Lappeenrannan teknillinen yliopisto
Teknistaloudellinen tiedekunta
Tietotekniikan koulutusohjelma

Diplomityö

Tuomo Kujala

MOBIILISOVELLUSTEN KESKITETYT JAKELUKANAVAT

Työn tarkastaja(t): Professori Kari Smolander
DI Tommi Kähkönen

TIIVISTELMÄ

Lappeenrannan teknillinen yliopisto
Teknistaloudellinen tiedekunta
Tietotekniikan koulutusohjelma

Tuomo Kujala

Mobiilisovellusten keskitetyt jakelukanavat

Diplomityö

2012

64 sivua, 10 kuvaa, 4 taulukkoa

Työn tarkastajat: Professori Kari Smolander
DI Tommi Kähkönen

Hakusanat: keskitetyt jakelukanavat, App Store, Google Play, Windows Phone Marketplace, Android, iOS, Windows Phone
Keywords: centralized distribution channel, App Store, Google Play, Windows Phone Marketplace, Android, iOS, Windows Phone

Diplomityössä tutkitaan mobiilisovellusten keskitettyjä jakelukanavia. Nämä uudet jakelukanavat ovat mahdollistaneet sovellusten uuden tyylisen ja helpon jakelun. Työssä tutkitaan itsenäisen kehittäjän näkökulmasta sovelluksen kehitystä Android-, iOS- ja Windows Phone -ohjelmistoalustoilla, ja sen julkaisua App Storessa, Google Playssa ja Windows Phone Marketplacella. Tavoitteena on tutkia huomataanko esimerkkisovelluksen kehityksessä ja julkaisussa merkittäviä eroja jakelukanavien välillä. Prosesseissa havaittiin eroja, mutta ei kuitenkaan niin merkittäviä, että jokin jakelukanava voitaisiin nostaa selkeästi toisten edelle helpompana tai rajoitteista vapaampana vaihtoehtona.

ABSTRACT

Lappeenranta University of Technology
Faculty of Technology Management
Degree Program in Information Technology

Tuomo Kujala

Centralized distribution channels for mobile applications

Master's Thesis

64 pages, 10 figures, 4 tables

Examiners: Professor Kari Smolander
DI Tommi Kähkönen

Keywords: centralized distribution channel, App Store, Google Play, Windows Phone Marketplace, Android, iOS, Windows Phone

This Master's Thesis examines the centralized distribution channels for mobile applications. These new channels grant an easy and new way for the developers to distribute mobile applications. Thesis researches the development of an application with Android, iOS and Windows Phone platforms and its release to the App Store, Google Play and to Windows Phone Marketplace from an independent developers point of view. The Goal of the research is to map out the key differences in the development and in the release of the example application between the different distribution channels. Some differences in the processes were found, but nothing so major that one distribution channel could be clearly preferred over another as an easier or more constraint free option.

ALKUSANAT

Työ on tehty Helsingin Kalliossa arki-iltaisin ja viikonloppuisin elämänrientoja ja –houkutuksia vältellen. Kiitos rakkaalle vaimolle tuesta ja ystäville ”pitäs tot dippaa tehdä”-ymmärryksestä.

SISÄLLYSLUETTELO

SYMBOLI- JA LYHENNELUETTELO

ADT	Android Development Tools
AOSP	Android Open Source Project
APK	Android Application Package
ARC	Automatic Reference Counting
AVD	Android Virtual Device
CLR	Common Language Runtime
DRT	Direct Run-time
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
MSDN	Microsoft Developer Network
REST	Representational State Transfer
SDK	Software Development Kit
SKU	Stock Keeping Unit
USB	Universal Serial Bus
VOIP	Voice Over Internet Protocol
XAML	Extensible Application Markup Language
XML	Extensible Markup Language

1 JOHDANTO

1.1 Tausta

Mobiililaitteiden kehitys on ollut viime vuosien aikana nopeaa. Laitteiden prosessoritehot, näyttöresoluutiot ja tietoliikennenopeudet ovat kasvaneet merkittävästi. Niistä on tullut oivallinen alusta toteuttaa itsenäisiä ja palvelimen kanssa kommunikoivia asiakassovelluksia. Tämä on avannut uusia mahdollisuuksia mobiilisovellusten kehitykselle.

Myös mobiilisovellusten jakelukanavissa on tapahtunut suuri muutos. Aiemmin mobiilisovellusten kehitys oli suurilta osin teleoperaattoreiden, laitevalmistajien ja muutaman sisällöntuottajan hallinnassa. Tähän suljettuun malliin ajavia tekijöitä olivat laiterajoitteet, kuten prosessoriteho ja muistin määrä, laitekannan monimuotoisuus sekä laitteiden jatkuva nopea kehitys. Applen ja Googlen tulo markkinoille on muuttanut perinteistä mobiiliteollisuutta ja edistänyt siirtymää suljetusta mallista avoimeen, jossa sovellusten kehittäjien kirjo on huomattavasti suurempi [1].

Kiinnostus mobiilikehitykseen on kasvanut myös itsenäisten kehittäjien ja pienempien yritysten keskuudessa. Ohjelmistoalustojen kehittäjien tarjoamat keskitetyt jakelukanavat, kuten Applen App Store, Googlen Google Play ja Microsoftin Windows Phone Marketplace, ovat muokanneet älypuhelinkäyttäjien käyttötottumuksia radikaalisti, sekä mahdollistaneet sovellusten helpon julkaisun ja jakelun. Ne ovat käytännössä luoneet uuden markkinasegmentin. Kolmansien osapuolien kehittämien sovellusten kysyntä ja tarjonta ovat kasvaneet voimakkaasti näiden jakelukanavien myötä.

1.2 Tavoitteet ja rajaukset

Työssä tutkitaan mobiilisovellusten kehittämistä ja niiden jakelua itsenäisen kehittäjän näkökulmasta. Diplomityön tekijällä oli ammattitausta mobiilikehityksestä, mutta ei merkittävää aiempaa kokemusta tutkimuksen kohteena olleista ohjelmistoalustoista ja niiden jakelukanavista. Työ toteutettiin ilman yrityksen tukea. Motivaationa työlle toimi halu oppia uusia teknologioita, ja halu ymmärtää näiden uusien jakelukanavien tarjoamia mahdollisuuksia itsenäiselle kehittäjälle.

Ohjelmistoalustoihin ja jakelukanaviin perehdyttiin toteuttamalla web-pohjaisen Tuntilistat.fi ajan- ja tehtävienseurantapalvelun mobiilisovellukset Android-, iOS- ja Windows Phone 7 -alustoille. Työ aloitettiin luomalla yleiskuva ohjelmistoalustojen ominaispiirteistä ja siitä mitä kehittäminen niillä vaatii. Tämän jälkeen kullakin alustalla toteutettiin oma versio Tuntilistat.fi-mobiilisovelluksesta ja julkaistiin se kyseisen alustan jakelukanavassa.

Tutkimuksen tavoitteena on kartoittaa merkittävimmät eroavaisuudet ja ominaispiirteet kehitystyössä eri alustoilla ja sovellusten julkaisussa niiden keskitetyissä jakelukanavissa. Huomataanko esimerkki sovelluksena toteutettavan Tuntilistat.fi-sovelluksen julkaisussa ja kehityksessä eroja jakelukanavien välillä? Asettaako joku jakelukanava tai alusta erityisiä rajoitteita julkaisulle? Mitä seikkoja itsenäisen kehittäjän tulee huomioida eri alustojen sovellusten kehityksessä ja jakelussa?

Työ on rajattu käsittelemään kolmen suosituimman ohjelmistoalustan jakelukanavaa: Applen App Storea, Googlen Google Playta ja Microsoftin Windows Phone Marketplacea [2]. App Store ja Windows Phone Marketplace ovat alustojensa ainoat jakelukanavat. Google Play ei ole alustansa ainoa jakelukanava, mutta se on selkeästi suosituin Android-sovellusten jakelukanava.

Ohjelmistoalustojen teknisessä tarkastelussa työ on rajattu käsittelemään kunkin alustan virallisesti tukemia ohjelmointikieliä ja tekniikoita. Työssä ei oteta kantaa kolmansien osapuolien tarjoamiin alustariippumattomiin kehityksen (engl. cross platform development) mahdollistaviin työkaluihin, kuten Corona SDK (Software Development Kit) , Flash CS5 ja PhoneGap.

1.3 Työn rakenne

Luvussa 2 esitellään keskitettyjä jakelukanavia yleisesti. Luvussa kuvaillaan jakelukanavien toimintaa, historiaa ja kehitystä, jakelukanavien ominaispiirteitä, sekä luodaan katsaus jakelukanavista aiemmin tehtyyn tutkimukseen. Luku 3 esittelee ohjelmistoalustat ja niiden kehitystyökalut. Luku on jaettu alustakohtaisiin alilukuihin ja viimeisessä aliluvussa esitellään alustojen merkittävimmät eroavaisuudet.

Luku 4 kuvaa kunkin jakelukanavan kehitys- ja julkaisuprosessin, ja niiden väliset erot. Luvussa 5 esitellään jakelukanavissa julkaistavien sovellusten hallinnointisivut, eli millaista lisätietoa sovelluksista on mahdollista antaa julkaisun tueksi, sekä mitä tietoa sovelluksesta on saatavilla kehittäjälle.

Luku 6 esittelee tutkimuksen käytännön osuuden. Siinä esitellään Tuntilistat.fi-palvelu, sekä diplomityössä toteutetut mobiilisovellukset. Luvussa esitellään myös sovelluksen julkaisu kussakin jakelukanavassa, julkaisussa havaitut erot, sekä muutamia julkaisun jälkeisiä havaintoja.

Luku 7 esittelee ja arvioi diplomityön tulokset. Luvussa 8 esitellään työn yhteenveto ja ideoita mahdollisista jatkotutkimuskohteista.

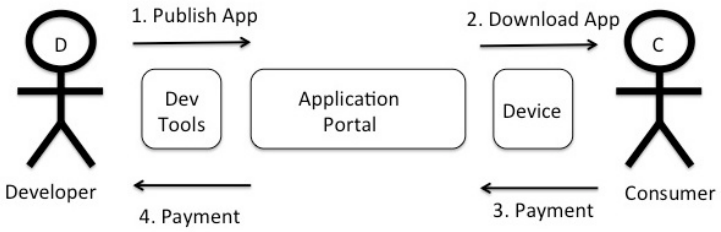
2 YLEISTÄ JAKELUKANAVISTA

Aiemmin älypuhelimen omistaja tyytyi pääasiallisesti puhelimen mukana tulleisiin ohjelmiin. Nyt suuri osa käyttäjistä haluaa, että kolmansien osapuolien tekemien ohjelmien saatavuus on hyvä ja niiden asentaminen on vaivatonta. [3]

Keskitetyistä jakelukanavista puhutaan myös usein kauppapaikkoina, sillä ne toimivat maksullisten sovellusten myyntikanavina loppukäyttäjälle. Toimintamalli on kaikilla alustoilla sama. Palveluntarjoajat Apple, Google ja Microsoft tarjoavat ja ylläpitävät jakelukanavaa, sekä määrittelevät niiden käyttöehdot. Jakelukanavien ylläpito ja kehitys rahoitetaan kehittäjiltä vaadittavilla kehityslisensseillä, jotka kehittäjien on ostettava omien sovellusten julkaisun mahdollistamiseksi. Tämän lisäksi jakelukanavien omistajat ottavat maksullisten sovellusten myynnistä prosentuaalisen korvauksen. Diplomityön tekohetkellä toukokuussa 2012 korvausten prosentuaalinen jako on kaikissa jakelukanavissa sama. Myydyn sovelluksen hinnasta 70% menee sovelluksen kehittäjälle ja 30% jakelukanavan ylläpitäjälle [4, 5, 6]. Jaosta on muodostunut alan standardi, eikä kukaan ole lähtenyt kilpailemaan sillä toistaiseksi.

Uudet jakelukanavat tarjoavat itsenäisten ohjelmistokehittäjien ja ohjelmistoyritysten sovelluksille saman näkyvyyden. Yksi keskitetty jakelukanava tarjoaa kehittäjille monia uusia mahdollisuuksia, mutta tuo myös mukanaan uusia haasteita. Itsenäisellä kehittäjällä ei aiemmin ole ollut mahdollisuutta jaella ohjelmaansa yhtä näkyvästi. Tähän tilaisuuteen on moni kehittäjä tarttunut, ja jakelukanavassa olevien sovellusten suuri määrä tekee sieltä erottumisen ja oman sovelluksen esille saamisen vaikeaksi. [3]

Holzer ja Ondrus kuvaavat mobiilisovellusten jakeluprosessia seuraavanlaisesti. Prosessilla tarkoitetaan koko ketjua, jossa kehittäjä kehittää sovelluksen ja julkaisee sen jakelukanavassa. Tämän jälkeen käyttäjä ostaa sovelluksen ja asentaa sen päätelaitteeseen. [7] Prosessin päävaiheet ovat esitelty kuvassa 1.



Kuva 1. Mobiilisovellusten jakeluprosessi Holzer ja Ondrus [7].

Jakelukanavaa hallinnoiva taho toimii rajapintana kehittäjän ja sovelluksen loppukäyttäjän välillä, aiemmin tämä rooli oli suurilta osin teleoperaattoreiden vastuulla.

Holzer ja Ondrus käyttävät Eric S. Raymondin määrittelemiä katedraali- ja basaarimalleja jaotellessaan ohjelmistoalustat suljettuihin ja avoimiin teknologioihin.

Suljetussa katedraalimallissa ohjelmistoalustan tarjoaja tekee kaikki strategiset päätökset yksin liittyen ohjelmistoalustan tulevaisuuteen. Tässä mallissa alustan tarjoajan pitää luonnollisesti palkata ohjelmistokehittäjät alustan kehittämiseen, eikä kehittäjäyhteisöjä voida hyödyntää apuna. [7]

Avoimessa basaarimallissa ohjelmistoalustan tarjoaja antaa kehittäjille pääsyn kaikkiin kehitysalustan ominaisuuksiin ja lähdekoodeihin. Avoimella lähdekoodilla pyritään luomaan kehittäjäyhteisö, joka auttaa alustan kehityksessä. [7]

Tämän jaottelun mukaisesti diplomityön käsittelemistä ohjelmistoalustoista Android perustuu basaarimalliin, ja iOS sekä Windows Phone 7 perustuvat katedraalimalliin.

Keskitetyssä jakelukanavamallissa sovellusten jakeluun tarjotaan vain yksi reitti, jonka kautta kaikki sovellukset julkaistaan ja jaellaan. Keskitetyn jakelukanavan etuina ovat muun muassa loppukäyttäjän sovellusten löytämisen ja lataamisen helppous, sekä kehittäjien sovellusten myymisen ja jakelun helpottaminen keskittämällä toiminta yhteen pisteeseen. Keskitetyt jakelukanavat keräävät kehittäjät ja loppukäyttäjän kaikki samaan paikkaan. [7]

Keskittämättömissä jakelukanavissa kehittäjät voivat vapaasti jaella sovelluksiaan valitsemassaan jakelukanavassa. Tämä malli poistaa kaikki keskitetyn mallin tuomat rajoitteet, mutta haittapuolena monimutkaistaa loppukäyttäjän kokonaiskuvan muodostamista ohjelmistoalustalle tarjolla olevista sovelluksista. Keskittämätön malli poistaa ohjelmistoalustan tarjoajan velvoitteen tarjota ja ylläpitää jakelukanavaa.

Ideologian suljettujen ja avointen järjestelmien suhteen heijastuu myös jakelukanaviin. App Store ja Windows Phone Marketplace ovat alustojensa ainoat jakelukanavat, toisin sanoen laitteisiin ei ole mahdollista asentaa sovelluksia mitään muuta kautta. Android eroaa kahdesta kilpailijastaan mahdollistamalla myös muita vaihtoehtoja sovellusten levittämiseen. Googlen Play, entiseltä nimeltään Android Market tarjoaa keskitetyn jakelukanavan, ja on suosituin jakelukanava Android-sovelluksille, mutta se ei ole Android käyttäjien ainoa mahdollisuus sovellusten lataamiseen. Osa Android-laitevalmistajista ja sisällöntarjoajista tarjoaa omia jakelukanaviaan. Tämän lisäksi Android-sovellusten jakelua ei ole rajoitettu millään tavalla, vaan kehittäjillä on mahdollisuus jaella sovelluksiaan myös esimerkiksi omien kotisivujensa kautta. Diplomityö on kuitenkin rajattu käsittelemään Android-alustan osalta vain Googlen tarjoamaan keskitettyä jakelukanavaa, eli Google Playta.

