Rolle Swanljung & Timo Kannala

Mobiilisovelluskehitysmenetelmien vertailu

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

25.11.2014

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Rolle Swanljung & Timo Kannala Mobiilisovelluskehitysmenetelmien vertailu 56 sivua + 8 liitettä 25.11.2014
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Lehtori Simo Silander Lehtori Juha Pekka Kämäri

Opinnäytetyössä käymme läpi, mitä mobiilisovelluskehitys on natiivina, HTML5:llä ja näiden yhdistelmänä, hybridinä. Selvitämme myös, mikä on paras vaihtoehto toteutukselle eri tilanteissa ja kuinka valittu toteutustapa vaikuttaa loppukäyttäjän käyttökokemukseen.

Käsittelemme ensin älypuhelinten historiaa ja kehitystä. Tämän jälkeen käsittelemme nykytilannetta ja pohdimme, mitä tulevaisuudessa voidaan odottaa.

Käymme läpi kolme merkittävintä tämän hetken mobiiliekosysteemiä. Ne ovat Apple, Microsoft ja Google. Selvitämme, mitä ekosysteemit pitävät sisällään ja miten ne eroavat toisistaan. Ekosysteemit ovat merkittävässä asemassa mobiilisovelluskehityksessä.

Mobiilisovelluskehitys on jaettu työssä kolmeen osaan. Natiivisovelluskehityksessä vertaillaan, miten sovellusten toteuttaminen eri alustoille tapahtuu. Aiheeseen liittyy muun muassa arkkitehtuuriin liittyvät eroavaisuudet ja ohjelmointikielet. Natiivi- ja hybridisovelluksista tehdään toteutukset Androidille.

Hybridisovelluskehitystä läpikäydessä todetaan, mitä hyviä ja huonoja puolia siinä on verrattaessa natiivi- ja HTML5-sovelluskehitykseen.

Puhdas HTML5-sovelluskehitys eroaa suuresti natiivi- ja hybridisovelluskehityksestä. Työssä selvitetään, mitä HTML5:llä pystytään toteuttamaan mobiililaitteille ja missä menevät sen rajat.

Teemme johtopäätökset sovelluskehitystavoista ja niiden eroista. Selitämme missä tilanteessa kannattaa käyttää mitäkin tapaa luoda sovelluksia. Lopuksi yhteenvetona käymme läpi lopputyön, sekä kerromme työskentelystämme.

Avainsanat	Mobiili, Sovelluskehitys, Natiivi, Hybridi, HTML5

Author(s) Title Number of Pages Date	Rolle Swanljung & Timo Kannala Comparison of Mobile Application Development Methods 56 pages + 8 appendices 25 November 2014
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Simo Silander, Senior Lecturer Juha Pekka Kämäri, Senior Lecturer

In this thesis we describe what is mobile development using either native, HTML5 or combinations of these, called hybrid processes. We also clarify which is the best option for implementation in different circumstances and how it affects the end users experience.

First we go through the history and development of smartphones. After that we cover the current situation and what can be expected in future.

We review the three most important mobile ecosystem of this day: Apple, Microsoft and Google. We find out what ecosystems entail and how they differ from each other.

Application development for mobiles has been divided into three parts. In the native application development part the comparison includes which way implementations differ on different platforms. Differences between architecture and programming languages among other things are related to this. In this thesis native and hybrid applications are Android implementations.

Pros and cons of native and HTML5 application development are discussed while explaining the hybrid application development.

Pure HTML5 application development differs greatly from the native and hybrid application development. This thesis explains what can be implemented for mobile devices using HTML5 and where are its limits.

We draw conclusions about the application development methods and their differences. We explain under which circumstances to use specific ways to create applications.

Keywords	Mobile, Software development, Native, Hybrid, HTML5

Sisällys

Lyhenteet

1 Johdanto 1

2 Mobiilikehitys yleisesti 1

2.1 Historia 1
2.2 Nykytilanne 3
2.3 Tulevaisuus 3

3 Ekosysteemit 5

3.1 Apple 5
3.2 Google 6
3.3 Microsoft 6

4 Sovelluskehitysmenetelmät 7

4.1 Natiivi 8
4.1.1 Windows Phone 8
4.1.2 Android 14
4.1.3 iOS 20
4.2 Hybridi 22
4.3 HTML5 23
4.4 Yhtäläisyydet ja eroavaisuudet 25

5 Sovelluksien toteutus 26

5.1 Sovelluksen määrittely 26
5.2 Toteutus 27
5.2.1 Natiivi 27
5.2.2 Hybridi 34
5.2.3 HTML5 41

6 Johtopäätökset 44

7 Yhteenveto 47

Lähteet 49

Liitteet

Liite 1. Android Java - MainActivity.java

Liite 2. Android Java - OttoMatti.java

Liite 3. Android Java - OttoMaattiDistance.java

Liite 4. Android Java - OttoMaattiDistanceComparator.java

Liite 5. Android Java - OttoMaattiListAdapter.java

Liite 6. Android XML

Liite 7. Hybrid - Ottopisteethybrid.js

Liite 8. Hybrid - index.html

Lyhenteet

ADT	Android development toolkit. Android-kehitystyökalu.
AOT	Ahead of time. Tavukoodin kääntäminen ennen suoritusta.
Alusta	Tarkoitetaan iOS, Android, Windows Phone, Ubuntu sekä muita puhelimissa käytettäviä käyttöjärjestelmiä.
API	Application programming interface. Ohjelmistorajapinta. Tätä käyttämällä saadaan koodissa käytettyä haluttuja valmiiksi tehtyjä toimintoja.
ASOP	Android Open Source Project. Avoimen lähdekoodin Android projekti.
B2B	Business to business. Yritysten välinen markkinointi.
CLR	Common Language Runtime on .Net:in virtuaalikone.
CPU	Central processing unit. Suoritin/Prosessori/Keskusyksikkö.
Ekosysteemi	Ekosysteemi tarkoittaa yhtenäisen ympäristön muodostamaa toiminnallista kokonaisuutta. Esimerkkinä käyttöjärjestelmä, kauppapaikka, sovelluskehitysvälineet, pilvi- ja muut yhteensopivat palvelut.
HAL	Hardware abstraction layer. Käyttöjärjestelmän osajärjestelmä, joka sisältää mahdollisuuden käyttää laitteiston resursseja.
IDE	Integrated development environment. Kehitysympäristö. Esimerkiksi Visual Studio, jolla voidaan tehdä, testata ja julkaista sovelluksia puhelimeen.
JIT	Just in time. Tavukoodin kääntäminen suorituksen aikana.

NDK	Native development kit. Natiivisovelluskehitystyökalut.
OEM	Original equipment manufacturer. Alkuperäinen laitevalmistaja.
PEP	Power management engine plug-in. Virranhallintamoottorin lisäosa.
PHP	Hypertext preprocessor. Ohjelmointikieli, jota käytetään web-palvelinympäristöissä.
SDK	Software development kit. Sovelluskehitystyökalut.
Sovelluskehys	Ohjelmistotuote, jonka päälle sovellus toteutetaan. Sisältää valmiiksi määriteltyjä sovellusosia, joita ei täydy määritellä uudestaan.
SV	Silicon vendor. Komponenttien toimittaja.
UEFI	Unified extensible firmware interface. Määrittelee laiteohjelmiston ja käyttöjärjestelmän välistä keskustelua.
URL	Uniform resource locator. Tietyn tiedon paikka internetissä, usein osoittaa www-sivuun.
WP	Windwos Phone, (käyttöjärjestelmä).
WYSIWYG	What you see is what you get. Editori jossa toteutuksen aikana näkyvä sisältö vastaa valmista tulostetta.
XAML	Extensible application markup language. Declarative (kuvaava) merkintäkieli.
Älypuhelin	Matkapuhelin, joka pystyy tietokoneen toimintoihin. Yleisesti ominaisuuksina kosketusnäyttö, internetyhteys ja käyttöjärjestelmä, jolla pystyy ajamaan ladattuja ohjelmia.

1 Johdanto

Opinnäytetyön tavoitteena on todeta, mikä sovelluskehitystapa on suotuisa eri tilanteissa. Tavat lajitellaan työssä natiiviin, hybridiin ja HTML5:een. Tarkastelemme mobiililaitteiden historiaa, nykytilannetta ja tulevaisuutta. Nykytilanteesta käydään läpi tarkemmin nykyiset merkittävät ekosysteemit ja kuinka niille mobiilisovelluskehitys tapahtuu.

Sovelluskehityksessä on tärkeää tietää, mihin tarkoitukseen sovellusta ollaan toteuttamassa, mikäli se ollaan julkaisemassa. Sovellusten ylläpito, päivittäminen ja saatavuus eroavat suuresti riippuen kehitysmenetelmästä.

Vertailu suotuisalle sovelluskehitystavalle tehdään useasta näkökulmasta. Näitä ovat mm. tarvittavat sovelluskehitysvälineet, oppimiskäyrä, sovelluksen ajaminen, ylläpito, jakaminen ja käyttökokemus. Vertailuissa otetaan huomioon, kuka kyseistä toteutusta on tekemässä ja mihin tarkoitukseen sitä tehdään.

Tämän lopputyön ohessa on toteutettu mobiilisovelluksia eri kehitysmenetelmillä. Ne on pyritty tekemään toiminnallisuuksiltaan mahdollisimman samanlaisiksi. Vertailemme toteutuksia ja niiden eroja. Timo Kannala on toteuttanut natiivisovelluksen ja Rolle Swanljung pääasiassa hybrdidi- ja HTML5-sovellukset. Opinnäytetyö on kirjoitettu yhdessä niin yleisen osuuden kuin vertailujen osalta.

2 Mobiilikehitys yleisesti

2.1 Historia

Älypuhelimien historia alkaa 90-luvulta, jolloin termiä on käytetty matkapuhelimen kuvauksessa. Vuonna 2007 Apple julkaisi ensimmäisen älypuhelimensa, iPhonen, kuluttajamarkkinoille. Tuosta alkoi älypuhelinten suosio kuluttajamarkkinoilla. Tätä ennen älypuhelimia käytettiin lähinnä liike-elämässä. [21;88.]

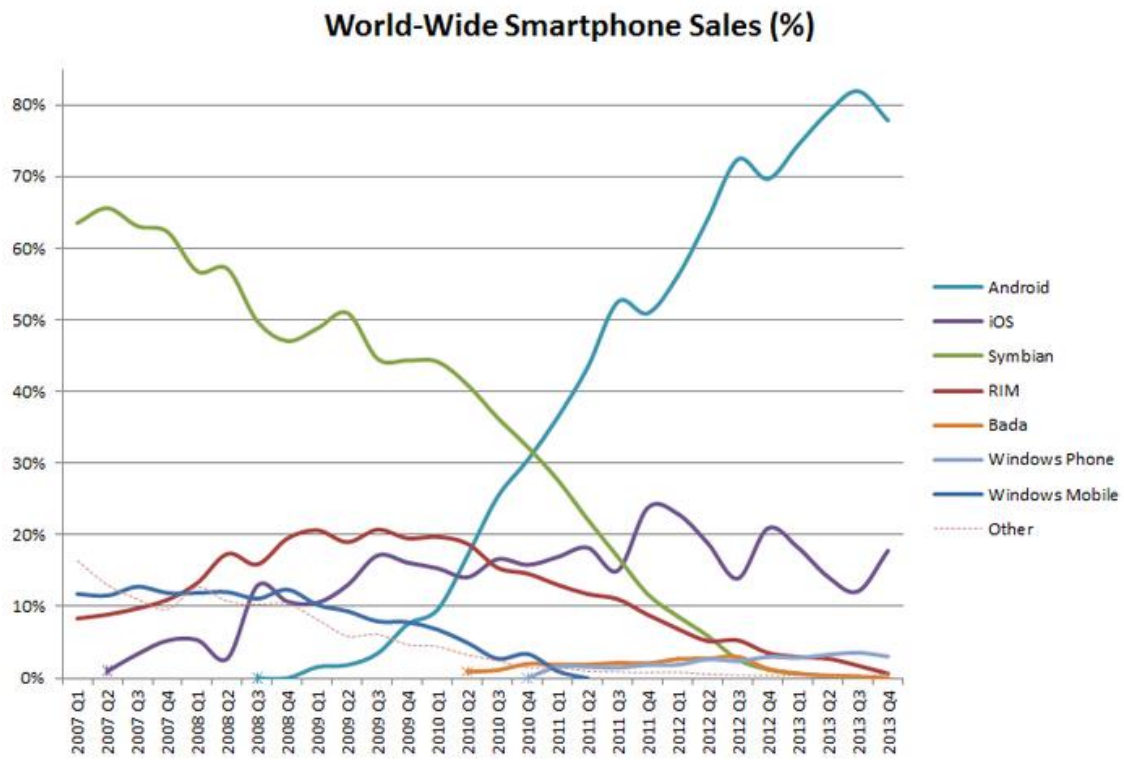
Ennen Applen tuloa markkinoille Nokia oli pitkään merkittävä tekijä mobiilimarkkinoilla. Symbian oli ensimmäisiä merkittäviä käyttöjärjestelmiä älypuhelimissa. Symbianin aikakausi kesti aina vuodesta 2001 vuoteen 2012 asti. Nokian ja useiden puhelinvalmistajien

voimin Symbianin markkinaosuus kävi jopa 65 prosentissa vuonna 2007 (kuva 1). Muiden valmistajien lopetettua alustan kehityksen tukemisen jatkoi Nokia yksin vielä alustan kehittämistä. Nokia 808 PureView oli viimeinen Nokian Symbian-tuote. [80;81;82;90.]

Vuonna 2008 HTC julkaisi ensimmäisen kuluttajamarkkinoille suunnatun Android-käyttöjärjestelmällä varustetun älypuhelimen. Puhelimen ominaisuuksiin kuuluu mm. GPS ja kosketusnäyttö. Mukana tuli myös silloinen Android Market (nykyinen Play). [66;94.]

Vuonna 2010 tulivat markkinoille ensimmäiset Windows Phone 7.0-käyttöjärjestelmällä toimivat puhelimet. [31;78.]

Ennen iOS-, Android- ja Windows Phone-käyttöjärjestelmiä mm. Blackberry- ja Symbian-käyttöjärjestelmät ovat olleet suuressa suosiossa. Näiden suosio on kuitenkin laskenut uudempien käyttöjärjestelmien tultua markkinoille. Uusien käyttöjärjestelmien suurena etuna vanhoihin verrattuna ovat olleet hyvin toimivat kauppapaikat, joista käyttäjät voivat helposti ladata kolmansien osapuolien sovelluksia. [2.]



Kuva 1. Eri käyttöjärjestelmien markkinaosuudet. [37-60.]

Kuvassa 1 on kaavio eri käyttöjärjestelmillä varustettujen puhelinten prosentuaalisesta myynnistä maailmalla. Kuten kuvasta nähdään, iOS ja Android valtasivat markkinoita hyvin nopeasti julkaisun jälkeen. Kuvassa RIM on aikaisemmin mainittu BlackBerry.

2.2 Nykytilanne

Tänä päivänä Android- ja iOS-käyttöjärjestelmillä toimivat älypuhelimet vievät merkittävän osan markkinoista. Vaikka maailmanlaajuisesti Windows Phonen markkinaosuus on pieni edellä mainittuihin verrattuna, Suomessa sen osuus on merkittävä. Tulevaisuudessa voidaan kuitenkin olettaa Windows Phonejen määrän laskevan myös Suomessa, koska yhdysvaltalainen Microsoft valmistaa niitä suomalaisen Nokian sijasta. [87;89.]

Älypuhelinlaitteiden ominaisuudet ovat nykypäivänä melko laajat ja laitteisto jatkuvasti kehittynyt. Näyttöjen tarkkuudet vastaavat tietokoneiden ja teräväpiirtotelevisioiden tarkkuutta. Tämän lisäksi keskusyksiköt ovat moniytimisiä ja muistia laitteista löytyy reilusti.

Markkinoilla on myös vähäisemmässä asemassa olevia käyttöjärjestelmiä, kuten BlackBerry, Sailfish, Nokia Asha ja niin edelleen. Näiden markkinaosuus varsinkin Suomessa on sen verran pienessä asemassa, ettei niitä käsitellä tässä työssä.

2.3 Tulevaisuus

Tulevaisuudessa uusien käyttöjärjestelmien tulo älypuhelinmarkkinoille on vaikeaa, koska nykyinen tarjonta on jo niin laaja. Mikäli uusilla käyttöjärjestelmillä haluttaisiin päästä markkinoille, olisi niillä hyvä olla valmiina kattavat ja toimivat ekosysteemit. Tämä voisi tapahtua esimerkiksi mahdollisuudella ajaa muiden isompien käyttöjärjestelmien sovelluksia tai pohjautumalla HTML5:een.

Tulevaisuudessa kiinnostavana kohteena on syytä mainita Ubuntu-käyttöjärjestelmä, koska Ubuntulla on jo valmiiksi laaja ekosysteemi ympärillään tietokoneissa ajettavan käyttöjärjestelmän vuoksi.

Ubuntu erottuu muista käyttöjärjestelmistä tekemällä puhelimesta myös pöytätietokoneen. Ubuntua käyttäessä puhelin kytketään telakkaan, johon on kytketty näyttö, näppäimistö ja hiiri. Tällöin ei erillistä tietokoneen keskusyksikköä tarvita. Puhelimen käyttökokemus vastaa tällöin hyvin pitkälle pöytätietokoneita. Turhat synkronoinnit puhelimen ja tietokoneen välillä voidaan unohtaa. Puhelin toimii normaalien älypuhelinten tapaan irrottamalla se telakasta. Kuten aiemmin mainittiin, jo nykyään puhelinten laitteisto vastaa osittain tietokoneita. Tulevaisuudessa saattavat esimerkiksi Apple ja Microsoft matkia Ubuntun mallia ja kehittää puhelimistaan myös tietokoneita. [95.]

