Haaga-Helia
ammattikorkeakoulu Oy

Mobiiliohjelmoinnin tekniikoiden vertailu

Terttu Koskela

Opinnäytetyö
Tietojenkäsittelyn
koulutusohjelma
2017

Tiivistelmä

Haaga-Helia ammattikorkeakoulu Oy

Tekijä(t)
Terttu Koskela

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko

Mobiiliohjelmoinnin tekniikoiden vertailu

Sivu- ja liitesivumäärä
22 + 0

Opinnäytetyö suoritetaan selvitysopinnäytetyönä, jonka tavoitteena on mahdollistaa opinnäytetyön tekeminen niille, joilla työ- tai muiden kiireiden vuoksi ei ole tarvetta panostaa opinnäytetyöhön merkittävästi. Opinnäytetyö suoritetaan kirjallisuusselvityksenä hyödyntäen internetsivustoja, joissa keskitytään mobiiliohjelmointiin. Mobiiliohjelmoinnista löytyy kattavasti tietoa internetistä, sillä mobiiliohjelmointi on hyvin ajankohtainen ja kehittyvä tällä hetkellä.

Selvityksen kohteena ovat älypuhelinten käyttöjärjestelmät ja niiden ominaisuudet ja yleisyys. Markkinoilla myytävien älypuhelinten käyttöjärjestelmistä yleisimmäksi on noussut Android-käyttöjärjestelmä. Applen iOs-käyttöjärjestelmän älypuhelimilla on myös merkittävä osa markkinoista. Muiden käyttöjärjestelmien, kuten esimerkiksi Window-mobiilikäyttöjärjestelmän, markkinaosuudet ovat pienentyneet merkittävästi.

Opinnäytetyön tavoitteena on koota eri lähteistä yleisimmät mobiiliohjelmoinnin tekniikat. Mobiiliohjelmoinnin tekniikoista käsitellään natiivi-, HTML-, hybridi- ja monialusta-ohjelmointia.

Selvitys sisältää myös katsauksen mobiilisovellusten vaatimuksiin ja mahdollisuuksiin. Mobiilisovellusten ohjelmoinnin eri tekniikoiden hyviä ja huonoja puolia vertaillaan. Kirjallisuusselvityksen avulla poimitaan keskeinen sisältö aiheesta ja luodaan niistä yhteenveto.

Asiasanat

mobiiliohjelmointi, natiivisovellus, HTML-sivusto, hybridisovellus, monialustasovellus, Android, iOs, Windows 10 Mobile

Sisällys

1 Johdanto

Opinnäytetyön tavoitteena on koota eri lähteistä mobiiliohjelmoinnin keskeisimmät tekniikat ja tarkastella niitä. Näistä ominaisuuksista kerrotaan menemättä syvemmälle tekniikoissa, vaan tarkoituksena on saada yleiskuva mobiiliohjelmoinnista, jonka avulla omaa selvitystä voidaan jatkaa.

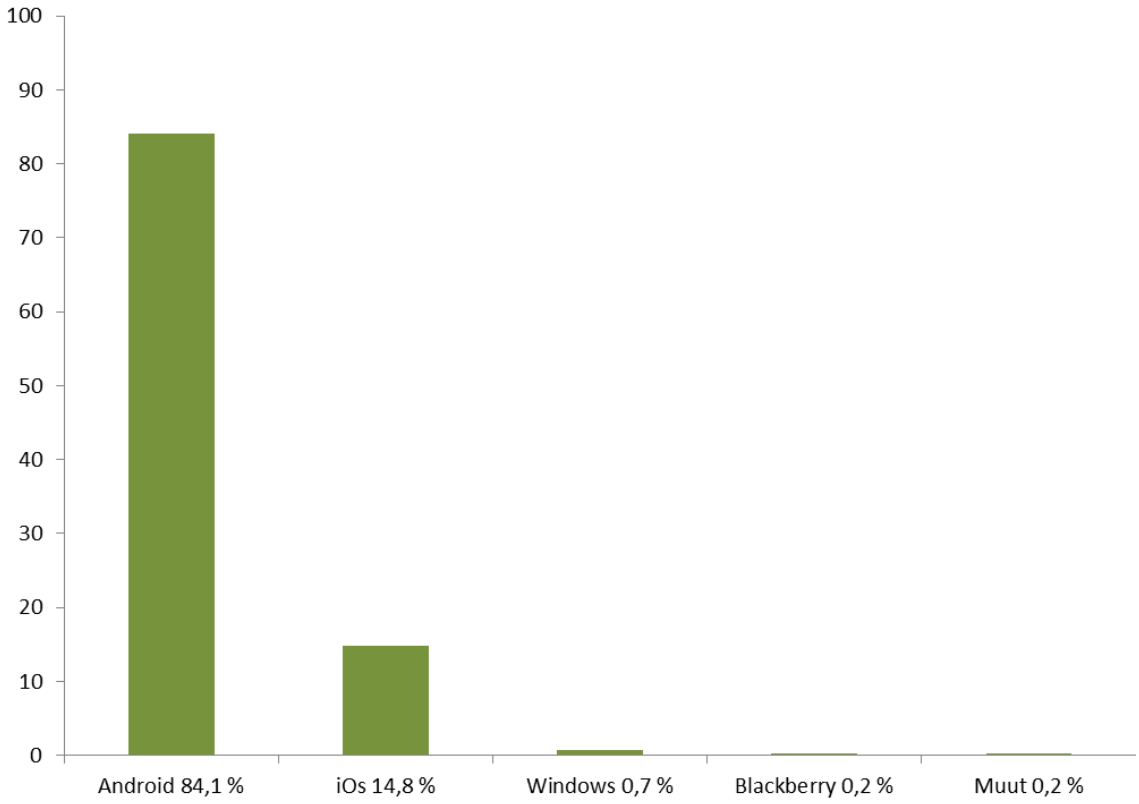
Mobiilikäyttöjärjestelmistä Googlen Android ja Applen iOs ovat älypuhelinmarkkinoiden johtavat käyttöjärjestelmät (Konttinen 2016). Älypuhelinten käyttöjärjestelmistä Androidilla oli vuoden 2016 ensimmäisellä neljänneksellä yli 80 %, iOs:llä lähes 15 % ja Microsoftin Windowsilla vajaan yhden prosentin markkinaosuus (BGR 2016). Android-käyttöliittymien ulkoasut vaihtelevat hyvin paljon eri laitevalmistajien välillä. iOs:ssa parasta on että sen käyttöjärjestelmä on yksinkertainen. iOs-käyttöjärjestelmän muokkaamismahdollisuudet ovat vähäiset. Windows 10 Mobile, Microsoftin uusin mobiilikäyttöjärjestelmä, on erilainen verrattuna Androidin tai iOs:n käyttöliittymiin. (Konttinen 2016.) Androidilla ja iOS:lla on paljon käyttäjiä ja tämän vuoksi myös kehittäjiä Android- ja iOs-alustoille on paljon. Panostus käyttöjärjestelmiin vaikuttaa käyttöjärjestelmien sovelluksien määrään ja laatuun. (Konttinen 2016.)

Nykyisin internetsivustoja selaillaan hyvin paljon mobiililaitteiden avulla ja sivujen kehittämisessä onkin tällä hetkellä tärkeää suunnitella sivuista responsiivisia eli että sivusto toimii sekä älypuhelimella, tabletilla että tietokoneella. Kun sivuistolla on paljon erilaisia toimintoja ja näkymiä, niin silloin responsiivinen sivusto ei riitä tekemään sivuston käytöstä riittävän miellyttävää kaikilla mahdollisilla internetlaitteilla. Kun responsiivinen sivusto tunnu riittävältä niin silloin parempi vaihtoehto on toteuttaa responsiivisen sivuston sijaan erillinen mobiilisovellus. (Vuorinen 2014.)

Natiivisovellusta tehdessä joudutaan ohjelmoimaan eri laitealustoille erikseen, koska laitealustat eivät ole keskenään yhteensopivia ja natiivikoodia ei voida käyttää kuin kunkin alustan omalle laitealustalle. Android-sovelluksien ohjelmointiin käytetään kehitystyökaluja, jotka Google tarjoaa Android-alustoille. iOS-natiivisovelluksien ohjelmointiin käytetään Applen Xcode kehitystyökaluja. HTML-sivustot toimivat kaikilla eri alustojen laitteilla. HTML-sivustojen kehittämiseen käytetään HTML-, CSS- ja JavaScript-tekniikoita. HTML-sivustolla ei pystytä natiivisovelluksen tavoin hyödyntämään täysin mobiililaitteen kaikkia toimintoja. Hybridi mobiilisovellus ohjelmoidaan samoja HTML-tekniikoita käyttäen kuin HTML-sivusto, mutta hybridin mobiilisovelluksen koodi paketoidaan hybridisovelluskehyksellä. (Vuorinen 2014.) Natiivisovellukset voidaan toteuttaa myös monialustasovelluskehyksillä. Monialustaohjelmoinnissa merkittävä osa ohjelmointikoodista voidaan hyväksikäyttää eri laitealustoille. (Kaasalainen 2015.)