2.1 Jakelukanavien historia ja kehitys

Tutkimuksen kohteena olleista keskitetyistä jakelukanavista ensimmäisenä markkinoille ehti Applen App Store. Se avattiin heinäkuun yhdestoista päivä vuonna 2008. App Storea ei siis ollut olemassa ensimmäisen iPhonen julkaisun aikana, joka esiteltiin tammikuussa vuonna 2007 ja julkaistiin myyntiin saman vuoden kesäkuussa. [3]

Ennen virallisen App Storen avaamista kysyntä kolmansien osapuolien kehittämille sovelluksille oli kova ja aktiivinen homebrew-yhteisö onnistui takaisinmallintamaan (engl. reverse engineering) iPhonen-ohjelmiston ja purkamaan siihen omia sovelluksia. Näiden sovellusten asentaminen vaati iPhonen ohjelmiston murtamista (engl. jailbreak). Yhteisö onnistui tekemään sovelluksia ilman Applen hyväksyntää ja ilman minkäänlaista dokumentaatiota. Monet näistä sovelluksista olivat hyvin viimeisteltyjä ja muistuttivat paljon nykyisin virallisesti App Storesta saavilla olevia sovelluksia. Tämän operaation

vaativuus antoi viitteitä sovellusten kysynnän ja kehittäjien mielenkiinnon suuruudesta alustaa kohtaan. Vuoden 2007 lokakuussa epävirallisten sovellusten ja niiden jakelukanava toimineen Cydia-nimisen kauppapaikan suosio oli niin suuri, että Apple julkisti aikeensa kehittää ja esitellä virallinen jakelukanava ja kehitystyökalut iOS-alustalleen. Viralliset kehitystyökalut oli ladattavissa maaliskuussa 2008. [3]

Applen App Store avattiin heinäkuussa 2008 ja siellä oli 550 sovellusta ladattavissa. Sovellusten ja latausten määrä lähti nopeasti jyrkkään nousuun. Kolme päivää julkaisun jälkeen sovellusten määrä oli 800 ja latausten määrä oli jo 10 miljoonaa. Huhtikuussa 2009 Apple ilmoitti App Storen latausten ylittäneen miljardin rajan. Samalla sovellusten kokonaismäärä oli kasvanut 35 000 kappaleeseen. [8] Maaliskuussa 2012 sovellusten määrä on 585 000 ja latausten määrä on ylittänyt 25 miljardia. [9]

Google julkaisi oman jakelukanavansa Android Marketin elokuussa 2008, ja se avattiin käyttäjille saman vuoden lokakuussa. Android Marketissa oli aluksi mahdollisuus jakaa vain ilmaisia sovelluksia, ja tuki maksullisille sovelluksille tuli vuonna 2009 alueittain. Amerikkalaiset ja Iso-Britannialaiset kehittäjät saivat mahdollisen maksullisten sovellusten jakeluun helmikuussa, ja 29 muuta maata syyskuussa. Vuonna 2012 ilmaiset sovellukset ovat saatavilla kaikissa maissa ja maksulliset 129 maassa. Google teki jakelukanavalleen uudelleen brändäyksen maaliskuussa vuonna 2012, jolloin sen nimeksi muutettiin Google Play. [10]

Windows Phone Marketplace ehti markkinoille viimeisenä ja avattiin lokakuussa 2010. Kesäkuussa 2012 Microsoft tiedotti Windows Phone Marketplacen ylittäneen 100 000 sovelluksen rajan [11]. Syyskuussa 2012 Microsoft muutti Windows Phone Marketplacen nimen Windows Phone Storeksi.

2.2 Jakelukanavien ominaispiirteitä

Selkeä ominaispiirre diplomityön käsittelemissä keskitetyissä jakelukanavissa on jaeltavien sovellusten alhainen hinta ja kokonaan ilmaisten sovellusten suuri määrä. App Storessa ilmaisten sovellusten osuus elokuussa 2012 oli hieman alle 50% [9]. Google Playssa ilmaisten sovellusten määrä on vieläkin korkeampi, 74% [12]. Windows Phone

Marketplace asettuu näiden kahden välimaastoon ja siellä ilmaisia sovelluksia on 66% [13].

Suosituin hinta maksulliselle sovellukselle on App Storessa ja Windows Phone Marketplacessa 0,99$/0,79€. App Storen maksullisista sovelluksista lähes 50% on hinnoiteltu 0,99$-kategoriaan. Windows Phone Marketplacella dollarin kategoriaan menee 61% maksullisista sovelluksista. Google Playssa maksullisista sovelluksista 33% maksaa 0,99$. Suosituin maksullisten sovellusten hintakategoria Google Playssa on 1-2,5$, johon menee 40% maksullisista Android-sovelluksista. [9,11,12] Yleisesti ottaen käyttäjät ovat siis tottuneet mobiilimarkkinoilla hyvin edullisiin, tai jopa ilmaisiin sovelluksiin.

Alhaiset hinnat tarkoittavat luonnollisesti sitä, että sovellusten myyntimäärien tulee olla todella suuria kehityskulujen kattamiseksi ja varsinaisen taloudellisen kannattavuuden saavuttamiseksi. Tämä piirre ajaa myös sovelluksien yksinkertaistamiseen. Sovellusten toiminta on usein hyvin kohdennettua ja suunniteltu täyttämään yhtä selkeää tarvetta. Alhaisten hintojen tavoittelu on turhauttanut monia kehittäjiä. Menestyneen iOS-sovelluksen Twittereficin tehnyt Graig Hockenberry kommentoi turhautumistaan seikkaan, että monien kehittäjien hienot ja monimutkaisemmat sovellusideat jäävät toteuttamatta kokonaan, sillä myyntiä ja latauksia saavat vain 0,99$ hintaiset sovellukset [3].

2.3 Aiemmat tutkimukset jakelukanavista

Keskitetyt jakelukanavat mobiilialalla ovat melko uusi ilmiö. Tutkimusta näistä niin kutsutuista kauppapaikoista ja niiden vaikutuksesta markkinoihin on kuitenkin jossain määrin tehty.

Yamakami tutki mobiilikauppapaikkojen syntyyn ja menestykseen vaikuttaneita tekijöitä. Tekijät jaettiin kolmeen ryhmään: teknologisiin, taloudellisiin ja hyväksyntään. Teknologisesti vaikuttavia tekijöitä ovat ohjelmistoalustalle tarjottavat kehitystyökalut, jotka helpottavat sovellusten tekemistä, kehittäjille tarjottava tekninen tuki muun muassa laadukkaan dokumentaation muodossa, sekä laitekannan samankaltaisuus, joka omalta osaltaan yksinkertaistaa sovelluskehitystä. Taloudellisesti vaikuttavia tekijöitä on laitteiden, eli loppukäyttäjien suuri määrä, sekä maksullisista sovelluksista laskuttamisen

helppous. Kolmantena ryhmänä olevalla hyväksynnällä tarkoitetaan käyttäjien valmiutta maksaa ladattavista sovelluksista. Yamakami tutkii myös mobiiliteollisuuden siirtymistä muutaman sisällöntarjoajan suljetusta mallista avoimempaan malliin, jossa sovelluksia on käytännössä kaikkien mahdollista kehittää ja julkaista. Suljettuun malliin ajaneita tekijöitä olivat erityisesti laiterajoitteet, laitekannan hajanaisuus ja nopeat suuret muutokset laitteissa. [1]

Yoon, Yoo ja Choi tutkivat sovelluksen kehittäjän ja jakelukanavan hallitsijan välillä tapahtuvaa voitonjakoa mobiilisovellustalouden nousua selittävänä tekijänä. Keskitettyjen jakelukanavien avulla sovellukset saadaan jaeltua suoraan loppukäyttäjille, jolloin perinteisestä jakeluketjusta saadaan tiputettua yksi väliporras pois. Myynnin perusteella tehtävä voitonjako on turvallinen ja reiluksi koettu vaihtoehto molemmille osapuolille. Voitonjaon nähdään myös kasvattaneen tarjottavien sovellusten määrää sekä laatua. Työssä selvitettiin myös optimaalista voitonjakoa sovelluksen kehittäjän ja jakelukanavan hallitsijan välillä. Applen markkinoille tuomaa 70%-30% voitonjakoa kehittäjän ja jakelukanavan haltijan välillä pidettiin molemmille osapuolille kannattavana ja sopivana mallina. [14]

Copeland tutki kauppapaikkojen roolia ystävänä ja vihollisena teleoperaattoreille. Kauppapaikkojen yleistyminen on muokannut ihmisten tottumuksia puhelimen valinnassa. Nyt monelle on tärkein kriteeri sovellukset joita laitteelle on saatavilla. Operaattoreille haitallisena tekijänä on kilpailijoiden tuonti kotiin, esimerkiksi Skypen ja muiden Voice over IP (VOIP) –palveluiden muodossa, jotka luonnollisesti vähentävät ihmisten perinteisen puheluiden soittamista. Operaattoreiden toisistaan erottuminen tulee myös entistä vaikeammaksi, sillä samat sovellukset ovat saatavilla kaikilla. Hyväksi asiaksi operaattorit mieltävät käyttäjien valmiuden maksaa sovelluksista, vaikkakin hinnat kauppapaikoilla ovatkin melko edullisia. Samoin sovellusten monipuolistuminen ja saatavilla olevien sovellusten määrän kasvaminen on hyvä asia. Ihmiset käyttävät operaattoreiden verkkoja dataliikenteessä entistä runsaammin. [15]

Kimbler tutki teleoperaattoreiden mahdollisuutta kasvattaa tulostaan kauppapaikkojen avulla hyödyntäen jo olemassa olevia jakelukanavia, kuten App Storea tai vaihtoehtoisesti

luomalla omia kauppapaikkoja. Operaattorit sijoittavat uusien ja entistä nopeampien verkkojen rakentamiseen suuria rahasummia, joten niillä on luonnollinen halu päästä osalliseksi mobiilimarkkinoiden muutokseen. Kimbler ei kuitenkaan näe, että operaattorit voisivat hyödyntää nykyisiä jakelukanavia lisätulojen lähteenä. Toisaalta täysin uuden oman jakelukanavan luominen on myös hyvin vaikeaa. Se vaatisi todella suuria panostuksia. Markkinoilla erottuminen, sekä kehittäjien ja käyttäjien mukaan saaminen riittävissä määrissä on hyvin vaikeaa. Operaattoreiden paras mahdollisuus hyötyä mobiilimarkkinoiden muutoksista on myydä käyttäjille uudempia puhelimia ja kattavampia dataliikennesopimuksia. [16]

3 OHJELMISTOALUSTOJEN JA KEHITYSTYÖKALUJEN ESITTELYT

Kehittäminen Android-, iOS- ja Windows Phone -alustoilla on hyvin erilaista. Ohjelmointikielet, kirjastot sekä työkalut eroavat kaikki merkittävästi toisistaan. Seuraavassa luodaan alustakohtainen katsaus ohjelmistoalustoihin, sovellusten rakenteisiin ja kehitystyökaluihin.

Kaikki kolme kehitysalustaa tarjoavat kehitystyökalut (engl. SDK Software Development Kit) kehittäjille ilmaiseksi. Microsoft tarjoaa kehitystyökalunsa Windows Vista- ja Windows 7 -käyttöjärjestelmiin. Applen kehitystyökalut toimivat Applen OS X -käyttöjärjestelmissä. Android-kehitystyökalut eivät puolestaan ole käyttöjärjestelmäriippuvaisia, vaan ne ovat tarjolla niin Windows-, OS X- kuin Linux-ympäristöihin.

Kehitystyökalut sisältävät ohjelmointiympäristön (engl. IDE Integrated Development Environment), emulaattorin tai simulaattorin sekä tarvittavat ohjelmistokirjastot sovellusten toteuttamiseen ja kääntämiseen.

3.1 Android

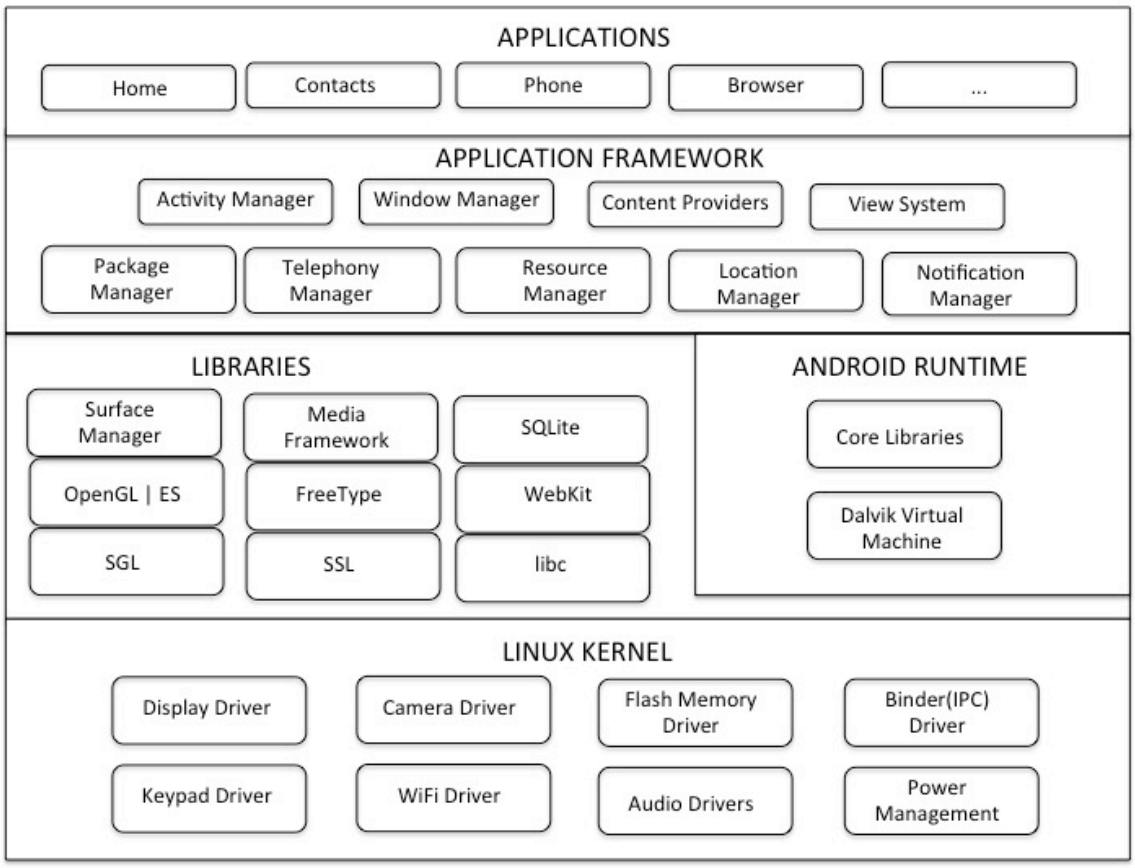
Android on Open Handset Alliancen kehittämä käyttöjärjestelmä mobiililaitteille. Open Handset Alliance perustettiin vuonna 2007 kehittämään ja edistämään avoimia standardeja mobiililaitteille. Liittoumaa perustamassa oli 34 laite-, ohjelmisto- ja televiestintäyritystä. Google johtaa liittoumaa, ja antoi sen pohjaksi vuonna 2005 Android inc.iltä ostamansa Android-käyttöjärjestelmän. [17]

Android alustan jatkokehitys ja ylläpito vastuu on Android Open Source Projectilla (AOSP). Android-lähdekoodi on julkaistu vapaaseen jakeluun Apache Software 2.0 lisenssillä, joten se on julkisesti saatavilla. [18]

3.1.1 Android-ohjelmistoalusta

Android on ohjelmistopino (engl. software stack), joka koostuu Linux-ytimestä (engl. kernel) ja joukosta C/C++-ohjelmistokirjastoista. Linux-ydin tarjoaa järjestelmän

kriittisimmät toiminnot, kuten tietoturvan (engl. security), muistinhallinnan, prosessien hallinnan, verkkoliikennearkkitehtuurin ja ajurimallin. Ydin toimii abstraktiona raudan ja muun ohjelmistoarkkitehtuurin välillä. Ohjelmistokirjastoihin pääsyn tarjoaa Application Framework -kerros, joka tarjoaa palvelut ja rajapinnat varsinaiselle sovelluskerrokselle. Sovelluskerroksella toimivat Android-käyttöjärjestelmän mukana tulevat sovellukset, kuten selain ja puhelinmuistio, laitevalmistajan tarjoamat ohjelmat, sekä muut jälkikäteen laitteeseen asennettavat eri kanavista saatavat ohjelmat. [10]



Kuva 2. Android-arkkitehtuuri [10].

Android-runtime koostuu ydinkirjastoista (engl. core libraries) ja Dalvik-virtuaalikoneesta. Ydinkirjastot ovat suurilta osin samat kuin Java-ohjelmointikielessä, ja lisänä on joukko Androidin omia ydinkirjastoja. Dalvik tarjoaa Android-ohjelmille suoritusympäristön. Jokainen Android-ohjelma suoritetaan erillisessä prosessissa, jossa on oma ilmentymä (engl. instance) Dalvikista. Dalvik on suunniteltu niin, että Android-laite pystyy suorittamaan useaa virtuaalikonetta samanaikaisesti. [10]

Androidin ydin, ohjelmakirjastot ja virtuaalikone ovat toteutettu C- ja C++-kielillä.

Varsinaiset Android-ohjelmat on kirjoitettu Java-ohjelmointikielellä. [10]

3.1.2 Ohjelmointikieli ja Android-sovelluksen rakenne

Android-kehityksessä ohjelmointikielenä on Java. Javassa kaikkea kohdellaan kuin ne olisivat olioita, vaikka muutoksen kohteena oleva tunnistin (engl. identifier) onkin viittaus. Java peittää tämän eron kielen käyttäjältä. Tämä yksinkertaistaa ja yhdenmukaistaa kielen syntaksia. Javassa ohjelmoija varaa muistia, muttei koskaan vapauta sitä. Muistin hallinta on automatisoitu ja vapautuksesta vastaa roskien keruu (engl. garbage collection). [19]

Android-sovellus koostuu Android Manifest -tiedostosta, sovelluskomponenteista (engl. Application Components) ja sovelluksen resursseista. Nämä muodostavat APK-tiedoston (Android Application Package), joka voidaan asentaa Android-laitteeseen.

Manifest-tiedosto luetaan ennen sovelluksen varsinaista käynnistämistä, ja se sisältää oleellista tietoa sovelluksesta:

- Sovelluksen käyttöoikeudet, eli mitä asioita sovelluksella on oikeus tehdä omassa hiekkalaatikossaan. Esimerkiksi oikeus käyttää verkkoliikennettä tai pääsy käyttäjän yhteystietoihin.
- Sovelluksen käyttämä ohjelmistorajapinnan minimiversio, eli käytännössä mikä on vanhin Android-alustan versio jolla sovellus toimii.
- Sovelluksen käyttämät laiteominaisuudet, esimerkiksi kamera, GPS, Bluetooth.
- Tieto ohjelmistokirjasoista joita vasten sovellus tulee linkittää, esimerkiksi Google Maps -kirjasto.
- Tieto sovelluksen komponenteista, eli mitä luokkia sovellukseen kuuluu.

Sovelluskomponentit ovat osia joista sovellus muodostuu. Jokainen komponentti on mahdollinen sisäänpääsypiste (engl. entry point) Android-järjestelmälle sovellukseen. Komponentit voidaan jakaa neljään osaan.

- Aktiviteetit (engl. activities) esittävät yksittäistä näkymää sovelluksessa jolla on käyttöliittymä. Sovelluksessa voi olla yksi tai useampi aktiviteetti.
- Palvelut (engl. services) ovat komponentteja jotka suorittavat pidempikestoisia operaatioita taustalla. Niillä ei ole käyttöliittymää.