Ubuntun kiinnostavuutta lisää se, että siinä toimivat tietokoneille tehdyt ohjelmat. Kyseessä on nimittäin sama käyttöjärjestelmä, joka pyörii tietokoneissa. Tällöin käyttöjärjestelmäversion valmistuttua puhelimille on käytössä heti laajasti ohjelmia. Tämä eroaa esimerkiksi Microsoftin tavasta tuottaa samoja ohjelmia tietokoneille ja puhelimille. Vaikka Microsoft tai kolmannen osapuolen sovelluskehittäjä tekevät sovelluksen tietokoneille, se ei suoraan toimi tai ole edes asennettavissa puhelimissa. Tällöin joudutaan tekemään vastaavat sovellukset erikseen tietokoneille ja puhelimille.

Muina kiinnostavina kohteina voidaan pitää vahvasti HTML5-teknologiaan eli web-teknologiaan tukeutuvia käyttöjärjestelmiä Firefox OS:ää ja Tizeniä. Sovellukset toteutetaan kyseisille käyttöjärjestelmille HTML5:llä. Näin ollen voidaan olettaa, että HTML5-sovellusten määrä kasvaa myös muilla käyttöjärjestelmillä. Tämä johtuu siitä, että HTML5-sovellukset toimivat alustariippumattomasti. Vain pienillä optimoinneilla sovellukset saadaan toimimaan laajasti kaikilla merkittävillä alustoilla.

Firefox OS on avoimen lähdekoodin käyttöjärjestelmä mobiililaitteille. Pääkehittäjänä toimii Firefox-selaimesta tunnettu Mozilla Corporation. Erikoisuutena ja etuna muihin kilpaileviin käyttöjärjestelmiin on, että se sallii HTML5-sovellusten kommunikoida suoraan laitteiston kanssa. Näin ollen myös koko käyttöliittymä on käytännössä yksi web-sovellus. Saadakseen ”Powered By Firefox OS” -sertifikaatin laitteeseen valmistajalta vaaditaan vain hyvin vähäiset vaatimukset laitteen sovellusten ja laitteiston suhteen. Toistaiseksi kaikki tällä hetkellä myynnissä olevat Firefox OS -laitteet kuuluvat älypuhelinten low-end-kategoriaan. Tämä tarkoittaa halvempia, halvemmalla laitteistolla ja vähemmillä ominaisuuksilla varustettuja puhelimia. [27;35;36.]

Tizen on Linux Foundationin tukema avoimen lähdekoodin projekti, jonka kehityksen suuntaa ohjaavat pääsiallisesti Intel ja Samsung. Tizenin on käyttöjärjestelmänä ja alustana, muista kilpailijoista poiketen, tarkoitus toimia useissa eri laitteissa, mutta Firefoxista tuttuun tapaan kehittäjille tarjottavassa sovelluskehityspaketilla voidaan luoda vain HTML5-teknologiaan perustuvia sovelluksia. Toistaiseksi Tizen-käyttöjärjestelmällä julkaistuja laitteita löytyy vain seuraavista kategorioista: mobiililaitteista, puettavasta elektroniikasta eli älykelloista sekä kameroista. Tulevaisuudessa Tizenin on tarkoitus löytyä myös televisioista, tableteista, tietokoneista, tulostimista, autoista sekä älykkäästä kodin elektroniikasta. Tizenin maalaamat tulevaisuuden suunnitelmat voivat hyvinkin onnistua suuren yhteistyökumppanien määrän sekä rahakkaiden tukijoiden ansiosta. [3;33;70;85.]

3 Ekosysteemit

Ekosysteemillä tarkoitetaan yhtenäisen ympäristön muodostamaa toiminnallista kokonaisuutta. Se sisältää esimerkkinä käyttöjärjestelmän, kauppapaikan, sovelluskehitysvälineet, pilvi- ja muut yhteensopivat palvelut. Näiden lisäksi tärkeässä asemassa osana ekosysteemiä ovat myös käyttäjät, kehittäjät ja operaattorit. Työssä tarkastellaan kolmen merkittävämmän käyttöjärjestelmäntarjoajan palveluita ja tuotteita. Nämä kolme ovat Apple, Google ja Microsoft.

3.1 Apple

Applen ekosysteemi eroaa Googlen ja Microsoftin ekosysteemeistä siten, että sillä on tiukemmat rajoitukset siitä, mitkä laitteet ja palvelut toimivat yhteen. Pelkästään Applen tietokoneiden käyttöjärjestelmä OS X on asennettavissa vain Applen omille tuotteille. Sovelluskehitys iOS-käyttöjärjestelmälle taas vaatii Xcodea, joka on asennettavissa vain OS X -käyttöjärjestelmälle. Näin ollen natiivi- ja hybiridisovelliskehitys iOS:lle tapahtuu aina Applen omilla tuotteilla.

Applen ja iOS-käyttöjärjestelmän virallisena kauppapaikkana toimii App Store. Kauppa julkaistiin vuonna 2008. App Store on esiasennettuna iPhonessa. Kaupassa on ladattavissa yli 1 200 000 sovellusta. App Store on jaettu osiin. Esimerkiksi Euroopassa kauppa

on jaettu maittain. Suomen App Storesta voi löytyä ohjelma, jota ei Ruotsissa ole saatavissa. App Storessa myydyistä sovelluksista tekijä saa hinnasta 70 % itselleen. [1;20;22;75.]

Applen pilvipalvelu on nimeltään iCloud. iCloudia voidaan käyttää muun muassa tiedon varmuuskopiointiin ja synkronointiin. Pilven kautta voidaan myös suoratoistaa mediaa eri laitteilla. Sen kautta voi myös paikantaa laitteen, mikäli laitteessa on virta päällä.

3.2 Google

Googlen ekosysteemissä on vähiten rajoitteita verrattaessa sitä Microsoftin ja Applen ekosysteemeihin. Tämä johtuu pääasiassa siitä, että Googlen historia juontuu hakukonepalvelusta. Googlen palvelut kuten Gmail ja Youtube ovat olleet käytössä Microsoftin ja Applen tuotteilla jo ennen Android-käyttöjärjestelmää.

Googlen ja Android-käyttöjärjestelmän virallisena kauppapaikkana toimii Google Play. Se tarjoaa sovelluksia, elokuvia, musiikkia, kirjoja ja pelejä. Google Playtä voi käyttää joko puhelimella tai selaimella. Kun palvelusta ostaa sovelluksia selaimella, asentuvat ne suoraan puhelimeen. Palvelussa on tarjolla yli 1 300 000 sovellusta.[17.]

Googlen pilven nimi on Google Drive. Palvelua voidaan käyttää varmuuskopiointiin ja tiedon synkronointiin. Se tarjoaa myös selainpohjaisen dokumenttien käsittelyn. [74.]

Android-sovelluskehitys tapahtuu Eclipse:llä. Eclipse toimii useilla laajasti käytetyillä käyttöjärjestelmillä muun muassa OS X:llä, Windowsilla ja Ubuntulla.

3.3 Microsoft

Microsoftin ekosysteemiin kuuluu laajalti Windows-ympäristöstä tuttuja palveluita. Näitä ovat muun muassa Office-ohjelmat, IE sekä OneDrive. Myös Microsoftin viihdepalvelut kuten pelitilit (Xbox-tunnus) voidaan synkronoida mobiililaitteisiin.

Microsoftin ja Windows Phone -käyttöjärjestelmän virallisena kauppapaikkana toimii Windows Phone -kauppa. Kaupan käyttö on kuluttajille ilmaista, mutta sovelluksien

myyjä joutuu maksamaan vuosittaisen summan, jotta voi laittaa sovelluksia myyntiin. [77.]

Microsoftin pilvi kulkee nimellä OneDrive (aikaisemmin SkyDrive). OneDriveä voidaan käyttää tiedon vamuuskopioimiseen ja synkronointiin. OneDrive sisältää muun muassa myös erilaisia verkkosovelluksia dokumenttien käsittelyyn, kuten Office Web Appsin. [84.]

Windows Phone-sovelluskehitys tapahtuu Visual Studiolla ja ohjelma toimii Windows-käyttöjärjestelmällä. Ubuntulla on myös mahdollista saada Visual Studio toimimaan, mutta sitä kannattaa käyttää mieluummin Windowsilla, koska silloin sen asennus on helpompaa ja toiminta varmempaa. Windows-käyttöjärjestelmän laitevaatimuksetkaan eivät ole kovin vaativia, joten tästäkin syystä on suositeltua käyttää Windowsia.

4 Sovelluskehitysmenetelmät

Sovelluskehitysmenetelmät alustoille voidaan jaotella kolmeen ryhmään: natiivi, hybridi ja HTML5. Natiivi- ja hybridisovelluskehtiys vaativat alustakohtaista natiivikielen osaamista. HTML5:llä toteutettu sovellus käyttää lähtökohtaisesti HTML:ää, JavaScriptiä ja CSS-tyylejä. Kaikki kolme alustaa, iOS, Android ja Windows Phone, tukevat HTML5:ttä, joten yksi HTML5-sovellus voidaan optimoida toimimaan kaikilla alustoille. HTML5-sovellukset eroavat natiivi- ja hybridisovelluksista siinä, ettei niitä jaeta kauppapaikkojen kautta, vaan ne toimivat verkkosivuina. HTML5 tukee sivujen lataamista puhelimeen, jolloin niitä on mahdollista käyttää myös offline-tilassa. [67;68;69.]

Sovelluskehitystä aloittaessa on hyvä tietää etukäteen sovelluksen vaatimat tarpeet. Riittääkö yhdelle alustalle toteutus, pitääkö sovelluksen päästä käsiksi puhelimen kontakteihin, halutaanko sovellusta myydä ja niin edelleen. Myös oma osaaminen on ratkaiseva tekijä. Oppimiskäyrät eri kehitystekniikoilla ja esimerkiksi pelkästään natiivitekniikoilla eroavat suuresti. Toisin sanoen HTML5 on nopeammin sisäistettävissä kuin natiivikielet ja natiivikielistä C# on helpoin oppia.

4.1 Natiivi

Natiivisovelluskehitys tapahtuu laitekohtaisella natiivikielellä. Natiivisti suoritettavien sovellusten rajapinta päivittyy usein verrattaessa HTML5-standardiin. Esimerkiksi Androidiin on tullut viimeisen neljän vuoden aikana yli kymmenen rajapintapäivitystä. Androidin API taso 7 on julkaistu tammikuussa 2010, kun nyt ollaan jo API tasolla 21. Uudet rajapinnat ovat tuoneet mukanaan uusia ominaisuuksia, muutoksia vanhoihin ja myös poistoja. Laitteiden kehittyessä tulee myös uutta laitteistoa, jota ei ole otettu huomioon vanhoissa rajapinnoissa. Näin ollen rajapintoja on hyvä päivittää tiiviiseen tahtiin, jotta raudan kaikki ominaisuudet saadaan käyttöön myös kolmansien osapuolien kautta. [6;14.]

Natiiviohjelmistorajapintojen kanssa voi olla lähes varma, että ne toimivat määritetyllä tavalla. Uusien rajapintojen kanssa on kuitenkin ongelmana se, että vanhemmat laitteet eivät ole aina päivitettävissä tukemaan niitä.

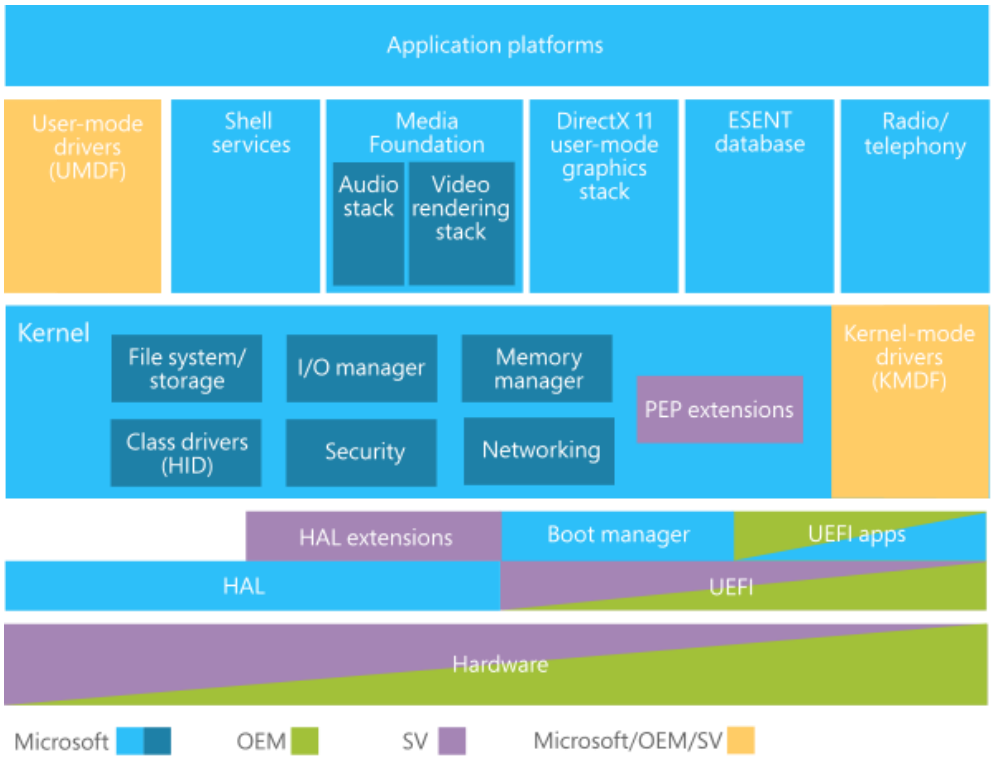
Natiivisovelluskehityksen etuna HTML5:n nähden ovat laajemmat rajapinnat. Natiivina toteutetussa sovelluksessa päästään paremmin kiinni laitteistoon. Esimerkiksi push-viestien lähettäminen laitteeseen puhtaalla HTML5:llä ei onnistu. Push-viestit ovat palvelimien pyyntöjä vuorovaikutteisille toiminnoille. [72.]

Virheenkorjausta varten on natiivisovelluskehitysvälineissä hyvät ominaisuudet. Kaikilla kolmella alustalla IDE:t näyttävät sovellusten rakenteen selkeästi. Näin ollen päästään laittamaan virheenkorjausta varten pysähdyspisteitä helposti haluttuihin kohtiin. Parametrien arvoja voidaan vaihtaa ajon aikana ja virheitä voidaan tuottaa hallitusti. Sovelluskehittäjä voi tällä tavalla tuottaa erilaisia virheitä ja toteuttaa niihin haluamansa virheenhallintamenettelyn. Virheenkäsittelyn testaaminen onnistuu tällä tavalla helposti ja voidaan varmistaa, ettei sovellus kaadu virheisiin.

4.1.1 Windows Phone

Windows Phonen arkkitehtuuri on pitkälti Microsoftin omaa käsialaa. Kuvassa 2 nähdään Windows Phonen arkkitehtuuri ja se, kuka on toteuttanut siitä mitkäkin osuudet. Nämä on esitetty kuvassa eri värityksillä. Komponentit ja niiden laiteohjelmisto ovat niiden valmistajien käsialaa, mutta niiden päälle toteutetut ajurit, toiminnot ja ohjelmisto ovat vähintään Microsoftin ja yhteistyökumppaneiden yhteistä toteutusta. Tietoturva ja sovellusalusta ovat täysin Microsoftin omaa toteutusta. Tämän perusteella voitaisiin olettaa,

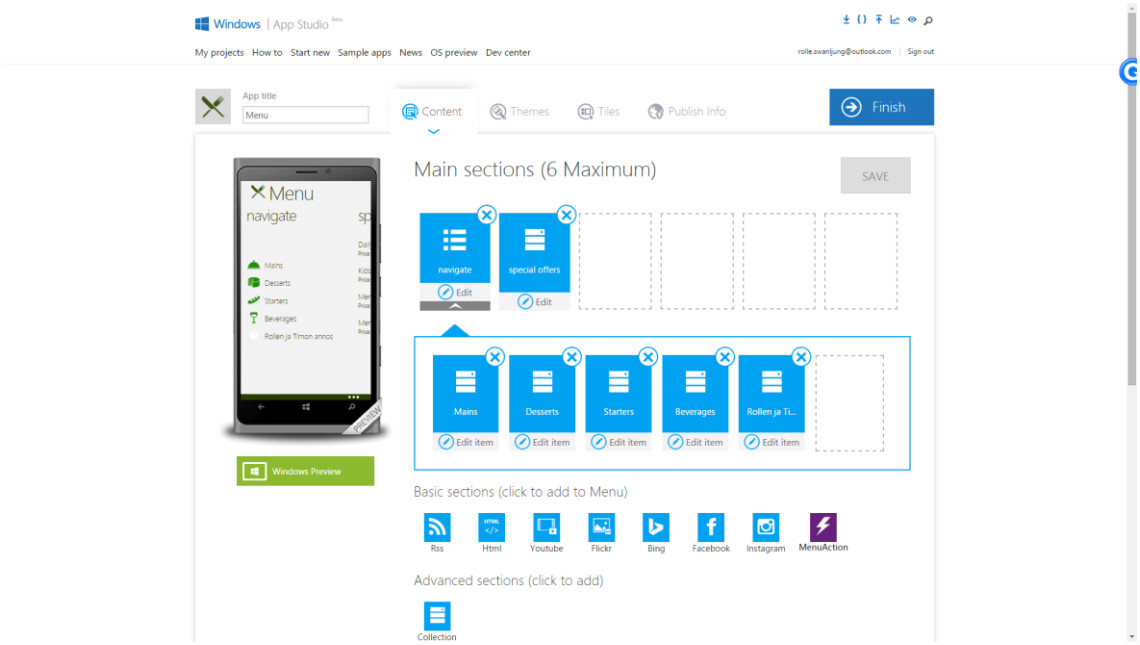
että puhelimesta saataisiin helposti kaikki tieto ja ominaisuudet ulos rajapintojen kautta. Näin ei kuitenkaan ole. Microsoftin mukaan se on tietoisesti rajoittanut API:a tietoturvasyistä. Kuulostaa kuitenkin erikoiselta, ettei sovelluskehittäjä esimerkiksi voi toteuttaa sovellusta, jolla pystyttäisiin vaihtamaan puhelimen värinähälytystä pois tai päälle, sillä sen ei pitäisi olla tietoturvariski.