2 Mobiilisovelluskehitys

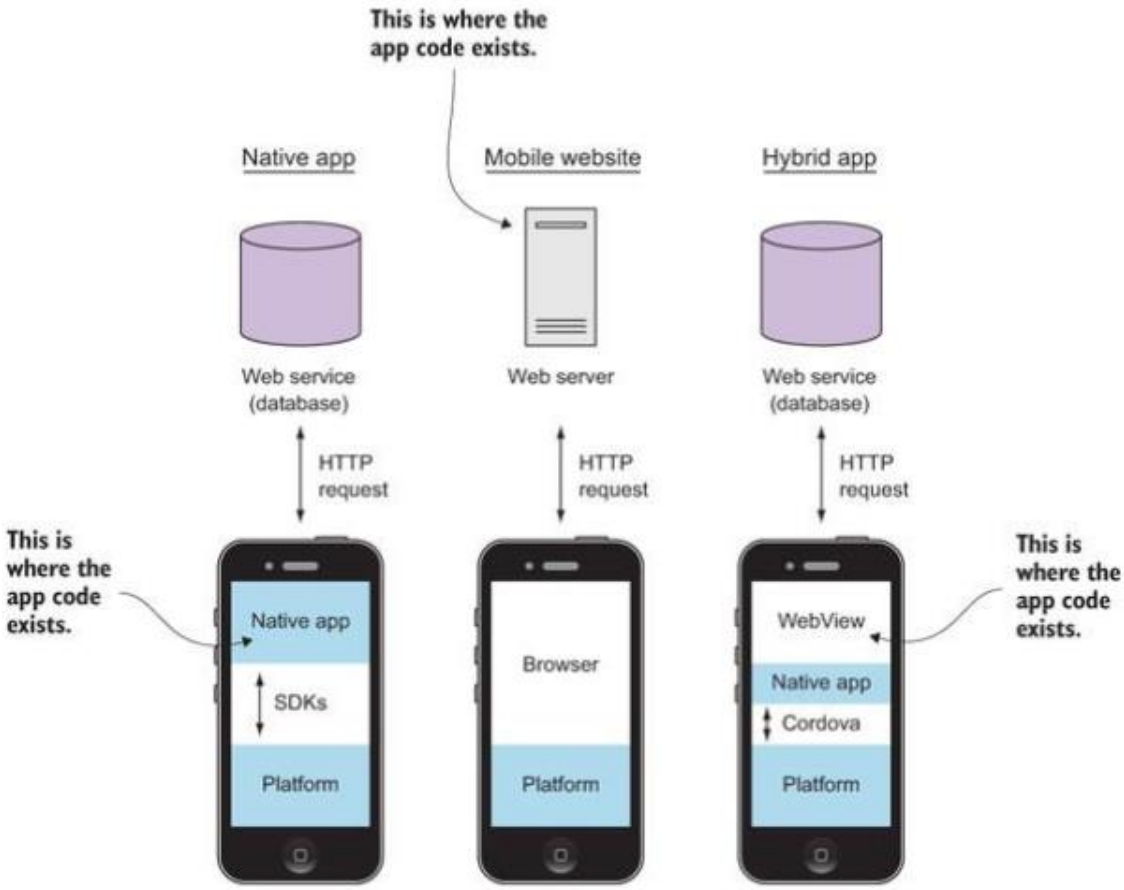
Mobiilisovelluksia kehitettäessä on tärkeää tietää älypuhelinten käyttöjärjestelmistä. Tällä hetkellä älypuhelinkäyttöjärjestelmistä eniten merkitseviä ovat Googlen Android, Applen iOS ja Microsoftin Windows 10 Mobile. (Konttinen 2016). Android ja iOS hallitsevat maailmanlaajuisesti käyttäjämäärissä (BGR 2016). Kuva 1. on esitetty älypuhelinten maailmanlaajuiset markkinaosuudet vuoden 2016 ensimmäiseltä neljännekseltä.



Kuva 1. Älypuhelinalustojen maailmanlaajuiset markkinaosuudet vuoden 2016 ensimmäiseltä neljännekseltä (BGR 2016).

Android-älypuhelimet ovat saavuttaneet hurjan markkinaosuuden älypuhelinmarkkinoista (BGR 2016). Microsoftin uusin mobiilikäyttöjärjestelmä, Windows 10 Mobile-käyttöjärjestelmä, on pyrkinyt pysymään älypuhelinmarkkinoilla, mutta ponnisteluista huolimatta Windows-käyttöjärjestelmä on menettänyt markkinaosuuttaan vuosi vuodelta (Konttinen 2016).

Tyypillisesti mobiiliohjelmointi jaetaan kolmeen eri vaihtoehtoon, natiiviohjelmointiin, HTML-sivuston ohjelmointiin tai hybridiohjelmointiin. Kuvassa 2 on esitetty natiivisovelluksen, HTML-sivuston ja hybridisovelluksen mobiilisovellusarkkitehtuurit (Nahkala 2016.) Natiivisovellukset voidaan toteuttaa myös monialustasovelluskehyksillä ja tällöin puhutaan monialustaohjelmoinnista (Kaasalainen 2015).



Kuva 2. Natiivisovelluksen, HTML-sivuston ja hybridisovelluksen mobiilisovellusarkkitehtuurit (Nahkala 2016).

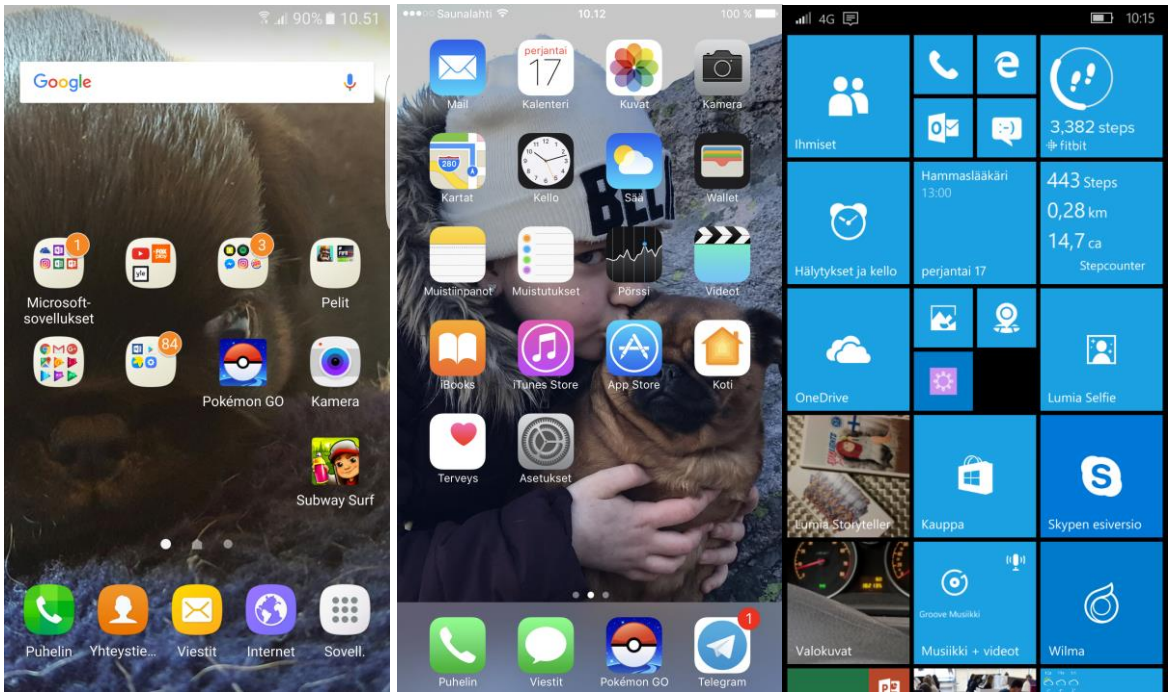
Mobiilisovelluksissa käsiteltävän tiedon pitää pystyä tallessa sovelluksen käytön aikana ja myös silloin, kun käyttäjä ei käytä sovellusta. Sovellusta käytettäessä tieto pysyy tallessa käyttömuistissa. Käyttäjän lopettaessa sovelluksen käytön, käyttömuistissa oleva tieto menetetään sovelluksen sammuessa. Käyttömuistissa olevan tiedon täytyy tallentua, kun sovellus suljetaan. Mobiililaitteissa on kaksi tiedon säilytystapaa. Tieto voidaan tallentaa puhelimen muistiin tai internetin palvelintietokoneelle. Mobiililaitteen muistiin voi laitealustasta riippuen tallentaa tietoa monella eri tavalla. Myös tiedontallennusformaatti muistiin tallentamisessa riippuu laitealustasta. Internetpalvelin on oikea valinta tiedon säilytykseen

silloin, kun tarvitsee käyttää jaettua sisältöä, jota päivitetään muualtakin kuin vain mobiililaitteesta. (Toivonen 2013.)

Internetyhteys on pakollinen sovellukselle, jonka pitää päästä palvelimella sijaitsevaan tietovarastoon. Ilman internetyhteyttä ei pystytä pitämään puhelimen tietoja ja palvelimen tietoja ajan tasalla. (Toivonen 2013.)

2.1 Android, iOs ja Windows käyttöjärjestelmät

Osalle ihmisistä käyttöjärjestelmän valinta mobiililaitetta ostettaessa on täysin selvää, mutta päätöksen tekoon on syytä käyttää aikaa jos ei ole täysin varma minkä käyttöjärjestelmän mobiililaitteelleen haluaa. Androidilla käyttöliittymät vaihtelevat paljon eri valmistajien väleillä, joten kannattaa tutustua myös eri laitevalmistajien käyttöliittymiin. (Konttinen 2016.) Androidin mobiililaitteita valmistavat Samsung, LG, HTC, Huawei, Acer, Sony, Motorola, OnePlus ja ZTE. Myös muidenkin valmistajien Android-mobiililaitteita on markkinoilla. (Phandroid 2016.) iOS-käyttöjärjestelmän parhaita puolia on sen varma toiminta ja käyttöjärjestelmän käyttäminen on miellyttävää. iOS on yksinkertainen käyttöjärjestelmä ja iOs-käyttöjärjestelmien mobiililaitteiden käyttämisen oppii helposti. iOs-käyttöjärjestelmästä ei voi muokata persoonallista toisin kuten Androidista voi. Sovellukset ovat erittäin tärkeitä älypuhelimissa ja iOs:n sovellusvalikoima on hyvin monipuolinen ja sovellukset ovat laadukkaita. iOS on ehdottomasti oikea valinta käyttöjärjestelmäksi, kun haluaa varmatoimintaisen mobiililaitteen ja on myös valmis maksamaan siitä enemmän kuin vastaavasta Android-mobiililaitteesta. Jos taas haluaa käyttöjärjestelmän, jonka voi muokata omannäköiseksi, niin silloin Android on mahdollisesti oikea valinta. Windows 10 Phone erottautuu aloitusnäytön tapahtumaruutujen rakenteessa. Windows 10 Phone:lla on erilainen ja tyylikäs käyttöliittymällä Androidiin ja iOs:iin verrattuna. Iso haitta Windows 10 Phonella on sen suppea ja huonolaatuinen sovellusvalikoima. (Konttinen 2016.) Kuvassa 3. on esitetty Android, iOs ja Windows käyttöjärjestelmien aloitusnäytöt.



