration = parseInt(response.track.duration);
                    if( track.duration > 5000 ) track.duration = track.duration / 1000;
                    if( response.track.album ){
                        track.albumName = response.track.album.title;
                        track.albumUrl = response.track.album.url;
                        track.albumMbid = response.track.album.mbid;
                        if( !track.mediumImageUrl )
                            extractImage( track, response.track.album );
                    }
                    if( response.track.artist ){
                        track.artistName = response.track.artist.name;
                        track.artistUrl = response.track.artist.url;
                        track.artistMbid = response.track.artist.mbid;
                    }
                    if( response.track.wiki ){
                        track.description = response.track.wiki.content;
                    }
                    if( response.track.playcount )
                        track.playcount = response.track.playcount;
                    associateTrackWithArtist( track );
                }
                if( typeof callback === "function" ) callback( track );
            }, function(response){
                if( typeof callback === "function" ) callback();
            });
        }

        this.fetchArtistInfo = function( id, name, callback ){
            var artist;
            var mbid;
            if( typeof id == "number" ){
                artist = this.getItemById( id );
                if( artist ) mbid = artist.mbid;
            }
            else mbid = id;

            var data = {
                autocorrect: 0
            };
            if( mbid ) data.mbid = mbid;
            else data.artist= name;
            if( m_lang ) data.lang = m_lang;

            lfmAjaxCall( METHOD_GETARTISTINFO, data, function(response){
                if( response.artist ){
                    if( !artist ){
                        artist = new Object();
                        extractCommonDataFields( artist, response.artist );
                        associateArtistWithTracks( artist );
                        m_artists.push( artist );
                    }
                    artist.bio = response.artist.bio.content;
                    artist.playcount = response.artist.stats.playcount;
                    artist.listeners = response.artist.stats.listeners;
                    artist.tags = response.artist.tags.tag;
                }
                if( typeof callback === "function" ) callback( artist );
            }, function(response){
                if( typeof callback === "function" ) callback();
            });
        }

        this.clearFetchedTopTracks = function(){
            var i = 0;
```

```
        while( i < m_tracks.length ){
            if( m_tracks[i].rank )
                delete m_tracks.pop();
            else i++;
        }
        m_tracksPagesFetched = 0;
    }

    this.clearFetchedTopArtists = function(){
        var i = 0;
        while( i < m_artists.length ){
            if( m_artists[i].rank ){
                clearArtistAssociation( m_tracks[j].artistId );
                delete m_artists.pop();
            }
            else i++;
        }
        m_artistsPagesFetched = 0;
    }

    this.clearBufferedDataItems = function(){
        var i = 0;
        while( i < m_tracks.length ){
            if( m_tracks[i].rank ) i++;
            else delete m_tracks.pop();
        }
        i = 0;
        while( i < m_artists.length ){
            if( m_artists[i].rank ) i++;
            else{
                clearArtistAssociation( m_artists[i].artistId );
                delete m_artists.pop();
            }
        }
    }

this.fetchMoreTracks = function( callback, errorCallback ){
        if( m_tracksPagesFetched < m_tracksTotalPages ){
            var parseData = function(r){
                if( typeof r !== "object" || r.error ){
                    if( typeof errorCallback === "function" )
                        errorCallback();
                }
                else{
                    m_tracksPagesFetched++;
                    var tracks = r.tracks;
                    var trackCount = parseInt(tracks.track.length);

                    var page = parseInt(tracks["@attr"].page);
                    var tracksPerPage = parseInt(tracks["@attr"].perPage);
                    var totalCount = parseInt(tracks["@attr"].total);
                    m_tracksTotalPages = parseInt(tracks["@attr"].totalPages);
                    var newTracks = new Array();

                    for( var i = 0; i < trackCount; i++ ){
                        var track = tracks.track[i];
                        var rank = (page-1)*tracksPerPage + i + 1;
                        var trackObj = new Object();
                        trackObj.rank = rank;