Kuva 2. Windows Phonen arkkitehtuuri. [100.]

Windows Phone -kehitys tapahtuu yleensä Visual Studiolla, johon ladataan Windows Phone SDK -laajennus. Visual Studiosta löytyy useita maksullisia versioita, mutta myös ilmainen mobiilisovelluskehitystä varten Visual Studio Express. Mikäli sovelluskehitystä halutaan tehdä Windows Phone 8:lle, vaaditaan Visual Studion asennusta varten Windows 8 -käyttöjärjestelmä tai uudempi.

Microsoft on tehnyt ohjelman mobiilisovelluskehitystä varten myös niitä henkilöitä ajatellen, jotka eivät osaa tuottaa omaa koodia. Tämä tapahtuu Windows App Studiolla (kuva 3). Windows App Studiota käyttämällä voi selaimella toteuttaa helposti sovelluksen.



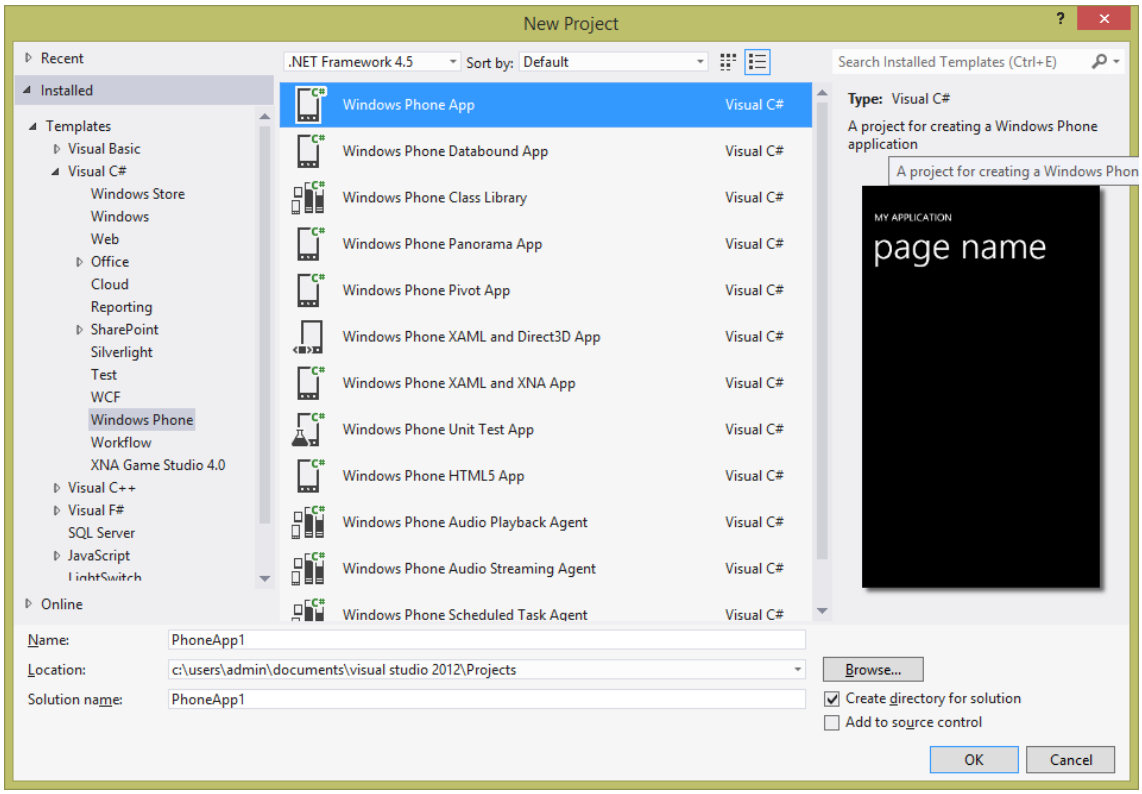
Kuva 3. Windows App Studion Menu-pohjalla voi helposti luoda ravintolalle ruokalistan. [98.]

Windows App Studio tarjoaa valmiita pohjia sovellusten luontiin. Vaikka pohjat ovat yksinkertaistettuja, helpottavat ne suuresti noviiseja ohjelmoijia luomaan sovelluksia.

Natiivisovelluskehitys Windows Phonelle tehdään joko käyttämällä C#-, Visual Basic-,.NET-, C++- tai HTML5-kieliä. HTML5:n osalta Microsoftin tarjoama API on puutteellinen, ja usein joudutaan turvautumaan kolmansien osapuolien ratkaisuihin. Microsoftin sivuilla on kuitenkin mahdollisuus kommentoida esimerkkejä, joissa kolmannet osapuolet korjaavat Microsoftin virheitä. Microsoft päivittää virheellisiä lähdekoodeja toimiviksi jatkuvasti.

Yleisesti C#:n sisäistämistä pidetään helpompana kuin muilla alustoilla käytettävien Javan ja Objective-C:n. Ero ei kuitenkaan ole niin merkittävä Javaan kuin Objective-C:hen. C# on korkean tason ohjelmointikieli, joka on niin sanottua hallittua koodia (engl. Managed code). .Net sovelluskehyksessä on mahdollista käyttää useampiakin korkean tason ohjelmointikieliä kuten esimerkiksi #J:tä, VB:tä, C++:aa. Nämä useat korkean tason ohjelmointikielet jakavat yhdistetyn luokkakirjaston ja ne on mahdollista kääntää välikielelle (engl. Common Intermediate Language). CLR puolestaan kääntää välikielen laitteen suoritettavaksi natiivikoodiksi. CLR-kääntäjä tarkistaa koodista mahdollisia virheitä sekä suorittaa roskankeräyksen, joten kehittäjän ei tarvitse välittää muistin hallinnasta. Tällaista teknologiaa käyttämällä saadaan ohjelmoinnista huomattavasti tehokkaampaa. [28;99;96.]

Visual Studiolla sovelluksen runko saadaan valmiina. Vaihtoehtoja on useita riippuen, minkälaista sovellusta ollaan toteuttamassa (kuva 4). Visual Studiosta löytyvä Windows Phone App on tyhjä runko, josta voi aloittaa toteutuksen.

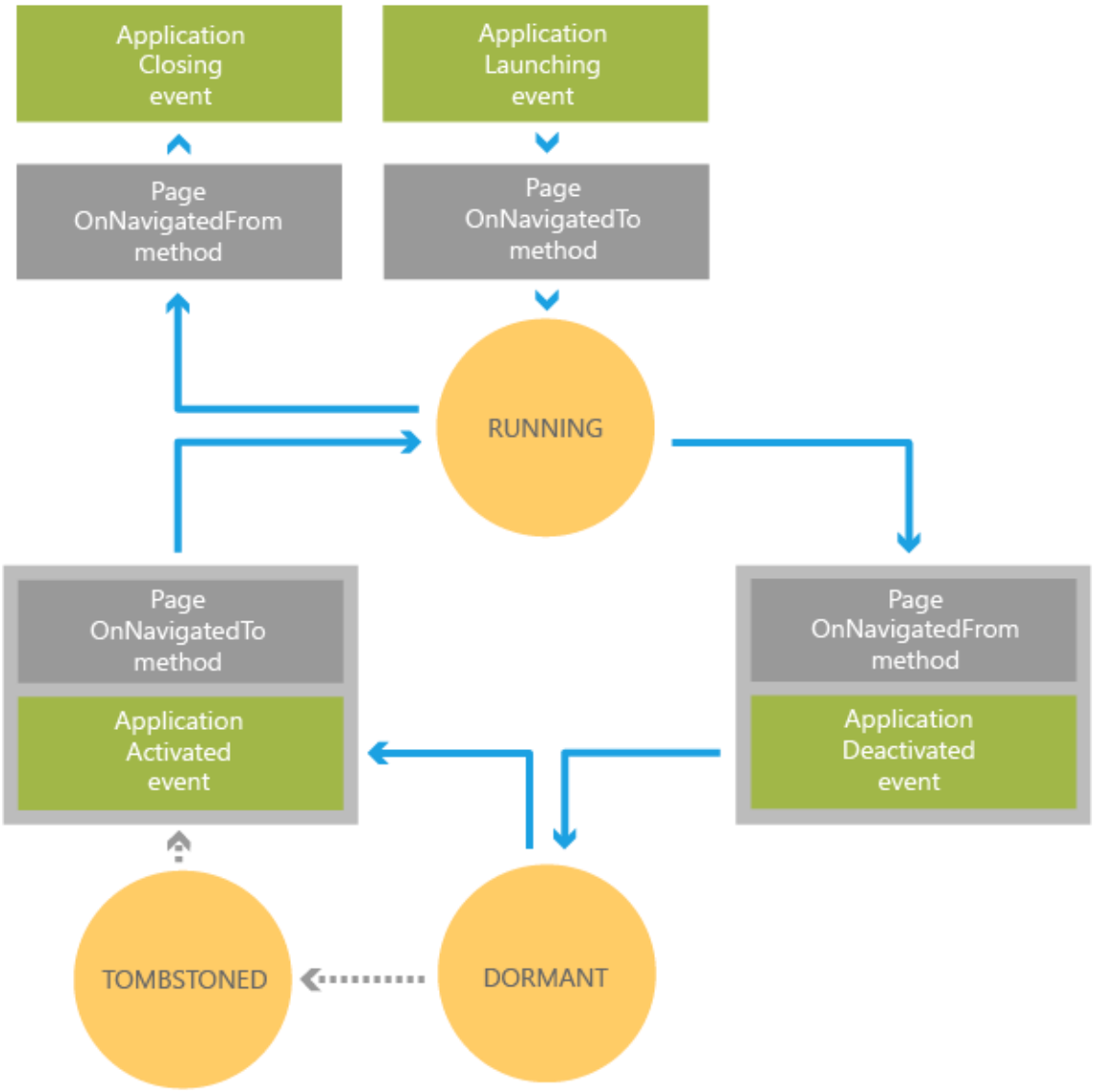


Kuva 4. Visual Studio tarjoaa useita erilaisia pohjia Windows Phone sovelluksen toteutusta varten.

Windows Phonen sovelluksia voidaan toteutusvaiheessa ajaa fyysisillä laitteilla tai emulaattoreilla. Windows Phone SDK sisältää mahdollisuuden ajaa mobiililaitteiden emulaattoreita. Näiden avulla on helppo testata sovellusta eri laitteilla, ilman niiden fyysistä omistamista. Ilman tätä ominaisuutta käyttöliittymän toteutus olisi paljon vaikeampaa, koska laitteiden näyttöjen koot vaihtelevat.

Windows Phonen sovellusten elinkaari eroaa iOS- ja Android-sovelluksista. Sovelluksesta poistuttaessa sulkematta sitä se siirtyy horrostilaan (kuva 5). Tällöin sovelluksen säikeet pysähtyvät. Tästä tilasta sovellus voidaan palauttaa ilman, että tiedot häviävät. Tiedot säilytetään väliaikaisesti muistissa. Mikäli muisti ei riitä säilyttämään tietoa, tiedot siirretään niin sanottuun hautakiveen (tombstone), mikäli se on ohjelmoitu sovellukseen.

Muistin loputtua tai sovelluksen sammutuksen jälkeen tiedot häviävät, ellei niitä tallenna hautakiveen. Sovellus voi ladata aikaisemman tilansa hautakivestä sen uudelleen käynnistyessä. Tässä on kuitenkin rajoituksensa. Hautakivessä voi säilyttää vain viiden sovelluksen tiedot samaan aikaan.



Kuva 5. Sovelluksen elinkaari Windows Phonella. [16.]

Kuten kuvasta 5 nähdään, sovellus voidaan toteuttaa myös ilman hautakivitoimintoa. Tällöin sovelluksen käynnistäminen tapahtuu aina kuin se käynnistettäisiin ensimmäistä kertaa. Tähänkin on poikkeuksia, jos tietoa haetaan esimerkiksi tietokannasta.

Sovelluksesta lähdettäessä kutsutaan aina OnNavigatedFrom-metodia. Tuossa metodissa voidaan tehdä tallennus hautakiveen tai muita sulkemisen yhteydessä haluttuja toimintoja.

Sovelluksen käynnistyessä tai sen siirtyessä aktiiviseen tilaan kutsutaan aina OnNavigatedTo-metodia. Tässä metodissa voidaan palauttaa tieto hautakivestä tai tehdä muita haluttuja toimintoja, kuten esimerkiksi tiedon päivitys internetistä.

Visual Studion versionhallintajärjestelminä toimii Microsoftin oma Team Foundation Server. Visual Studiosta löytyy myös verkossa pyörivä versio Visual Studio Online. Tässä versiossa versionhallintajärjestelminä toimivat Git Hub ja Team Foundation Version Control. Molemmista löytyy ilmainen rajoitettu versio. Visual Studio Onlinella voidaan olla yhteydessä Visual Studioon, Eclipseen ja Xcodeen. Sovelluskehitys on tällä mahdollista useille alustoille ja useilla kielillä. [97.]

Visual Studio 2013 pitää sisällään Blend for Visual Studion (aikaisemmin erillisenä sovelluksen toimineen Expression Blendin). Tällä pystyy toteuttamaan XAML-pohjaisia käyttöliittymiä Windows Phoneille. Työkalua voidaan käyttää myös prototyyppien toteuttamiseen. Blend for Visual Studio perustuu WYSIWYG-tekniikkaan, ja näin ollen on kätevä työkalu esimerkiksi käyttöliittymien suunnittelijoille, joilla ei ole erityisemmin sovelluskehitystaustaa.

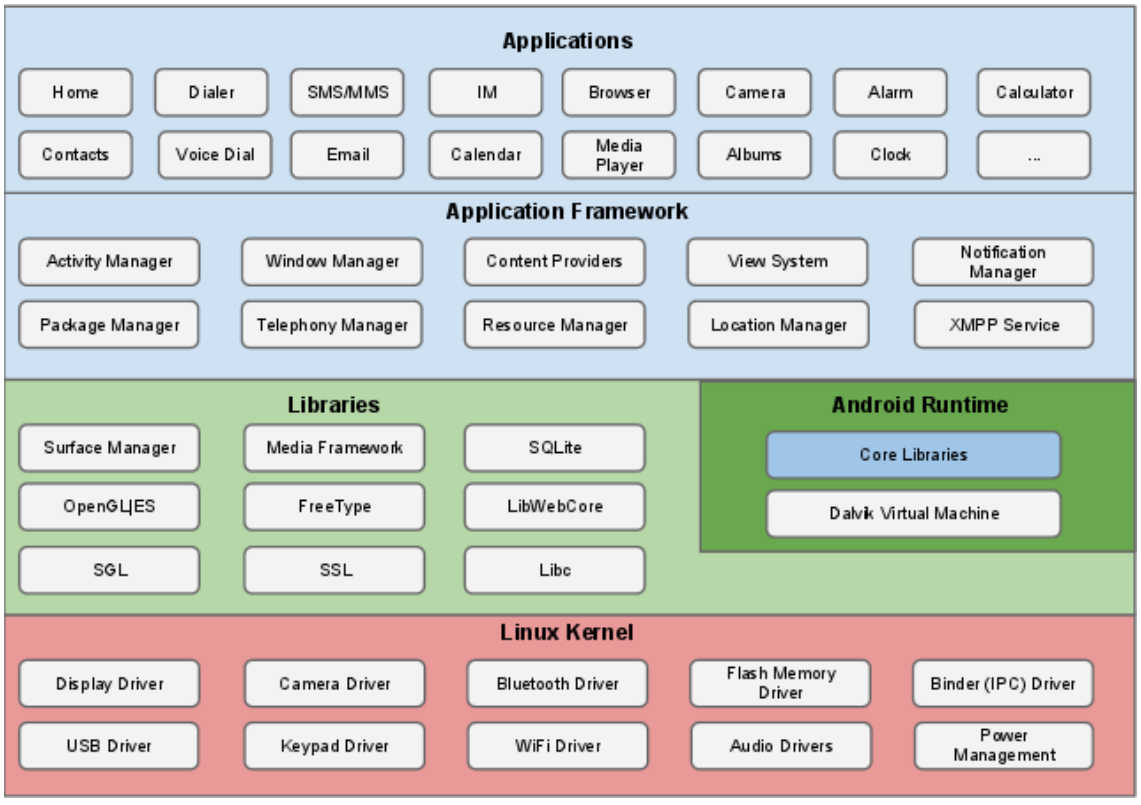
Windows Phonen beta-testausta voidaan toteuttaa kahdella tapaa. Sovellus lähetetään jakoon Windows Dev Centerin kautta, jolloin sitä ei näy vielä kaupassa. Sovelluksen toteuttaja voi valita joko beta-julkaisun tai kohdennetun sovellusjakelun. Beta-julkaisussa vain määritetyt käyttäjät voivat asentaa sovelluksen. Kohdennetun sovelluksen voi ladata kuka vain tietyn URL:in takaa. [25.]

Valmis natiivisovellus Windows Phonelle voidaan julkaista kauppaan. Kaupassa sovellusten julkaisu vaatii tunnukset ja vuosittain perittävän maksun. Rekisteröityminen maksaa tällä hetkellä Suomessa yksityishenkilölle 14 euroa. Sovelluksen julkaisija saa itse määritellä sovelluksen hinnan. Sovellusta julkaistaessa sovellus käy läpi sertifiointivaatimukset. Vaatimukset on jaoteltu sovelluskäytäntöihin, sisältökäytäntöihin, sovelluksen esittämiin vaatimuksiin, teknisiin vaatimuksiin ja lisävaatimuksiin, jotka on erityisesti suunnattu Windows Phonelle. Sovelluksen vaatimusten läpikäynti kestää 5-7 arkipäivää. Mikäli sovellus läpäisee tarkistuksen, se julkaistaan Windows Phone -kaupassa. Tämän

jälkeen sovelluksen myyjä voi seurata muun muassa latausten määrää eri laitteille eri päivinä. [4;18;30.]