Kuva 3. Android, iOs ja Windows käyttöjärjestelmien aloitusnäytöt. Vasemmalla Marshmallow Android 6.0.1 käyttöjärjestelmän ruudunkaappaus Samsung Galagy S6 EDGE älypuhelimesta, keskellä iOs 10.2.1 käyttöjärjestelmän ruudunkaappaus Applen iPhone 7 Plus älypuhelimesta, oikealla Windows 10 Mobile käyttöjärjestelmän ruudunkaappaus Nokia Lumia 830 älypuhelimesta.

Android-käyttöjärjestelmän kehitys aloitti vuonna 2003 Android Inc. toimesta. Android Inc. markkinaideana oli kehittää digitaalisten kameroiden ohjelmistoalusta. Kehityksen suuntaus muutui kuitenkin mobiilikäyttöjärjestelmiin markkinoiden muuttumisen vuoksi. Android-käyttöjärjestelmää Android Inc. kehitti muutaman vuoden ajan. Google osti yrityksen vuonna 2005. Google lähti mukaan mobiilikäyttöjärjestelmämarkkinoille vuonna 2007. Silloin julkaistiin Android-käyttöjärjestelmän ensimmäinen versio. Android-käyttöjärjestelmän kehittyminen perustuu pitkälti avoimeen lähdekoodiin käyttöön, koska kaikkien on mahdollista tutustua avoimeen lähdekoodiin ja muokata koodia. Android-ohjelmistoalusta pohjautuu Linux-ytimeen. Lähdekoodit on julkaistu avoimella Apache-lisenssillä (Apache License) ja käyttöjärjestelmä on avoin ja maksuton. (Pääkkönen 2014.)

iOs käyttöjärjestemä on Applen tuoteperheen laitteissa, kuten iPhone-älypuhelimissa, iPad-tableteissa, ja iPod-musiikinsoittimissa, käytetty käyttöjärjestelmä (Apple 2016). Käyttöjärjestelmän nimi oli ensin iPhone OS, mutta nimi vaihtui iOs:ksi, kun Apple toi markkinoille iPadin vuonna 2010 (Sani 2012). Ensimmäinen iOs-pohjainen älypuhelin julkaistiin vuonna 2007. iOS-käyttöjärjestelmä pohjautuu samaan Unix-ytimeen, kuin Applen Mac-työpöytätietokoneiden käyttöjärjestelmä Mac OS X. (Pääkkönen 2014.)

Microsoftin uusimman käyttöjärjestelmän nimi on Windows 10 Mobile (Windows Central 2017). Microsoftin aiempien käyttöjärjestelmien nimiä olivat Windows Phone ja Windows Mobile. Käyttöjärjestelmän ensimmäinen versio julkaistiin vuonna 2010. (Pääkkönen 2014.) Microsoft on vuosi vuodelta menettänyt osuuttaan älypuhelinmarkkinoilta ja on luovuttamassa ja lopettamassa yrittämisen olla merkittävä käyttöjärjestelmä älypuhelinmarkkinoilla. Microsoft keskittyy jatkossa yritysmarkkinoille. Kuitenkaan yhtiö ei ole vielä maininnut varsinaisesta vetäytymisestä kuluttajamarkkinoilta. Microsoft on lopettamassa laitteiden valmistuksen omissa yksiköissään. (Hynninen 2016.) Keväällä 2016 Nokia teki sopimuksen HMD Global –nimisen yhtiön kanssa. HMD Global alkaa valmistamaan älypuhelimia, matkapuhelimia ja tabletteja Nokian tavaramerkkiä käyttäen. Alkuvuodesta 2017 ilmestyvistä Nokia-merkkisissä puhelimissa käyttöjärjestelmänä on Android eikä enää Windows, kuten aiemmin. (Järvinen 2016.)

2.2 Natiivi mobiilisovellus

Kun sovellus tuotetaan natiivisti, joudutaan ohjelmakoodi ohjelmoimaan eri mobiilikäyttöjärjestelmille erikseen, koska käyttöjärjestelmät vaativat eri ohjelmointikielillä toteutettua ohjelmointia. Android-sovellukset ohjelmoidaan Java-ohjelmointikielellä Googlen tarjoamilla kehitystyökaluilla, objektiivisella C-ohjelmointikielellä (Objective-C) ohjelmoidaan iOs-sovellukset Applen iOs SDK (Software Development Kit) sekä Applen Xcode -kehitystyökalujen avulla. (Vuorinen 2014.) Windows 10 Mobilen natiiviohjelmoinnissa kehitysalustana on Visual Studio ja ohjelmointikielenä C# (C sharp) tai C++ ohjelmointikielet. (Lehto 2016.) Natiivisovelluskehitykseen liittyy paljon myös asioita, jotka pitää ottaa huomioon sovellusta kehitettäessä, kuten esimerkiksi tietyt käyttöehdot ja säännöt, joihin on sitouduttava. Androidin sovelluskauppa Google Play vaatii Google Play -kehittäjätilin, jotta voi sovelluksia voi julkaista ja hallinnoida Goggle Playssä. Ennen kuin iOs-natiivisovellukset julkaistaan, suoritettaan sovelluksen tarkistusprosessi ja vasta hyväksytyn tarkistusprosessin jälkeen sovellukset päästetään Applen sovelluskauppaan App Storeen (Vuorinen 2014).

Android-natiivisovelluksia ohjelmoidaan Java- ja XML (Extensible Markup Language) -ohjelmointikielillä. Javalla ohjelmoidaan toiminnot ja XML:llä ohjelmoidaan käyttöliittymän ulkoasu (Rantakeisu 2016). Android-sovelluskehityksessä tarvittava työkalu on Android SDK (Software Development Kit) ja se sisältää kaikki Android-sovelluksen luontiin tarvittavat kirjastot. (Rantakeisu 2016.) Android Studion voi ladata Android Developersin verkkosivuilta https://developer.android.com/studio/index.html. Android Studio sisältää Android

SDK:n. (Mullis 2016.) Yleisimmin Android SDK:ta käytetään ohjelmistosovellus kehitystyökalu Eclipsen kanssa. Android SDK asennetaan osaksi Eclipse kehitysympäristöä. Eclipsen voi ladata Eclipsen verkkosivuilta http://www.eclipse.org/downloads/. (Rantakeisu 2016.)

iOs-natiivisovelluksia ohjelmoidaan objektiivisella C-ohjelmointikielellä eli Objective-C kielellä. Objective-C on C-kielen laajennos. Objective-C kielessä on olio-pohjainen ajattelu toisin kuin perinteisessä C-ohjelmointikielessä ei ole olio-pohjaista ajattelua. Objective-C kielessä on toimintoja olioiden kuvaamiseen ja viestien välittämiseen oliolta toiselle. Esimerkkinä voisi mainita että tarvitaan päivämääränkäsittelijäolio, jos kehittäjä halua tehdä merkkijonosta päivämäärän. Päivämääränkäsittelijäoliolle annetaan kalenteriolio, joka edustaa gregoriaanista kalenteria, islamilaista kalenteria tai jotakin muuta käytössä olevaa kalenteria. (Sani 2012.)

iOs-natiivisovelluksen kehitysväline on Applen Xcode. Xcode sisältää koodieditorin, debuggerin, versionhallinnan ja visuaalisen rakennusvälineen käyttöliittymille. Tärkeintä Xcodessa on, että ainoastaan sillä voidaan paketoida sovellus App Store -jakelua varten. Kaikki muut toiminnot voi tehdä muillakin välineillä. Suurin osa Xcoden toiminnoista löytyy yhdestä ikkunasta. Erillinen sovellus on simulaattori, iOs Simulator. Simulaattorin avulla sovelluskehittäjä voi testata sovelluksia iOS ympäristössä asentamatta oikeaa iOS-laitetta. Olennainen lisätyökalu on myös Instruments. Se analysoi sovelluksen suorituskykyä ja varoittaa mahdollisista pullonkauloista. Xcoden voi ladata ilmaiseksi Mac App Storesta ja se on käytettävissä vain Applen Mac-tietokoneissa. (Sani 2012.) Käytössä on myös maksullinen macincloud.com-pilvipalvelu, jossa Mac-tietokone on maksullisesti käytettävissä internetin kautta (Ching 2016). Apple esitteli Swift-ohjelmointikielen vuonna 2014. Apple yrittää saada Swift-ohjelmointikielestä iOs-alustojen ohjelmointikielen korvaten Objective-C-ohjelmointikielen, koska Swift-ohjelmointikielellä sovellusten kehittäminen iOs-käyttöjärjestelmille on helpompaa, selkeämpää ja tehokkaampaa kuin Objective-C-ohjelmointikielellä. (Hynninen 2014.) iOs:n kanssa ei tarvitse ajatella ohjelmointia laitteiston kannalta juurikaan, koska Applella pieni laitekanta. (Toivonen 2013).

Windows 10 Mobilen natiiviohjelmoinnissa kehitysalustana käytetään Microsoftin Visual Studiota ja ohjelmointikielenä C# tai C++ ohjelmointikieliä. Ilmainen Community-versio Visual Studiosta on ladattavissa osoitteesta https://www.visualstudio.com/downloads/. Visual Studioon on hankittavissa myös maksulliset lisenssit. (Lehto 2016.)

Mobiilialustojen natiivisovellusten ohjelmointikielet ja kehitystyökalut on esitetty yhteenvetona taulukossa 1.

Taulukko 1. Mobiilialustojen natiivisovellusten ohjelmointikielet ja kehitystyökalut (Andström 2016).

Käyttöjärjestelmä	Ohjelmointikieli	Kehitysalusta
Android	Java	Android SDK, Android Studio, Eclipse
iOs	Objective-C / Swift	Xcode
Windows 10 Mobile	C# / C++	Microsoft Visual Studio

2.3 HTML-sivusto

Internetin mobiilikäytön lisääntyminen on luonut tarpeen suunnitella HTML-sivustoista responsiivisia. Responsiivisen HTML-sivuston käyttäminen on helppoa päätelaitteesta riippumatta, koska responsiivinen HTML-sivusto tunnistaa käytettävän laitteen ja mukauttaa ulkoasun, sisällön ja toiminnallisuudet automaattisesti käytössä olevan laitteen mukaan. Sama HTML-sivusto toimii kaikilla käyttöjärjestelmillä ja kaikilla laitteilla. Internetselaimien kehittyminen on mahdollistanut HTML-sivustojen kehittämisen erityisesti mobiilikäyttöön optimoiduksi. (Vuorinen 2014.)