- Sisällön tarjoajat (engl. content providers) hallinnoivat jaettua sovellusdataa. Niitä voidaan käyttää sovelluksen datan tallentamiseen tai vastaavasti datan jakamiseen muiden käyttöön.
- Lähetyksen vastaanottajat (engl. broadcast receivers) ovat komponentteja, jotka kuuntelevat käyttöjärjestelmältä tulevia viestejä. Sovellus voi esimerkiksi reagoida näytön sammumiseen, kuvan ottamiseen tai akun varauksen laskemiseen näiden viestien avulla.

Sovellusresursseissa on sovelluksen lähdekoodien lisäksi tarvitsemat muut resurssit, kuten mahdolliset kuva- ja äänitiedostot, sekä kaikki sovelluksen käyttöliittymään muotoiluun liittyvät tiedostot. Yleisin tapa määritellä käyttöliittymä Android-sovellukseen on Extensible Markup Language (XML) layout -tiedosto, jossa näyttökomponenttien hierarkia ilmaistaan XML-tiedoston hierarkian avulla. XML-elementtien nimet vastaavat sen esittämää Java-luokkaa. Resurssit tarjoavat helpon tavan erottaa varsinainen toteutus käyttöliittymästä, ja mahdollistaa näin sovelluksen helpon muokattavuuden eri laitteistoille ja kielille. [20]

3.1.3 Android-kehitystyökalut

Android SDK starter -paketti ei sisällä kaikkia kehitykseen tarvittavia työkaluja. Se sisältää tärkeimmät kehitystyökalut, joiden avulla käyttäjä voi ladata loput haluamansa lisäpaketit.

Toisin kuin iOS ja Windows Phone 7 SDK:t Android SDK ei sisällä varsinaista kehitysympäristöä (IDE). Eclipse on Androidin virallinen IDE, ja siinä Android-kehityksen mahdollistava liitännäinen (engl. plugin) on nimeltään ADT (engl. Android Development Tools). ADT mahdollistaa kooditemplaattien ja emulaattorin käytön Eclipsessä.

Android-kehitystyökalujen mukana tulee Android SDK Manager, jonka avulla kehittäjä voi asentaa haluamansa ja tarvitsemansa lisäpaketit kehitysympäristöönsä. Android SDK on rakennettu modulaarisesti niin, että Android-alustan eri versiot, lisäosat (engl. add-ons), työkalut, esimerkit ja dokumentaatio on omia moduuleitaan, joten niiden asentaminen irrallisina kokonaisuuksina olisi mahdollisimman helppoa.

Android SDK Manager hakee oletusarvoisesti paketteja kahdesta pakettivarastosta (engl. repository): Android-pakettivarastosta ja kolmansien osapuolien lisäosista. Android-pakettivaraston paketit ovat luokiteltu seuraavasti:

- *SDK Tools*. Sisältää työkalut debuggaamiseen, sovelluksen testaamiseen ja muita utility-työkaluja. Nämä työkalut tulevat Android SDK starter -paketin mukana.
- *SDK Platform-tools*. Sisältää alustariippuvaiset työkalut sovellukset kehittämiseen ja debuggaukseen. Nämä päivittyvät yleensä vain uuden Android-version julkaisun yhteydessä.
- *Android platforms*. Jokaisesta julkaistusta Android-alustan versiosta on oma pakettinsa. Nämä paketit sisältävät täydellisen Android-kirjaston, systeemi-imagen, esimerkkikoodia ja emulaattorin skinit (engl. skin)
- *USB Driver for Windows*. Tämä paketti mahdollistaa oikealla Android-laitteella testaamisen Windows-käyttöjärjestelmissä. Linux ja Mac OS X eivät tarvitse erillispakettia.
- *Samples*. Sisältää esimerkki koodeja ja sovelluksia eri Android-versioille.
- *Documentation*. Sisältää Android-ohjelmistokehyksen rajapintojen dokumentaation. [20]

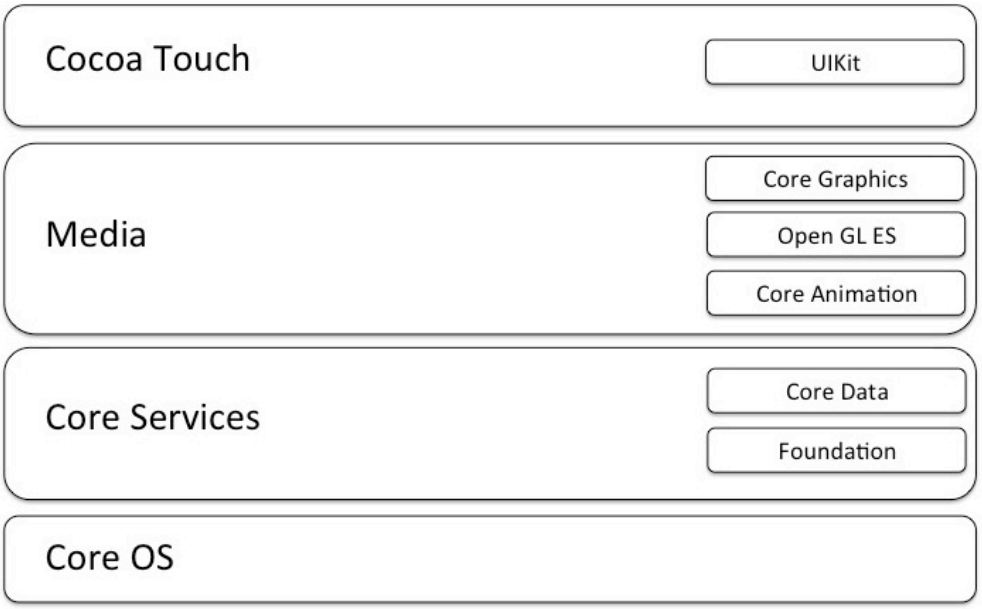
Android-emulaattori toimii kehitysympäristössä Android-virtuaalilaitteen (engl. Android Virtual Device), eli AVD:n sisällä. Android-työkalut mahdollistavat usean, toisistaan erilaisen AVD:n luonnin, mikä helpottaa sovelluksen testausta. Kehittäjä voi näin emuloida sovelluksensa toimintaa usealla eri resoluutiolla ja muistimäärällä. [20]

3.2 iOS

iOS on Applen kehittämä ja ylläpitämä käyttöjärjestelmä mobiililaitteille. Se julkaistiin vuonna 2007 alunperin nimellä iPhone OS. Apple ei lisensoi käyttöjärjestelmäänsä muille, vaan valmistaa kaikki iOS-käyttöjärjestelmää käyttävät laitteet itse. [21]

3.2.1 iOS-ohjelmistoalusta

iOS-arkkitehtuuri koostuu neljästä kerroksesta, ja on arkkitehtuuriltaan hyvin samankaltainen Applen tietokoneiden käyttämän OS X -käyttöjärjestelmän kanssa. Kuva 3 esittelee iOS-arkkitehtuurin.



Kuva 3. iOS-arkkitehtuuri [21].

Core OS -kerros on arkkitehtuurin alimpana, ja sen perustana toimii OS X -ydin Mach 3.0. Mach 3.0:n perustana on puolestaan ollut Darwin, joten iOSin voidaan sanoa olevan UNIX-käyttöjärjestelmä. Core OS -kerros koostuu C-kielisestä koodista, ja vastaa muun muassa muistinhallinnasta, levyoperaatioista ja verkkoliikenteestä.

Core Services -kerros tarjoaa paljon samoja toimintoja kuin Core OS -kerros, mutta nostaa abstraktiotasoa, ja mahdollistaa olio-ohjelmointimallisen toteutuksen rajapinnan käyttäjälle.

Media-kerros tarjoaa nimensä mukaisesti multimediapalvelut, kuten kuvien, videoiden ja musiikin avaamisen ja toistamisen. Ylimpänä oleva Coco Touch –kerros tarjoaa palvelut ja rajapinnat vuorovaikutukseen käyttäjän kanssa. [21]

3.2.2 Ohjelmointikieli ja iOS-sovelluksen rakenne

iOS-kehitys tehdään Objective-C-kielellä. Kochan kuvailee Objective-C -kielen historiaa kirjassaan Programming in Objective-C. [22]

Brad J Cox kehitti Objective-Cn 1980-luvun alussa, ja sen perustana oli SmallTalk-80 ohjelmointikieli. Objective-C tehtiin kerroksena C-kielen päälle, jolla mahdollistettiin olio-

ohjelmointi. Vuonna 1988 ohjelmistoyritys NeXT Software lisensoi Objective-C-kielen ja kehitti sen ohjelmistokirjastoja sekä kehitysympäristön nimeltään NEXTSTEP.

Vuonna 1994 NeXT Computers ja Sun Microsystems julkaisivat standardoidun määrittelyn NEXTSTEP-ympäristöstä nimeltään OPENSTEP, jota puolestaan käytettiin ilmaisten jakeluiden GNUStep ja LinuxSTEP perustana.

Joulukuun 20. päivä 1996 Apple ilmoitti NeXT Softwaren hankinnasta itselleen, ja aikomuksestaan käyttää NEXTSTEP ja OPENSTEP ympäristöjä sen seuraavan käyttöjärjestelmän, OS Xn perustana. [22]

Tästä lähtien Apple on käyttänyt Objective-Ctä kehitysympäristöjensä suositeltuna kielenä, jolloin se oli luonnollinen valinta myös iOS-kehitykseen.

iOS-sovellus rakentuu niin kutsutusta bundlesta (engl. bundle). Bundle on kansio levyjärjestelmässä, joka ryhmittelee sovelluksen kaikki resurssit samaan paikkaan. Sovellus-bundle muodostuu ajotiedostosta (engl. app executable),
Info.plist-tiedostosta (engl. the information property list file), sovelluksen kuvakkeista, käyttöliittymätiedostosta tai -tiedostoista, mahdollisesta erillisestä asetus-bundlesta ja resurssitiedostoista.

- *Ajotiedosto*. Sisältää sovelluksen käännetyn lähdekoodin
- *Info.plist-tiedosto*. Sisältää tiedot sovelluksen konfiguraatiosta, sovelluksen versionumeron ja nimen, tiedon sovelluksen oikeuksista ja tiedon sovelluksen vaatimuksista kohdelaitteelta, kuten käyttöjärjestelmäversio. Yleisesti käyttöjärjestelmä tietää info.plist-tiedoston perusteella miten, ja minkälaisessa vuorovaikutuksessa se voi sovelluksen kanssa olla.
- *Sovelluksen kuvakkeet*. Kuvakkeisiin kuuluvat sovelluksen ikoni, josta sovellus voidaan käynnistää, sekä käynnistyskuva joka näytetään sovelluksen taustakuvana ennen kuin se varsinaisesti käynnistyy (engl. launch image/splash screen).
- *Käyttöliittymätiedosto*. Tiedosto voi olla kahdessa eri muodossa. iOS 5 -versiossa tullut yksittäinen storyboard-tiedosto, joka koostaa kaikki sovelluksen näkymät

yhteen tiedostoon, tai vaihtoehtoisesti kokoelma nib-tiedostoja, joista jokainen esittää sovelluksen yhtä näyttöä.

- *Asetus-bundle (engl. settings bundle).* Tämä ei ole pakollinen tiedosto, mutta se tarvitaan mikäli sovellus haluaa mahdollistaa omien asetusten muokkaamisen käyttöjärjestelmän yleisten asetusten kautta.
- *Resurssitiedostot.* Tiedostot voidaan jakaa kahteen osaan: Yleiset ja lokalisoitavat resurssit. Yleisissä resursseissa sijaitsevat sovelluksen käyttämät kuva, ääni, video ja muut vastaavat tiedostot. Lokalisoitavissa resursseissa sijaitsevat aluekohtaisesti muutettavat tiedostot, kuten käyttöliittymällä esitettävät tekstit. [23]

3.2.3 iOS-kehitystyökalut

Applen nimeämiskäytäntö kehitystyökalujen suhteen eroaa Microsoftin ja Googlen vastaavista. iOS-kehitystyökalut jaellaan Xcoden mukana, joka on iOS-kehityksessä käytetty IDE. Applen käsitteillä Xcode on siis kehittäjien täydellinen työkalukokoelma, joka sisältää Xcode IDEn, työkalut, iOS-simulaattorin ja iOS SDK:n (vaadittavat ohjelmakirjastot). Vastaavasti Microsoft ja Google käyttävät yleisesti termiä SDK, jonka yhtenä osana on IDE.

iOS-kehitystyökalut voidaan jakaa kahteen osaan. Xcode IDE on ensimmäinen osa ja varsinainen kehitysympäristö, jolla iOS-kehitys tehdään. Lähdekooditiedostojen muokkauksen tukeminen, kuten ennakoiva syöttö, täydennys, korjausehdotukset, refaktorointi sekä kääntäminen ovat kaikki yleisiä piirteitä kehitysympäristöissä. Xcode IDEn osana on myös iOS-simulaattorin käyttö, integrointi versionhallintaan, sekä Interface Builder, jonka avulla iOS-sovellusten käyttöliittymä rakennetaan. Toinen osa on sovellusten analysointityökalu Instruments, jonka avulla sovelluksen muistin ja prosessorin kuormitusta voidaan analysoida. Sen avulla voidaan myös simuloida erilaisia tilanteita, kuten muistin loppumista laitteesta, joiden testaaminen ja toistaminen normaalisti on hankalaa. [24]

3.3 Windows Phone 7

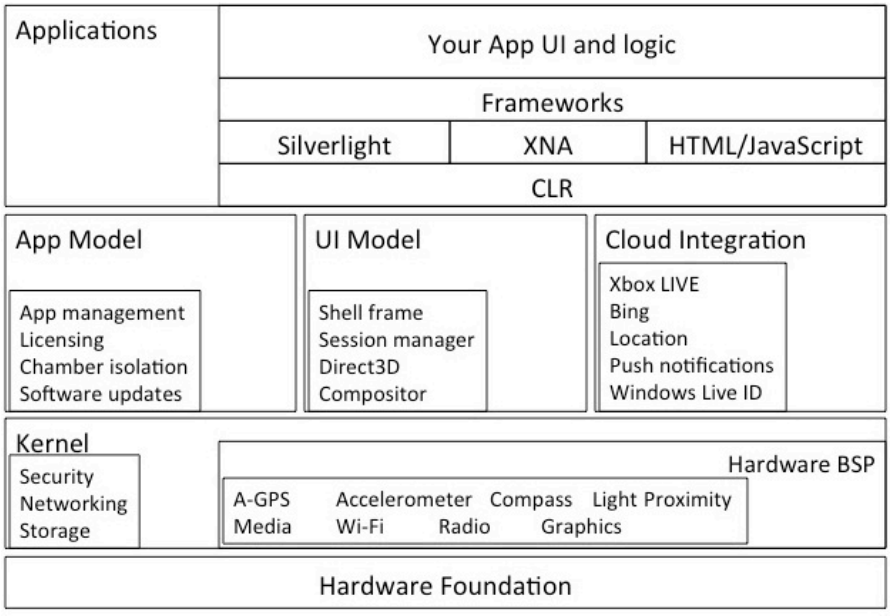
Windows Phone on Microsoftin kehittämä mobiilikäyttöjärjestelmä. Microsoft julkaisi käyttöjärjestelmän vuonna 2010 jälkimmäisellä vuosipuoliskolla. Windows Phone on

Microsoftin edellisen mobiilikäyttöjärjestelmän Windows Mobilen itsenäinen seuraaja. Minkäänlaista yhteensopivuutta näiden kahden käyttöjärjestelmän välillä ei ole. [25]

Windows Phone alustalla on kaksi ohjelmistokehystä sovellusten kehittämiseen. Silverlight-ohjelmistokehys XAML-pohjaisten (Extensible Application Markup Language) sovellusten tekoon ja XNA-ohjelmistokehys pelien ja muuten graafisesti vaativien sovellusten tekemiseen. Vuonna 2010 julkaistussa ensimmäisessä Windows Phone 7 -versiossa nämä kaksi ohjelmistokehystä oli eroteltu täysin tosistaan, mutta vuonna 2011 julkaistu versio 7.5 mahdollistaa molempien ohjelmistokehysten käytön samassa sovelluksessa. [26]

3.3.1 Windows Phone 7 -ohjelmistoalusta

Alustan arkkitehtuuri on kuvattu kuvassa 4. C-kielellä toteutettu ydin tarjoaa perustoiminnallisuuden kuten tietoturvan, tietoliikenteen ja tiedostojenhallinnan. Tämän päälle on kuvattu sovellus- ja näyttömallit sekä pilvi-integraatiomalli. Ylimpänä on varsinainen sovelluskerros, jolla sovellukset toimivat. Tällä kerroksella toimivat Silverlight- ja XNA-sovelluskehykset, joita käyttäen sovellukset tehdään. CLR, eli Common Language Runtime on Microsoftin .NET-tuoteperheen virtuaalikone, joka tarjoaa sovelluksille ajoympäristön.



Kuva 4. Windows Phone arkkitehtuuri [27].

3.3.2 Ohjelmointikieli ja Windows Phone 7 -sovelluksen rakenne

Windows Phone 7 -kehityksessä käytettävä ohjelmointikieli on C#. Silverlight-ohjelmistokehystä käyttävien sovellusten käyttöliittymä määritellään XAML-kielellä.