4.1.2 Android

Android ja Windows Phone eroavat iOS:sta sisältämällä virtuaalikoneen. Kuvassa 6 nähdään, kuinka Android-käyttöjärjestelmän ohjelmistopino koostuu Linux-kernelistä, Java-virtuaalikoneesta, kirjastoista sekä ohjelmistokehyksestä. Kaikki sovellukset mukaan lukien järjestelmäsovellukset ja kolmannen osapuolen sovellukset suoritetaan virtuaalikoneella, joka sijaitsee Java-kirjastojen päällä. Androidissa ei käytetä Javan omaa virtuaalikonetta vaan varta vasten Android-projektia varten luotua mobiililaitteelle optimoitua Dalvik-virtuaalikonetta. [10.]



Kuva 6. Arkkitehtuurillinen kuva Android-laitteen ohjelmistopinosta. [10.]

Android 4.4:ssä (KitKat) on mahdollista valita käyttöön vaihtoehtoisesti ART (Android runtime) -virtuaalikone. Uudessa Android 5.0:ssa (Lollipop) on Dalvik jo korvattu ART:lla. ART tarjoaa pääasiassa parannuksia käyttöjärjestelmän suorituskykyyn ja vianmäärityk-

seen. Suorituskykyä on saatu paranemaan mm. kehittämällä roskankeräystä sekä muuttamalla Java-tavukoodin kääntämisen JIT:stä AOT:iin. Suurin hyöty saadaan juuri koodin kääntämismekanismia muuttamalla. AOT:ssa Java-koodi käännetään tavukoodiksi asennuksen yhteydessä ja .dex-tiedostoista(.class) muodostetaan suoritettava paketti. JIT:ssa puolestaan tavukoodi käännetään suorituksen yhteydessä vain, mikäli se on tarpeen. [8;91.]

Android-sovelluskehitys Eclipsellä tapahtuu pääasiassa Javalla. Sitä voidaan myös tehdä C- ja C++-kielillä, mutta se ei ole suositeltavaa ilman pätevää syytä. Pelkästään halu käyttää näitä kieliä ei ole riittävä syy. Syyksi kelpaa esimerkiksi keskusyksikköä paljon rasittavien toimintojen toteutus, kuten pelimoottorien toteutus.

Androidin ja Googlen kanta sovellusten asentamiseen on hyvin avoin. Sovelluksia pystyy lataamaan muualtakin kuin Play-kaupasta. Androidin API tarjoaa myös laajan pääsyn kiinni puhelimen ominaisuuksiin ja resursseihin. Nämä yhdessä takaavat mahdollisuuden tehdä monipuolisia sovelluksia, mutta sisältävät myös tietoturvariskin käyttäjille.

Sovellusta asentaessa Android-laitteeseen sovellus ilmoittaa, mitä suojattuja rajapintoja se käyttää. Tällaisia ovat esimerkiksi kamera, gps, kontaktit ja muut puhelimessa olevat ominaisuudet ja tieto, mitä käyttäjä ei välttämättä halua luovuttaa. Loppukäyttäjällä on tällöin tieto mahdollisista tietoturvariskeistä. Tällä tavalla Google jakaa loppukäyttäjille huomattavasti enemmän vastuuta sovellusten asennusten suhteen kuin Microsoft tai Apple. Näin ollen Googlen ei täydy tehdä tarkkoja tarkastuksia sovelluskaupan sovelluksille, koska vastuu niistä jää itse käyttäjille. Käyttäjä ottaa siis tietoisen riskin antaessaan sovelluksen käyttää esimerkiksi kameraa luottaen sovelluksen käyttävän sitä vain kuvatuissa tilanteissa. Vastaavia vaatimuksia sovelluksien käyttämistä rajapinnoista ei Applen ja Microsoftin laitteissa esitetä, koska sovellusten tarkistusprosessi ennen julkistamista ekosysteemien virallisille kauppapaikoille on paljon vaativampi.

Android-sovelluksia on mahdollista kehittää lukuisilla eri kolmannen osapuolen sovelluskehitystyökaluilla. Kehittäjäyhteisön virallisesti tukema IDE on kuitenkin ADT:lla varustettu Eclipse. Eclipsestä on toteutettu versiot Linuxille, Mac OS X:lle sekä Windowsille. Lähitulevaisuudessa tulee myös saataville virallinen versio Android Studiosta, joka julkaistiin beta-testaukseen Google-kehittäjä konferenssissa 2013. [11;12;34;61;73.]

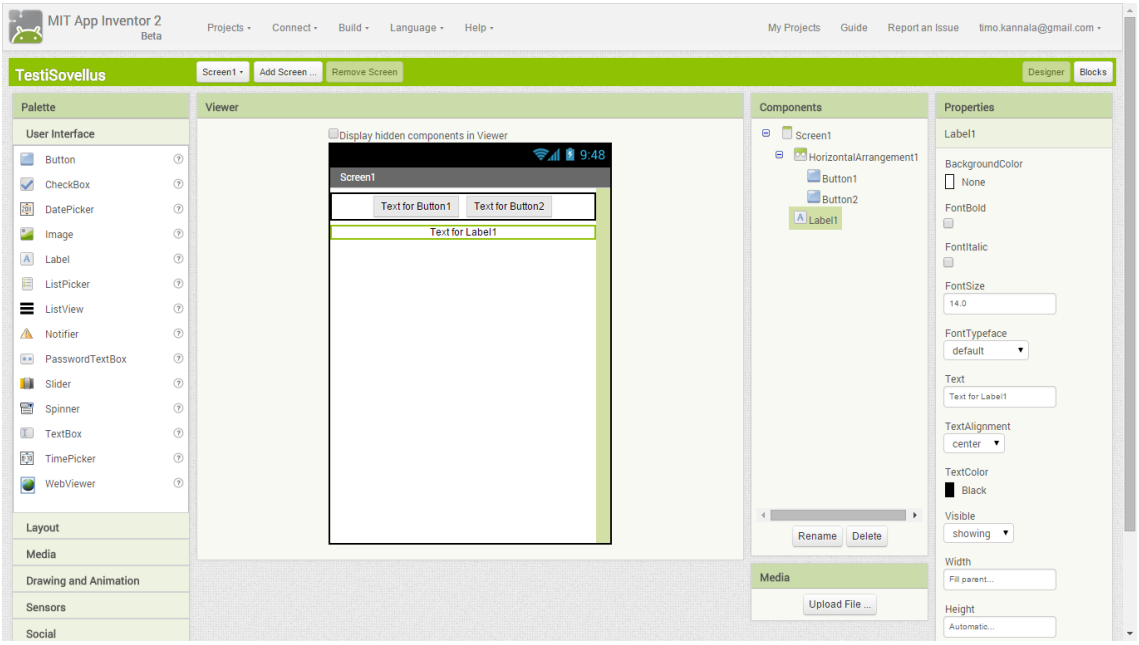
Android Studio on Googlen kehittämä tuote, joka perustuu JetBrainsin IntelliJ IDEA Java IDE:een. Android Studio on avointa lähdekoodia niin kuin myös IntelliJ IDEA. Eclipsen ja ADT:n yhdistelmä verrattuna Android Studio tarjoaa paljon uusia ominaisuuksia sekä useita parannuksia (taulukko 1). Sisäänrakennettu tuki Google Cloud Messageihin ja App Engineen sekä laajennettu tuki Googlen muihin palveluihin helpottavat sovelluksen integrointia Googlen tarjoamien pilvipalveluin kanssa. App Studiossa on myös työkalu laitteen suorituskyvyn, käytettävyyden sekä versioyhteensopivuuksien tarkkailuun. Sitä ei Eclipsestä ei löydy. NDK-tuki, jolla voidaan toteuttaa sovelluksia C- ja C++ -kielillä, puuttuu vielä Android Studiosta, mutta on tulossa. Android Studio käyttää myös uudempaa Gradle-koontityökalua. Tähän on kerätty hyviä puolia Ant- ja Maven –koontityökaluista. [9;11.]

Taulukko 1. Android Studion ja Eclipsen ominaisuusvertailu. [11.]

Ominaisuus	**Android Studio**	**Eclipse**
Koonti	Gradle	Ant
Mavenin koonnin riippuvuusmekanismi	Kyllä	Ei
Asennus-paketin käännökset useille eri sukupolville (Android Wear-sukupolvi)	Kyllä	Ei
Kehittynyt koodin täydennys sekä refaktorointiominaisuudet	Kyllä	Ei
Graafinen käyttöliittymätyökalu	Kyllä	Kyllä
Sovelluspaketin allekirjoituksen hallinta	Kyllä	Kyllä
NDK-tuki	Tulossa	Kyllä

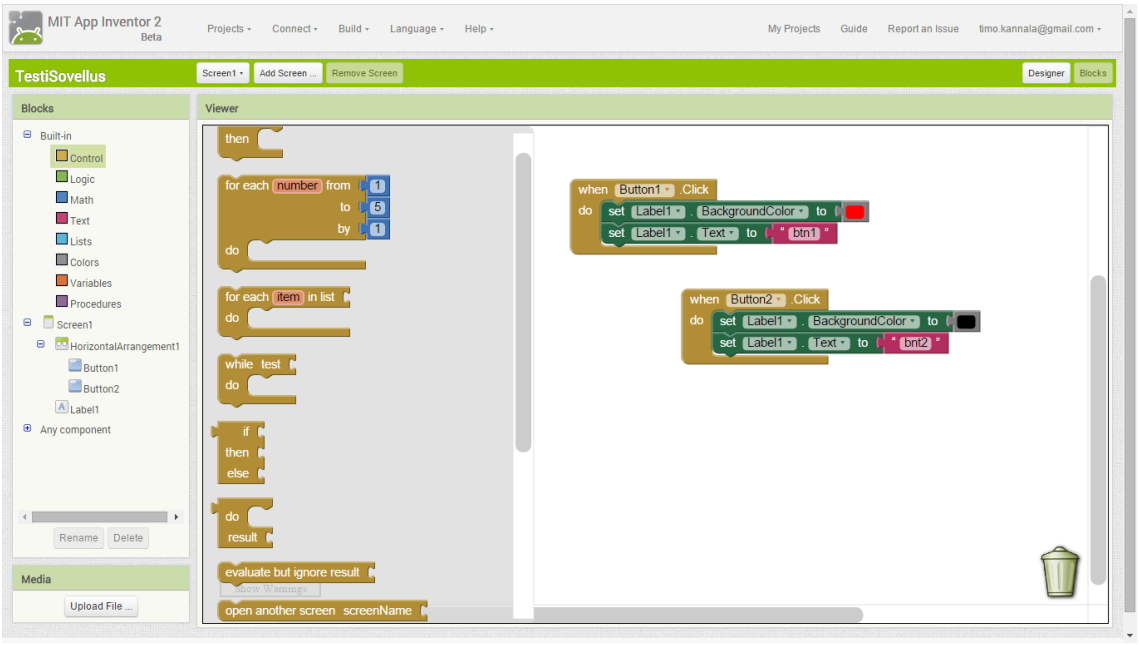
Android-sovelluksia on myös mahdollista luoda käyttämällä MIT:n ylläpitämää selaimella toimivaa App Inventor BETA -pilvisovellusta, joka näkyy kuvassa 7. App Inventor sopii hyvin noviisille ohjelmoijalle, koska alkuun pääsemiseen riittää vain Googlen tunnus eikä varsinaista ohjelmointia tarvitse osata. Luotavan sovelluksen virheenkorjaukseen on tarjolla useampi vaihtoehto, sovellusta voi testata joko USB-kaapelin avulla, langattomasti tai emulaattorilla. Huomioitavaa on, että palvelu on vielä beta-testausvaiheessa. [62.]

Sovellusta luotaessa kehittäjä ei pääse vaikuttamaan sovellusprojektin tekniseen rakenteeseen, vaan kehittäjälle tarjotaan vain minimaalinen näkymä lähinnä käyttöliittymän suunnitteluun. Osa sovelluksen konfiguraatiosta generoituu itsestään, mikä Eclipsessä pitää tehdä yleensä manuaalisesti: esimerkiksi aktiviteetti-instanssien elinkaarenhallinta ja erilaisten medioiden versiointi eri näytöille.



Kuva 7. App Inventorin graafisen suunnittelun näkymä

Käyttöliittymää voidaan rakentaa graafisesti Eclipsestä tutulla tavalla raahaamalla elementtejä valittuihin näyttöihin ja antamalla niille tarvittavat lisäominaisuudet.



Kuva 8. App Inventorin Blocks -näkymä sovelluksen logiikan luomiseen.

Sovelluksen logiikan luomista varten on palvelussa Blocks-näkymä (kuva 8). Blocks–näkymää käyttämällä sovelluksen käyttöliittymäelementteihin voidaan luoda logiikkaa

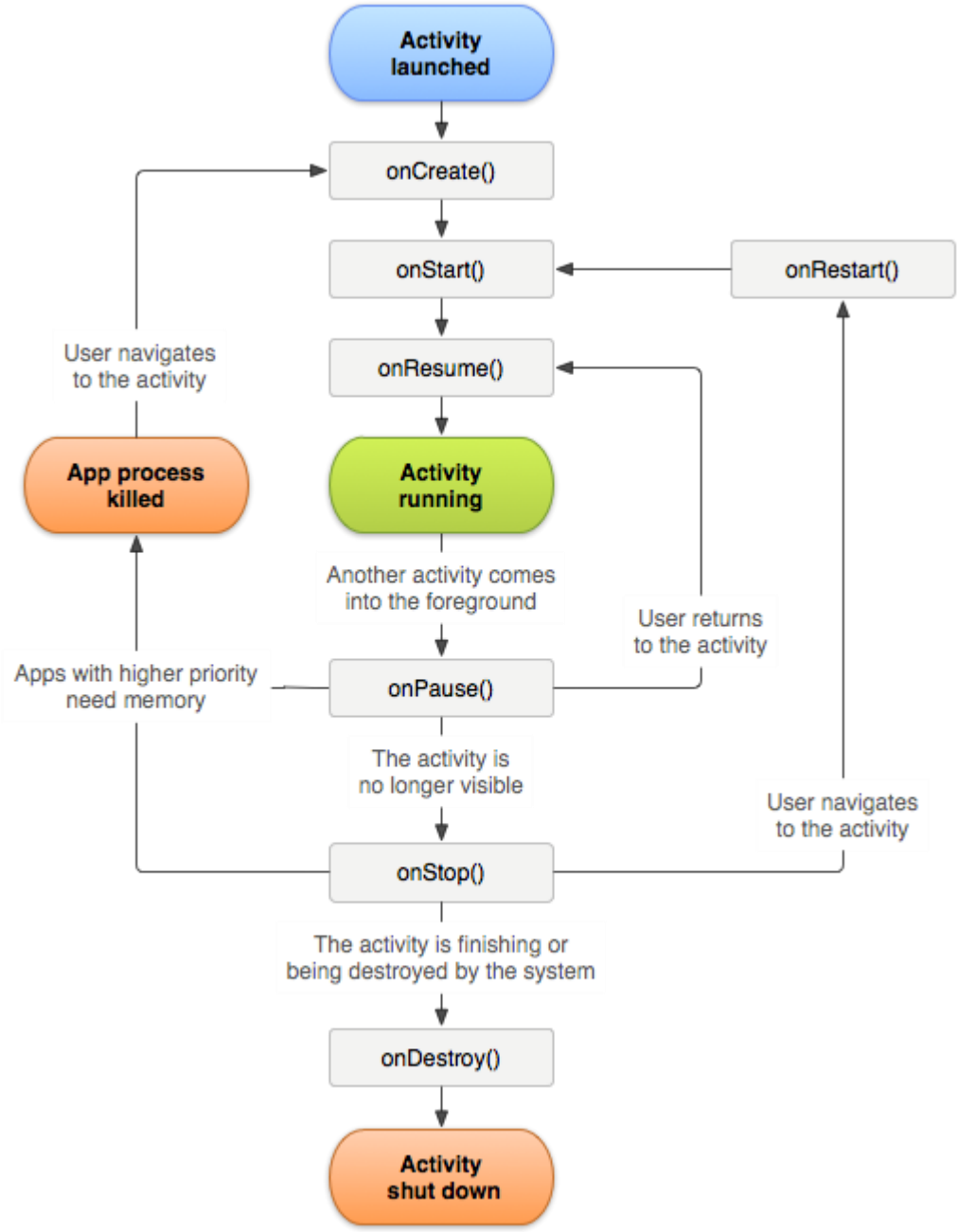
käyttäen valmiita koodipalikoita. Valmiita koodipalikoita käyttäen pystyy luomaan hyvin monimutkaisiakin logiikoita aina tiedostojen lukemiseen saakka.

Android-sovellukset perustuvat aktiviteetti-instansseihin. Näyttöä käännettäessä tai sovelluksesta ja ikkunasta toiseen siirryttäessä aktiviteettien instanssit liikkuvat eri tilojen välillä. Aktiviteetti-instanssin tilan muuttuessa Android-järjestelmä kutsuu sarjan erilaisia elinkaarimetodeita (kuva 8/taulukko 2). Elinkaarimetodeilla kehittäjä määrittelee, miten joku tietty aktiviteetti käyttäytyy, kun liikutaan sovellusten tai aktiviteettien välillä. Aktiviteetin luomiseen ja jatkamiseen liittyvissä metodeissa määritellään ja alustetaan käyttöliittymä sekä muut komponentit. Lopuissa metodeissa määritellään puolestaan data, jota halutaan tallentaa joko tilapäisesti tai pysyvästi. Voidaan esimerkiksi määritellä, miten videota suoratoistava sovellus toimisi puhelun saapuessa. Sovelluksen pysähtyessä suljetaan verkkoyhteys ja pysäytetään video. Jatkettaessa avataan verkkoyhteys uudestaan ja jatketaan kohdasta, johon jäätiin.