HTML (Hypertext Markup Language) on kuvauskieli HTML-sivustojen luomiseen (Ilves, Vihavainen 2015). HTML-kuvauskielen uusin versio on HTML5 (Hartto 2014). HTML-kielellä kuvataan HTML-sivun sisältö jarakenne. Sisältöön kuuluvat sivuston sisältämät tekstit ja kuvat. HTML-sivujen rakenne kuvataan elementeillä. Sisäkkäin ja peräkkäin olevat elementit määrittelevät HTML-sivujen rakenteen. (Ilves, Vihavainen 2015.)

Elementit täytyy erotella merkeillä. Kaikki elementit, lukuun ottamatta tyhjiä elementtejä, avataan pienempi kuin merkillä ja suljetaan suurempi kuin merkillä. Esimerkiksi juuriele-

mentillä <html> aloitetaan HTML-dokumentti ja juurielementillä </html> lopetetaan HTML-dokumentti. Elementtien sisälle voi laittaa muita elementtejä. (Ilves, Vihavainen 2015.) HTML-sivuston perusrakenteessa juurielementillä <html> aloitetaan HTML-dokumentti. Juurielementin jälkeen lisätään dokumentin metatietoa sisältävä elementti <head>. Dokumentin metatietoa sisältävän elementin sisälle lisätään metatietoelementti <meta> ja se täytyy sulkea </meta>. Dokumentin metatietoa sisältävän elementin sisälle lisätään myös dokumentin ulkoinen otsikkoelementti <title> ja se täytyy myös sulkea </title>. Metatietoelementin ja otsikkoelementtien jälkeen tulee myös dokumentin metatietoa sisältävä elementti sulkea </head>. Kun dokumentin metatietoa sisältävä elementti on suljettu niin sen jälkeen voidaan avata koko elementin runko elementillä <body>. Koko elementin runko suljetaan elementillä </body>. Lopuksi juurielementillä </html> lopetetaan HTML-dokumentti. HTML-dokumentissa elementit on suljettava samassa järjestyksessä kuin ne on avattu. Tyhjille elementeille, esimerkiksi rivinvaihtoelementille
, ei tarvitse kirjoittaa lopetusta. Valinnaisesti voi käyttää /-merkkiä, jolloin rivinvaihtoelementin tulisi muotoon
. (Ilves, Vihavainen 2015.)

CSS (Cascading Style Sheets) on kieli HTML-sivuston tyylin määrittelyyn. CSS-tyylitiedostoissa sijaitsevat HTML-sivuston tyylimäärittelyt ja niillä määritellään miltä HTML-sivun elementit näyttävät. HTML-sivustolla voi olla vain yksi tyylitiedosto tai niitä voi olla myös useita useita. Tyylimäärittelyt voi kirjoittaa myös HTML-dokumenttiin, mutta selkeyden vuoksi tyylimäärittelyt on parempi pitää erillisenä CSS-tiedostoissa. Tyylitiedoston HTML-dokumentin saa käyttöön, kun HTML-dokumenttiin lisätään tyylitiedoston lataava elementti <link> dokumentin metatietoa sisältävän elementin <head> ja metatietoelementin <meta> sisälle (Ilves, Vihavainen 2015.)

DOM (Document Object Model) eli dokumenttiobjektimalli on ohjelmointirajapinta dynaamisten HTML-sivustojen tuottamiseen. DOM määrittelee, kuinka HTML- tai XML-dokumenttien sisältämät elementit välittävät tietoa toistensa kanssa. (Luoto 2008.) DOM rakentaa hierarkkiseen dokumenttipuun jokaisesta HTML-dokumentissa esiintyvästä elementistä tehden niistä joko solmuja tai lehtiä (Pyykkönen 2015). DOM-puun solmuja ovat oksat ja lehtiä ovat solmut, joista ei lähde oksia. (Ilves, Vihavainen 2015.) JavaScriptillä pääsee solmuihin, lehtiin ja niiden välisiin suhteisiin käsiksi. (Pyykkönen 2015). JavaScriptillä voidaan muuttaa suoraan DOM-puun sisältöä. (Luoto 2008). JavaScriptin getElementsByTagName-attribuutin kutsulla saadaan listan kaikista HTML-sivuston elementeistä (Ilves, Vihavainen 2015).

JavaScript-ohjelmointikielellä lisätään dynaaminen toiminta HTML-sivustolle. JavaScriptin ohjelmakoodia suoritetaan komento kerrallaan, ylhäältä alas, vasemmalta oikealle. Ja-

vaScript-lähdekoodit kannattaa erottaa HTML-dokumentista omaksi tiedostokseen aivan kuten CSS-tyylimäärittelyt. JavaScript-tiedoston pääte on .js ja JavaScript-tiedoston viitataan HTML-dokumentissa skriptielementillä script. Elementin script attribuuttilla src kerrotaan lähdekooditiedoston sijainti. (Ilves, Vihavainen 2015.)

JavaScript-ohjelmointikieli toimii vain selaimessa. JavaScript pystyy antamaan käyttäjälle suoraa palautetta ilman turhia lataustaukoja toisin kuin palvelinpuolen ohjelmat joutuvat keskustelemaan palvelimen kanssa ja lataamaan tietoja. JavaScript mahdollistaa HTML-sivuston älykkään ja vuorovaikutteisen toiminnan käyttäjän kanssa. Yhdessä CSS-tyylitiedoston kanssa voi JavaScriptillä muokata myös HTML-sivuston ulkonäköä. (Pyykkönen 2015.)

jQuery on tällä hetkellä yleisimpiä JavaScript-kirjastoja. Google, Dell, WordPress sekä monet muut suuret yritykset käyttävät sitä. (Sovelto 2016.) jQuery on avoimen lähdekoodin JavaScript-kirjasto. Sen ensimmäinen versio julkaistiin vuonna 2006. (Pääkkönen 2014.) jQueryn avulla on mahdollista yksinkertaistaa HTML-dokumentin ja tapahtumien käsittelyä, animointia, DOM-puun muokkaamista ja tehdä sivuille dynaamisia käyttöliittymäkomponentteja (Sovelto 2016). jQueryllä on helppo luoda HTML-sivuston ulkoasua JavaScript-komennoilla. Kirjaston avulla mahdollista käyttää HTML-sivustoilla yksinkertaisia kutsuja, jotka ilman kirjaston käyttöä vaatisivat useita rivejä JavaScript-koodin komentoja. HTML-sivustolla voi olla esimerkiksi näppäin, jota painamalla voidaan toteuttaa haluttuja toimintoja. Monimutkaisten asioiden helpompi käyttäminen HTML-sivustoilla on mahdollistaa JQueryn avulla, kuten esimerkiksi AJAX:in (Asynchronous JavaScript And XML). (Pääkkönen 2014). AJAX on tekniikka, jonka avulla voidaan siirtää tietoa selaimen ja palvelimen välillä ilman, että koko www-sivua täytyy ladata uudelleen (Ohjelmistoputka 2008).

jQueryn muutamalla koodirivillä saa aikaan isoja muutoksia. Esimerkkinä voisi mainita että sivun taustavärin muuttaminen vie vain yhden koodirivin jQueryn avulla. Testaaminen ja korjaaminen ovat helpompaa ohjelmointikoodin vähyyden vuoksi. jQueryn metodit on tehty yhteensopiviksi useimpien selainten kesken. Metodit toimivat samalla tavalla useimmissa selaimissa ja myös tämä helpottaa erittäin paljon kehittämistä ja testaamista. (Nahkala 2016.)

HTML-sivuston kävijä lataa jQueryn koneelleen. jQuerystä on tarjolla eri versiota, kuten esimerkiksi pakattu ja ei-pakattu versio. Suorituskyky on alhaisempi pakatussa versiossa. Pakatun version koko on huomattavasti pienempi kuin ei-pakatun version ja pienemmän koon vuoksi pakattu versio on mahdollisesti parempi valinta. jQuerya on saatavilla myös

CDN-versio (Content Delivery Network). Tällöin tiedosto haetaan CDN:stä eikä tiedosto sijaitse omalla palvelimella. Ohjelmointikoodi on muistettava pitää ajan tasalla CDN-versiossa, koska sivusto voi mahdollisesti hajota uuden version ilmestyessä. (Pyykkönen 2015.)

Ember.js on avoimen lähdekoodin JavaScript-sovelluskehys. Ember.js on MVC (Model-View-Controller) -arkkitehtuurimallin mukaan jaettu osiin. (Hartto 2014.) MVC-arkkitehtuurissa sovelluksen toiminnot on jaettu mallille, näkymälle ja käsittelijälle (Portimo 2014). Mallikerros on sovelluksen tietorakennekerros, näkymäkerros on sovelluksen ulospäin näkyvä kerros ja käsittelijä- eli ohjainkerros sisältää sovelluksen suoritusrakenteen. Kun Ember.js-sovelluskehyksen MVC-arkkitehtuuria verrataan muihin MVC-arkkitehtuuriin pohjautuviin sovelluskehyksiin, kuten esimerkiksi Ruby on Rails -sovelluskehykseen, Ember.js ei sisällä täysin samaa rakennetta vaan osa MVC-mallista on toteutettu monissa eri kerroksissa. Ember.js-sovelluskehyksellä on paljon avoimen lähdekoodin kehittäjiä ja käyttäjiä. Mukana on myös Yahoo, joka käyttää projekteissaan Ember.js -sovelluskehystä. Ideologisella tasolla Ember.js eroaa monista muista JavaScript-sovelluskehyksistä painottaessaan käytäntöä eikä konfiguraatiota. (Hartto 2014.)

Angular.js on avoimen lähdekoodin JavaScript-ohjelmistokehys, jota Google ylläpitämä. (Aucor 2016). Angular.js-työkalujen avulla voi rakentaa oman käyttötarpeensa mukaisen sovelluskehyksen (Hartto 2014). Angular.js on mahdollisesti oikea sovelluskehys yhden HTML-sivun varaan rakennetun SPA-arkkitehtuurin (Single-Page-Application) HTML-sivustolle. Angular.js tarjoaa sovelluskehittäjille paljon ohjelmointikäytäntöjä, kuten TDD (Test Driven Development) eli testivetoinen kehitys ja MVC-arkkitehtuuri. (Aucor 2016.) Angular.js muokkaa DOM-puuta suoraan. DOM-puun muokkaaminen suoraan on raskasta. Angular.js-sovelluskehys tarkastaa DOM-puun tiedot vaikka siihen ei oli edes tehty muutoksia. (Hamppula 2015.)