Microsoft esitteli C#-ohjelmointikielen vuonna 2001. Se perustuu C-kieleen, ja on syntaksiltaan hyvin samankaltainen C- ja C++-kielien kanssa. Toisin kuin C- ja C++-ohjelmia, C#-ohjelmia ei käännetä tietylle prosessorityypille, jossa ohjelma suoritetaan. C#-koodi käännetään .NET-ympäristöön, jolloin se on suoritettavissa kaikissa järjestelmissä, jotka tarjoavat .NET-ajoympäristön. Tämä vähentää merkittävästi kielen ympäristöriippuvuutta. Javan tapaan C# tarjoaa automaattisen muistinhallinnan, jossa kehittäjän ei tarvitse itse vapauttaa varaamaansa muistia. [28]

XAML on Microsoftin kehittämä käyttöliittymienmäärittelykieli. Sen perustana on ollut XML ja sitä käytetään laajalti .NET-ohjelmistokehykseen perustuvissa teknologioissa. XAML määrittelee käyttöliittymän komponentit hierarkkisesti. Juurielementti määrittelee pääelementin ja sen sisälle voidaan määritellä komponentille ominaisuuksia (engl. properties), tai kokonaan uusia elementtejä. [29]

Kuvassa 5 on määritelty näytöllä piirrettävä nappula XAML-kielellä. Nappulan taustaväri on sininen ja nappulan teksti punainen. Nappulassa lukee teksti ”Click Me”.

```
<Button>
  <Button.Background>
    <SolidColorBrush Color="Blue"/>
  </Button.Background>
  <Button.Foreground>
    <SolidColorBrush Color="Red"/>
  </Button.Foreground>
  <Button.Content>
    Click Me
  </Button.Content>
</Button>
```

Kuva 5. XAML-kielellä määritelty nappula.

Silverlight on Microsoftin ohjelmistoalusta- ja selainriippumaton .NET-ohjelmistokehyksen toteutus, joka kehitettiin alunperin tarjoamaan monipuolisia

multimediaominaisuuksia verkkoon. Myöhemmin se laajeni ja toi mahdollisuuden kehittää itsenäisiä työpöytäsovelluksia. Silverlight-ohjelmistokehys otettiin pohjaksi myös Windows Phone -sovelluksille. [30]

Windows Phone 7 -sovelluksen rakenne riippuu siitä, onko kyseessä Silverlight- vai XNA-muotoinen projekti. Tässä käsitelty esimerkki on rakennettu Silverlight-ohjelmistokehyksellä. Visual Studio -kehitystyökaluilla luotu Silverlight-projekti koostuu lähdekooditiedostoista, käyttöliittymätiedostoista, erilaisista kuvakkeista, sekä erilaisista ominaisuus (engl. properties) ja viittaus (engl. references) kansioista. Nämä tiedosto pakataan XAP-tiedostoon, joka voidaan asentaa puhelimeen tai jakaa eteenpäin. Alapuolella on kuvattu Visual Studio -kehitystyökalulla luodun projektin sisältämät tiedostot ja niiden roolit:

- *App.xaml- ja App.xaml.cs-tiedostot*. Nämä tiedostot luetaan sovelluksen käynnistyessä. Ne alustavat sovelluksen resurssit ja näyttävät sovelluksen käyttöliittymän. Tiedostot muodostavat parin, xaml-päätteinen tiedosto määrittelee sovelluksen käyttöliittymän ja cs-tiedosto toimintalogiikan. Tiedostopäätteet viittaavat XAML- ja C#-kieliin.
- *MainPage.xaml ja MainPage.xaml.cs*. Tiedostopari määrittelee sovelluksen päänäkymän ja sen toimintalogiikan.
- *ApplicationIcon.png*. Sovelluksen kuvake, joka näytetään puhelimen sovelluslistalla.
- *Background.png*. Määrittelee sovelluksen kuvakkeen, joka näytetään puhelimen aloitusnäytöllä.
- *SplashScreenImage.jpg*. Tiedosto joka näytetään sovelluksen käynnistyessä, varsinaisten sovellusresurssien latautuessa. Tämä kuva on usein sovelluksen ensimmäisen näkymän kaltainen, jolla käyttäjälle luodaan vaikutelma sovelluksen välittömästä käynnistymisestä.
- *AppManifest.xml*. Tiedosto sisältää sovelluksen asennettavan XAP-tiedoston muodostamisen vaatimia tietoja.
- *AssemblyInfo.cs*. Tiedosto sisältää sovelluksen nimi ja versiotiedot, jotka sisällytetään sovelluksen käännökseen.

- *WMAppManifest.xml.* Tiedosto sisältää Windows Phone Silverlight-sovelluskohtaista metatietoa.
- *References-kansio.* Kansio sisältää viittaukset kirjastoihin joita sovellus käyttää. [31]

3.3.3 Windows Phone 7 -työkalut

Windows Phone SDK -kehitystyökalut sisältävät Visual Studio -kehitysympäristön, joka sisältää tarvittavat komponentit Windows Phone -kehitykseen, emulaattorin sovelluksen testaukseen ja debuggaamiseen, sekä työkaluja sovelluksen suorituskyvyn ja muistin käytön analysointiin.

Keskeisimmät osat ovat:

- Visual Studio 2010 Express for Windows Phone, joka on Microsoftin Visual Studio -tuoteperheen Windows Phone -kehitykseen tarkoitettu IDE.
- Windows Phone -emulaattori, jonka avulla oikeaa puhelinta voidaan emuloida sovellusten testauksessa. Emulaattorissa on myös työkalut, joilla voidaan emuloida puhelimen kallistelua, useamman pisteen kosketusta, GPS-sijaintia ja muita rautariippuvaisia ominaisuuksia
- Windows Phone Profiler, jonka avulla sovelluksen suorituskykyä voidaan analysoida muun muassa muistin ja prosessorin kuormituksen suhteen.
- Silverlight 4 SDK ja DRT (Direct Run-time)
- Windows Phone SDK 7.1 Assemblies ja tarvittavat lisäosat (engl. extensions) XNA Game Studiolle
- Microsoft Expression Blend -käyttöliittymien suunnittelutyökalu, joka mahdollistaa Silverlight-sovellusten käyttöliittymien tekemisen Visual Studio IDEn ulkopuolella.
- Microsoft Advertising SDK tarjoaa mahdollisuuden kehittäjälle sisällyttää mainostilaa sovellukseensa. [32]

3.4 Erot alustojen välillä

Merkittävimmät erot ohjelmistoalustojen välillä on kartoitettu taulukossa 1. Android-

kehitys ei ole käyttöjärjestelmäriippuvainen, toisin kuin iOS- ja Windows Phone 7 -kehitys. Ohjelmointikielenä Android-alustalla on Java, iOS-alustalla Objective-C ja Windows Phone 7 alustalla C#.

Sovelluksen käyttöliittymä määritellään Androidilla XML-elementein ja Windows Phone 7 XAML-kielen avulla. iOS-sovelluksen käyttöliittymän määrittelylle ei ole erillistä kieltä.

Android- ja Windows Phone 7 -alustat tarjoavat kehityksen tueksi emulaattorit, joilla sovellusta voidaan testata. iOS-kehitystyökalut eivät tarjoa emulaattoria, vaan simulaattorin. Emulaattoriympäristössä yritetään jäljitellä varsinaisen laitteen rautaominaisuuksia (engl. hardware), kuten prosessorin nopeutta ja muistimäärää ohjelmistoympäristön (engl. software) lisäksi, simulaattorin mallintaessa pelkän ohjelmistoympäristön.

Ohjelmistoalustojen kielet ovat olio-ohjelmointikieliä. Syntaksierojen lisäksi merkittävin ero on muistinhallinta, joka on Androidin Java ja Windows Phone 7:n C#-kielissä automatisoitu. iOS-kehittäjän tulee hallita muisti itse. Merkittävä muutos tähän tosin tuli vuonna 2011 julkaistun iOS5-version mukana. Apple esitteli ARC:n (Automatic Reference Counting) helpottamaan muistinhallintaa iOS-kehityksessä. Uudessa mallissa kehittäjä määrittelee varattavaan muistiin osoittavan osoittimen (engl. pointer), joka vahvaksi (engl. strong) tai heikoksi (engl. weak). Vahvasti osoitettu muisti pidetään varattuna kunnes siihen ei enää ole osoituksia. Heikon osoittimen varaama muisti pysyy varattuna niin kauan, kuin siihen osoittaa heikon osoittimen lisäksi jokin vahvaksi määritelty osoitin.

Kaikkien kolmen alustan sovelluksia suositellaan vahvasti testattavaksi varsinaisella laitteella ennen sovelluksen julkaisua. Emulaattoriympäristö mallintaa laiteympäristöä, mutta se ei kuitenkaan usein vastaa lopullisen laitteen suorituskykyä. Raudalla testaaminen onnistuu Androidilla ilman kehityslisenssiä. iOS- ja Windows Phone 7 -laitteet vaativat laitteen rekisteröimisen kehityskäyttöön, joka puolestaan vaatii kehityslisenssin. Vasta rekisteröinnin jälkeen sovelluksen asentaminen laitteelle on mahdollista.

Android- ja iOS-alustat ovat julkaistu vuonna 2007 ja niiden SDKt olivat saatavilla vuonna

2008. Windows Phone 7 -alusta on kilpailijoitaan kolme vuotta nuorempi, sillä se tuli markkinoille vuonna 2010.

Taulukko 1. Erot alustojen välillä.

	Android	iOS	Windows Phone 7
Käyttöjärjestelmä (työkaluille)	Windows, Linux, OS X	OS X	Windows
Ohjelmointikieli	Java	Objective-C	C#
käyttöliittymämäärittely	XML	-	XAML
Simulator/emulator	emulator	simulator	emulator
Muistinhallinta	automatisoitu	man., iOS 5ARC	automatisoitu
Raudalla testaus	ilmaista	vaatii lisenssin	vaatii lisenssin
Kehityslisenssin hinta	25 $	99 $/vuosi	99 $/vuosi
Julkaisuvuosi	2007	2007	2010

4 KEHITYS- JA JULKAISUPROSESSIT

Lähdemateriaalia ja ohjeita kehittämisen aloittamiseen on saatavilla hyvin. Niin Apple, Google kuin Microsoft tarjoavat ohjelmistokirjastodokumentaation lisäksi opetusvideoita, artikkeleja ja kattavan määrän erinäisiä tutoriaaleja.

Applen iOS Dev Center (https://developer.apple.com/devcenter/ios) toimi keskitettynä aloitus- ja kokoomapisteenä kaikelle iOS-kehitykseen liittyvälle. Sieltä on ladattavissa iOS-kehitystyökalut, sekä luettavissa ohjeita ja artikkeleita. iOS Dev Center tarjoaa myös keskustelupalstat, joissa rekisteröityneet käyttäjät voivat keskustella tai pyytää apua.

Android Developers –sivusto (http://developer.android.com) tarjoaa resurssit Android-kehitykseen. App Hub (http://create.msdn.com) on Microsoftin sivusto Windows Phone 7-kehitykseen. Kaikki kolme sivustoa ovat sisällöiltään hyvin samankaltaisia, ja ne tarjoavat toisiaan vastaavat resurssit sovellusten kehitykseen.

Apple ja Microsoft verifioivat sovellukset ennen niiden julkaisua App Storeen ja Windows Phone Marketplaceen. Joten kaikki niissä jaeltavat sovellukset ovat yksitellen hyväksytty julkaistaviksi jakelukanaviin. Googlen Play ei toteuta vastaavaa hyväksymisprosessia, vaan kehittäjien julkaisemat sovellukset julkaistaan suoraa käyttäjien ladattaviksi. Googlen Play kuitenkin suorittaa jälkiseurantaa ja poistaa jakelukanavasta sovellukset jotka todetaan käyttäjäehtojen vastaisiksi.

Seuraavassa tarkastellaan kunkin kanavan julkaisuprosessia, sekä sen asettamia vaatimuksia ja prosessin yleistä etenemistä.

4.1 Google Play

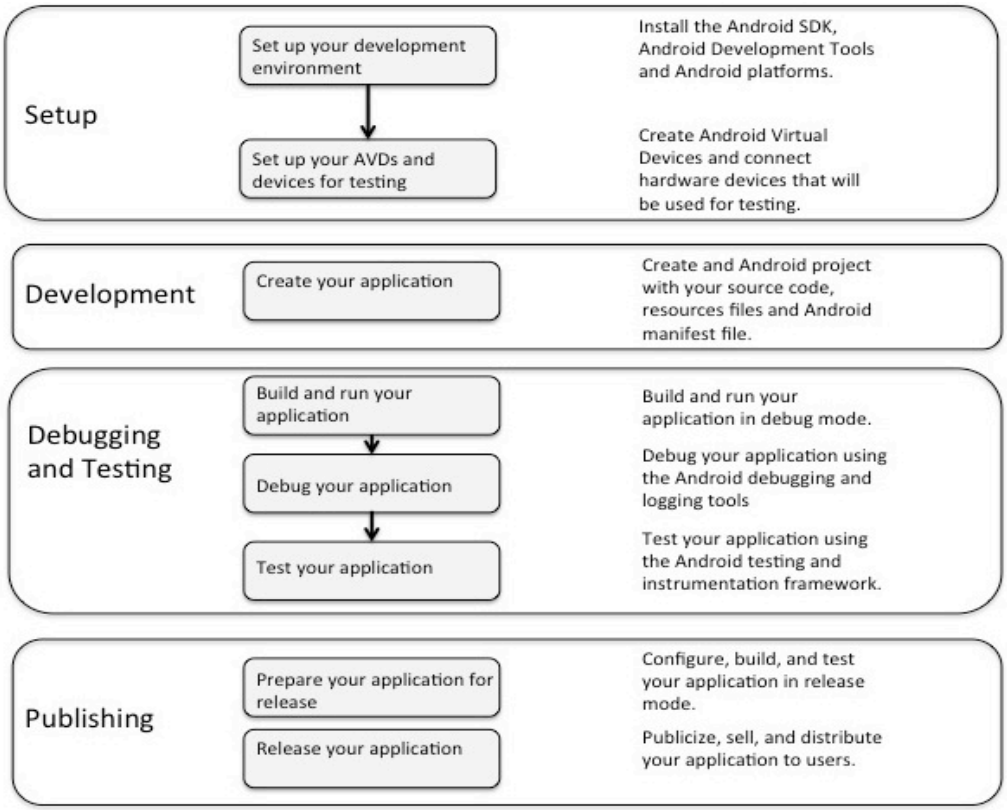
Android-sovelluksen kehitys- ja julkaisuprosessia on kuvailtu Android Dev Guide-dokumentaatiossa kuvan 6 mukaisesti. Prosessi on jaettu neljään päävaiheeseen. Vaiheet ovat asennus ja valmistelu (engl. setup), kehitys (engl. development), debuggaus ja testaus (engl. debugging and testing), sekä julkaisu (engl. publishing).

Ensimmäinen vaihe on Setup, joka sisältää kehitysympäristön asennuksen ja käyttöönoton. Tässä vaiheessa luodaan myös virtuaaliset Android-laitteet (AVD), joilla kehitettävää sovellusta voidaan testata emulaattorissa, sekä yhdistetään mahdolliset varsinaiset Android-laitteet kehitystyökaluihin.

Toinen vaihe on Development, jossa luodaan Android-projekti, lähdekoodit, muut sovelluksen vaatimat resurssit sekä Android manifest -tiedosto.

Kolmas vaihe, Debugging and Testing, muodostuu sovelluksen kääntämisestä ja testauksesta emulaattorilla tai Android-laitteella. Projektista käännetään debugattava apk-paketti, jota voidaan ajaa emulaattorissa tai laitteella. Sovelluksen toiminnallisuus ja käyttäytyminen varmistetaan eri tilanteissa, sekä mahdollisia ongelmatilanteita selvitetään debug-työkalujen avulla.

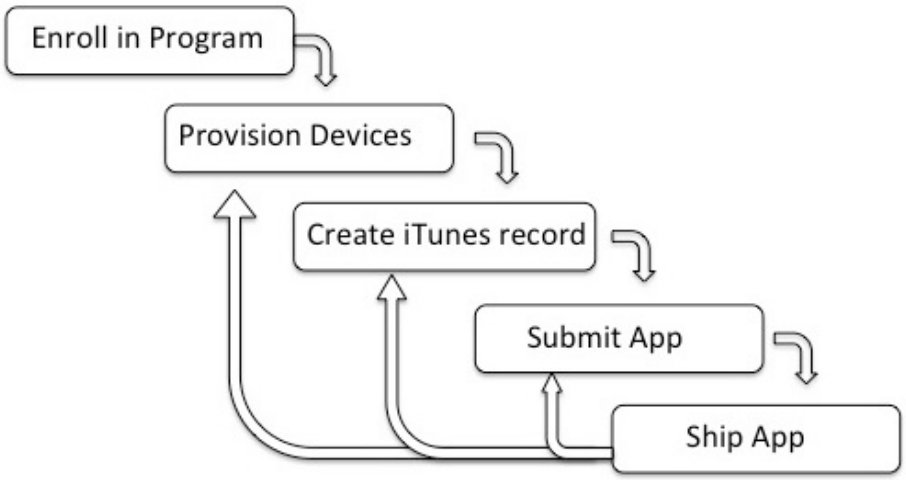
Viimeinen vaihe on Publishing. Tässä vaiheessa sovellus konfiguroidaan ja käännetään julkaisua ja jakelua varten. Tämän vaiheen jälkeen sovellus on käyttäjien saatavilla. [33]



Kuva 6. Android kehitys- ja julkaisuprosessi [33].

4.2 App Store

Sovelluksen toteutuksen ja testauksen valmistuttua, lähetetään sovellus ja siltä vaadittavat lisätiedot (engl. metadata) Applen arviointiin. Mikäli sovellus ja sen lisätiedot täyttävät niille asetetut ehdot, voidaan sovellus julkaista App Storeen. iOS Developer Library kuvaa sovelluksen kehitys ja julkaisuprosessia kuvan 7 mukaisesti.



Kuva 7. App Store -julkaisuprosessin kuvaus [34].

Ensimmäisessä vaiheessa (Enroll in Program) kehittäjä liittyy Applen iOS-kehittäjäohjelmaan (engl. iOS Developer Program). Tämän jälkeen kaikki Applen tarjoama iOS-kehitykseen liittyvä materiaali ja työkalut ovat käytettävissä. Tässä vaiheessa kehittäjä antaa itsestään tietoa Applelle, kuten yhteystiedot ja allekirjoittaa sopimuksen, jossa sitoudutaan toimimaan sovellusten kehityksessä ja julkaisussa sopimuksen määrittelemin ehdoin. Rekisteröitymisen yhteydessä määritellään myös toimiiko kehittäjä itsenäisenä kehittäjänä vai yrityksenä. Tämän vaiheen jälkeen kehittäjä voi aloittaa sovellusten kehittämisen.