Taulukko 2. Android-aktiviteetin elinkaarimetodit selitettyinä.

onCreate()	Kun aktiviteetti aukeaa
onStart()	Kun aktiviteetti aukeaa
onRestart()	Kun aktiviteetti aukeaa
onResume()	Kun aktiviteettiin palataan
onPause()	Kun toinen aktiviteetti tulee näytölle
onStop()	Kun aktiviteetti ei enää ole näkyvillä
onDestroy()	Kun aktiviteetti suljetaan tai järjestelmä tuhoaa sen

Taulukossa 2 on selitetty Android-sovelluksen elinkaaren metodit, jotka näkyvät kuvassa 9.



Kuva 9. Android-sovelluksen elinkaari.

Eclipsessä versionhallintaohjelmina toimii yleisesti Subversion, CVS tai EGit. Jokainen näistä ohjelmista on ilmainen ja niistä löytyy graafinen käyttöliittymä. Tallennuspaikkana käytetään usein GitHubia.

Android-sovellusten julkaiseminen, myynti ja beta-testaus tapahtuu Google Play Developer Consolen kautta. Tämä on sovelluksen toteuttajalle tarkoitettu sivusto Google Playsta.

Sovelluskehittäjä voi laittaa sovelluksen beta-testattavaksi Google Play Developer Consolen kautta. Kehittäjä määrittää testaajat Google-ryhminä tai Google+ -yhteisöinä. Tämän jälkeen kehittäjä saa linkin, jonka välittää eteenpäin testaajille. Linkin kautta voi ilmoittautua testaajaksi.

Julkaisemalla sovellus Google Play Developer Consolen kautta vie näkyminen Play-kaupassa vain muutamia tunteja. Julkaisija voi seurata sovellukseen liittyviä erilaisia tilastoja Developer Consolen kautta. Mikäli sovellus on laitettu ilmaisena myyntiin, siitä ei voi tehdä maksullista jälkikäteen. Sovelluksen tekijä saa 70 % myyntihinnasta myydessään sovelluksia Google Playssa.

4.1.3 iOS

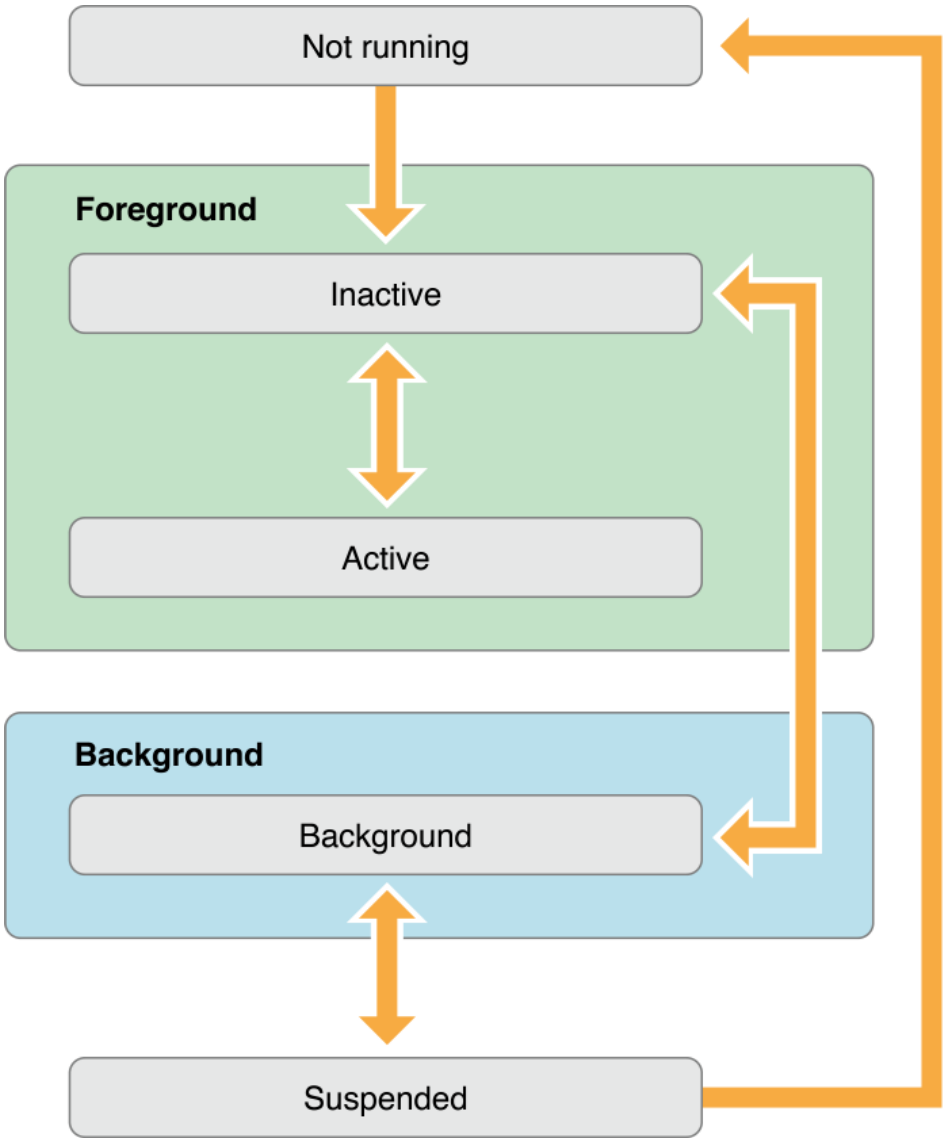
iOS-sovelluskehitys tapahtuu Xcodella ja iOS SDK:lla. Pääsovelluskehityskielinä toimivat Swift ja Objective-C. Tämän lisäksi voidaan käyttää C- ja Objective-C++-kieliä. C ja Objective-C++ toimivat yhdessä Swift-kielen kanssa. Myös Swiftiä ja Objective-C:tä voidaan ajaa rinnakkain. [24.]

Kuten aiemmin käytiin läpi, Xcode toimii vain OS X -käyttöjärjestelmällä. Näin ollen sovelluskehitystä varten tarvitaan Applen tietokone. Tätä ei taas ole mahdollista ajaa virtuaalikoneena Windows- tai Ubuntu-ympäristöissä.

Xcodesta löytyy iOS SDK:n kanssa vastaavat virheenkorjaus- ja emulaattoritoiminnot, toisin kuin Visual Studiosta ja Eclipsestä. Xcodessa emulaattoria kutsutaan vain simulaattoriksi. XCode tarjoaa myös jatkuvasti näkyvillä CPU:n ja muistin käytön. Näin pystytään jatkuvasti tarkkailemaan sovelluksen resurssien käyttöä.

Xcodessa oletuksena versionhallintajärjestelmänä toimii GitHub. GitHubiin on saatavissa ilmainen rajoitettu käyttäjätunnus. Ilmaisella käyttäjätunnuksella ei saa yhtään yksityistä säilytyspaikkaa. Kaikki lähdekoodit tulevat näin ollen julkisiksi.

iOS-sovellusten elinkaari eroaa Windows Phonen ja Androidin elinkaarista (kuva 9). Sovelluksen käynnistyttyä se siirtyy pois päältä-tilasta toimettomaan-tilaan ja siitä aktiiviseen-tilaan. Aktiivisesta siirrytään aina toimettomaan-tilaan ja siitä joko tausta-ajoon tai sovelluksen keskeytykseen. Siirtyminen tausta-ajoon riippuu siitä, onko se ohjelmoitu käyttöön sovelluksessa. Keskeytyksestä siirrytään aina pois päältä tilaan. [23.]



Kuva 10. iOS-sovelluksen tilojen vaihtuminen. [92.]

iOS-sovelluksia voidaan testata omalla laitteella ja simulaattorilla. Jos sovellusta halutaan beta-testata vielä ennen julkaisua, tähän on kolme vaihtoehtoa. Kaksi vaihtoehtoa tapahtuu iTunes Connectin kautta. Sovellus ladataan Xcodella iTunes Connectiin, johon kirjautumalla voi kutsua testaajia testaamaan sovellusta. iTunes Connectin kautta testatessa kutsun voi tehdä joko 25 tutulle käyttäjälle tai 1000:lle satunnaisesti valitulle käyttäjälle. [26.]

Kolmas beta-testaus vaihtoehto vaatii testaajien laitteiden rekisteröinnin. Sovelluskehittäjä voi rekisteröidä 100 laitetta vuodessa. Sovelluksessa itsessään on myös sertifikaatit

rekisteröidyistä laitteista. Näin ollen uusien testaajien tullessa joukkoon pitää heidän laitteensa rekisteröidä erikseen, sertifioida sovellukseen ja päivittää testattavan sovelluksen uusi versio sisältäen uudet laitteet.

Sovelluksen julkaiseminen voidaan tehdä kolmella tapaa. Julkaisemalla se App Storeen, B2B-julkaisu tai ad hoc -jakelu. Ad hoc-jakelu tapahtuu 100:lle iOS-laitteelle sähköpostin kautta tai omalta serveriltä jaettuna. [1.]

B2B-julkaisu tehdään iTunes Connectin kautta. Tätä tehdessä pitää tunnistaa asiakkaat ja antaa julkaisupäivä. Asiakkailta vaaditaan massaostos-tunnus Applelle. Kun tunnus on tehty ja tunnistettu, näkyy sovellus heillä Volume Purchase Program -kaupassa, josta he valitsevat ostomäärän. [32.]

Tunnetuin tapa jakaa sovelluksia on App Storen kautta. Ilmaisten sovellusten jakaminen ei maksa sovelluskehittäjälle mitään, eikä näin ollen kuluja synny.

4.2 Hybridi

Hybridisovellus tarkoittaa sovellusta, jonka käyttöliittymä toteutetaan pääasiassa HTML5:llä. Itse sovelluksen runko taas on aina natiivi. Nykyään myös toiminnallisuudet voidaan toteuttaa pitkälle HTML5:llä. Hybridisovellusten toteuttaminen eroaa hieman alustojen osalta. Hybridisovellukset vaativat aina vähintään samat sovelluskehitysvälineet kuin natiivisovelluskehitys. Tämä johtuu sovellusten natiivirungoista.

Mikäli samaa hybridisovellusta ollaan toteuttamassa useammalle alustalle, on suositeltavaa käyttää valmista sovelluskehystä. Tällöin samoja lähdekoodeja voidaan toistaa eri alustoille toteutusta tehtäessä enemmän kuin ilman sovelluskehyksen käyttöä.

Vaikka kaikilla kolmella alustalla voi toteuttaa Cordova-sovelluskehyksellä hybridisovelluksia (entinen PhoneGap), eroja tulee natiivikoodeihin. Cordova tarjoaa API:n jolloin samoilla JavaScript-komennoilla saadaan aikaan toimintoja, jotka muuten eroavaisivat alustakohtaisesti. Näin ollen sama koodi voidaan uudelleen käyttää toisessa eri alustan hybridisovelluksessa, mikäli molemmissa on Cordova käytössä. [15.]

Hybridisovellusten etu pelkkään natiiviin on mahdollisuus luoda samalta näyttävä sovellus kaikilla alustoilla. Samalta näyttävä sovellus HTML5:llä toteutettuna on raskaampi pyörittää kuin natiivisti toteutettu. Tämä johtuu siitä, että selain vaatii omat resurssit ja sen selainmoottori tekee työn. Natiivina työn tekee laitteisto. Hybridisovelluksissa on mahdollisuus kuitenkin käyttää sekaisin natiivi- ja HTML5-koodia, joten edut saadaan irti molemmista sovelluskehitystavoista.

4.3 HTML5

HTML on merkintäkieli, jonka selaimen selainmoottori tulkitsee näkyväksi verkkosivuksi. HTML-standardia ylläpitää W3C. HTML5:stä puhuessa voidaan tarkoittaa eri asioita. Se voi tarkoittaa HTML-versiota 5.0. Yleisemmin sitä käytetään kuitenkin yleisnimityksenä HTML:n, JavaScriptin ja CSS:n yhdistelmästä.

Vaikka HTML5 onkin suhteellisen uusi (2014 suositeltu W3C:n puolesta), on se kuitenkin lunastanut paikkansa ja kehitystä tapahtuu työkalujen osalta jatkuvasti. Myös HTML5-sovelluskehyksiä löytyy useampia. Tämä kuitenkin hidastaa kehitystä verrattaessa yhden kehyksen toteutukseen. Suurimpia mobiilialan HTML5-sovelluskehyksiä ovat Cordova (aikaisemmin PhoneGap) ja JQuery Mobile. [93.]

HTML-standardin päivitys versiosta 4.01 versioon 5.0 kesti lähemmäs kymmenen vuotta. HTML5:een on tuotu paljon uusia ominaisuuksia, mitä aikaisemmasta versioista on puuttunut. Näitä ovat mm. web socketit, web storaget, canvasit ja useita erilaisia multimediaominaisuuksia. [71.]

Vaikka HTML-versiot ovat päivittyneet hitaasti, se ei tarkoita, että versiot toimisivat kaikilla selaimilla. HTML näytetään sitä tulkitsevan selainmoottorin esittämällä tavalla. Eri selaimissa voi olla käytössä eri selainmoottorit. Näin ollen sama HTML5-sovellus saattaa näyttää ja toimia eri tavalla esimerkiksi iPhonella ja Windows Phonella. Oletuksena iPhonessa on Safari- ja Windows Phonessa Internet Explorer-selain, ja näissä on eri selainmoottorit. Erilaisia mobiiliselaimia löytyy useita kymmeniä. [71.]

HTML5:n etu natiivi- ja hybiridsovelluskehitykseen nähden on alustariippumattomuus ja se, ettei sovellusta tarvitse erikseen ladata. Samaa sovellusta voi ajaa monilla eri alus-

toilla ensin optimoimalla se sopivaksi. HTML5:n oppimiskäyrä on myös natiivikieliin nähden paljon helpompi. Lähes poikkeuksetta natiivikieliä osaavat sovelluskehittäjät osaavat toteuttaa myös HTML5-sovelluksia.

HTML5-sovelluskehityksen suuri ongelma on tuen puute itse käyttöjärjestelmien valmistajilta. Useat HTML5-sovelluskehitystyökalut ovat avointa lähdekoodia. Näyttää siltä, että niiden hiominen huippuunsa ei ole isoille yrityksille kannattavaa, koska se ei tuo niille tuloja. Tässä tapauksessa se olisi myös työkalujen toteuttamista kilpailijoille. Tämän takia HTML5-sovelluskehitystyökalut ovatkin ontuvia verrattaessa natiivisovelluskehitystyökaluihin.

Sovellus voidaan toteuttaa pelkkää tekstieditoria käyttäen. Usein kuitenkin käytetään IDE:jä, jotka helpottavat työskentelyä ilmoittamalla esimerkiksi sulkematta jääneistä elementeistä.

HTML5-sovellusten päivitys tapahtuu huomaamattomasti loppukäyttäjälle. Sivu päivitetään ilman, että käyttäjä saa siitä mitään erityisiä ilmoituksia. HTML5-sovellusta ei voi kuitenkaan laittaa keskitettyihin kauppapaikkoihin myyntiin, joten mainostaminen niille pitää tehdä muuta kautta. Jos HTML5-sovellusta halutaan myydä, on siihen hyvä toteuttaa esimerkiksi maksulliset käyttäjätunnukset.

Virheiden etsiminen HTML5-sovelluksissa on usein haastavaa. Esimerkiksi Firefox-selaimen FireBug-lisäosalla se on mahdollista, mutta se ei ole läheskään verrattavissa natiivisovelluskehitysympäristön tarjoamiin virheenkorjausvälineisiin. IDE:issä on suuria eroja, ja niiden toiminnallisuudet vaihtelevat laajasti. Ongelmaksi uudelle kehittäjälle saattaa syntyä, se millä sovelluskehityksen tekee, tekstieditorilla vai jollain tietyllä IDE:llä.

Usein käyttäjillä HTML5-sovellusten kanssa ongelmaksi muodostuu se, etteivät ne toimi kuin natiivisovellukset. Vaikka sovellus saattaa näyttää päältä päin natiivisovellukselta, käyttöjärjestelmäkohtaiset ominaisuudet eivät välttämättä toimi. Esimerkkinä sovelluksessa navigointi ei toimikkaan samalla tavalla kuin natiivissa ja nipistystoiminto zoomaakin, vaikka käyttäjä voisi olettaa sen tekevän jotain muuta.

4.4 Yhtäläisyydet ja eroavaisuudet

Suuria eroja on siinä, miten natiivi- ja hybridisovellukset jaetaan kauppapaikkojen kautta verrattuna HTML5-sovelluksiin, jotka jaetaan muita reittejä. Tämä tarkoittaa, että sovelluksien päivitys tapahtuu myös kauppapaikkojen kautta natiivi- ja hybridisovelluksissa. HTML5-sovellukset voidaan päivittää koska vain ilman kauppapaikan tarkistuksia.

HTML5:n oppiminen on nopeampaa kuin natiivikielien. HTML5:n eduksi voidaan laskea myös se, että yhdellä kielellä voidaan toteuttaa kolmelle (ja useammallekin) alustalle sovelluksia, kun taas natiivisovelluksissa jokainen alusta vaatii oman kielensä.

Yksi HTML5-sovellus voi pyöriä usealla alustalla. Natiivisovellukset pitää tehdä aina alustakohtaisesti eikä lähdekoodeja pystytä kierrättämään. Hybridisovellus pystyy uusiokäyttämään suurimman osan lähdekoodeista, mikäli toteutus on tehty se mielessä pitäen. Jos taas ei, ei ole välttämättä kannattavaa toteuttaa hybridisovellusta ollenkaan. Tähän on kuitenkin poikkeus, mikäli sovelluksen pitää esimerkiksi yrityksen brändin takia näyttää samalta kaikilla alustoilla.

Natiivisovellukset ovat nopeampia kuin hybridi- ja HTML5-sovellukset. Jos sovellus vaatii paljon laiteresursseja, on syytä valita kehitysmenetelmäksi natiivi.