React.js on uusimpia JavaScript-sovelluskehyksiä. Erona edellä esiteltyihin Angular.js- ja Ember.js-sovelluskehyksiin on React.js-sovelluskehyksen MVC -arkkitehtuuri. React.js toteuttaa MVC-arkkitehtuurista vain näkymäkerroksen. Sivuston toiminnallisuus toteutetaan komponenttien avulla. Selaimen näkymät toteutetaan React.js-kehyksessä komponenttipohjaisen arkkitehtuurin mukaan. React.js tukee keskeisesti selaimen näkymien toteuttamista ja näkymät perustuvat niin sanottuun virtuaaliseen HTML-dokumentinobjektimallin (DOM) rakenteeseen. React.js-kehyksen käyttäminen käyttöliittymien toteutuksessa toimii erittäin hyvin. (Hamppula 2015.)

Backbone.js on vanhimpia MVC-mallin toteuttavia JavaScript-sovelluskehyksiä, mutta sen käyttö vähenee vuosi vuodelta. Lukuisia muitakin JavaScript-sovelluskehyksiä on mahdollista valita HTML-sivuston rakentamiseen ja kannattaa tutustua laajasti eri sovelluskehyksiin jos suunnittelee HTML-sivuston rakentamista. (Hartto 2014).

2.4 Hybridi mobiilisovellus

Hybridi mobiilisovellus tarkoittaa HTML-tekniikoilla toteutettua HTML-sovellusta, joka paketoidaan kullekin tuettavalle mobiilikäyttöjärjestelmälle niin että mobiililaitteen toiminnot saadaan sovelluksen käyttöön (Vuorinen 2014). Hybridi-sovelluksien kehitykseen käytetään perinteisiä internetsivujen tekoon käytettyjä web-teknologioita, kuten HTML-, CSS- ja JavaScript-kieliä. Web-teknologioiden lisäksi tarvitaan ohjelmistosovelluskehys. (Andström 2016.)

Apache Cordova on ilmainen ja avoimen lähdekoodin ohjelmistokehys, jolla voidaan toteuttaa hybridimobiilisovelluksia. Apache Cordovan avulla HTML-sovellus paketoidaan natiivisovellukseksi. Apache Cordovalla tehdyt sovelluksen voi lisätä mobiilikäyttöjärjestelmien sovelluskauppoihin aivan kuten natiivisovellukset. Tällä hetkellä Apache Cordova on suosituin ohjelmistokehys hybridimobiiliohjelmoinnissa. Cordovalla pystytään tekemään HTML-, Javascript- ja CSS-kielellä ohjelmia älypuhelimille, jotka toimivat kaikilla ohjelmistokehyksen tukemilla mobiilikäyttöjärjestelmillä. (Vuorinen 2014.)

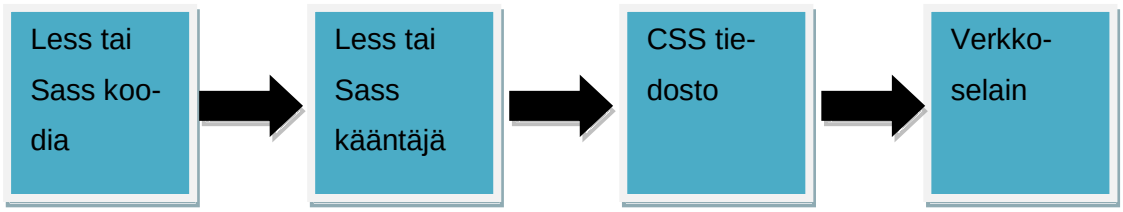
Adoben PhoneGap on ilmainen ja avoimen lähdekoodin paketointikehys hybridisovellusten ohjelmointiin. Se on joka on rakennettu Apache Cordovan päälle. PhoneGapiä käytettäessä voi valita JavaScript-kirjaston, jota käyttää. Valittavia kirjastoja ovat jQuery, plainJS ja Angular.js. PhoneGap-ohjelman voi ladata omalle koneelle ja pilvipalvelun kautta kääntäminen on myös mahdollista, jolloin SDK:ta ei tarvitse asentaa itselle paikallisesti omalle tietokoneelle jokaista alustaa varten. (Andström 2016.)

Ionic on ilmainen avoimen lähdekoodin ohjelmistokehys, joka on rakennettu Apache Cordovan päälle. Ionic käyttää JavaScript-ohjelmointikieltä ja Angular.js-sovelluskehystä. Ionic on sidottu MVVM-suunnittelumalliin (Model-View-ViewModel eli malli-näkymä-näkymämalli). (Andström 2016.) MVVM-arkkitehtuurimallissa on selvä ero käyttöliittymän, kontrollien ja näiden logiikan välillä. Komponentit ovat toisistaan irrallaan, joten komponenttien vaihtamien ja muokkaamisen on mahdollista rikkomatta muita komponentteja. Komponentit toimivat itsenäisesti ja niitä voidaan testata erillään muista. (Suntio 2016.)

Ionic-ohjelmistokehyksellä voidaan ohjelmoida hybridisovelluksia ilman puhelinta tai emulaattoria. Kehittäminen onnistuu suurilta osin verkkoselaimen avulla. Kaikkia puhelimien ominaisuuksia, kuten kameraa tai värinää ei kuitenkaan pysty testaamaan pelkän verkkoselaimen välityksellä. Google Chromen kehittäjän työkalujen avulla pystyy kehittäjä debuggaamaan eli korjaamaan virheitä sovelluksen ohjelmakoodista helpommin, kuin käyttämällä komentoriviä. DOM-elementtien ja HTTP-kutsujen tutkiminen onnistuu myös Google Chromen kehittäjän työkalujen avulla. (Nahkala 2016.)

AppBuilderissa ja PhoneGapissa on paljon samoja ominaisuuksia. Niissä on pilvipalvelukääntäjä, komentorivikäyttöliittymä, selainkäyttöliittymä ja Windowsilla toimiva graafinen käyttöliittymä. AppBuilderissa on lisäosa Visual Studion ohjelmointiympäristöön ja Sublime Text -tekstieditoriin. (Andström 2016.)

Kendo UI on tarkoitettu lähinnä ammattilaiskäyttöön. Kendo UI:ssä on yli 70 graafisen käyttöliittymän elementtiä eli käyttöliittymäkomponenttia ja useita valmiita ulkoasuteemoja. Kendo UI tukee Angular.js-, jQuery- ja Bootstrap-sovelluskehyksiä. Bootstrap käyttää HTML-, CSS- ja JavaScript-kieliä. Bootstrap on suunniteltu erityisesti mobiilikäyttöliittymien tekemiseen. Bootstrap tarjoaa tuen suositulle Less ja Sass (Syntactically Awesome Stylesheets) CSS-ennakkoprosessointimoottoreille. CSS-ennakkoprosessointimoottoreiden avulla CSS:ää voidaan kirjoittaa ohjelmointikielellä, joka käännetään CSS:ksi. Kuvassa 4. on esitetty Less ja Sass CSS-ennakkoprosessoinnin toiminta. (Andström 2016.)



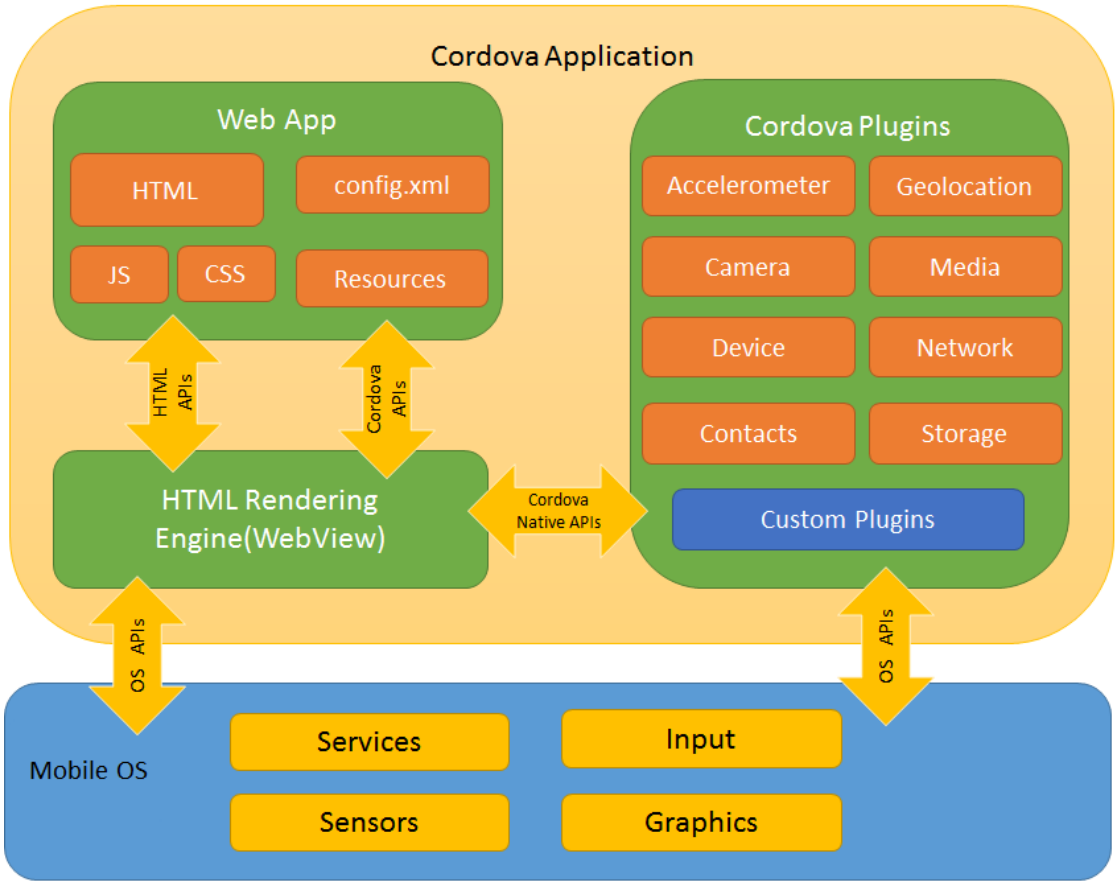
Kuva 4. Less ja Sass CSS-ennakkoprosessointimoottoreiden toiminta (Andström 2016).