                        extractCommonDataFields( trackObj, track );
                        trackObj.duration = parseInt(track.duration);
                        if( trackObj.duration > 5000 )
                            trackObj.duration = trackObj.duration / 1000;
                        trackObj.artistMbid = track.artist.mbid;
                        trackObj.artistName = track.artist.name;
```

```
                        trackObj.artistUrl = track.artist.url;
                        associateTrackWithArtist( track );
                        m_tracks.push( trackObj );
                        newTracks.push( trackObj );
                    }

                    if( typeof callback === "function" )
                        callback( totalCount, tracksPerPage, page, newTracks );
                }
            }

            var data = {
                page: m_tracksPagesFetched+1,
                limit: ENTRIES_PER_PAGE
            };
            lfmAjaxCall( METHOD_GETTOPTRACKS, data, parseData, function(response){
                if( typeof errorCallback === "function" ) errorCallback();
            });
        }
    }

    this.fetchMoreArtists = function( callback, errorCallback ){
        if( m_artistsPagesFetched < m_artistsTotalPages ){
            var parseData = function(r){
                if( typeof r !== "object" || r.error ){
                    if( typeof errorCallback === "function" )
                        errorCallback();
                }
                else{
                    m_artistsPagesFetched++;
                    var artists = r.artists;
                    var artistCount = parseInt(artists.artist.length);

                    var page = parseInt(artists["@attr"].page);
                    var artistsPerPage = parseInt(artists["@attr"].perPage);
                    var totalCount = parseInt(artists["@attr"].total);
                    m_artistsTotalPages = parseInt(artists["@attr"].totalPages);
                    var newArtists = new Array();

                    for( var i = 0; i < artistCount; i++ ){
                        var artist = artists.artist[i];
                        var rank = (page-1)*artistsPerPage + i + 1;
                        var artistObj = new Object();
                        artistObj.rank = rank;
                        extractCommonDataFields( artistObj, artist );
                        associateArtistWithTracks( artist );
                        m_artists.push( artistObj );
                        newArtists.push( artistObj );
                    }

                    if( typeof callback === "function" )
                        callback( totalCount, artistsPerPage, page, newArtists );
                }
            }

            var data = {
                page: m_artistsPagesFetched+1,
                limit: ENTRIES_PER_PAGE
            };
            lfmAjaxCall( METHOD_GETTOPARTISTS, data, parseData, function(response){
                if( typeof errorCallback === "function" ) errorCallback();
            });
        }
    }

    this.getItemById = function( id ){
```

```
            if( id ){
                for( var i = 0; i < m_tracks.length; i++ )
                    if( m_tracks[i].id == id ) return m_tracks[i];
                for( var i = 0; i < m_artists.length; i++ )
                    if( m_artists[i].id == id ) return m_artists[i];
            }
            return null;
        }
    }
```

C Cordova-liitännäinen

musicapp.h

```
#ifndef MUSICAPP_H
#define MUSICAPP_H

#include "../cplugin.h"
#include "../pluginregistry.h"

#include <QObject>
#include <QString>

#include <qplatformdefs.h>

class MusicApp : public CPlugin{
    Q_OBJECT
public:
    explicit MusicApp();
    void init();
public slots:
    void openUrl( int scId, int ecId, const QString& url );
private:
    static MusicApp* m_musicApp;
};

#endif // MUSICAPP_H
```

musicapp.cpp

```
#include "musicapp.h"
#include <QDesktopServices>
#include <QUrl>

MusicApp* MusicApp::m_musicApp = new MusicApp;

MusicApp::MusicApp() : CPlugin(){
    PluginRegistry::getRegistry()->registerPlugin("com.cordova.MusicApp",this);
}

void MusicApp::init(){
}

void MusicApp::openUrl(int scId, int ecId, const QString& url ){
    Q_UNUSED(ecId)
    QDesktopServices::openUrl ( QUrl(url) );
    callback(scId, "");
}
```

plugins.xml

```
<?xml version="1.0" encoding="utf-8"?>
<plugins>
    <plugin name="MusicApp" value="com.cordova.MusicApp"/>
</plugins>
```