Jo pelkästään tarvittavat ominaisuudet voivat rajata HTML5:n pois, esimerkiksi jos sovelluksen pitää nähdä käyttäjän kalenteri. Tällöin vaihtoehdoiksi jää vain natiivi- ja hybridisovelluskehitys. Ominaisuuksia, joihin ei pystytä puhtaalla HTML5:llä pääsemään käsiksi, ovat muun muassa kalenteritiedot, kontaktit, kamera (toimii Androidilla ainoastaan), NFC, värinähälytys, erilaiset kosketusnäytön toiminnot kuten nipistys, taustalla pyörivät palvelut, OpenGL/Direct3D, laitteen tietovarastot, push-viestit, jne. Näiden ominaisuuksien puute johtuu siitä, että HTML5-sovellukset ajetaan selaimessa ja niin sanotussa hiekkalaatikossa tietoturvasyistä.

Luotettavan tietoturvan osalta niin natiivi- ja hybridi- kuin HTML5-sovellusten kyse on sovelluskehittäjästä. Kaikilla tavoilla on mahdollista luoda varmasti tietoturvallisia sovelluksia. Natiivi- ja hybridisovellusten etuna voidaan pitää kuitenkin kolmansien osapuolien vaikeutta päästä kiinni lähdekoodeihin ja sitä kautta löytämään aukkoja sovelluksesta.

Sovelluskehitysvälineet ovat selkeästi valittavissa natiivisovelluskehityksessä. Tämä on suuri etu, mikäli ollaan tutustumassa uuteen sovelluskehitysmenetelmään. Erilaisia HTML5-sovelluskehitystyökaluja löytyy paljon, ja sopivan löytäminen voi tuntua turhauttavalta. Hybridisovelluskehitystä voidaan tehdä niin natiivisovelluskehitystyökaluilla kuin kolmansien osapuolien IDE:illä.

Hybridisovelluskehityksen huonona puolena voidaan pitää riippuvuutta kolmansiin osapuoliin. Kun alustat päivittyvät, eivät vanhat hybridisovellukset välttämättä toimi uusilla alustoilla. Hybridisovelluksissa käytetyt rajapinnat saattavat vaatia päivitystä ennen kuin niitä voidaan taas käyttää uusilla alustoilla. Natiivi- ja HTML5-sovelluskehitys voidaan aloittaa heti uusien alustapäivitysten jälkeen.

5 Sovelluksien toteutus

Luomme kolme sovellusta käyttäen natiivi-, hybridi- ja HTML5-sovelluskehitysmenetelmiä. Toteutamme natiivi- ja hybridisovellukset vain yhdelle alustalle. Olemme valinneet alustaksi Androidin.

Toteutamme kolme samankaltaista sovellusta eri kehitysmenetelmillä. Tällöin pystymme vertailemaan, miten samojen toimintojen toteutus eroaa eri menetelmillä. Tällöin sovellus pitää määrittää niin, että se käyttää ominaisuuksia, jotka voidaan toteuttaa myös pelkällä HTML5:llä. Emme voi tällöin toteuttaa esimerkiksi mitään kontakteja tai erilaisia laitteistoa käyttäviä sovelluksia, koska HTML5 ei tue sitä.

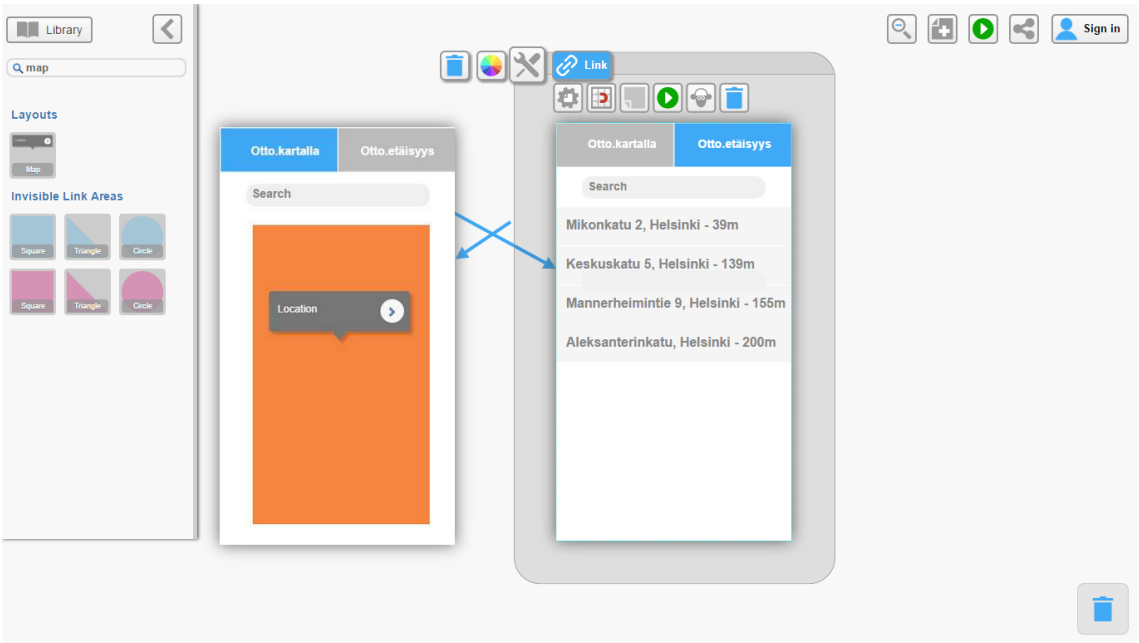
5.1 Sovelluksen määrittely

Toteutamme sovelluksen, joka näyttää lähimmät pankkiautomaatit kartalla ja tekstinä. Sovelluksessa oletuksena etsitään lähimmät automaatit käyttäjän senhetkisestä sijainnista. Automaattien sijainnit voidaan ladata CSV-tiedostona Otto-automaattien sivuilta.

Sovellus käyttää gps-, kartta- ja wlan-rajapintoja. Gps toimii oletuspaikantimena käyttäjän sijainnille. Mikäli gps ei ole käytössä, pyritään sijainti paikantamaan lähimmän tukiaseman perusteella.

Sovelluksesta toteutetaan ensin prototyyppi, jonka pohjalta sovelluksia aletaan toteuttamaan. Prototyypin ideana on saada selkeä kuva siitä, miltä valmis toteutus voisi näyttää. Tällä tavalla voi helposti vertailla, minkälainen navigaatio ja ulkoasu sovellukselle tehdään. Esimerkiksi välilehdet toimivat paremmin kuin sivun vaihto näytön sivusta vetämällä. Näin siksi, että käyttäjä voi vahingossa siirtää karttanäkymää sivuun, eikä pääse haluamalleen tekstisivulle.

Kuvassa 11 näkyy sovelluksen prototyyppi. Sovelluksessa tulee olemaan kaksi sivua välilehdillä eroteltuina. Ne ovat karttanäkymä- ja etäisyysnäkymäsivut. Klikkaamalla välilehtiä siirrytään aina klikatulle sivulle. Karttanäkymässä nähdään oma sijainti ja lähimmät pankkiautomaatit. Prototyyppi on tehty https://www.fluidui.com/editor/live/-sivuilla, eikä siinä ollut mahdollista käyttää karttaa pohjana, joten oranssi alue kuvaa karttaa. Tekstinäkymässä nähdään etäisyysjärjestyksessä lähimpien automaattien katuosoitteet.



Kuva 11. Sovelluksen prototyyppi, jonka pohjalta sovellusta lähdetään toteuttamaan.

5.2 Toteutus

5.2.1 Natiivi

Natiivikielellä toteutettu sovellus tehdään käyttäen Eclipseä sekä Eclipsen ADT-liitännäistä. Eclipsen ADT-liitännäinen sisältää Android-SDK managerin, jolla voidaan ladata

kaikki kehityksessä tarvittavat työkalut, kirjastot ja levykuvat Android-virtuaalikonetta varten. Tarvittaviin kirjastoihin luetaan AOSP-kirjastot sekä Googlen kirjastot, joilla tuetaan liitettävyyttä Googlen omiin palveluihin, ja tukikirjastot joilla voidaan taata sovellusten yhteensopivuus myös vanhemmissa Android-versioissa. Google Play Services on kirjastoista se, millä otetaan käyttöön Google-palveluita kuten Google Maps.

Google Play Services on nykyisin Android-laitteissa oleva hyvin keskeinen apukirjasto, joka toimii järjestelmätason prosessina, mutta voidaan kuitenkin päivittää Google Playn kautta. Google Play Services on Googlen oma tuote, joka julkaistiin vuonna 2012. Se on siitä lähtien kasvavassa määrin pyrkinyt lisäämään kirjastonsa kattavuutta kaikkiin omiin palveluihinsa. Se sisältää myös muita apukirjastoja. [7;63.]

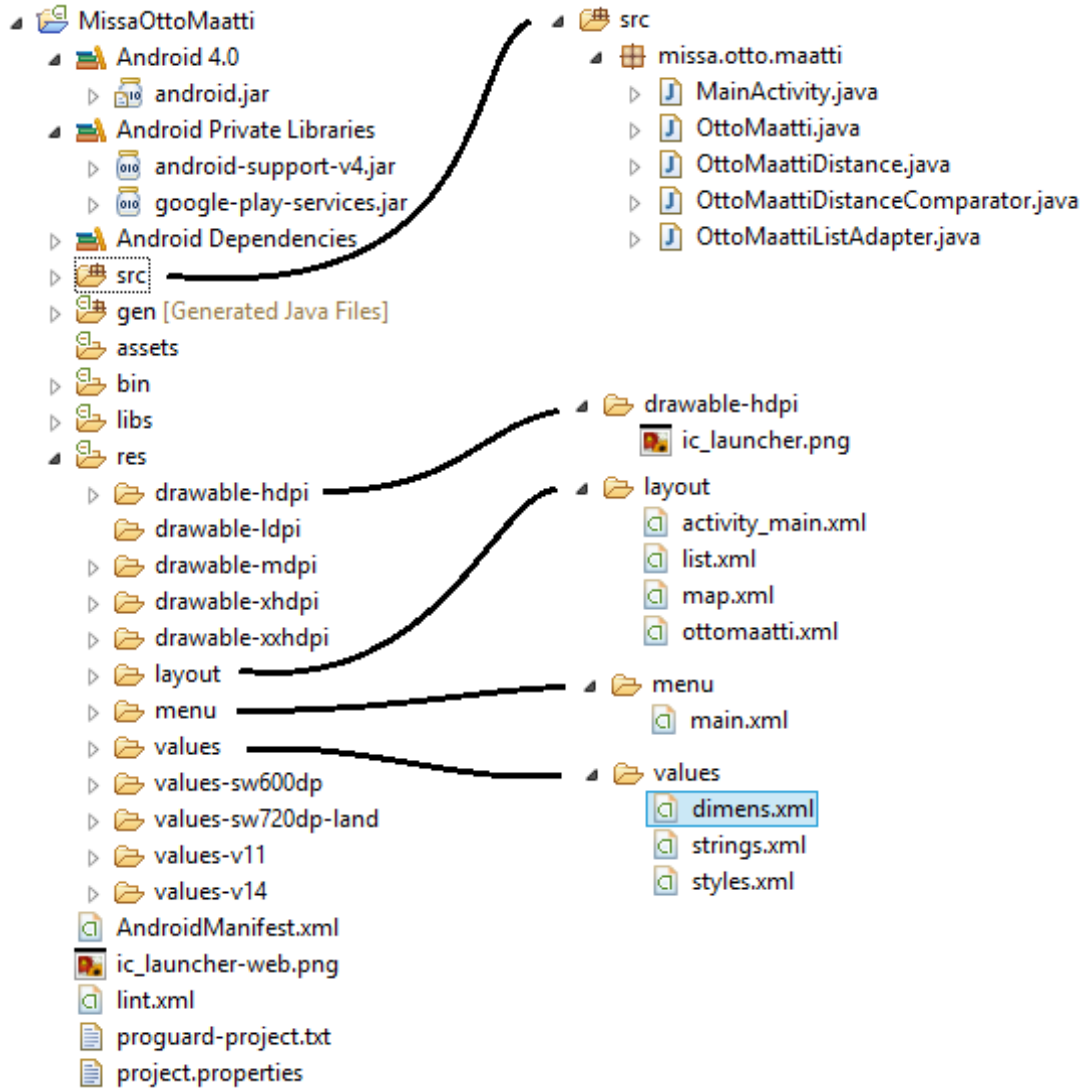
Taulukko 3. Google-palvelut, joihin Google Play Services tarjoaa rajapinnat. [64.]

Google Maps	Kartta ja sijaintipalvelut
Google Play In-App Billing	Sovellusten sisäiset ostokset
Google+	Sosiaalinen media
Google Wallet Instant Buy	Maksutapahtuminen käsittely palvelut
Google Cloud Platform	Sovelluksen integrointi pilveen
Google Analytics	Analytiikkapalvelut
Google Cloud Messaging	Kevyt viestittely ns. Push viestit
Google Mobile Ads	Mobiilimainostus
Google Cloud -tallennus	Käyttäjän datan tallennus pilveen

Tällaisen apukirjaston avulla Googlen on mahdollista päivittää Android-laitteita itsenäisesti laitevalmistajista riippumatta. Näin voi taistella Androidin pirstaloitumista vastaan. Asian kääntöpuolena on muuttuminen Google-tuotteiksi eli suljetuksi lähdekoodiksi. Julkisuudessa on keskusteltu juuri tästä ilmiöstä, että Google olisi ikään kuin tiukentamassan otetta Androidista, ja juuri siksi useita Googlen palveluita/sovelluksia on viime vuosina brändätty uusiksi. [65.]

Sovellusta luotaessa Androidille on lisättävä sovelluksen projektiin Google Play Services–kirjasto, mikäli jotain edellä mainituista rajapinnoista on käytössä. Mobiilisovelluksissa kirjastojen kuten paikannuspalveluiden/karttojen käyttö on hyvin yleistä. Se puolestaan aiheuttaa sovellusten yhteensopimattomuutta muiden AOSP:iin perustuvien käyttöjärjestelmien kuten Fire OS:n ja Nokia X Platformin kanssa. Yhteensopivuusongelmat johtuvat Google Play Services -kirjastosta, mikä on asennettuna vain Google Android-laitteissa. [13;83.]

Kun sovelluksen Android-projekti on luotu ja tarvittavat kirjastot projektiin on lisätty, näyttää projektin perusrakenne kuvan 12 mukaiselta. Kuvassa näkyvät myös jälkeenpäin lisätyt luokat ja näkymät.



Kuva 12. Android-sovelluksen projektin rakenne.

Sovelluksen ohjelmoitava logiikka luodaan src-kansioon. Android-sovellukset perustuvat aktiviteetteihin. Tässä sovelluksessa niitä on vain yksi kappale. Aktiviteetti on esitelty liitteessä 1. Aktiviteetin vastuu on olla vuorovaikutuksessa käyttäjän kanssa kuten näyttää ikkunat käyttäjälle. [5.]

Näytettävien ikkunoiden sivupohjat nappuloineen eli siis ikkunat ilman dynaamista sisältöä määritellään layouts-kansioon. Layouts-kansioon voidaan myös määritellä muitakin rautalankoja kuten listanäkymän yksittäinen rivi.