Toinen vaihe (Provision Devices) mahdollistaa sovellusten testaamisen varsinaisilla iOS-laitteilla. Apple suosittelee sovellusten testaamista varsinaisella iOS-laitteella, vaikkei se varsinaisesti ole vaatimus sovellusten julkaisulle. Laitteella testaaminen vaatii laitteen rekisteröimistä kehityskäyttöön. Tämä tapahtuu liittämällä laite tietokoneeseen, ja lataamalla kehittäjän lisenssiin sidotun provisointi-profiilin laitteeseen. Tämän jälkeen kehittäjä voi ladata kehityssertifikaatin, jolla testattava sovellus tulee allekirjoittaa (engl.

signing), ennen kuin se on siirrettävissä ja suoritettavissa laitteella.

Kolmannessa vaiheessa (Create iTunes Record) kehitettävästä sovelluksesta tehdään tietue iTunes Connectiin, joka sisältää lisätietoa sovelluksesta. Nämä tiedot näytetään sovelluksesta App Storessa kun se on julkaistu. Näytettäviin tietoihin kuuluu muun muassa sovelluksen nimi, kuvaus sovelluksesta, sovelluksen kuvake, ruutukauppauksia sovelluksesta sekä sen kehittäjän yhteystiedot.

Neljännessä vaiheessa (Submit App) sovellus lähetetään Applelle hyväksyttäväksi. Kolmannen vaiheen tietojen tulee olla täytettynä, ja lähetettävä sovellus on allekirjoitettava erillisellä jakeluun tarkoitetulla sertifikaatilla. Sovellus lähetetään pakattuna archieve-tiedostona, jonka tekeminen onnistuu Xcode-kehitysympäristöllä. Tämän jälkeen Apple arvioi sovelluksen sen määrittelemien vaatimusten mukaan, jotka ovat kuvattu App Store Review Guideline dokumentissa [35].

Viimeisessä vaiheessa (Ship App) sovellus julkaistaan App Storeen, mikäli se on läpäissyt julkaisulle asetetut kriteerit. Kehittäjä voi valita sovelluksen automaattisesti julkaistavaksi arvioinnin jälkeen, tai halutessaan määritellä julkaisulle tietyn päivän. [34]

4.3 Windows Phone Marketplace

Windows Phone Marketplacen prosessi kehityksen aloituksesta sovelluksen julkaisuun on hyvin samankaltainen Applen vastaavan kanssa. Microsoft kuvailee MSDN (Microsoft Developer Network) -dokumentaatiossaan julkaisu- ja kehitysprosessin koostuvan seruaavista vaiheesta: Kehittäjäksi rekisteröityminen, sovelluksen kehitys ja testaus, sertifioinnin esivaatimusten kerääminen, sovelluksen arvioitavaksi jättö ja sertifiointi, Windodws Phone Markeplace -linkkien luonti sekä sovelluksen päivittäminen ja sovelluksen tuki.

Ensimmäinen vaihe on rekisteröityä Microsoftin App Hub -sivustolla Windows Phone -kehittäjäksi. Samalta sivustolta tehdään myös sovellusten julkaisu ja kaikki sovellukseen liittyvien tietojenpäivitys. Rekisteröityminen mahdollistaa myös Windows Phone 7 -laitteiden ”avaamisen” kehityskäyttöön, joka mahdollistaa omien sovellusten asentamisen

niihin.

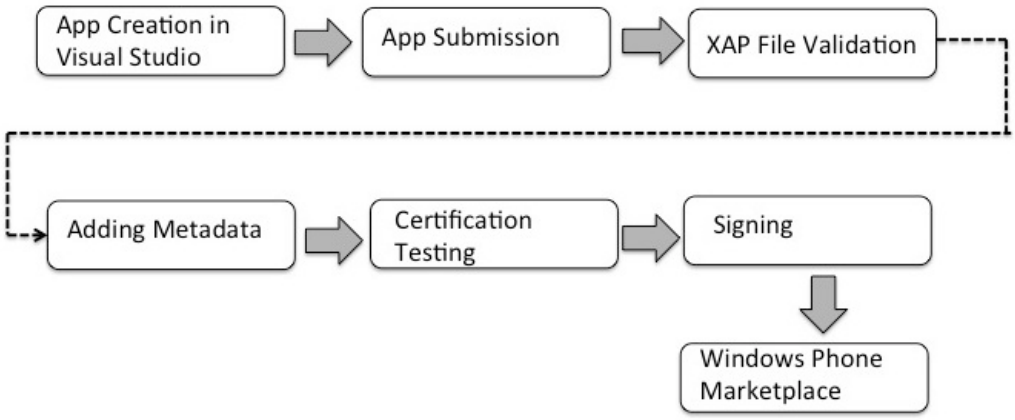
Toinen vaihe muodostuu kehitettävän sovelluksen tekemisestä ja testaamisesta. Kehitystyö tehdään Visual Studio -työkalulla, joka mahdollistaa myös sovelluksen testaamisen emulaattorilla ja varsinaisella laitteella. Visual Studio tekee kehitettävästä projektista XAP-tiedoston, joka voidaan asentaa Windows Phone 7 –laitteeseen, joka on rekisteröity kehityskäyttöön. Testauksen helpottamiseksi Microsoft tarjoaa Marketplace Preparation Test Kit nimisen työkalun, joka sisältää automatisoituja sekä manuaalisia-testejä, joita kehittäjä voi ajaa. Nämä testit varmistavat, että sovellus ja sen resurssit täyttävät esiehdot julkaisulle.

Kolmas vaihe prosessissa on sertifioinnin esivaatimusten kerääminen, jossa käydään tarkistuslista läpi, ja laaditaan oikean kokoiset kuvakkeet sovelluksen ikonille, Market Place -kuvaukseen ja kerätään muut tarvittavat resurssit. Varsinaiselle sovellukselle ei tarvitse tässä vaiheessa tehdä mitään, sillä Visual Studio tekee jokaisen käännöksen yhteydessä .XAP-tiedoston, joka sovelluksesta jätetään arvioitavaksi.

Seuraavassa vaiheessa sovellus lähetetään Microsoftille hyväksyttäväksi. Tätä ennen tulee tarkistaa, että sovellus täyttää sisällöllisesti Microsoftin määrittelemät vaatimukset julkaistaville sovelluksille. Nämä vaatimukset ovat julkaistu MSDN-dokumentaation dokumentissa Application Certification Requirements for Windows Phone. MSDN-dokumentaatio kuvailee myös suunnitteluperiaatteita, joiden mukaan Windows Phone 7 –sovelluksen käyttöliittymä suositellaan tehtäväksi. [36]

Tämän jälkeen sovellus on valmis lähetettäväksi Microsoftin arviointiin App Hub -sivuston kautta. Sovelluksen tiedot kuten nimi ja sen kuvaus kirjataan sovelluksen tietoihin, ja lopuksi Visual Studion tekemä XAP-tiedosto ladataan sivustolle.

Julkaisun jälkeen on mahdollista tehdä Windows Phone Marketplace -linkkejä, jotka avaavat Marketplacen suoraan julkaistun sovelluksen sivulle. Näitä linkkejä kehittäjä voi käyttää sovelluksen jakelun ja mainostamisen helpottamiseen. Viimeiset vaiheet muodostuvat sovelluksen tuesta ja mahdollisista päivityksistä. [36]



Kuva 8. Windows Phone 7 -kehitys- ja julkaisuprosessi [36].

4.4 Erot kehitys ja julkaisuprosessien kuvauksissa

Prosessit ovat kaikilla kolmella alustalla hyvin samankaltaiset. Prosessien kuvauksissa on painotettu hieman eri asioita, mutta lähes samat vaiheet löytyvät niistä kaikista. Kolme eroavaisuutta julkaisuprosesseissa voidaan kuitenkin havaita, jotka ovat kuvattu seuraavassa.

Ensimmäisenä eroavaisuutena Android-julkaisuprosessinkuvaus ei ota kantaa jakelukanavaan. Google Play mainitaan yhtenä mahdollisuutena, mutta sovelluksen jakelureitti on sovelluksen kehittäjän päätettävissä. App Storen ja Windows Market Placen ollessa kehitysalustansa ainoat mahdolliset jakelukanavat, ovat ne kiinnitetty luonnollisesti myös julkaisuprosessin kuvauksiin tiiviimmin.

Toinen merkittävä ero on niin kutsuttu arviointimenettely, jossa jakelukanavaa hallitseva taho arvioi ja hyväksyy sovellukset ennen niiden julkaisua kyseisen alustan jakelukanavassa. Apple ja Microsoft arvioivat kaikki App Storeen ja Windows Marketplaceen julkaistavat sovelluksen ennen kuin ne julkaistaan kyseiseen kanavaan, ja ovat käyttäjien saatavilla. Google ei vastaavaa arviointia tee, vaan sovelluksen julkaistaan Google Playssä suoraan kehittäjän toimesta. Sovelluksille tehdään osittain automatisoitua jälkiseurantaa ja Google poistaa sovelluksia, joissa kehittäjien julkaisusopimuksen tekemisvaiheessa hyväksymiä käyttäjäehtoja rikotaan.

Kolmas ero liittyy sovellusten allekirjoittamiseen (engl signing), jossa kyseinen sovellus

allekirjoitetaan digitaalisella sertifikaatilla. Tällä allekirjoituksella sovelluksen tekijä pystytään aina tunnistamaan. Android- ja iOS-alustoilla tämä allekirjoittamisen suorittaa itse kehittäjä, kun taas Windows Phone 7 -alustalla allekirjoituksen hoitaa Microsoft, siinä vaiheessa kun sovellus on läpäissyt arvioinnin ja on muuten valmis julkaistavaksi Windows Phone Marketplaceen.

5 SOVELLUSTEN JA TIETOJEN HALLINTA

Jakelukanavat tarjoavat sovellusten hallintasivustot, joiden avulla sovelluksia sekä niihin liittyviä tietoja tarkastellaan ja hallitaan. App Store –julkaisukanavan tietoja hallitaan iTunes Connect –sivustolta, Google Play –kanavassa hallintasivuston nimi on Android Developer Console ja Windows Phone Marketplace –kanavan sivusto on nimeltään App Hub. Seuraavassa esitellään sivustot ja tarkastellaan niiden eroavaisuuksia.

5.1 Google Play

Android Developer Console on kehittäjän sivusto hallita Google Play -jakelukanavaan julkaistuja sovelluksia. Sivustolla rekisteröitynyt kehittäjä voi päivittää omia tietojaan, päivittää julkaisemiensa sovelluksia ja niiden tietoja, sekä julkaista uusia sovelluksia jakeluun. Sivusto toimii myös Googlen ja kehittäjän välisenä tiedonjakokanavana, jossa Google ilmoittaa sovelluksen julkaisuehtoihin tai kehitykseen vaikuttavista muutoksista.

Sovelluskohtaisista tiedoista pääsee muokkaamaan Google Playssä näkyviä tietoja ja sovelluksesta, kuten sen kuvausta, kuvakaappauksia ja ikoneita. Osion kautta sovellukseen voi julkaista päivityksiä tai vetää sovellus pois jakelusta. Sovelluksen tiedoista löytyy myös käyttäjien jättämät arvostelut sovelluksesta sekä sovelluksen aiheuttamat virhelokit.

Sovelluskohtaisista tilastoista saa tietoa sovelluksen latausmääristä, sekä kyseisellä ajanhetkellä aktiivisena olevista asennuksista. Tilastoista näkee myös yksityiskohtia sovelluksen lataajien Android-käyttöjärjestelmäversiosta ja -laitteista, maista, kielistä sekä operaattoreista. Kaikkia tietoja pystyy tarkastelemaan valitsemallaan ajanjaksolla. Raportit pystyy myös viemään CSV-tiedostona, jolloin niitä on mahdollisuus muokata omien tarpeidensa mukaan. Tilastot näyttävät myös tietoa sovelluksen julkaisukategoriasta yleisellä tasolla, kuten kyseisen kategorian suosituimmat Android-versiot laitteissa.

5.2 iTunes Connect

iTunes Connect on sivusto, jonka kautta App Storessa julkaistavien sovellusten, ja niihin liittyvien tietojen hallinta tehdään. Sivusto toimii myös tiedonvälityskanavana kehittäjän ja Applen välillä.

Sovellustenhallinta-osiosta pääsee muokkaamaan ja tarkistelemaan julkaistujen ja julkaisua odottavien sovellusten tietoja, kuten sovelluksen kuvausta, hinnoittelua, kuvakaappauksia, ladattujen binääritiedostojen versiohistoria ja muuta App Storessa julkaistavaa tietoa. Samasta osiosta kehittäjä saa myös tietoa sovelluksen loppukäyttäjiltä. Sovelluksen käyttäjien lähettämät virhelokit ja käyttäjäarvostelut ovat saatavilla sovelluksen tiedoissa.

Sopimukset, verot ja tilitys -osiossa hallitaan kehittäjän ja Applen välisiä sopimuksia. Sitä kautta voi tarkastella vanhoja ja tehdä uusia sopimuksia. Allekirjoitettavia julkaisusopimuksia on kolme kappaletta. Ilmaiseksi jaeltaville ja maksullisille sovelluksien julkaisulle on omat sopimuksensa, jonka lisäksi sovellusten sisällä mainonnan mahdollistava iAd Network vaatii erillisen sopimuksen. Näitä sopimuksia tehdään tarpeen mukaan. Tuntilistat.fi-sovellus julkaistiin ilmaiseksi, joten sille riitti ilmaisten sovellusten jakelun mahdollistava sopimus. Tähän osioon syötetään myös tili- ja verotustiedot, joiden avulla Apple tekee maksullisten sovellusten myynnistä muodostuvat tilitykset.

Raportointi ja myynnin seuranta -osioita on kaksi. Toinen mahdollistaa sovellusten päivittäisen ja viikoittaisen myyntimäärien seurannan, ja toinen on kuukausittainen raportti, joka sisältää myös Applen kehittäjälle tekemät myynnistä saatavat tilitykset.

iTunes Connect toimii myös käyttäjien hallintakanavana, jota kautta kehitystiimiin voi lisätä uusia kehittäjiä tai testikäyttäjiä. Tämä mahdollistaa esimerkiksi lisättyjen käyttäjien sovelluksen testaamisen omilla laitteillaan.

5.3 App hub

App Hub -sivustolta löytyvä my dashboard -osio mahdollistaa Windows Phone Marketplacella julkaistavien sovellusten ja niiden tietojen hallinnan. My dashboard -osio koostuu My Payouts-, App Highlights-, Reports-, Submit a new app-, How To- ja Notifications-osioista.

My Payouts -osiosta löytyy kehittäjälle maksetut korvaukset sovelluksien myynnistä. Korvaukset ovat jaettu viimeiseksi maksettuun ja historiaan, josta näkee kaikki kehittäjälle maksetut korvaukset.

App Highlights -osio listaa kehittäjän sovelluksen tai sovelluksien latausmäärät viimeiseltä kuukaudelta, sekä mahdolliset käyttäjien lähettämät virhelokit. Sovelluksen nimeä painamalla aukeaa sovelluksesta tarkemmat tiedot, kuten sovelluksen käyttäjien jättämät arvostelut, hinnoittelutiedot, sovelluksen elinkaari ja sovelluksen lisätiedot. Sovelluksen elinkaari -sivulta kehittäjä voi päivittää sovellusta, sen tietoja sekä hinnoittelua. Lisätiedot sisältävät sovelluksen kuvauksen ja ruutukaappaukset Matketplacella, ikonit, versiotiedot ja muut vastaavat sovelluksen arvosteluun jätettäessä syötettävät tiedot. Näitä tietoja ei kuitenkaan ole mahdollisuutta muuttaa ilman erillistä Microsoftin suorittamaa sovelluksen uudelleen arviointia.

Reports-osiosta kehittäjä voi seurata sovelluksen latausten määrä, sekä sovelluksen kaatumisraportteja ja niiden kehitystä. Submit a new app -sivulta kehittäjä voi jättää uusia sovelluksia Microsoftin arvioitavaksi julkaisua varten. How To -osio sisältää julkaisuun ja sitä tukevien operaatioihin liittyviä ohjeita. Notifications sisältää kehittäjän tietoihin tai julkaistuihin sovelluksiin liittyviä huomioita, jotka odottavat kehittäjän toimenpiteitä.

5.4 Hallintasivustojen erot

Sovellustenhallinta ja sovelluksista saatavilla olevat tiedot ovat alustojen välillä pitkälti toistensa kaltaiset. Kaikki jakelukanavat kertovat sovelluksen latausmäärät ja niiden kehitystä pystyy seuraamaan. Virhelokit sovellusvirheistä on myös kaikissa jakelukanavissa saatavilla.

Google Play tarjoaa kuitenkin latauksista yksityiskohtaisimmat tiedot. Se näyttää maakohtaisia latausmääriä, sovellusten lataajien käyttöjärjestelmäkielet, sovellusten lataajien teleoperaattorit ja ehkä merkittävimpänä tietona sovelluksen aktiiviset asennukset. Aktiivisten asennusten perusteella kehittäjä voi arvioida sovelluksensa todellisia käyttäjämääriä. App Store ja Windows Phone Marketplace eivät näitä tietoja kerro.