Sencha Touch –hybridisovelluskehyksessä on yli 50 graafisen käyttöliittymän elementtiä. Se hyödyntää MVC-arkkitehtuurimallia ja JavaScript-kirjastoista käytetään Ext.js-kirjastoa. Sencha Touch tarjoaa käytettävän laitteen mukaan mukautuvan eli responsiivisen käyttöliittymän muokkautumisen, tehokkaat animaatiot ja laitteelle sopivan kuvan vierityksen eli kuvan liikuttelu laidasta toiseen siten, että toisesta laidasta kuva poistuu ja toisesta laidasta tulee lisää kuvaa. Tietojen käsittelyyn löytyy Sencha Touchista mallit, jotka helpottavat

palvelimelta tulevan tiedon esittämistä. Kaavioiden ja taulukoiden tekemiseen on valmiina sisäänrakennetut toiminnot. (Andström 2016.)

Mobile Angular UI on Sencha Touchin kaltainen sovelluskehys. Mobile Angular UI käyttää Bootstrap- ja Angular.js sovelluskehyksiä. Mobiiliohjelmoinnissa tarvittavia ominaisuuksia on lisätty Bootstrap-kehykseen ja tätä muokattua versiota kutsutaan nimellä Angular UI:n Bootstrap. Mobile Angular UI -sovelluskehys on ilmainen ja avoimen lähdekoodin sovelluskehys. (Andström 2016.)

Phonegap, Ionic, AppBuilder, Kendo UI, Sencha Touch ja Mobile Angular UI käyttävät Apache Cordovan - ohjelmistokehystä kääntämään sovelluksen kohdealustalle ja rakentamaan WebViewissä pyörivän sovelluksen. WebViewissä sovellus renderöi hybridisovellukseen sisältyvän web-sovelluksen. Apache Cordovalla käännetyn hybridisovelluksen sovellusarkkitehtuuri on esitetty kuvassa 5. (Andström 2016.)



Kuva 5. Apache Cordovan sovellusarkkitehtuuri (The Apache Software Foundation 2017).

Web-sovellus on sovelluskehyksellä kehitetty ohjelmakoodi. Web-sovelluksessa JavaScript-, HTML- ja CSS-tiedostot pysyvät muuttumattomina. index.html tiedostossa on määritelty sovelluksen tarvitsemat tiedostot. Sovelluksen config.xml tiedosto pitää sisällään sovellukselle tärkeitä tietoja siitä että kuinka sovelluksen kuuluu toimia. (Andström 2016.)

Apache Cordovan lisäosat ovat keskeisiä. Lisäosat luovat rajapinnan, jonka avulla Cordovan ja natiivien ominaisuuksien on mahdollista keskustella keskenään. Lisäosat luovat sidoksen laitteen API:in (Application Programming Interface) eli ohjelmoinnissa käytettävään rajapintaan. Tämä mahdollistaa laitteen rajapinnan käytön suoraan JavaScript-koodista. Apache Cordovan kehitystiimi on tehnyt ydinlisäosat, joka liittyvät laitteen yleisimpiin toimintoihin, kuten esimerkiksi akunkeston seurantaan, kameraan ja yhteystietoihin. Apace Cordovan lisäosien lisäksi on myös olemassa muiden tahojen kehittämiä lisäosia. Lisäosien tekemien itse on myös mahdollista ja se onkin paras vaihtoehto, jos on lisännyt omatekoisia natiivikomponentteja kohdealustalle. (Andström 2016.)

Mobiilisovelluksia voi kehittää ilman kolmannen osapuolen työkaluja käyttäen Apache Cordovaa. Cordovan kanssa kannattaa käyttää Phonegap, Ionic, AppBuilder, KendoUI, Sencha Touch tai Mobile Angular UI sovelluskehyksiä tai vaihtoehtoisesti voi itse luoda sovelluskehyksen. (Andström 2016.)

Cordovalla kehitittäessä on kaksi vaihtoehtoa, monialustatyötapa ja alustakeskeinen työtapa. Monialustatyötavalla on helppo tukea useita käyttöjärjestelmiä minimaalisella käyttöjärjestelmäkohtaisella kehityksellä. Cordova CLI:tä (Cordova Command Line Interface) käytetään monialustatyötavassa. Kun halutaan tukea useaa alustaa mahdollisimman vähäisellä alustakohtaisella kehityksellä, on monialustatyötapa hyödyllinen. Kun työtapana on alustakeskeinen, niin silloin keskitytään kerrallaan vain tiettyyn käyttöjärjestelmään ja sovellusta pyritään muokataan vain hyvin vähän. Kun sovellusta halutaan muokata alustan omaa SDK:ta hyödyntäen, niin silloin alustakeskeinen työtapa on oikea valita. (Andström 2016.)

2.5 Monialustasovellus

Natiivisovelluksia voidaan toteuttaa myös monialustasovelluskehyksillä. Monialustasovelluskehys mahdollistaa sen, että merkittävä osa sovelluksen koodista voidaan käyttää uudelleen eri laitealustoille. Tällaisia kehyksiä ovat esimerkiksi Xamarin ja Qt. (Kaasalainen 2015.)

Xamarin ei käytä web-tekniikoita sovelluskehitykseen vaan sovellukset kirjoitetaan C#-ohjelmointikielellä ja käännetään alustoihin sopivaksi. Sovellukset ovat kääntämisen jälkeen natiivisovelluksia. HTML-sivustot käyttävät selainmoottoria toimintoihin, mutta käännetyt monialustasovellukset käyttävät toimintoihin mobiililaitteen resursseja. Xamarinin runtime ja luokkakirjastot toimivat Android, iOS, ja Windows 10 Mobile käyttöjärjestelmillä. Xamarinilla on siis vain kaksi markkinoitavaa tuotetta, Xamarin.iOs iOs-käyttöjärjestelmälle ja Xamarin.Android Android-käyttöjärjestelmälle, koska Windows 10 Mobilen natiiviohjelmointi käyttää myös C#-ohjelmointikieltä kuten Xamarin. Xamarin.iOs ja Xamarin.Android on rakennettu Mono:n päälle. Mono on avoimen lähdekoodin versio .NET-ohjelmistokehyksestä. (Lehto 2016.)

Xamarin toiminta perustuu Android-alustalla ajonaikaiseen kääntämiseen. Koodien kääntäminen ennenaikaisesti tekee mahdolliseksi sen että Xamarinilla tehdyt sovellukset toimivat myös iOs-käyttöjärjestelmällä, koska ajonaikaiseen kääntäminen on kiellettyä iOs-käyttöjärjestelmällä. C#-ohjelmointikieli on natiivi ohjelmointikieli Windows 10 Mobile -käyttöjärjestelmälle ja tästä johtuen Xamarin sovellukset ovat valmiiksi natiiveina Windows 10 Mobilelle. Natiiviohjelmoinnilla ja Xamarinin monisovellusohjelmoinnilla on paljon yhtäläisyyksiä. Eroavaisuutena natiiviohjelmointiin verrattuna on käytettävä C#-ohjelmointikieli. Sekä Xamarinin tarjoamilla työkaluilla että natiivisovellusten kehitykseen tarkoitetuilla työkalulla on mahdollista toteuttaa sovelluksen käyttöliittymä. (Nikunen 2016.) Xamarilla kehittäminen maksaa $1000 - 1800 vuodessa, riippuen lisenssistä. (Lehto 2016.)

Qt-monialustasovelluskehyksen sovellukset ohjelmoidaan C++-ohjelmointikielellä. Qt:ssä käytettävään C++-ohjelmointikieleen on lisätty ominaisuuksia. Kuitenkin Qt-kehyksen oma kääntäjä kääntää lisättyjä ominaisuuksia sisältävän C++-ohjelmointikoodin standardin mukaiseksi C++-koodiksi. Qt:lla saadaan natiiveja mobiilisovelluksia. Android-alustalla Qt-kehyksellä toteutettujen sovellusten toiminnan tekee mahdolliseksi Java-ohjelmointirajapintojen käyttö. C++ ja Objective-C –ohjelmakoodeista kääntäjä tekee yhdistelmän iOs-käyttöjärjestelmälle sovelluksille. Qt on ilmainen avoimen lähdekoodin sovelluskehys. Myynnissä on myös kaupallisia lisenssejä. Jos käyttää Qt:n maksutonta versiota, niin silloin tuotettu sovellus täytyy julkaista tiettyjen lisenssien mukaan ja avointa lähdekoodia käyttäen. Qt:n tuettuja alustoja ovat muun muassa Android, iOs, Windows, Linux, OS X ja BlackBerry. (Nikunen 2016.)

3 Tutkimus ja tulokset

Tällä hetkellä mobiilisovelluksille on hyvin paljon vaihtoehtoisia toteutustapoja. Voidaan lähes puhua teknisten ratkaisumallien ylitarjonnasta, joka voi sekoittaa asiaan perehtymättömän ja hiukan perehtyneemmänkin. (Kaasalainen 2015.)

3.1 Tutkimusongelma ja menetelmä

Tutkimusongelmana oli selvittää kappaleessa 2 käsiteltyjen mobiiliohjelmoinnin tekniikoiden hyvät ja huonot puolet sekä riskit ja mahdollisuudet. Selvityksen kohteena oli selvittää mihin tilanteeseen ja milloin suositellaan kutakin mobiiliohjelmoinnin tekniikkaa. Tutkimusongelmaa käsitellään kirjallisuusselvityksenä hyödyntäen tuoreimpia sivustoja, jotka keskittyvät mobiiliohjelmointiin. Lähteenä käytettiin internetissä julkaistuja opinnäyte- ja insinööritöitä sekä mobiiliohjelmointia käsitteleviä nettisivustoja.