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity$ListSctionFragment" >
    <TextView
        android:id="@+id/info"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/infoTxt"
        android:paddingBottom="10dip" />
    <ListView
        android:id="@+id/nearByAtmList"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_below="@+id/info"
        android:fastScrollEnabled="true" />
</RelativeLayout>
```

Kuva 13. Sovelluksen listanäkymän sivupohjan määritelmä (list.xml).

Listanäkymän sivupohjassa on vain kaksi elementtiä, TextView sekä ListView. Näkymän TextView-elementtiin on sisällöksi määritelty infoTxt-niminen staattinen teksti, ListView-elementin sisältö puolestaan määritellään dynaamisesti. Listaan ei kuitenkaan viedä OttoMaatti-olioita (esitelty liitteessä 2) vaan OttoMaattiDistance-olioita (esitelty liitteessä 3), OttoMattiDistance-olio sisältää OttoMaatti-olion lisäksi myös etäisyyden sijaintiin sekä toteuttaa Comparable-rajapintaa, jotta etäisyysvertailu voidaan toteuttaa helposti. Varsinaista datan vertailua varten on luotu OttoMaattiDistanceComparator-olio (esitelty liitteessä 4). Datan lisäämiseksi ListView-elementtiin (Kuva 13) pitää käyttää ArrayAdapteria.

Androidissa on perusadaptereita, mutta niitä voi soveltaa lähinnä yhden rivin/merkkijonon asettamiseen, joten helpointa on luoda adapteri omaan tarpeeseen. Adapterin avulla voidaan listaan liitettävälle oliojoukon yksittäiselle oliolle määrittää esitystyyli. Adapterissa voidaan siis määrittää pohja ja miten data pohjassa esitetään. Adapteri on esitelty liitteessä 5. Adapterin pohja on esitelty liitteessä 6 kohdassa *ottomaatti.xml*.

Sovelluksen res-kansioon eli resurssit kansioon sisällytetään kaikki sovelluksen mediatiedostot sekä XML-määritelmät. Layouts-kansio sisältää sivupohjat, muut näkymiin tarvittavat pohjat ja menu-kansiovalikot. Values-kansioon voidaan puolestaan sisällyttää

sovelluksessa käytettäviä globaaleja arvoja. Globaaleja arvoja voivat olla teemoissa käytettävät värit tai staattisesti näytettävät tekstit. Staattisilla teksteillä on mahdollista toteuttaa sovellukselle helposti monikielisyyden tuki.

Jokaisella Android-sovelluksella on oltava manifest-tiedosto, joka manifesti syntyy oletuksena projektin juureen. Manifesti esittää kaiken oleellisen tiedon sovelluksesta kuten:

- Sovelluksen Java-paketin nimen. Nimi toimii sovelluksen yksilöivänä tunnusteena.
- Kuvailee sovelluksen komponentit eli aktiviteetit, palvelut, massatiedotukset (broadcast receivers) ja (content providers), joista koko sovellus koostuu. Nimeää luokat, joita kukin komponentti toteuttaa ja julkaisee niiden kyvyt. Nämä ilmoitukset kertovat Android-järjestelmälle, mitä komponentit ovat ja millä ehdoilla ne avataan.
- Määrittelee, mitkä prosessit ajavat sovelluksen komponentteja.
- Ilmoittaa, mitä oikeuksia sovellus tarvitsee päästäkseen API:n suojattuihin osiin ja ollakseen vuorovaikutuksessa muihin sovelluksiin.
- Ilmoittaa, mitä oikeuksia muut sovellukset tarvitsevat ollakseen sovelluksen komponenttien kassa vuorovaikutuksessa.
- Ilmoittaa minimi API-tason, mitä sovellus tarvitsee.
- Listaa kirjastot, mitä vasten sovellus on linkitettävä.

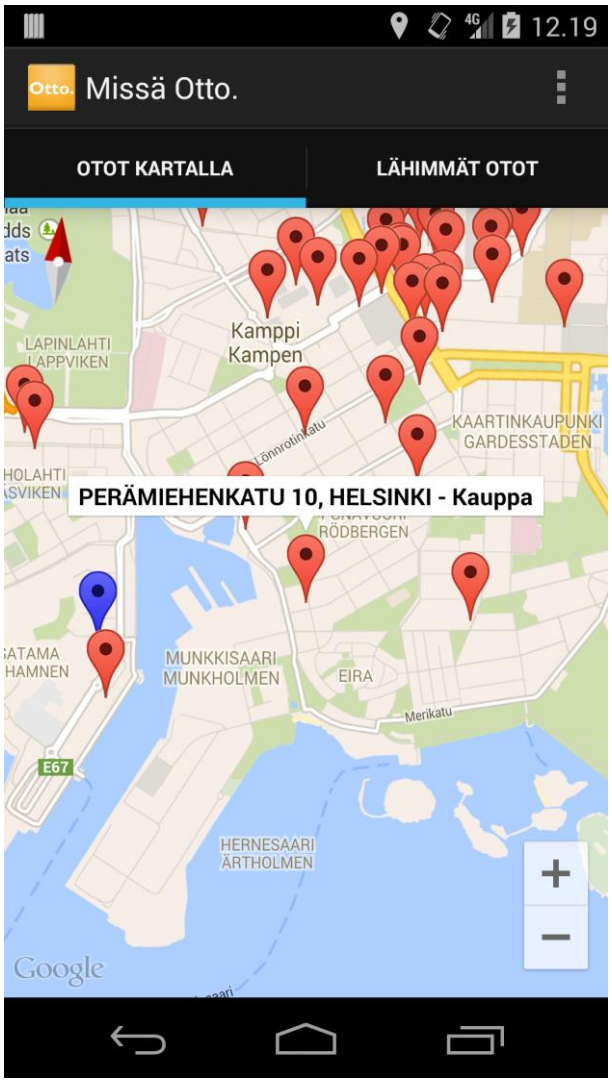
[19.]

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="missa.otto.maatti"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk android:minSdkVersion="14"
              android:targetSdkVersion="21" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="missa.otto.maatti.permission.MAPS_RECEIVE"/>
    <uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
    <permission android:name="your_package_name.permission.MAPS_RECEIVE"
                android:protectionLevel="signature"/>
    <uses-feature android:glEsVersion="0x00020000" android:required="true"/>
    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme">
        <activity
            android:name="missa.otto.maatti.MainActivity"
            android:label="@string/app_name"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <meta-data android:name="com.google.android.gms.version"
                   android:value="@integer/google_play_services_version" />
        <meta-data android:name="com.google.android.maps.v2.API_KEY"
                   android:value="                                          " />
    </application>
</manifest>
```

Kuva 14. Natiivisovelluksen, manifest-tiedosto

Natiivisovelluksen manifest-tiedosto (kuvassa 14) vaatii huomattavasti paljon enemmän oikeuksia suojattuihin API:n kuin vertailtava hybridtoteutus Microsoft Bing Maps-kartoilla. Sovellus vaatii useiden paikantamiseen liittyvien oikeuksien lisäksi myös oikeudet OpenGL-kirjastojen käyttöön sekä kirjoitusoikeudet tallennustilaan. Kartta API:iin autentikointi tapahtuu kummassakin vertailtavassa sovelluksessa hyvin samantapaisesti. Bing Mapsin ja Google Mapsin käyttö vaatii kehittäjältä rekisteröitymisen kartta API:n käyttäjäksi ja tämän jälkeen on mahdollista anoa autentikoitumisavain (API key) kartta-API:iin. Natiivisovelluksen karttakirjastoja käytettäessä API:n autentikoitumisavain sijoitetaan manifestitiedostoon.

Kuvassa 15 näkyy Androidille toteutetun natiivisovelluksen karttanäkymä. Kartalla sininen piste kuvaa omaa sijaintia ja punaiset pankkiautomaatteja. Klikkaamalla pisteitä tulee lisätietona osoite ja paikka esimerkiksi kauppa, satama, laivaterminaali tai kauppakeskus.



Kuva 15. Androidille toteutetun natiivisovelluksen karttanäkymä.

Kuvassa 16 näkyy Androidille tehdyn sovelluksen tekstinäkymä. Natiivitoteutuksessa listaan voidaan pistää kaikki pankkiautomaatit, silti sovellus toimii sulavasti. Klikkaamalla pankkiautomaattia siirrytään karttanäkymässä kyseisen automaatin kohdalle.

Kuva 16. Androidille natiivisti toteutetun sovelluksen tekstinäkymä.

5.2.2 Hybridi

Hybridisovelluksen toteutamme käyttäen Cordovaa. Se tarjoaa laajan JavaScript API:n sovellusten toteutukseen. Windows-ympäristössä Cordova vaatii Node.js ja Git-clientin asennuksen.

Node.js:stä ja Git-clientistä löytyy molemmista itseasentuvat mediat. Näiden asentamisen jälkeen voidaan asentaa Cordova seuraavasti. Komentorivillä annetaan seuraava komento: *npm install –g cordova*.

Tällöin Node.js hakee paketit Git-clienttiä käyttäen ja asentaa Cordovaan vaadittavat paketit. Parametri –g on tärkeä, koska se tekee Cordova-asennuksesta globaalin. Tällöin sitä voi käyttää muualtakin kuin Node.js-asennuskansiosta. Tämän jälkeen voidaan luoda Cordova-sovellus. Komentorivillä mennään kansioon, missä säilytetään lähdekoodeja ja luodaan projekti komennolla: *cordova create lopputyo timo.rolle.ottopisteet Ottopisteet*.

Komento luo uuden kansion nimeltä ”lopputyo”-kansioon, jossa säilytetään lähdekoodeja. Projektin nimeksi tulee Ottopisteet. Luonnin jälkeen siirrytään komentorivillä uuteen kansioon ja ajetaan komento: *cordova platform add android*.

Tämä määrittelee, mille alustoille sovellusta ollaan toteuttamassa. Komennossa voidaan vaihtaa android- ios-, amazon-fireos-, blackberry10- tai wp8-sanoiksi, riippuen mille alustalle toteutusta ollaan tekemässä. Komento voidaan ajaa myös useammalle kuin yhdelle alustalle toteutusta varten.

Kun alusta on valittu, voidaan projekti avata Eclipsessä. Tämä tapahtuu sen tuonnilla (Import). Projekti on heti käännettävissä ja ajettavissa emulaattorissa. Käytämme työskennellessä Genymotion -emulaattoria. Tämä on kevyempi ja monipuolisempi, kuin Android-SDK:n mukana tullut emulaattori. Näin ollen sovelluskehitys onnistuu hieman tehottomammallakin kannettavalla tietokoneella. Genymotion tarjoaa useita eri laitepohjia emulaattoreiksi ja ne näkyvät Eclipsessä vastaavasti kuin fyysiset laitteet eli ei emulaattoreina.

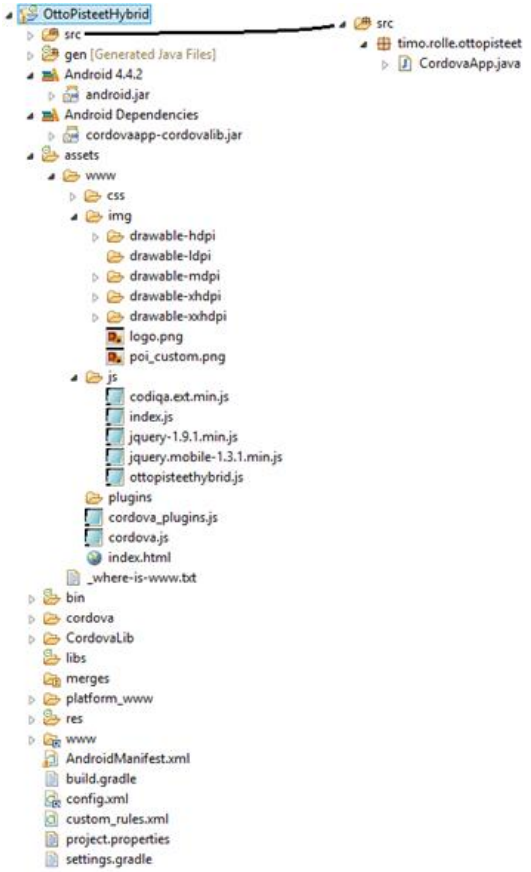
Sovelluksen pohja on toteutettu käyttäen Codiqan selainpohjaista sovelluskehitysympäristöä. Sitä käyttäen voidaan toteuttaa sovelluksen runko hyvin yksinkertaisesti. Sivutus ja sovellukseen tarvittavat komponentit voidaan raahata suoraan paikoilleen. Runko on ladattu sivulta ja kopioitu jo luotuun Cordova-projektiin.

Toteutus käyttää Bing Maps Ajax Control v7.0:aa. Tämä on verkkosovelluksia varten toteutettu versio Bing Mapsista. Sitä voidaan käyttää useilla eri alustoilla hybridisovelluksissa. Sen käyttö vaatii sovellukseen tunnuksen, jolla palveluun voidaan tehdä kutsuja.

Sovelluskohtaista tunnistautumista ei kuitenkaan tarvita, kuten Google Mapsia käyttäessä natiivisovelluksissa. Käyttämämme karttaversio on ilmaisversio, joten sitä ei voi laittaa kauppapaikkaan myyntiin. Mikäli kartoista ottaisi maksullisen version, se toimisi vastaavasti ja sitä voitaisiin myydä.

Sovelluksessa ainoat muutokset natiiviosuuteen pitää tehdä AndroidManifest.xml-tiedostoon. Tiedostossa määritellään sovellukselle lupa käyttää verkkoyhteyttä, paikannusta ja lukea sekä kirjoittaa tiedostojärjestelmään. Tämän lisäksi tiedostoon on määritelty, ettei sitä voi käyttää horisontaalisesti. Tällä määrityksellä säästytään ylimääräisten tyylien tekemiseltä. Sovellus toimii oletuksena myös horisontaalisesti, mutta karttanäkymä jää ikävän pieneksi. Cordova-projektissa on valmiiksi luotu natiiviosuus HTML5-sivujen näyttämiseen, joten siihen ei täydy tehdä muutoksia.

Kuvassa 17 näkyy hybridisovelluksen rakenne. Omat lähdekoodit ja data ovat assets/www-kansiossa ja sen alakansioissa. Cordova-projektissa tulee mukana cordova.js ja cordova_plugins.js javascript-tiedostot. Cordova.js sisältää mm. funktioita, joilla voidaan tarkistaa, mikä laite ja käyttöjärjestelmä ovat käytössä. Cordova_plugins.js sisältää määritykset lisäosille, joita ovat ”File” ja ”File-transfer”. Niitä käytetään csv-tiedoston lataamiseen. Codiqa.ext.min.js, jquery-1.9.1.min.js ja jquery.mobile-1.3.1.min.js-tiedostot tulivat Codiqa:lla tuodusta rungosta. Niiden avulla saadaan muun muassa välilehdet toimimaan. Omat lähdekoodit ja sovelluksen pääasiallinen logiikka ovat ottopisteethybrid.js- ja index.html-tiedostoissa.



Kuva 17. Cordova-projektin rakenne avattuna.

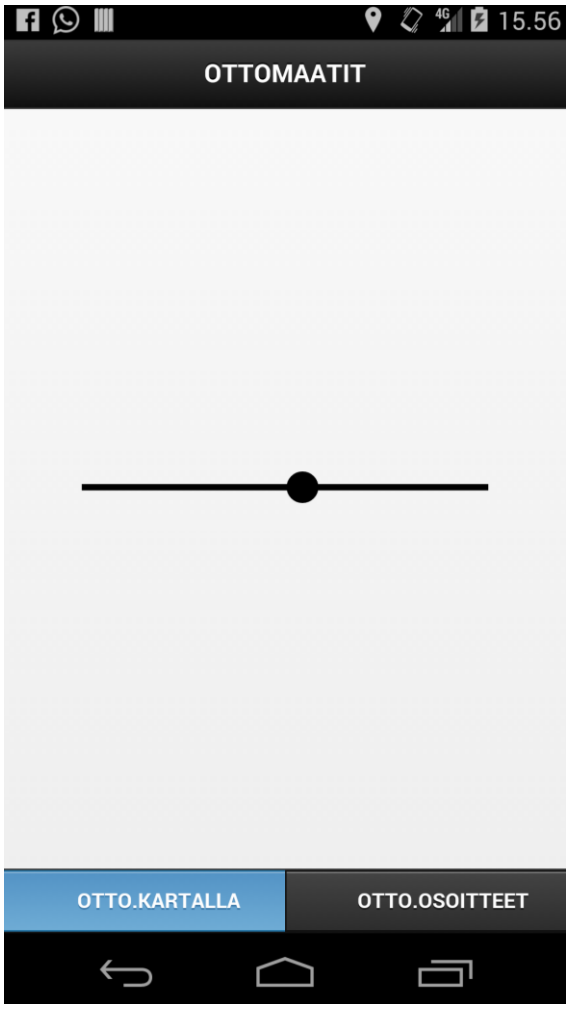
JQuery on hyvin suosittu JavaScript-kirjasto, jota käyttämällä voidaan toteuttaa helposti erilaisia välilehti-, lista-, pudotusvalikko- ja muita näkymiä toiminnallisuuksineen. Tästä on tehty myös erityisesti mobiilisovellusten toteutukseen suunnattu lisäkirjasto jQuery.mobile. [76.]

Sovelluksen latauduttua kutsutaan init-funktiota. Ensimmäiseksi tarkistetaan, löytyykö csv-tiedosto puhelimesta. Jos sitä ei löydy, se ladataan www.otto.fi-sivustolta ja otetaan käyttöön. Tämän jälkeen kutsutaan nykyistä sijaintia. Kun nämä lataa, on näkyvissä vielä latausnäkymä. Sijaintitiedon latauduttua piilotetaan latausruutu ja piirretään kartta laitteen näytön koon mukaan. Tämän jälkeen merkataan oma sijainti. Vasta sitten luetaan csv-tiedostosta automaattien sijainnit. Tämä on jaettu kahteen funktioon. Ensimmäisessä merkataan karttaan automaattien sijainnit. Toisessa funktiossa automaatit järjestetään etäisyyden mukaan järjestykseen. Lähimmät 30 automaattia lisätään jQueryn append-toiminnolla HTML-listaan. Liitteessä 7 näkyvät itse tehdyt JavaScript-funktiot.

JavaScriptillä ei voi tehdä niin sanottuja cross-domain kyselyjä, ellei niitä ole erikseen sallittu sivulla, johon kyselyt tehdään. Cordovan lisäosilla ”File” ja ”File-transfer” pystymme kuitenkin lataamaan tiedoston cross-domainista, koska se käyttää puhelimen natiiviominaisuuksia. Näitä käyttämällä voimme JavaSriptillä tarkistaa, löytyykö automaattien tiedot sisältävä csv-tiedosto puhelimesta. Mikäli tiedostoa ei löydy, se ladataan www.otto.fi-sivustolta ja otetaan käyttöön. Jos tiedosto löytyy, voidaan se heti ottaa käyttöön. [29.]

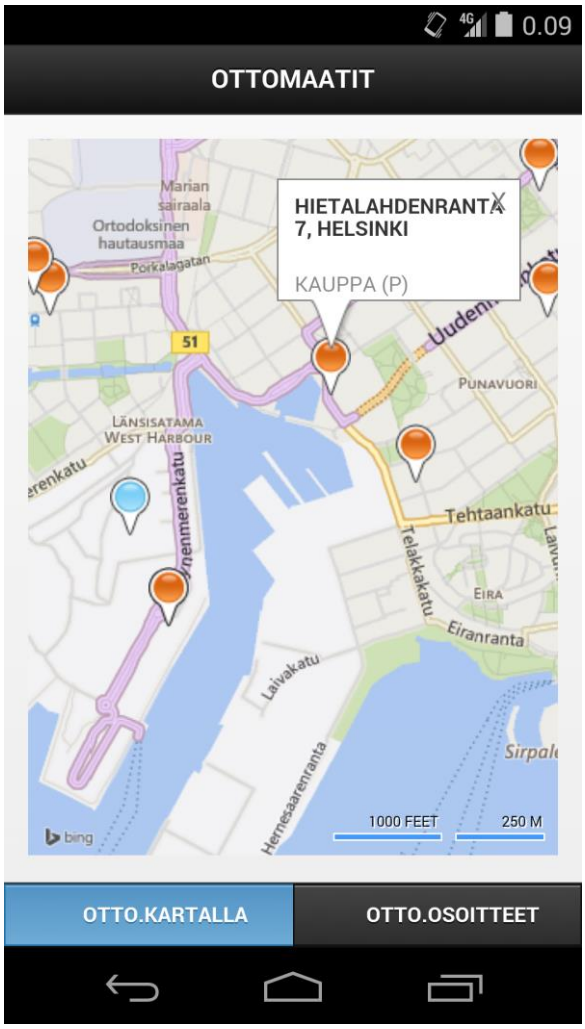
Projektin HTML-osuus on hyvin yksinkertainen. Se muodostuu index.html:stä, joka sisältää kommentteineen vain 110 riviä. Sovelluksessa käytettävät molemmat sivut on määritelty samaan tiedostoon div-elementteinä sisältäen data-role=”page” -tarkennuksen. Tämän lisäksi molemmille sivuille on lisätty identtiset välilehti-elementit. Liittessä 8:n näkyy index.html, tiedoston rakenne.