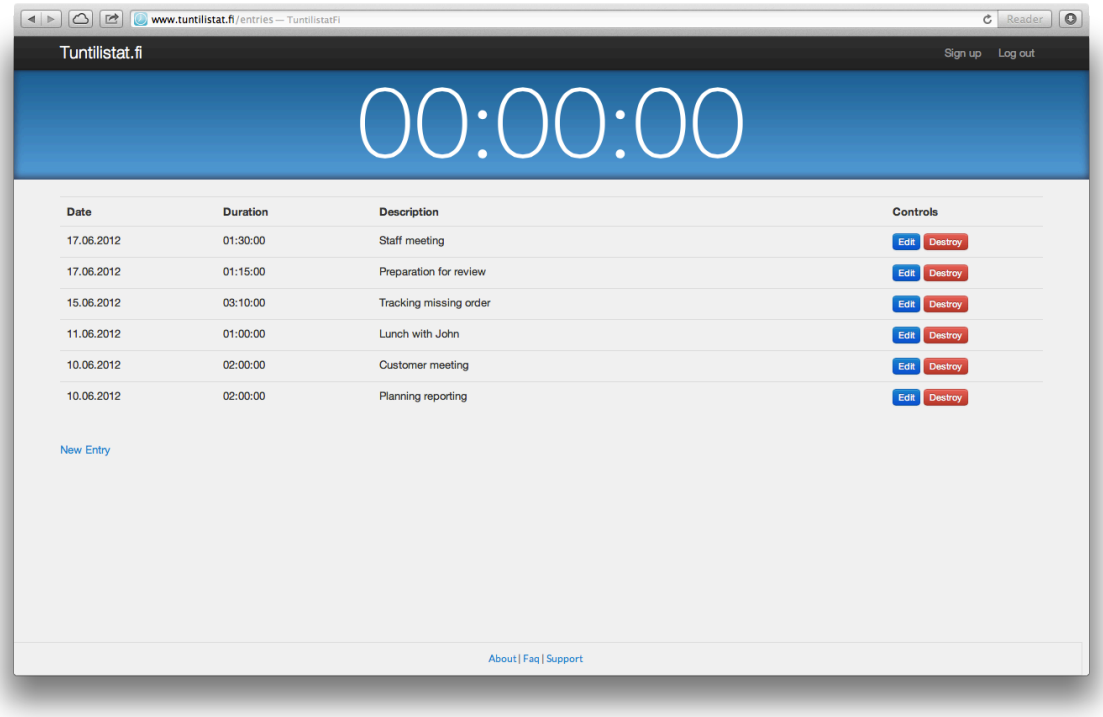
Sovelluksen oheistietojen päivitys on Google Playssä kahta muuta jakelukanavaa vaivattomampaa. Google Play -sovellusten tietojenpäivitys onnistuu kehittäjältä suoraan, kun kahdessa muussa kanavassa tietojen päivitys vaatii jakelukanavan hallitsijan arvioinnin. Tämä luo tietojenpäivitykseen muutaman päivän viiveen.

6 SOVELLUSTEN KEHITTÄMINEN

Diplomityön käytännön osuus toteutettiin kehittämällä Tuntilistat.fi-palvelun mobiilisovellukset eri ohjelmistoalustoille, ja julkaisemalla ne kunkin alustan jakelukanavaan. Seuraavassa esitellään Tuntilistat.fi-palvelu, sekä toteutetut mobiilisovellukset ja niiden julkaisu jakelukanaviin. Kappaleessa kuvataan myös eri alustojen kehitystyössä ja julkaisussa ilmenneitä erityispiirteitä.

6.1 Tuntilistat.fi palvelun esittely

Tuntilistat.fi on web-pohjainen tuntien- ja tehtävienkirjausjärjestelmä. Se on suunniteltu yksityisten ihmisten ja pienyritysten käyttöön. Käyttäjän tulee rekisteröityä ja luoda henkilökohtainen tili palveluun. Tämän jälkeen hän voi kirjautua palveluun, ja syöttää tehtäviä järjestelmään. Tuntilistat.fi mahdollistaa tehtävien muokkauksen ja seurannan. Palvelu on vielä kehityksen alla ja tällä hetkellä ilmainen käyttää.



Kuva 9. Tuntilistat.fi-sivusto.

Järjestelmä on toteutettu Representational State Transfer (REST) -arkkitehtuurin mukaisesti ja sovelluksen tietoliikenne perustuu JavaSript Object Notation (JSON) -

viesteihin. Nämä keskeisimmät tekniikat vaikuttivat myös mobiilisovellusten toteutukseen.

Roy Fielding määrittelee REST-arkkitehtuurin rajoitteiden kautta seuraavasti:

- Asiakas/Palvelin (engl. client-server), lähtökohta erottaa asiakkaan ja palvelimen ongelmat toisistaan. Järjestelmien siirto alustojen välillä (engl. porting) helpottuu, skaalautuvuus paranee, sekä asiakas ja palvelinkomponenttien toisistaan riippumaton kehitys mahdollistuu.
- Tilattomuus (engl. stateless), asiakkaan ja palvelimen välinen liikenne tulee olla tilatonta. Toisin sanoen jokaisen asiakkaan palvelimelle lähettämä pyyntö tulee sisältää kaikki tarpeellinen tieto sen käsittelyyn ja ymmärtämiseen, eikä palvelimelle tallennettua tietoa tule hyödyntää sen käsittelyyn.
- Välimuistin (engl. cache) käyttö, verkkoliikenteen tehostamiseksi palvelimen vastaus asiakkaan pyyntöön voidaan jättää asiakkaan välimuistiin, jolloin sen sisältämää tietoa voidaan käyttää seuraavaan samaan pyyntöön. Vastaukset tulee merkitä onko välimuistin käyttö sallittua vai ei.
- Yhdenmukaiset rajapinnat (engl. Uniform Interface), komponenttien roolien ja rajapintojen tulee olla selkeät ja yhdenmukaiset. Tämä yksinkertaistaa järjestelmän arkkitehtuuria, ja kannustaa komponenttien itsenäiseen kehittämiseen.
- Kerroksittainen järjestelmä (engl. Layered system), parantaa internetin vaatimaa skaalautuvuutta, ja mahdollistaa hierarkkisista kerroksista koostuvan arkkitehtuurin.
- Code-On-Demand, mahdollistaa asiakkaan toiminnallisuuden laajentamisen palvelimelta ladattavalla suoritettavalla koodilla. Tämä mahdollistaa asiakkaan toteutuksen pitämisen yksinkertaisena, ja laajennettavana tarpeiden mukaan. [37]

JavaScript Object Notation eli JSON on tekstipohjainen avoin standardi tiedon välitykseen. Se on suunniteltu ihmiselle helposti luettavaksi ja tietokoneelle kevyeksi jäsentää. Se koostuu avain- ja arvopareista, jotka ovat eroteltu kaksoispisteellä toisistaan. Näitä pareja voi olla samassa viestissä useita ja ne erotellaan toisistaan pilkuilla. Kokonaisen JSON-viestin rajaa aaltosulut. [38]

Alapuolella oleva esimerkki kuvaa Tuntilistat.fi-palvelun JSON-viestistä, joka sisältää

yhden palveluun syötetyn tehtävän tiedot. Esimerkki viestissä on sisäkkäisiä avainarvopareja. Entry-kentän arvo (engl. value) on toinen JSON-rakenne, joka koostuu kentistä: ”created_at”, ”description” ja ”duration”.

{entry: {created_at : 04052012, description : ”opiskelua”, duration : 3500}}

6.2 Mobiilisovellusten suunnitteluperiaatteet ja esittely

Diplomityössä toteutettiin Tuntilistat.fi-web-sovelluksen mobiiliversiot Android-, iOS- ja Windows Phone 7 -alustoille. Mobiilisovellusten logiikka ja toiminnallisuus pyrittiin pitämään mahdollisimman yhdenmukaisina. Sovellus muodostuu kolmesta näkymästä, jotka ovat kuvattu seuraavassa:

- Päänäkymä, jossa näkyvät käyttäjän syöttämät tehtävät, sekä niiden luomispäivämäärät ja kestot listana.
- Tehtävän syöttönäkymä, jossa käyttäjä voi luoda uuden tai muokata jo olemassa olevan tehtävän tietoja.
- Asetukset näkymä, jossa käyttäjä syöttää tuntilistat.fi-käyttäjätunnuksen ja salasanansa. Tästä näkymästä käyttäjä pystyy myös rekisteröitymään palvelun käyttäjäksi.

Kuva 10 esittelee sovelluksen kaksi näkymää iOS-versiossa. Vasemmalla puolella on päänäkymä ja oikealla tehtävien syöttönäkymä.

Kuva 10. iOS-sovelluksen päänäkymät.

Alustojen eroavaisuuksista johtuen yhtenäistä koodia sovellusten välillä ei pystytty hyödyntämään. Suuren rajoitteen yhteisen koodin jakamiseen toi luonnollisesti jo eriävät ohjelmointikielet: C#, Java ja Objective-C. C#- ja Java-kielen syntaksi on hyvin pitkälti samankaltainen, mutta alustojen toisistaan eroavat rajapinnat ja ohjelmistokirjastot tekevät yhteisen koodin luomisen mahdottomaksi.

Sovellusarkkitehtuuri pyrittiin pitämään sovelluksissa samankaltaisena. Kussakin ohjelmistoalustassa on kuitenkin vahvat ominaispiirteensä, jotka omalta osaltaan vaikuttavat myös sovelluksen arkkitehtuuriin. Arkkitehtuurin suunnittelu lähtökohdaksi otettiin sovelluksen logiikan ja datan erottaminen näyttökerroksesta. Sovelluksissa käytettiin Model-View-Controller (MVC) suunnittelumallia.

MVC on yksi yleisimmin käytetty suunnittelumalli ohjelmistokehityksessä. Sen kehitti 1970-luvun lopulla Trygve Reenskaug SmallTalk-ohjelmointikielelle. MVC-malli muodostuu kolmesta roolista: malli (engl. model), näkymä (engl. view) ja ohjain (engl. controller). Malli on olio joka esittää jotain tietoa kohdealueesta. Se sisältää kaiken muun tiedon ja toiminnallisuuden paitsi käyttäjälle näytettävän käyttöliittymän. Näkymä esittää mallin tiedon käyttäjälle näytöllä. Suunnittelumallin kolmas osa ohjain, vastaa mallin

muutosten välittämisestä näytölle sekä käyttäjän syötteen välittämisestä näytöltä malliin. [39]

Toinen sovellusten toteutuksessa järjestelmällisesti käytetty suunnittelumalli oli Singleton. Sitä käytettiin kaikissa kolmessa sovelluksessa mallin luomiseen.

Singleton-suunnittelumalli varmistaa, että luokasta luodaan vain yksi ilmentymä, ja tarjotaan globaali tapa kaikille päästä käsiksi tähän samaan ilmentymään [40]. Singleton-mallin käyttö tuntui loogiselta, sillä jokaisen näkymän ohjain pystyi tarpeen mukaan hakemaan tarvitsemansa tiedon samasta mallin ilmentymästä.

Tietoliikenne mobiili-sovellusten ja web-järjestelmän välillä on toteutettu https-protokollan yli JSON-viestein. REST-arkkitehtuurin mukaisesti mobiilisovellus voi lähettää pyynnön esimerkiksi osoitteeseen https://tuntilistat.fi/entries, jolloin järjestelmä lähettää https-vastauksen, jossa kyseisen käyttäjän kaikki tehtävät ovat JSON-muodossa. Tämän jälkeen sovellus lukee JSON-viestistä datan malliinsa, josta se esitetään käyttöliittymälle ohjaimen välittämänä.

Sovellukset haluttiin pitää mahdollisimman yksinkertaisena. Tästä syystä mobiilisovellukset eivät tallenna päätelaitteisiin muita tietoja kuin käyttäjätunnuksen ja salasanan. Mobiilisovellukset siis käyttävät suoraan web-järjestelmän dataa. Samasta syystä julkaistut sovellukset kohdennettiin pääasiallisesti mobiilipäätelaitteiden käyttöön, ja niiden toteutuksessa ei huomioitu tablet-laitteita. Sovellukset voi asentaa ohjelmistoalustan version mukaisiin tablet-laitteisiin, mutta ne eivät hyödynnä erityisesti tablet-koneille suunnattuja komponentteja ohjelmistoalustoissa. Taulukko 2 esittää sovellusten käyttämien ohjelmistoalustojen ja kehitystyökalujen versiot.

Taulukko 2. Ohjelmistoalusta- ja työkaluversiot.

	Android	iOS	Windows Phone 7
Ohjelmistoalusta	2.3.3	iOS 5	7.5 (Tango)
IDE-versio	Eclipse 3.5.2 (ADT 11.0)	Xcode 4.4.1	Visual Studio Express 2010
Käyttöjärjestelmä	OS X 10.7	OS X 10.7	Windows 7

6.3 Sovellusten kehitystyö

Ohjelmistoalustat voidaan katsoa olleen diplomityön tekijälle entuudestaan täysin vieraita. Java ja C#-ohjelmointikielet olivat entuudestaan jossain määrin tuttuja, vaikka pääasiallinen ohjelmointikokemus tekijällä oli C++-kielestä. Objective-C oli syntaksiltaan ja ideologialtaan kolmesta toteutuskielestä entuudestaan kaikista tuntemattomin.

Sovelluksen yleinen ensimmäinen prototyyppi toteutettiin Android-alustalla, joka kuitenkin päädyttiin käytännössä uudelleen kirjoittamaan kokonaan muista alustoista saatujen kokemusten ja näkemysten muuttumisesta johtuen. Kokonaisuudessaan Android-alustaan tutustumiseen käytettiin 23 tuntia, joka muodostui Android DevCenterin materiaalien tutustumisesta ja esimerkkien kokeilusta. Ensimmäinen Android prototyypin tekemiseen sovelluksesta meni 50 tuntia. Julkaistun ja uudelleen kirjoitetun Android version tekoon meni 70 tuntia.
Ensimmäisen Android-prototyypin jälkeen toteutettiin Windows Phone 7 -versio. Windows Phone alustaan tutustuminen tehtiin MSDN Channel9 -videoiden ja esimerkkisovelluksiin tutustumalla. Aikaa alustaan tutustumiseen käytettiin 25 tuntia, ja varsinaisen sovelluksen tekoon 54 tuntia.

Viimeisenä toteutusalustana, ennen Android-sovelluksen uudelleen kirjoitusta, oli iOS. Siihen tutustuminen tapahtui Applen iTunes -kaupassa ilmaiseksi tarjottavien Standfordin yliopiston järjestämän kurssin materiaaleihin perehtymällä. Alustaan tutustumiseen käytettiin 15 tuntia ja sovelluksen varsinaiseen toteutukseen 77 tuntia.

Android-kehitykseen kului eniten aikaa. Tämä ei kuitenkaan tekijän kokemusten perusteella suoranaisesti kerro Android-alustan olevan aloituskynnykseltään tai muutenkaan hitain kehitysalusta. Suuri osa ajan kulumisesta selittyy kehitysjärjestyksellä, sillä sovellus muovautui ja monien ongelmien ratkaisumallit selkeytyivät kehitysiteraatioiden välillä. Kirjoitettujen koodirivien määrä oli Windows Phone 7-alustalla pienin, ja sen kehitys onnistui myös lyhyimmässä ajassa. Kehityksen haastavuuden absoluuttinen arviointi on kuitenkin hyvin vaikeaa. Kehittämisen helppouteen vaikuttaa luonnollisesti kehittäjän aikaisempi kokemus kyseisistä tekniikoista, sekä saatavilla olevan lähdemateriaalin laatu. Taulukko 3 esittää ohjelmistoalustakohtaisen

ajankäytön.

Taulukko 3. Alustakohtainen ajankäyttö.

	Android	iOS	Windows Phone 7
Alustaan tutustuminen	23h	15h	28h
Sovelluksen kehitys	70h + 50h (prototyyppi)	77h	54h

6.4 Kehityslisenssin hankinta

Sovelluksen julkaisu kussakin jakelukanavassa vaatii voimassa olevan kehityslisenssin. Applen ja Microsoftin lisenssit ovat uusittava vuosittain, kun Google Play -lisenssin hankinta tapahtuu kertaluonteisella maksulla.

App Store- ja Windows Phone Marketplace -julkaisulisenssin hinta on 99$ vuodessa. Euromääräiset hinnat lisensseille on App Storessa 79€ ja Windows Phone Marketplacella 75€. Google Play -lisenssin kertamaksu on 25$. Google ei ole määritellyt lisenssille eurohintaa, vaan se maksetaan dollareissa. Google Playn edeltäjä Android Market oli alun perin ilmainen, mutta Google otti kertaluonteisen maksun käyttöön luodakseen sovelluksen julkaisulle pienen kynnyksen, ja näin pyrkimällä nostamaan sovellusten yleistä laatua [41].

Rekisteröityminen ja kehityslisenssin hankinta onnistui kuhunkin jakelukanavaan ongelmitta. Apple mainitsee, että rekisteröinnin käsittely voi kestää 24 tuntia [42], mutta minkään kehityslisenssin kohdalla käsittely viive ei ollut kahta tuntia suurempi.

6.5 Sovellusten julkaisu

Sovellukset julkaistiin kauppapaikoissa kehitystyön päätyttyä. Julkaisun esivaatimuksena oli kehityslisenssin hankinta ja jakelukanavan julkaisulle asettamien ehtojen noudattaminen. App Store -julkaisu arvelutti etukäteen eniten, sillä jopa 60% ensimmäisistä julkaisuista hylätään erinäisten syiden takia [43]. Apple asettaa julkaisulle paljon vaatimuksia, jotka ovat kuvattu App Store Review Guidelines -dokumentissa.

6.5.1 App Store -julkaisu

App Store Review Guideline on käytännössä pitkä lista asioita jotka aiheuttavat

sovelluksen hylkäyksen ja näin estävät sovelluksen julkaisun App Storeen. Dokumentin tyyli on hyvin vapaamuotoinen, ja sen alussa todetaan etteivät kaikki hylkäysperusteet ole yksiselitteisiä. Johdannossa todetaan että sovellukset, jotka näyttävät hätäisesti tehdyiltä, tai eivät tarjoa loppukäyttäjälle aidosti hyödyllistä toiminnallisuutta voidaan hylätä. Samoin arvostelussa voidaan hylätä sovellukset jotka menevät ”rajan” yli. ”Raja” määritellään sillä, että arvostelija huomaan sen nähdessään. Johdanto-kappaleen jälkeen on pidempi yksiselitteinen kieltolista, jonka luettelemia asioita sovellus ei saa rikkoa. [35]

Sovelluksen julkaisu App Storeen tehdään iTunes Connect -sivuston kautta. Sivustolle kirjautumisen jälkeen uudelle sovellukselle syötetään sen perustiedot: sovelluksen kieli, nimi, SKU-numero (engl. Stock Keeping Unit) ja bundle-tunnistin. SKU-numeroa käytetään sovelluksen myynnin seuraamiseen App Storessa ja bundle-tunnistinta sovelluksen binäärien päivittämiseen.