3.2 Tulokset

Mobiiliohjelmoinnista löytyy kattavasti tietoa, sillä mobiiliohjelmointi on hyvin ajankohtainen ja kehittyvä tällä hetkellä. Noin neljännes internetin käytöstä tapahtuu mobiililaitteilla. Internetin käytön mobiililaitteilla uskotaan vain kasvavan ja mobiilikäytettävyys onkin HTML-sivustoille erityisen tärkeää. Monet aikaisemmin vain tietokoneella toimineet ohjelmistot on mahdollista korvata mobiilisovelluksilla. (Klopal 2014.)

Natiivimobiilisovelluskehitys tarkoittaa sovelluksen ohjelmoimista jokaiselle alustalle erikseen. Natiivimobiilisovelluskehityksessä täytyy tehdä myös käyttöliittymän suunnittelu jokaiselle tuettavalle alustoille erikseen sovelluskoodin lisäksi. Natiivisovelluksella päästään hyödyntämään älypuhelinkäyttöjärjestelmien rajapintoja ja puhelimien toimintoja täysin. Natiivi mobiilisovellus tarjoaa parhaan suorituskyvyn ja käyttökokemuksen. Hyvänä esimerkkinä voisi mainita pelit, jotka vaativat riittävästi suorituskykyä pelin pyörittämiseen mobiililaitteessa sekä miellyttävän käyttökokemuksen tarjoamiseen. Pelit toteutetaankin lähes aina natiivilla kehityksellä. (Nahkala 2016.) Natiivi mobiilisovellus toimivat parhaiten juuri peleille ja sovellukseille, jotka käyttävät puhelimen toimintoja, kuten esimerkisi kameraa, kiihtyvyysanturia tai puhelimen tietoja (Appliware 2016).

Natiivisovellus toteutetaan käyttäen kehitettävän alustan työkaluja ja tekniikoita. Natiivisovelluksen kehittäjä saa ohjelmointialustan toiminnot käytettäväksi ja on mahdollista tehdä

käyttäjäkokemuksesta hyvin sujuva. Natiivisovelluskehitys on kallis tapa kehittää mobiilisovellus, jos sovelluksen tehdään usealle alustalle ja koodi kirjoitetaan kaikille tuettaville alustoille erikseen niiden omalla ohjelmointikielellä. (Kaasalainen 2015.) Joitakin samankaltaisuuksia eri alustoille tehtyjen natiivien mobiilisovelluksien toteutuksissa on, mutta pääpiirteissään eri alustoille rakennetut mobiilisovellukset ovat itsenäiset ja erilliset toteutukset (Appliware 2016).

Natiivien mobiilisovellusten plussat:

- Optimointi voidaan suorittaa käyttöliittymän ja suorituskykyn mukaan ja käyttökokemuksesta tulee sujuva (Appliware 2016).
- Natiivisovellus voi hyödyntää mobiilialustan toimintoja ja rajapintoja, kuten esimerkiksi puhelimen sijaintitietoja, tiedostojen tallentamista, kameraa, kiihtyvyysanturia, puheluita ja tekstiviestejä (Hurja 2016).
- Sovellus voidaan jakaa sovelluskauppojen kautta. Android-sovellusten sovelluskauppa on Google Play, iOs-sovellusten App Store ja Windows-sovellusten Windows 10:n sovelluskauppa. (Hurja 2016.)
- Sovelluksen käyttöön ei vaadi pakosti internetyhteyttä (Appliware 2016).
- Mahdollisuus käyttää taustajärjestelmiä (Appliware 2016).
- Sujuva käyttökokemus erityisesti paljon grafiikkaa sisältävissä sovelluksissa, kuten peleissä (Vuorinen 2014).
- Voidaan hyödyntää puhelimen suorituskykyä täysin (Hurja 2016).

Natiivien mobiilisovellusten miinukset:

- Useiden eri sovellusten kehittäminen ja ylläpito on kallista (Appliware 2016).
- Työresurssien tarve on suuri (Appliware 2016).
- Sovellusversioiden hallinta on monimutkaista (Appliware 2016).
- Toimii vain tietyillä käyttöjärjestelmillä ja mahdollisesti tietyillä laitteilla (Klopal 2014).

Jos natiiviohjelmoinnissa jätetään jonkun käyttöjärjestelmän sovellus rakentamatta, menetetään joukko asiakkaita. Nykyisin tekemällä vain Android- ja iOS-sovellukset saa jo hyvin suuren mahdollisen käyttäjäryhmän sovellukselleen. (Appliware 2016.)

HTML-sivusto tarkoittaa käyttäjän näkökulmasta normaalia internetsivua. Käyttäjän huomaa eron natiivisovelluksiin verrattuna siinä, että sovellusta ei asenneta puhelimeen. Kehittäjä ei kuitenkaan pysty hyödyntämään täysin kaikkia puhelimen ominaisuuksia. Esimerkiksi puhelimen yhteystietoihin ei HTML-sivusto pääse, mutta muun muassa kameran

käyttö on sen sijaan mahdollista. (Nahkala 2016.) Kuitenkin HTML-sivustolla on sijaintitiedon käyttäminen mahdollista (Appliware 2016).

HTML-sivustoja tehdessä tulevat mobiililaitteiden ruutukokojen erot vahvasti esiin. HTML-sivusto on testattava hyvin eri näyttökoon omaavilla mobiililaitteilla. (Kaasalainen 2015.) Kolmansien osapuolien, kuten Applen tai Googlen, hyväksyntää ei tarvitse HTML-sivustojen julkaisuun (Kaasalainen 2015). Natiivisovelluksiin verratuna HTML-sivusto eroaa myös sen osalta, että päivitys ei vaadi sovellusten lataamista kauppapaikoille uudestaan (Nahkala 2016). Palvelimelle sijoitettua HTML-sivustoa voidaan muokata helposti ja nopeiden testauksien ja korjauksien tekeminen on mahdollista (Kaasalainen 2015).

HTML-sivuston plussat:

- Laajat mahdollisuudet käyttää puhelimen rajapintoja, kuten esimerkiksi puhelimen sijaintitietoja, tiedostojen tallentamista tai kameraa (Appliware 2016).
- Kaikille laitteille ja käyttöjärjestelmille on yhtenäinen lähdekoodi (Hurja 2016).
- Nopea ja kustannustehokas sovelluskehitys (Hurja 2016).
- Helppo jatkokehitys myös julkaisun jälkeen (Hurja 2016).
- Ylläpitokustannukset ovat natiivisovelluksen ylläpitokustannuksiin verrattuna alhaiset (Appliware 2016).
- Erillisiä jakelukanavia ei tarvita, jolloin laitevalmistajien säännöt eivät ole rajoituksena (Hurja 2016).

HTML-sivuston miinukset:

- Puhelimen tietojen käyttäminen on rajallista (Appliware 2016).
- Suorituskyvy ja käyttökokemus on huonompi kuin natiivisovelluksilla (Appliware 2016).
- HTML-sivuston käyttö tarvitsee internet-yhteyden (Appliware 2016).

Hybridisovelluksissa käytetään web-teknologioita (HTML, CSS ja JavaScript). Käytännössä hybridisovellus on verkkosivu, joka on asennettu mobiililaitteeen. Erona hybridisovelluksissa HTML-sivustoihin verrattuna on että ohjelmistokehykset, kuten esimerkiksi Apache Cordova, joiden päälle hybridiohjelmistokehykset ovat rakennettu, mahdollistavat mobiililaitteen käyttöjärjestelmän rajapintojen käytön. Hybridimobiilisovelluksella on mahdollista hyödyntää mobiilialustan rajapintoja lähes samoin kuin että sovellus olisi kehitetty natiivisti. (Nahkala 2016.)

Hybridi-sovelluksen avulla kehitys- ja ylläpitokustannukset pysyvät kohtuullisena. Viime vuosina hybiridi-sovellusten kehitysvälineet ovat kehittyneet huimasti. (Appliware 2016.)

Hybridi-sovelluksen plussat:

- Samalla ohjelmointikoodilla saadaan mobiilisovellus kaikille alustoille (Appliware 2016).
- Sovellus voidaan jakaa helposti ja laajasti sovelluskauppojen kautta (Hurja 2016).
- Kehitys- ja ylläpitokustannukset ovat natiivisovellukseen verrattuna alhaiset (Appliware 2016).
- Hybridisovellus tukee lähes kaikkia laitteen sensoreita ja ominaisuuksia (Klopal 2014).

Hybridi-sovelluksen miinukset:

- Puhelimen tietojen käyttäminen on rajallista (Appliware 2016).
- Vanhoilla mobiililaitteilla saattaa suoritusteho olla puutteellinen (Klopal 2014).
- Käyttökokemus ei välttämättä vastaa natiivisovellusta (Klopal 2014).

Monialustasovelluksessa sovelluskehys muuttaa kirjoitetun ohjelmointikoodin mobiilialustalla toimivaksi paketiksi mahdollistaen natiivitason suoritusnopeuden. Monialustasovelluskehysten avulla säästetään paljon työresursseista, koska sama ohjelmointikoodi voidaan kääntää eri mobiilialustoille sopivaksi (Kaasalainen 2015.) Monialustasovellukset luovat pohjimmillaan natiivin sovelluksen (Nahkala 2016).

4 Pohdinta

Erilaisia vaihtoehtoja mobiilisovelluksen toteuttamiseksi on useita. Oikean totetustavan valintaan ei ole helppo antaa yleispätevää ohjetta vaan siihen vaikuttaa sovelluksen tarpeet, käyttöympäristöstä ja vaaditut ominaisuuden. Käytettävissä olevat resurssit ja budjetti on syytä ottaa huomioon myös mobiilisovelluksen toteuttamistapaa valittaessa. (Klopal 2014.)

4.1 Johtopäätökset ja suositukset

Opinnäytetyn tarkoituksena oli tarkastella mobiilialustoja, mobiiliohjelmoinnin tekniikoita ja vertailla eri tekniikoita kirjallisuusselvityksenä hyödyntäen nettisivustoja, joissa keskitytään mobiiliohjelmointiin. Mobiilialustoista löytyi helposti tietoja. Ainoastaan Windows 10 Mobile –käyttöjärjestelmästä oli tällä hetkellä vaikea tehdä johtopäätöksiä että onko kyseinen mobiilialusta poistumassa kokonaan markkinoilta. Nykyisin käytössä olevista mobiilikäyttöjärjestelmistä Windows 10 Mobilella on vielä merkittävä asema, mutta tällä hetkellä myytyjen älypuhelimen määrissä Windows 10 Mobile -käyttöjärjestelmän osuus on hyvin pieni.