Toisessa vaiheessa syötetään sovelluksen haluttu julkaisupäivämäärä. Mikäli sovellus ei ole läpäissyt Applen arviointia ennen valittua päivämäärää, se julkaistaan heti arvostelun päätyttyä. Tuntilistat.fi-sovellukselle valittiin julkaisuajankohdaksi suoraan arvioinnista -vaihtoehto. Tässä vaiheessa valitaan myös sovelluksen hinnoittelu. Hinnoitteluperiaate valitaan valmiista kategorioista, joiden valuuttakurssit ovat kiinnitetty etukäteen. Alin maksullinen kategoria on 0,99$ ja 0,79€. Julkaistavalle sovellukselle voi tarjota myös määräalennuksia koululaitoksille. Tuntilistat.fi jaeltiin ilmaisena sovelluksena, joten hinnoittelu ja alennuskategorioita ei valittu erikseen. Sovelluksen kohdemaita on myös mahdollista rajata. Kehittäjä saa valita minkä maan App Storeissa kyseinen sovellus on saatavilla. Oletusarvoisesti sovellus on saatavilla kaikissa maissa.

Seuraavassa vaiheessa syötetään sovelluksen versiotiedot kuten versionumero ja tekijänoikeuksien omistajuus, sekä valitaan kategoriat jotka parhaiten kuvaavat sovelluksen käyttötarkoitusta. Kategorioita on mahdollisuus valita kaksi, pää- ja toissijainen kategoria. Valittujen kategorioiden perusteella käyttäjät etsivät sovelluksia App Storesta. Tuntilistat.fi-sovelluksen pääkategoriaksi valittiin liiketoiminta (engl. business) ja toissijaiseksi kategoriaksi elämäntyyli (engl. lifestyle). Tässä vaiheessa on mahdollisuus kirjoittaa sovelluksen arvioijalle viesti asioista, mitkä mahdollisesti

vaikuttavat tai auttavat arviointiprosessia. Vaadittavat asetukset tai perustelut sovelluksen normaalista poikkeavaan käytettävyyteen tulee mainita tässä. Esimerkkisovellus vaati käyttäjätunnuksia Tuntilistat.fi-paveluun, joten testauksen mahdollistavat käyttäjä- ja tilitiedot syötettiin tähän.

Kehittäjä kirjoittaa kuvauksen sovelluksen toiminnasta ja käyttötarkoituksesta, sekä määrittelee sovelluksen käyttöikärajan. Nämä kentät näytetään App Storessa sovelluksen tiedoissa. Avainsanat kenttään syötetään avainsanoja, joiden perusteella sovellus esiintyy App Storen -avainsanahauissa. Sovellukselle tulee antaa myös www- ja sähköpostiosoite, joita kautta käyttäjät voivat saada apua ja tukea sovelluksen käytössä. Sovellukset jotka tarjoavat automaattisesti uusittavia tai ilmaisia tilauspalveluita, vaaditaan myös www-osoite, josta löytyy käyttäjän yksityisyydensuojaamisen tae (engl. privacy policy url). Kaikille julkaistaville sovelluksille tarjotaan oletuksena Applen tarjoama loppukäyttäjän lisenssisopimus (engl. end user license agreement), mutta kehittäjällä on myös mahdollisuus määritellä oma lisenssisopimus niin halutessaan.

Seuraavassa vaiheessa sivustolle ladataan App Storen vaatimat kuvakaappaukset ja kuvakkeet sovelluksesta. Sovelluksesta vaaditaan kuvake, jota näytetään App Storessa hakujen ja latausten yhteydessä. Kuvakaappauksia sovelluksesta vaaditaan yksi, tämä näytetään sovelluksen tiedoissa. Lisäksi on mahdollista ladata neljä muuta kuvakaappausta näytettäväksi.

Kun kaikki tarpeelliset tiedot sovelluksesta on syötetty, siirtyy sovellus Ready to Upload Binary -tilaan, jolloin varsinaisen sovelluksen lataaminen arviointiin on mahdollista. Sovelluksen binääritiedostot voi lähettää joko erillisen Application Loader -työkalun avulla, tai suoraan kehitystyökalu XCodesta. Tuntilistat.fi-sovellus ladattiin arvioitavaksi XCoden kautta.

Apple pitää kirjaa ja päivittää sivustolleen sovelluksen julkaisun arviointiprosessin kestosta. Toukokuussa vuonna 2012 ilmoitetun tilaston mukaan yli 90 prosenttia arviointiin jätetyistä sovelluksista saatiin käsiteltyä viiden arkipäivän sisällä [44]. Diplomityössä julkaistu sovellus jätettiin Applen arviointiin perjantaina 20.4.2012 ja se

julkaistiin App Storessa torstaina 26.4.2012. Arviointi siis saatiin tehtyä ennakoidun ajan mukaisesti, ja sovellus läpäisi arvioinnin ensimmäisellä yrityksellä. Apple kommunikoi arvioinnin edistymistä tilannepäivityksillä. Sovellus ladattiin arviointiin 20.4 ja sen tila muutettiin arvioinnissa olevaksi 26.4. Saman päivän aikana arviointi saatiin tehtyä ja sovellus julkaistiin App Storeen.

6.5.2 Windows Phone Marketplace -julkaisu

Windows Phone Marketplace -julkaisua helpottaa Microsoftin kehitystyökaluihin kuuluva Marketplace Testkit. Testkit määrittelee testitapauksia, jotka sovelluksen tulee läpäistä, jotta se olisi julkaisukelpoinen. Osa testitapauksista on automatisoituja, ja osa manuaalisesti suoritettavia. Manuaalisissa testitapauksissa on selkeät askeleet, joita kehittäjän tulee suorittaa arvioitavaksi jätettävälle sovellukselle. Näissä testeissä muun muassa varmistetaan sovelluksen oikeaoppinen toiminta eri tilanteissa, kuten puhelun tullessa ja kalenterihälytyksen aikana.

Windows Phone marketplace -julkaisu tehdään App Hub -sivuston kautta. Ensimmäisessä vaiheessa syötetään sovelluksen nimi, valitaan ladataanko sovellus julkiseen jakeluun vai yksityiseen testauskäyttöön, valitaan jaeltavan sovelluksen xap-paketti ja syötetään sovelluksen versionumero.

Seuraavassa vaiheessa syötetään sovelluksen kuvaus ja muut yleiset tiedot. Sovellukselle valitaan pääkategoria, ja sen alta mahdollisesti löytyvä alakategoria. Sovelluksen tukemat kielet luetaan suoraan xap-paketista, ja kaikille tuetuille kielille voidaan syöttää erikseen seuraavat kauppapaikalla näytettävät tiedot: lyhyt kuvaus sovelluksesta, yksityiskohtainen kuvaus, hakusanat joiden avulla sovellusta voidaan etsiä kauppapaikalta, www-osoite, käyttäjä- ja sopimusehdot sovellukselle sekä sähköpostiosoite sovelluksen tuelle. Windows Phone Marketplace -julkaisu vaatii tuekseen kolme eri kokoista sovelluskuvaketta: mobiiliversio-kauppapaikasta vaatii ison ja pienen kuvakkeen sovelluksesta, ja kolmas koko tarvitaan kauppapaikan pöytätietokone-version käyttöön. Sovelluskuvakkeiden lisäksi ladataan sovelluksesta vähintään yksi kuvakaappaus ja maksimissaan kahdeksan kappaletta. Sovellukselle voi halutessaan ladata taustakuvan, jota käytetään sovelluksen markkinoinnissa, mikäli se päätyy Microsoftin mainoskampanjaan.

Kolmannessa vaiheessa julkaisua määritellään sovelluksen hintakategoria ja saatavuusalue. Tuntilistat.fi-sovellus jaettiin ilmaisena, joten hinnaksi määriteltiin nolla euroa.

Viimeisessä vaiheessa syötetään sovelluksen arviointiin ja testauksen liittyvät lisätiedot, kuten verkkopalveluihin mahdollisesti tarvittavat käyttäjätunnukset. Tässä vaiheessa valitaan myös sovelluksen julkaisuajankohta. Julkaisu voidaan valita tapahtuvan heti arviointiprosessin onnistuttua, tai se voidaan valita myös myöhemmin määriteltäväksi, jolloin kehittäjä voi suorittaa julkaisun haluamanansa ajankohta arviointi hyväksynnän jälkeen. Tuntilistat.fi valittiin julkaistavaksi suoraan arviointiprosessista.

Microsoft toteaa sovellusarvioinnin kestävän keskimääräisesti viisi arkipäivää. Tuntilistat.fi-sovellus jätettiin arviointiin 20.4.2012 ja julkaistiin Windows Phone Marketplacella 25.4.2012. Arviointi kesti siis kolme arkipäivää.

Kokonaisuudessaan Windows Marketplace -julkaisuprosessi vaikutti suoraviivaisemmalta ja selkeämmältä operaatiolta kuin App Store -julkaisu.

6.5.3 Google Play -julkaisu

Sovellusten Google Play -julkaisu tehdään Android Developer Consolen kautta. Ensimmäisessä vaiheessa ladataan julkaistavan sovelluksen APK-tiedosto ja mahdolliset laajennustiedostot, mikäli sovelluksen APK ylittää sille asetetun 50 megatavun ylärajan.

Seuraavassa vaiheessa syötetään kaikki muu sovelluksen julkaisulta vaadittava tieto. Tieto on ryhmitelty kuuteen kokonaisuuteen. Ensimmäinen kokonaisuus on sovelluksen tukiresurssit. Kuvakaappauksia sovelluksesta ladataan vähintään kaksi ja maksimissaan kahdeksan kappaletta. Sovelluskuvake, joka näytetään sovelluksen yhteydessä kauppapaikassa vaaditaan myös. Kehittäjän on myös mahdollista ladata sovelluksen markkinointia tukeakseen kaksi erikokoista kuvaketta, joita Google Play voi käyttää erilaisissa mainoskampanjoissa. Google Play mahdollistaa myös esittelyvideon lataamisen sovelluksesta, joka on kestoltaan 0,5 - 2 minuuttia. Mikäli sovelluksen käyttäjälle halutaan tarjota yksityisyydensuoja menettelyehdot (engl. privacy policy) syötetään sen osoite tähän. Kehittäjällä on mahdollisuus jättäytyä kaikkien Google Playn järjestämien

mainoskampanjoiden ulkopuolelle niin halutessaan.

Toinen kokonaisuus on sovelluksesta Google Playssa näytettävät tiedot. Sovelluksen tukemat kielet valitaan, ja tiedot syötetään kielikohtaisesti. Sovelluksen tietoihin annetaan nimi, kuvaus sovelluksesta, muutokset viimeisimpään versioon (mikäli kyseessä on päivitys), lyhyt kuvaus jota voidaan käyttää sovelluksen mainostamiseen ja valitaan sovelluksen tyyppi ja kategoria. Tyyppi on joko sovellus tai peli, ja kategoria valitaan pidemmästä listasta.

Kolmas kokonaisuus on julkaisun tiedot. Kehittäjä valitsee sovelluksen kopiosuojauksen, eli onko sovellusta mahdollista kopioida Android-laitteesta. Sovellukselle määritellään sopiva käyttöikäraja sen sisällön mukaan neljästä kategoriasta. Viimeiseksi valitaan hinta ja alueellinen saatavuus.

Neljäs kokonaisuus on kehittäjän yhteystiedot, jotka koostuvat kotisivuista, sähköpostiosoitteesta ja puhelinnumerosta. Viides kokonaisuus on sovelluksen mahdollinen tuki Google Cloud Messagingille, joka mahdollistaa verkkopalvelun ja sovelluksen välisen liikenteen optimoinnin tietyissä tilanteissa. Tuntilistat.fi ei käyttänyt tätä ominaisuutta.

Viimeinen kokonaisuus on sopimukset, joissa kehittäjä vahvistaa, että sovellus noudattaa Android Content Guidelinea –ohjeistusta [45], joka on Googlen Android-sovelluksille määrittelemä sisältösopimusta. Kehittäjän on myös vahvistettava tietoisuutensa julkaistavan sovelluksen olevan velvollinen noudattamaan Yhdysvaltojen lain asettamia vaatimuksia maasta vietävälle vientituotteille.

Sovellus on valmis julkaistavaksi, kun kaikki tarpeellinen tieto on syötetty. Google Play ei suorita sovelluksille arviointia ennen julkaisua, joten sovelluksen julkaisuprosessi on nopea. Tuntilistat.fi-sovellus oli ladattavissa Google Playstä kolme tuntia sovelluksen julkaisun jälkeen. Google Play -käyttäjäehdoissa ei tarkemmin oteta kantaa julkaisun kestoon, vaan niissä todetaan että sovellusten prosessointi vie yleensä muutaman tunnin.

6.6 Erot julkaisussa

Sovelluksen julkaisuprosessin suurin ero on julkaisussa esiintyvä viive, joka aiheutuu App Store- ja Windows Phone Marketplace -jakelukanavissa suoritettavasta sovellusten arvioinnista ennen julkaisua. App Store -julkaisussa esiintynyt viive sovelluksen arviointiin jättämisen ja varsinaisen julkaisun välillä oli neljä arkipäivää. Windows Phone Marketplace julkaisussa vastaava viive oli kolme arkipäivää. Google Play julkaisussa esiintynyt viive oli vain kolme tuntia.

Toinen kehittäjälle hieman lisätyötä aiheuttava ero on sovelluksen käyttäjälle tarjottavan tuen vaatimuksissa. App Store vaatii sähköpostiosoitteen lisäksi kotisivun, jolta käyttäjä voi saada tukea sovelluksen käyttöön. Käytännössä Tuntilistat.fi-sovelluksen iOS-version tuelle tarkoitettu kotisivu on hyvin yksinkertainen, jossa on lueteltu sovelluksen perusominaisuudet ja sähköpostiosoite, josta käyttäjä voi ongelmatilanteissa pyytää apua. Muissa jakelukanavissa pelkkä sähköposti osoite riittää.

Julkaistavalle sovellukselle määritellään App Storessa ja Windows Phone Marketplacella pää- ja alikategoria, kun Google Playssa määritellään vain yksi kategoria. Sovelluskuvakkeiden ja kuvakaappausten määrissä on pieniä eroja. Suurin ero on Windows Phone Marketplacen vaatimassa kolmessa eri kokoisessa kuvakkeessa, jotka vaaditaan Marketplacen eri versioihin.

Sovelluksen kuvaukselle on määritelty 4000 merkin tila App Store- ja Google Play -jakelukanavissa, kun Windows Phone Marketplace -kuvauksen on mahduttava 2000 merkkiin. Avainsanoja ei Google Playssa määritellä erikseen, vaan haut perustava sovelluksen kuvaukseen. Windows Phone Marketplace avainsanoja saa märittää viisi kappaletta ja App Storen vastaava rajoitus on 2-100 merkkiä.

Google Play on jakelukanavista ainoa, joka mahdollistaa sovelluksen esittelystä videon. Muut jakelukanavat eivät tarjoa tätä mahdollisuutta.

Sovellusten kokorajoitteissa on huomattavia eroja. Windows Phone Marketplace asettaa julkaistavan sovellukselle 225 megatavun rajoitteen. App Storen vastaava rajoite on 2

gigatavua. Google Playssä julkaistavan sovelluksen APK-paketin koko on rajoitettu 50 megatavuun, mutta tämän lisäksi sovellukselle voi ladata kaksi maksimissaan kahden gigatavun kokoista sisältöpakettia. Sisältöpaketit voivat sisältää esimerkiksi paljon tilaa vaativia videoita ja musiikkia. Alla oleva taulukko 4 esittää sovelluksen julkaisun merkittävimmät erot.

Taulukko 4. Erot jakelukanavien julkaisussa.

	Android	iOS	Windows Phone 7
Julkaisuviive	3h	4 arkipäivää	3 arkipäivää
Sähköposti tuki	Kyllä	Kyllä	Kyllä
Kotisivu tuki	Ei vaadita	Kyllä	Ei vaadita
Kategoriat	1	Pää- ja alakategoria	Pää- ja alakategoria
Kuvakaappaukset	2 - 8	1 - 5	1 - 8
Sovelluskuvakkeet	1- 3	1	3
Sovelluksen kuvaus	max 4000 merkkiä	max 4000 merkkiä	max 2000 merkkiä
Vaatimusten kuvaus julkaisulle	Google Play	Review Guideline	App Hub Help
Metatietojen lokalisointi	Tuettu	Tuettu	Tuettu
Avainsanat	Ei määritellä erikseen	2-100 merkkiä	5 sanaa
Export compliancy / cryptography	Vaaditaan	Vaaditaan	Ei vaadita
Mainosvideo	Valinnainen	Ei tukea	Ei tukea
Sovelluksen maksimi koko	50Mt APK + 2 x 2GB	2 GB	225 Mt

6.7 Julkaisun jälkeisiä havaintoja

Sovelluksia ei mainostettu tai markkinoitu, joten odotukset niiden latausmääriin olivat hyvin maltilliset. Havaintoja haluttiin tuntemattoman kehittäjän ensimmäisen sovelluksen julkaisusta ja sen saamista latausmääristä. Sovellusten kuvaukset, kuvakkeet, hakusanat ja kaikki jakelukanavassa saatavilla oleva tieto pyrittiin pitämään mahdollisimman samankaltaisena. Jakelukanavat asettivat tälle luonnollisesti muutamia rajoitteita, kuten toisistaan hieman poikkeavat kategoriajaottelut.

Jakelukanavien välillä huomattiin kuitenkin merkittäviä eroja latausmäärissä. Sovellus sai selkeästi eniten latauksia Windows Phone Marketplacella, jossa latauksia kertyi ensimmäisen kuukauden aikana 80 kappaletta. Seuraavaksi eniten latauksia tuli App Storessa, jossa sovellusta ladattiin ensimmäisen kuukauden aikana 20 kertaa. Google Play kanavassa latauksia tuli vain kolme kappaletta. Latausmääriin vaikuttavia tekijöitä on vaikea, tai jopa mahdoton arvioida. Koska sovelluksen nimi on suomenkielinen, voidaan suurimman osan latauksista olettaa tulleen Suomesta. Maakohtaisia lataustilastoja tarjoaa jakelukanavista kuitenkin vain Google Play. Maakohtaisia laitekantoja ei ole saatavilla,

joten käyttäjämäärien vaikutusta latausten määrään ei voida arvioida. Windows Phone latausten määrää voi selittää sovelluksen hyvä näkyvyys jakelukanavassa, sillä Windows Phone Marketplace sovellusten määrä on huomattavasti pienempi kuin kahdessa muussa jakelukanavassa. Tuntilistat.fi-sovellus löytyy diplomityön kirjoitushetkellä, kesäkuussa 2012 Windows Phone Marketplacelta näkyvältä paikalta liiketoiminta-kategorian ilmaiset sovellukset pääsivulta.

Julkaisun jälkeen iOS-sovelluksessa huomattiin virhe, joka oli aiheutunut sovellukseen viimeisistä muutoksista. Muutos kohdistui palvelimen ja sovelluksen välisen virhetilanteiden käsittelyyn, ja sen vaikutuksia ei testattu riittävän huolellisesti. Virhe aiheutti sovelluksen sulkeutumisen tilanteessa, jossa käyttäjällä ei ole yhtään tehtävää syötetty. Sovellus kuitenkin läpäisi Applen arvioinnin ongelmitta, vaikka sovelluksen toiminnallisuuden kokeileminen kyseisessä tilanteessa voisi mieltää olevan testauksen kannalta perustoiminnallisuutta.

7 TULOKSET JA NIIDEN ARVIOINTI

Kehitystyökalut on saatavilla kaikille alustoille ilmaiseksi. iOS ja Windows Phone 7 kuitenkin asettavat rajoituksen kehitystyössä käytettävälle käyttöjärjestelmälle. iOS kehityksen ollessa mahdollista vain Applen OS X -käyttöjärjestelmällä, ja vastaavasti Windows Phone 7 -kehityksen vaativan Microsoftin Windows käyttöjärjestelmän. Android sovellusten kehitys onnistuu niin Linux-, OS X- kuin Windows-käyttöjärjestelmillä.

Kehitystyö eri alustojen välillä eroaa merkittävästi toisistaan. Ohjelmointikielet ja kirjastot ovat täysin erilaiset. Helppous ja lähestyttävyys kuhunkin ohjelmistoalustaan on hyvin vaikeasti mitattavissa. Yksi selkeä tekijä aloituskynnyksen madaltamiseen on kuitenkin kehittäjälle entuudestaan tuttu ohjelmointikieli. Android-kehityksessä pääasiallisena ohjelmointikielenä on Java, iOS-kehityksessä Objective-C ja Windows Phone 7 -kehityksessä C#. Diplomityön tekijä ei kuitenkaan kokenut uusien kielien olevan ylitsepääsemätön este kehitystyön aloittamiselle. Kaikki kielet olivat kuitenkin olio-ohjelmointikieliä, ja C# sekä Java vielä syntaksiltaan samankaltaisia. Eniten kielistä eroaa Objective-C selkeästi erilaisella syntaksillaan, ja muistinhallinnaltaan jota ei ole automatisoitu, niin kuin C# ja Java-ohjelmointikielissä. Diplomityössä toteutetun Tuntilistat.fi-sovelluksen kohdalla kehitystyössä ei havaittu rajoitteita eri alustojen välillä.

Julkaisuprosessien suhteen suurin ero on sovellusten arviointi ennen niiden julkaisua jakelukanavassa. Google Play ei suorita sovelluksille ennakkoarviointia, kuten App Store ja Windows Phone Marketplace. Tämä aiheuttaa sovelluksen julkaisulle ennakkoarvioinnin suorittavissa kanavissa viiveen. Diplomityössä julkaistujen sovellusten kohdalla viive oli Windows Phone Marketplace -julkaisussa kolme arkipäivää ja App Store –julkaiussa neljä arkipäivää. Toinen merkittävä ero on julkaisulta vaadittava lisenssi, joka on Google Playssä 25$ kertamaksu, kun App Store- ja Windows Marketplace -julkaisu vaativat voimassa olevan vuosittain uusittavan kehityslisenssin, jonka hinta on 99$ vuodessa. Muilta osin julkaisuprosessit ovat hyvin samankaltaiset. Julkaisut vaativat hieman erilaisia tietoja sovelluksesta, kuvakkeita ja kuvakaappauksia sovelluksesta, mutta erot näiden suhteen eivät ole kovinkaan merkittäviä. Pientä lisätyönä aiheuttavana erona voidaan pitää sovelluksen tuelle vaadittavaa kotisivua, jonka App Store -julkaisu vaatii, kun muissa

kanavissa sovelluksen tuelle riittää sähköpostiosoite.

Kehittäjän maksullisten sovellusten myynnistä saama prosenttiosuus on kaikissa kolmessa jakelukanavassa sama. Kehittäjä saa sovelluksen tuotosta 70% ja jakelukanavan omistaja 30%. Voitonjaon perusteella ei täten mitään jakelukanavaa voida pitää parempana vaihtoehtona. Yoon, Yoo ja Choi pitivät tutkimuksessaan tätä voitonjakoa, ainakin App Storen –tapauksessa optimaalisena [14], mutta on mielenkiintoista nähdä lähteekö joku jakelukanavista kilpailemaan kehittäjistä ja sovelluksista tarjoamalla erilaisia malleja voitonjakoon.

Latausten määrä kolmessa jakelukanavassa vaihteli suuresti. Windows Phone Marketplace latauksia tuli ensimmäisen kuukauden aikana 80 kappaletta, App Storessa 20 kappaletta ja Google Playssa kolme kappaletta. Kolme kuukautta julkaisun jälkeen Windows Phone Matketplacella sovelluksen lataus jatkui tasaisena, kokonaismäärän ollessa 231 kappaletta. App Storessa latauksia tuli hieman lisää, kokonaismäärän ollessa 31 kappaletta. Google Playssa latausten määrä ei kasvanut.

Sovellusten valtavan määrän takia menestyminen ja erottuminen jakelukanavassa on vaikeaa. Varsinainen haaste ei havaintojen perusteella ole sovelluksen kehitys ja julkaiseminen jakelukanavassa. Kuten Yakamani toteaa tutkimuksessaan, on mobiiliteollisuudessa siirrytty suljetusta muutaman pelurin mallista avoimempaan malliin [1]. Todellinen haaste on näkyvyyden ja latausten saaminen omalle sovellukselle.

Diplomityössä toteutetun Tuntilistat.fi-sovelluksen toteutuksessa ja julkaisussa ei havaittu alusta- tai jakelukanavakohtaisia rajoituksia. Merkittävin ero oli lopulta sovelluksen latausten määrässä, joka oli Windows Phone Marketplacessa selkeästi suurin. Koska diplomityössä kuitenkin julkaistiin vain yksi sovellus, ei tästä pidä tehdä liian pitkälle meneviä johtopäätöksiä. Työssä tehtyjen havaintojen perusteella mitään jakelukanavaa ei voida selkeästi nostaa toisten edelle.

8 YHTEENVETO

Diplomityössä tutkittiin itsenäisen kehittäjän näkökulmasta mobiilisovellusten kehitystä ja jakelua keskitetyissä jakelukanavissa. Työssä haluttiin kartoittaa merkittävimmät eroavaisuudet ja mahdolliset rajoitteet sovelluksen kehityksessä ja julkaisussa eri alustoilla. Tutkimus suoritettiin kehittämällä Tuntilistat.fi-mobiilisovellus Android-, iOS- ja Windows Phone 7 -alustoilla, ja julkaisemalla sovellukset kunkin alustan keskitetyssä jakelukanavassa.

Sovelluksen kehitys eri alustoilla eroaa merkittävästi toisistaan. Erot syntyvät eri ohjelmointikielistä, ja kunkin alustan omista ohjelmistokirjastoista. Esimerkki sovelluksen toteutuksessa ei ilmennyt teknisiä rajoitteita alustojen välillä. Ohjelmistoalustoja ei tulosten perusteella pystytty laittamaan paremmuusjärjestykseen, vaan tietyn alustan suosiminen toisen sijaan koettiin olevan kunkin kehittäjän omista mieltymyksistä kiinni.

Diplomityössä toteutetun Tuntilistat.fi-mobiilisovelluksen toteutuksessa ei esiintynyt teknisiä rajoitteita minkään ohjelmistoalustan kanssa. Sovellus oli luonteeltaan melko yksinkertainen. Yksi jatkotutkimuskohde voisi olla ohjelmistoalustojen tekniset rajoitteet. Tarjoaako joku alusta toisia kattavammat rajapinnat ja mahdollistaa täten asioita mitkä ei muilla alustoilla ole mahdollisia. Rajoitteet voisivat liittyä esimerkiksi sovellusten moniajoon tai tulevien puheluiden kaappauksiin ja niin edespäin.

Julkaisuprosesseissa merkittävin ero on jakelukanavien suorittama sovellusten esiarviointi. App Store ja Windows Phone Marketplace suorittavat sovelluksille arvioinnin ennen julkaisua. Kaikki näiden jakelukanavien sovellukset ovat Applen tai Microsoftin hyväksymiä. Google Playssa ei vastaavaa hyväksymiskäytäntöä ole. Arviointi tuo sovelluksen julkaisuun muutaman päivän viiveen. Esimerkki sovellus läpäisi arvioinnit ensimmäisellä yrityksellä, ja ongelmia ei koettu julkaisun suhteen missään jakelukanavassa.

Jakelukanavan asettamissa esivaatimuksissa julkaistavalle sovellukselle on havaittavissa pieniä eroja, mutta ei mitään niin merkittävää, mikä asettaisi kanavat selkeään

paremmuusjärjestykseen. Työn lopputuloksena todettiin että varsinainen haaste ei ole sovelluksen kehitys ja julkaisu vierailla ohjelmointikielillä ja eri jakelukanavissa, vaan jakelukanavissa erottuminen ja oman sovelluksen esille saaminen sovellusten valtavassa määrässä. Toinen jatkotutkimuskohde voisi olla oman sovelluksen latausmäärien kasvattaminen jakelukanavissa. Miten sovelluksen hinta vaikuttaa latausmääriin? Mitä keinoja sovellusten näkyvyyden lisäämiseen ja markkinointiin on yleisesti? Mitkä näistä keinoista ovat rajallisilla resursseilla toimivien pienten yritysten ja yksityisten kehittäjien saatavilla?

LÄHTEET

1. Yamakami, T., A Three-dimension Analysis of Driving Factors for Mobile Application Stores: Implications of Open Mobile Business Engineering, Workshops of International Conference on Advanced Information Networking and Applications, Biopolis, Singapore, 2012
2. The Windows Club -verkkosivu, Windows Phone overtakes RIM to become 3rd largest Mobile OS in Key 8 countries, saatavissa http://www.thewindowsclub.com/windows-phone-overtakes-rim-3rd-largest-mobile-os-key-8-countries, viitattu 6.10.12
3. Stevens, C., Appillionaires: Secrets from Developers Who Struck It Rich on the App Store, 3 edition, Wiley, USA, 2011
4. Apple Developer -verkkosivu, iOS Developer Program, saatavissa https://developer.apple.com/programs/ios/distribute.html, viitattu 5.9.2012
5. Google Play -verkkosivu, Transaction Fees, saatavissa https://support.google.com/googleplay/android-developer/bin/answer.py?hl=en&answer=112622&ctx=cb&src=cb&cbid=-lh2kreo5o7g6, viitattu 5.9.2012
6. Microsoft-verkkosivu, MSDN Windows Phone Help, saatavissa http://msdn.microsoft.com/en-us/library/windowsphone/help, viitattu 5.9.2012
7. Holzer, A., Ondrus, J., Mobile application market: A developer's perspective, Telematics and Informatics 28 (2011) 22–31, Elsevier, United Kingdom, 2010
8. McCann, T., The Art of the App Store: The Business of Apple Development, 1 edition, Wrox, USA, 2011
9. 148apps.bizz-verkkosivu, App Store Metrics, saatavissa http://148apps.biz/app-store-metrics, viitattu 6.9.2012
10. Meier, R., Professional Android 4 Application Development, 3 edition, Wrox, USA, 2012
11. Blandford, R., All About Windows Phone –verkkosivu, 100,000 apps published to Windows Phone Marketplace, saatavissa http://allaboutwindowsphone.com/news/item/14960_100000_apps_published_to_Windo.php , viitattu 6.9.2012

12. AppBrain-verkkosivu, Free vs. paid Android apps, saatavissa http://www.appbrain.com/stats/free-and-paid-android-applications, viitattu 6.9.2012
13. O´Dell, J, VentureBeat-verkkosivu, Windows Phone hits 100K apps, but who's getting the money?, saatavissa http://venturebeat.com/2012/06/05/100000-windows-phone-apps/, viitattu 6.9.2012
14. Yoon, Y. S., Yoo, J., Choi, M., Revenue Sharing is the Optimal Contractual Form for Emerging App Economy?, International Conference on ICT Convergence 2010, Jeju Island, Korea, 2010
15. Copeland, R., Telco App Stores - Friend or Foe?, International Conference on Intelligence in Next Generation Networks, Saksa, 2010
16. Kimbler, K., App Store Strategies for Service Providers, International Conference on Intelligence in Next Generation Networks, Saksa, 2010
17. Open handset alliance -verkkosivu, Industry Leaders Announce Open Platform for Mobile Devices, saatavissa http://www.openhandsetalliance.com/press_110507.html, viitattu 6.9.2012
18. Android open source project -verrkosivu, About the Android Open Source Project, saatavissa http://source.android.com, viitattu 6.9.2012
19. Eckel, B., Thinking in Java, 4 edition Prentice Hall, USA, 2006
20. Android Developers -verkkosivu, Application Fundamentals, saatavissa http://developer.android.com/guide/topics/fundamentals.html, viitattu 6.9.2012
21. Mark, D., Nutting, J., LaMarche, J., Beginning iOS 5 Development: Exploring the iOS SDK, 1 edition, Apress, USA 2011
22. Kochan, S., Programming in Objective-C, 4 edition, Addison-Wesley Professional USA, 2011
23. Apple Developer -verkkosivu, iOS App Programming Guide, saatavissa https://developer.apple.com/library/ios/#documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/AppArchitecture/AppArchitecture.html#//apple_ref/doc/uid/TP40007072-CH3-SW12, viitattu 6.9.2012
24. Apple Developer -verkkosivu, Developer Tools, saatavissa https://developer.apple.com/technologies/tools/features.html, viitattu 6.9.2012
25. Cameron, R., Pro Windows Phone 7 Development, 1 edition, Apress, USA 2011

26. Microsoft MSDN -verkkosivu, Application Platform Overview for Windows Phone, saatavissa http://msdn.microsoft.com/en-us/library/ff402531(v=vs.92).aspx, viitattu 6.9.2012
27. Sengupta, S., Microsoft Channel 9 -verkkosivu, , Windows Phone 7 Series Architecture Deep Dive, saatavissa http://channel9.msdn.com/Events/TechEd/NorthAmerica/2010/WPH313, viitattu 6.9.2012
28. Bako, J., Introduction to C# Joes 2 Pros, CreateSpace Independent Publishing Platform, USA 2010
29. Dalal, M., Ghoda, A., XAML Developer Reference, Microsoft Press, USA, 2011
30. Microsoft MSDN -verkkosivu, Silverlight Overview, saatavissa http://msdn.microsoft.com/en-us/library/bb404700(v=vs.95).aspx, viitattu 6.9.2012
31. Microsoft MSDN -verkkosivu, Creating Windows Phone Applications with Microsoft Visual Studio 2010 Express for Windows Phone, saatavissa http://msdn.microsoft.com/en-us/windowsphonetrainingcourse_yourfirstwp7applab_topic2, viitattu 6.9.2012
32. Microsoft MSDN -verkkosivu, Windows Phone SDK Tools, saatavissa http://msdn.microsoft.com/en-us/library/ff402523(v=vs.92).aspx, viitattu 6.9.2012
33. Android Developers -verkkosivu, Android Dev Guide, saatavissa http://developer.android.com/distribute/index.html, viitattu 6.9.2012
34. Apple Developer -verkkosivu, About Your First App Store Submission saatavissa https://developer.apple.com/library/ios/#documentation/ToolsLanguages/Conceptual/YourFirstAppStoreSubmission/AboutYourFirstAppStoreSubmission/AboutYourFirstAppStoreSubmission.html#//apple_ref/doc/uid/TP40011375], viitattu 6.9.2012
35. Apple Developer -verkkosivu, App Store Review Guidelines, saatavissa https://developer.apple.com/appstore/resources/approval/guidelines.html, viitattu 20.9.2012
36. Microsoft MSDN -verkkosivu, Application Certification Requirements for Windows Phone, saatavissa http://msdn.microsoft.com/en-us/library/hh184843(v=vs.92), viitattu 6.9.2012

37. Fielding, R., Representational State Transfer (REST), saatavissa http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm, viitattu 6.9.2012
38. Json.org-verkkosivu, Introducing JSON, saatavissa http://www.json.org/, viitattu 6.9.2012
39. Fowler, M., Patterns of Enterprise Application Architecture, 1 edition, Addison-Wesley Professional, USA, 2012
40. Gamma, E., Helm, R., Johnson, R., Vlissides, J., Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional, 1 edition,USA 1994
41. Google Play -verkkosivu, Developer Registration, saatavissa http://support.google.com/googleplay, viitattu 6.9.2012
42. Apple Developer -verkkosivu, Apple Developer Program Enrollment, saatavissa https://developer.apple.com/programs/start/standard/, viitattu 6.9.2012
43. Wooldridge, D., Schneider, M., The Business of iPhone and iPad App Development: Making and Marketing Apps that Succeed, 2 edition, Apress, USA, 2011
44. Apple Developer -verkkosivu, App Store Approval Process, saatavissa https://developer.apple.com/appstore/resources/approval/index.html, viitattu 6.9.2012
45. Google Play –verkkosivu, Google Play -kehittäjien ohjelmasäännöt, saatavissa http://play.google.com/about/developer-content-policy.html, viitattu 20.9.2012