Eri mobiiliohjelmoinnin tekniikoista löytyi erittäin helposti tietoa. On vaikea tehdä yhtenäisiä ohjeita mikä olisi kullekin sovellukselle oikea toteutustapa. Opinnäytetyön tuloksista keskeisin on että natiiviohjelmoinnilla saavutetaan parhaat tulokset kalliimmalla hintalapulla. Tosin taas hybridi- ja monialustaohjelmoinnilla voidaan helposti löytää oikeaa sovelluskehystä käytettäessä hyvinkin nopea ja helppo tapa tehdä natiivin käyttökokemuksen antava sovellus.

Mobiilisovellukselle on ehdottoman tärkeää että se pystyy antamaan miellyttävä käyttökokemuksen käyttäjälleen. Vain hyvin toimivalla mobiilisovelluksella on mahdollisuudet menestyä mobiilisovellusmarkkinoilla. Esimerkkinä voisi mainita kesällä 2016 ilmestyneen Pokemon Go –mobiilipelin. Pokemon Go –peli on toteutettu natiivisovelluksena Android- ja iOs-käyttöjärjestelmille. Noin kahden viikon ajan pelin julkaisun jälkeen, peli toimi hyvin huonosti. Peliä joutui käynnistämään uudelleen usean kerran pelin jumittumisen takia. Pokemon Go –pelin ongelmat saatiin kuitenkin parin ensimmäisen viikon jälkeen korjatuksi ja peli alkoi toimimaan erittäin hyvin. Pelistä tuli erittäin suosittu.

4.2 Opinnäytetyöprosessi ja oma oppiminen

Olin aikaisemmin tutustunut iOs, Android ja Windows älypuhelinkäyttöjärjestelmien natiiviohjelmointeihin sekä hydridiohjelmointiin Android-käyttöjärjestelmälle käyttäen PhoneCap-sovelluskehystä ja Apache Cordovan –moottoria. Valinnanmahdollisuudet yllättivät tutustuessani HTML-, hybridi- ja monialustaohjelmointiin. Tarjolla on hyvin paljon erilaisia sovelluskehyksiä ja kirjastoja, jotka helpottavat huomattavasti sovelluksen tekoa. Mobiiliohjelmointi vaikuttaa hyvin kiinnostavalta, mutta pidän haastavana oikean toteutustavan valintaa, koska vaihtoehtoisia toteutustapoja on hyvin paljon ja yleistä ohjetta tuohon valintaan ei oikeastaan ole.

Lähteet

Andström, I. 2016. Metropolia ammattikorkeakoulu. Insinöörityö. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/114497/ilpo_andstrom.pdf?sequence=1. Luettu 7.12.2016.

The Apache Software Foundation, 2017. Overview. Luettavissa: https://cordova.apache.org/docs/en/latest/guide/overview/. Luettu 20.3.2017.

Apple. 2016. System requirements for Continuity on Mac, iPhone, iPad, iPod touch, and Apple Watch. Luettavissa: https://support.apple.com/en-us/HT204689. Luettu 20.11.2016.

Appliware. 2016. Miten tehdä mobiilisovellus? Luettavissa: http://appliware.fi/miten-tehda-mobiilisovellus/. Luettu 8.12.2016.

Aucor. 2016. AngularJS. Luettavissa: https://www.aucor.fi/angularjs/. Luettu 7.12.2016.

BGR. 2016. Whatever you do, don't buy these smartphones Luettavissa: http://bgr.com/2016/05/23/smartphone-market-share-q1-2016/. Luettu 11.9.2016.

Ching, C. 2016. 1. Introduction to the tools and materials. Luettavissa: http://codewithchris.com/1-introduction-to-the-tools-and-materials/. Luettu 5.12.2016.

Hamppula, M. 2015. Metropolia ammattikorkeakoulu. Modernit web-teknologiat pelikehityksessä. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/101729/hamppula_marko.pdf?sequence=1. Luettu 8.3.2017.

Hartto, N. 2014. Metropolia Ammattikorkeakoulu. HTML5-pohjaiset mobiilisovellukset. Insinöörityö. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/74216/Hartto_Nico_html5_pohjaiset_mobiilisovellukset.pdf?sequence=1. Luettu: 7.3.2017.

Hurja. 2016. Mobiilisovelluksen kehitys. Luettavissa: http://www.hurja.fi/fi/palvelut/mobiiliohjelmointi. Luettu 8.12.2016.

Hynninen, T. 2014. Opi tuntemaan Swift: Applen uusi ohjelmointikieli. Luettavissa: http://teemuhynninen.fi/2014/07/opi-tuntemaan-swift-applen-uusi-ohjelmointikieli/. Luettu 5.12.2016.

Hynninen, T. 2016. Microsoft vetäytyy älypuhelimien kuluttajamarkkinoilta – jatkaa yrityksien parissa. Luettavissa: http://www.mobiiliblogi.com/2016/10/01/microsoft-vetaytyy-alypuhelimien-kuluttajamarkkinoilta-jatkaa-yrityksien-parissa/. Luettu18.3.2017.

Ilves, K., Vihavainen, A. 2015. Web-selainohjemointi. Luettavissa: https://web-selainohjelmointi.github.io/. Luettu 7.3.2017

Järvinen, J. 2016. Nokian nimissä valmistettu Android-puhelin ei ehdi enää pukinkonttiin. Luettavissa: http://yle.fi/uutiset/3-9279139. Luettu 6.12.2016.

Kaasalainen, J. 2015. Miten valita sovellukselle oikea toteutustapa – perustietoa teknologiaviidakosta. Luettavissa: http://www.taiste.fi/blogi/miten-valita-sovellukselle-oikea-toteutustapa-perustietoa-teknologiaviidakosta. Luettu 11.9.2016.

Konttinen, E. 2016. Huippupuhelimien käyttöjärjestelmät vertailussa: Android 6.9, Windows 10 ja iOS 9.2. Luettavissa: http://mobiili.fi/2016/01/15/vertailussa-ios-android-windows/. Luettu 15.11.2016.

Klopal. 2014. Mobiiliaplikaatioiden kehitys: HTML5, natiivi vai hybridi? Luettavissa: https://www.klopal.fi/2014/09/01/mobiiliaplikaatioiden-kehitys-html5-natiivi-vai-hybridi/. Luettu 8.3.2017.

Lehto, M. 2016. Hämeen ammattikorkeakoulu. Mobiilisovellustyypit ja niillä kehittämisen vaihtoehdot. Opinnäytetyö. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/106545/Lehto_Mikael.pdf?sequence=1. Luettu 6.12.2016.

Luoto, V. 2008. Tampereen ammattikorkeakoulu. Asiakaspisteiden suunnittelu ja toteutus. Tutkintotyö. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/10243/Luoto.Vesa.pdf?sequence=2. Luettu 19.3.2017.

Mullis, A. 2016. How to install the Android SDK (Software Development Kit). Luettavissa: http://www.androidauthority.com/how-to-install-android-sdk-software-development-kit-21137/. Luettu 20.11.2016.

Nahkala, T. 2016. Tampereen ammattikorkeakoulu. Mobiilisovellusten alustariippumattomat ohjelmistokehykset. Opinnäytetyö. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/120488/Nahkala_Toni.pdf?sequence=1. Luettu: 7.3.2017.

Nikunen, P. 2016. Tampereen ammattikorkeakoulu. Monialustainen mobiilikehitys Xamarin-alustalla. Opinnäytetyö. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/118762/Nikunen_Pyry.pdf?sequence=1. Luettu 8.12.2016.

Ohjelmistoputka. 2008. Opasarkisto: AJAX: AJAX - Asynchronous JavaScript and XML. Luettavissa: http://www.ohjelmointiputka.net/oppaat/opas.php?tunnus=ajax. Luettu 7.3.2017.

Phandroid. 2016. Android Phones Guide. Luettavissa: http://phandroid.com/phones/. Luettu 11.9.2016.

Pääkkönen, A. 2014. Kajaanin ammattikorkeakoulu. Alustariiippumaton mobiilisovellus Bluetooth Low Energy –kommunikointiin. Insinöörityö. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/77032/Paakkonen_Antti.pdf?sequence=1. Luettu 11.9.2016.

Portimo, H. 2014. Lahden ammattikorkeakoulu. Python ja Django webkehityksessä. Opinnäytetyö. Luettavissa: http://www.theseus.fi/bitstream/handle/10024/79177/Portimo_Henri.pdf?sequence=1. Luettu 7.12.2016.

Pyykkönen, V-M. 2015. Hämeen ammattikorkeakoulu. jQuery web-sovelluksen apuvälineenä. Opinnäytetyö. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/96356/Pyykkonen_Veli-Matti.pdf?sequence=1. Luettu 7.3.2017.

Rantakiesu, M. 2016. Mikan Android blogi. Luettavissa: http://www.mikarantakeisu.fi/internetsivut/content/android-ohjelmien-tekeminen. Luettu 20.11.2016.

Sani, I. 2012. iOs-ohjelmoinnin alkeet 1/3 Tekniikat ja työkalut. Luettavissa: http://mikropc.net/nettilehti/pdf/3008201244.pdf. Luettu 3.12.2016.

Sovelto. 2016. Moderni web-sivusto jQuerylla. Luettavissa: https://www.sovelto.fi/kurssit/kurssi/3336/Moderni-web-sivusto-jQuerylla. Luettu 7.12.2016.

Suntio, J. 2016. Kymenlaakson ammattikorkeakoulu. Treenipäiväkirjasovelluksen toteutus Windows 10 –laitteille. Opinnäytetyö. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/108237/joni_suntio.pdf?sequence=2. Luettu 7.12.2016.

Toivonen, J. 2013. PhoneCap-työvälineohjelmistojen kehitys älypuhelimiin. Turun ammattikorkeakoulu. Opinnäytetyö. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/62810/Toivonen_Jesse.pdf?sequence=1. Luettu 11.9.2016.

Vuorinen, C. 2014. Kolme tapaa kehittää mobiilisovellus. Luettavissa: http://w3.fi/kolme-tapaa-kehittaa-mobiilisovellus/. Luettu 11.9.2016.

Windows Central. 2016. Windows 10 Mobile. Luettavissa: http://www.windowscentral.com/windows-10-mobile. Luettu 8.12.2016