Kuvassa 18 näkyy Androidille toteutetun hybridisovelluksen latausruutu. Sovelluksen latautuminen on kiinni puhelimen tehoista ja verkkoyhteyden nopeudesta, se vaihtelee paljonkin laitekohtaisesti. Tämän takia sovellukseen on toteutettu latausruutu. Se on toteutettu HTML:ää ja CSS:ää käyttäen. Kun kartta on ladattu, piilotetaan kyseinen div-elementti, joka sisältää latausnäkymän, jQueryllä.



Kuva 18. Cordovalla toteutetun hybridisovelluksen latausruutu.

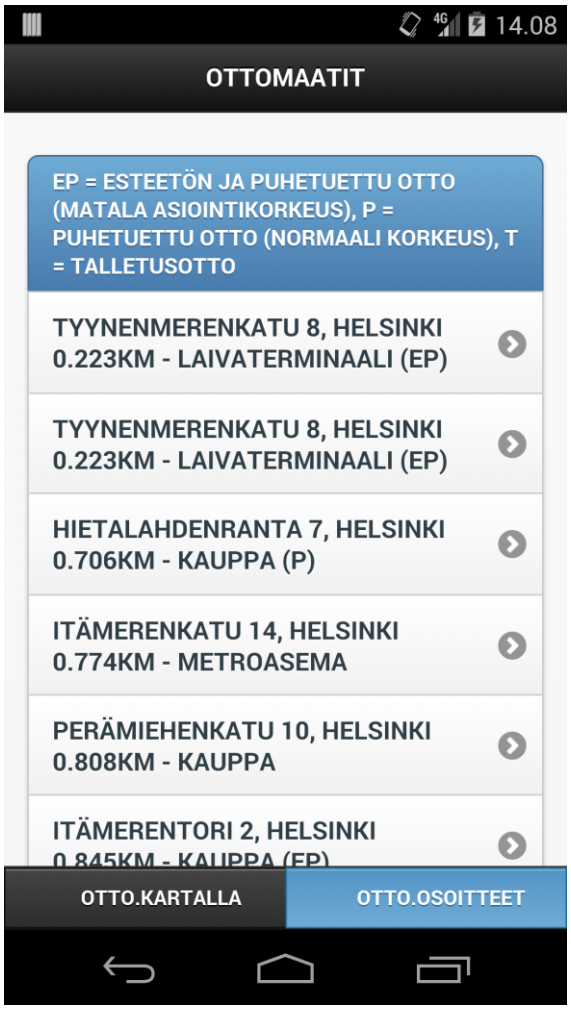
Kuvassa 19 näkyy hybridisovelluksen karttanäkymä. Näkymässä sinisellä näkee oman sijainnin, oranssit pisteet ovat pankkiautomaatteja. Klikkaamalla oransseja pisteitä tulee näkyviin lisätieto, jossa näkee osoitteen sekä minkälaisessa paikassa automaatti sijaitsee. Mikäli lisätiedot eivät mahdu näkymään, siirretään kartan keskipistettä sen verran, että koko lisätieto saadaan mahtumaan.



Kuva 19. Cordovalla toteutetun hybridisovelluksen karttanäkymä.

Kartta, pisteet kartalla ja lisätiedot on toteutettu täysin JavaScriptillä. Ainoastaan kartan div-elementti, johon kartta piirretään, on toteutettu valmiiksi HTML:llä. Oranssit pisteet ovat karttapalvelun oletuskuvalla, ne tulevat suoraan Bing Maps-palvelun API:sta. Sininen piste on erikseen sovellusta varten tehty kuva, joka on mukana projektin rakenteessa.

Kuvassa 20 näkyy hybridisovelluksen tekstinäkymä. Toisin kuin natiivisovelluksessa, pitää lähimpien pankkiautomaattien listaa rajata reilusti. Käyttöliittymä toimii todella hitaasti, mikäli kaikki automaatit listattaisiin. Pelkästään 30:nen automaatin näkymälläkin huomaa selvää hidastumista käyttöliittymässä. Klikkaamalla listasta automaattia siirtyy näkymä takaisin kartalle, kartta keskitetään klikattuun automaattiin.



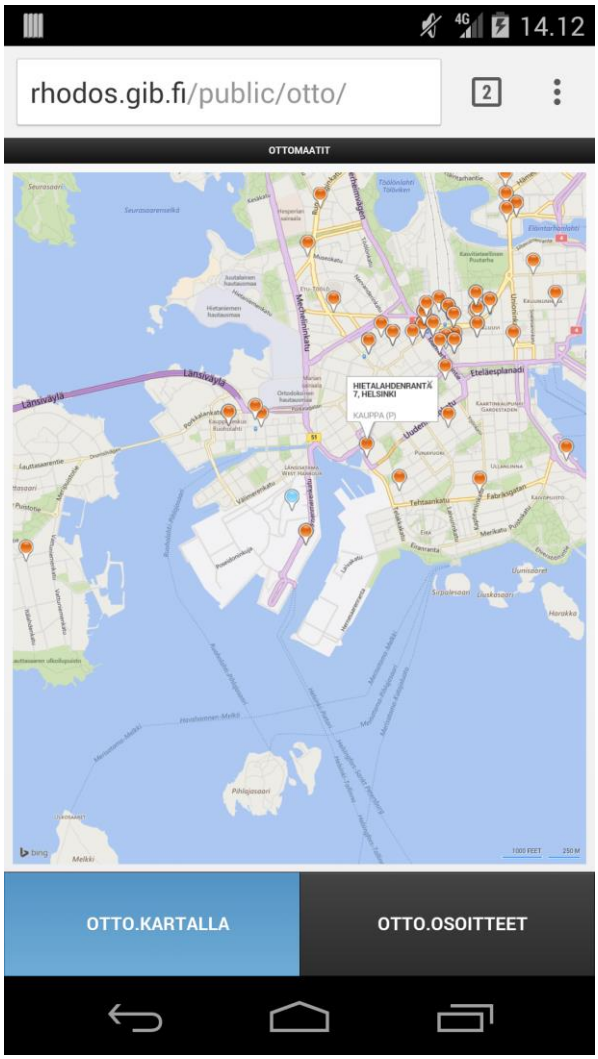
Kuva 20. Cordovalla toteutetun hybridisovelluksen tekstinäkymä.

Etäisyyksien laskemiseksi on tehty JavaScript-funktio, joka laskee kahden pisteen välisen etäisyyden ja palauttaa tämän kilometreissä. Pisteinä ovat aina oma sijainti ja automaatti. Tämän tiedon perusteella lista voidaan erillisellä funktiolla järjestää näyttämään lähimmät automaatit ensimmäisenä. Lista kirjoitetaan JavaScriptillä dynaamisesti HTML-muotoon.

5.2.3 HTML5

HTML5-toteutus on tehty kopioimalla hybridisovelluksen assets/www-kansion sisältö alakansioineen ja sisältöineen. Se on ladattu pavelimelle sellaisenaan. Ainoat muutokset, joita vaaditaan sovelluksen toimimiseksi, ovat Cordova-skriptien poistaminen ja csv-tiedoston lataaminen. Sovellus toimii sellaisenaan, mutta vaatii optimointeja muun muassa käyttöliittymään.

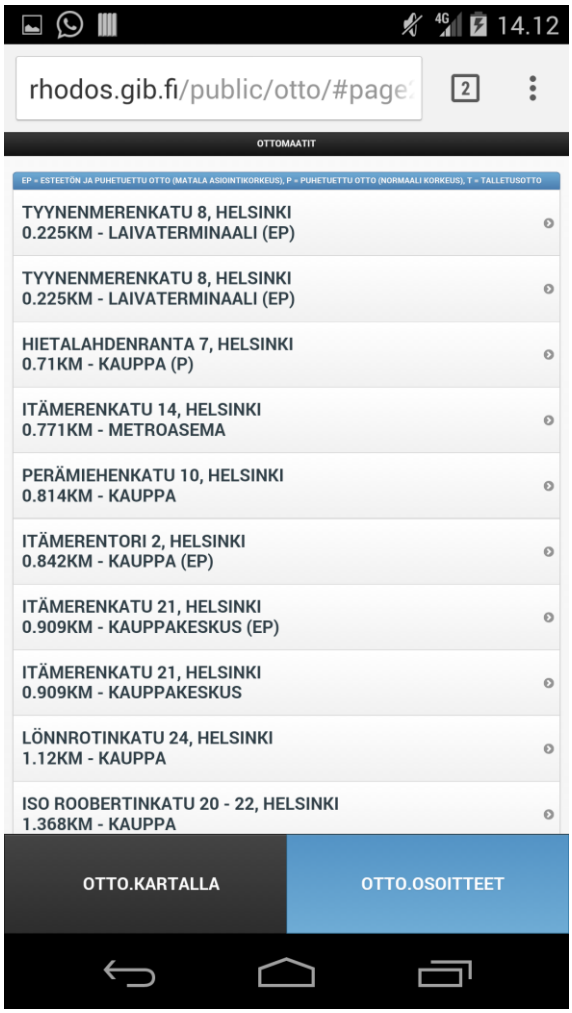
Kuten kuvasta 21 nähdään, pisteet kartalla näkyvät hyvin pieninä. Myös välilehdet olivat pienentyneet liian pieniksi käytettäväksi. Näitä joutui korjaamaan CSS-tiedostoja muuttamalla. Muutokset joudutaan tekemään nappien ja tekstin kokoon.



Kuva 21. HTML5-sovelluksen karttanäkymä.

Sovellus kysyy käyttäjältä lupaa käyttää puhelimen sijaintia. HTML5-sovellukseen ei voi lisätä natiivi- ja hybridisovelluksissa olevaa AndroidManifest.xml-tiedostoa ja siihen määritettyjä oikeuksia päästä käsiksi sijaintiin. Tämän takia oikeudet pitää kysyä joka kerta uudestaan. Laitteesta ja selaimesta riippuen lupa voidaan tallentaa sivulle kuitenkin pysyväksi.

Kuvassa 22 listan kohteet näkyivät hyvin pienellä ja niitäkin joutui muuttamaan CSS-tiedostoissa isommiksi. Toiminnallisuudet vastaavat kuitenkin hybridisovellusta, lukuun ottamatta csv-tiedoston lataamista.



Kuva 22. HTML5-sovelluksen tekstinäkymä.

HTML5-sovelluksen tiedostot ovat palvelimella, jossa olisi mahdollista käyttää myös PHP:tä. PHP mahdollistaisi CSV-tiedostosta uuden version lataamisen palvelinpäässä aina sivulle tultaessa. Tällöin tiedot olisivat automaattisesti ajan tasalla jokaisella kerralla, kun sivu ladataan. Tämä voitaisiin toteuttaa myös muilla palvelinpäässä pyörivillä kielillä.

Verkkosovellusten yksi iso ongelma on selainmoottorien eri tapa näyttää tyylitiedostoja. Internet Explorer-selain WP8:lla eroaa suuresti Androidin Chromesta ja iOS:n Safarista. IE tulee usein jäljessä tyylitiedostojen kanssa, eikä näytä dataa halutulla tavalla.

6 Johtopäätökset

Natiivisovelluskehityksen huonona puolena pidetään sen toimivuutta vain valitulla alustalla. Tässäkin on kuitenkin etunsa. Sovelluksen testaaminen ja optimointi yhdelle alustalle on helpompaa, kuin usealle alustalle. Natiivisovelluksen voi joutua myös optimoimaan useille erikokoisille näytöille. Tällöin kaikkea näytön tilaa ei välttämättä voida hyödyntää tehokkaasti. WP:lle ja iPhonelle kummallekin sovelluskehittäjän tarvitsee optimoida sovellus vain neljälle eri näytölle eli tietyn resoluution ja pikselitiheyden omaavalle näytölle. Androidin osalta eri variaatioita näyttöjen resoluution ja pikselitiheyden osalta taas voi olla satoja. [79;86.]

HTML5-sovellusta tehtäessä testasimme sitä iOS-, Android- ja WP8-laitteilla. Usein joku ominaisuus ei toiminut yhdellä laitteella. Kun ongelma saatiin korjattua, toinen ominaisuus toisella laitteella hajosi. Vastaavat tilanteet voivat johtaa siihen, että tehdään alustakohtaisia lähdekoodeja esimerkiksi if-lauseilla. Tällöin kuitenkin osa niin sanotusta alustariippumattoman koodin hyödystä häviää. Lähdekoodin optimointi taas kaikilla alustoilla toimivaksi voi viedä reilusti pidempään kuin usean esimerkiksi natiivisovelluksen tekeminen.

Hybridisovelluksissa on sama ongelma kuin HTML5:llä. Mikäli täysin samaa lähdekoodia halutaan käyttää, voi sen optimointi viedä tuskallisen paljon aikaa. Etuna ovat kuitenkin aina alustakohtaiset sovellukset, joten lähdekoodin ei tarvitse olla täysin sama. Jokaisella alustalla on oma sovellus, joten JavaScriptejä, jotka eivät ole esimerkiksi Cordovan tukemia, voidaan tehdä alustakohtaisesti.

Hybridisovelluksien kanssa ollaan riippuvaisia kolmannesta osapuolesta. Aina uusien käyttöjärjestelmäversioiden tullessa markkinoille joudutaan odottamaan, kunnes kolmas osapuoli saa API:nsa toimimaan uusien toimintojen ja muutosten kanssa. Tämä aiheuttaa viivästymistä sovellusten päivittämisessä. Hybridisovellusalustat joudutaan myös päivittämään usein, sillä ne sisältävät API:n kaikille kolmelle (ja useammalle) alustalle. Versioita siis tulee useammin ulos kuin tulisi vain yhdelle alustalle.

Sovelluksia tehdessä päädyimme kuitenkin suosimaan HTML5-pohjaisia menetelmiä paremmiksi vaihtoehdoiksi, siitäkin huolimatta, että sovelluksen optimointi eri selainmoottorille on pahimmissa tapauksissa verrattavissa toisen sovelluksen tekoon. Sovelluskehitysmenetelmää valitessa tulisi päähuomio kiinnittää sovelluksen ominaisuuksiin.

Mikäli sovelluksen ominaisuudet vain on toteutettavissa HTML5:llä, tulisi sovelluksen toteutusta harkita sillä. Etuna HTML5-sovelluksille on todella laaja sovelluskehitysverkosto.

Taulukko 4. Vertailukaavio kehityksessä ilmi tulleille erovaisuuksille.

Näkökohta	Natiivi	Hybridi (Cordova)	HTML5-sovellus
Tarvittavat taidot kehitykseen	Haastavaa	Helppoa	Helppoa
Sovelluksen jakaminen sovelluskaupan kautta	Kyllä	Kyllä	Ei
Sovelluksen päivitys	Hidas	Hidas	Nopea
Sovelluksen oikeudet mobiililaitteen rajapintoihin	Laajat	Laajat	Rajoitetut
Sovelluksen käyttö verkottomassa tilassa	Kyllä	Kyllä	Kyllä*
Käyttökokemus	Sulava	Rajoittunut	Rajoittunut

Taulukkoon 4 olemme keränneet työn sovelluksia toteutettaessa ilmi tulleita eroavaisuuksia. Taulukossa käsitellään niin sovelluskehitykseen kuin itse sovelluksen ominaisuuksiin tulevia eroavaisuuksia.

Tarvittavien kehitystaitojen osalta natiivisovellusten tekeminen on haastavampaa, kuin hybridi- ja HTML5-sovellusten. Natiivikielet eroavat toisistaan ja niiden oppiminen on vaikeampaa omien kokemuksiemme perusteella. HTML5:ssä on selkeästi rakenne, ominaisuudet ja tyylit jaoteltu omiin kieliinsä. Rakenne muodostuu HTML:stä, ominaisuudet JavaScriptistä ja tyylit on määritelty CSS-tiedostoissa.

Sovellusten jakaminen ja myyminen tapahtuu natiivi- ja hybridisovellusten osalta kauppapaikkojen kautta. Kauppapaikoissa sovelluksen myynti on helppoa ja ne ovat suoraan käyttäjien löydettävissä. Sovelluksia päivittäessä sovelluksen koko paketti joudutaan päivittämään ja sille tehdään aina tarkistus kauppapaikkakohtaisesti. Tämä vie oman aikansa. HTML5-sovellukset toimivat verkossa ja käyttäjät eivät löydä niitä helposti ilman erillistä mainostusta tai hakukonetta. Niiden päivitys on kuitenkin nopeampaa ja tapahtuu suoraan sovellustiedostoja muokkaamalla.

Kaikki sovellusominaisuudet ovat natiivisti käytettävissä. Hybridinä toteutettuna voidaan käyttää niin natiivi- kuin HTML5-kieliä. Näillä päästään natiivia vastaavasti kiinni kaikkiin

ominaisuuksiin, mutta HTML5:n ja natiivikielen yhdistäminen voi muodostua hyvinkin hankalaksi. Tätä varten voidaan käyttää valmiita sovelluskehyksiä kuten Cordovaa. HTML5:llä ominaisuudet ovat rajoittuneet. Sillä ei päästä kiinni puhelimen laitteistoon kuin osittain. Esimerkiksi kameran käyttö HTML5:llä ei toimi kuin uusimmissa Android-laitteissa.

Sovelluksia voidaan käyttää verkottomassa tilassa kaikilla kehitystavoilla. HTML5:llä verkotonta tilaa varten joudutaan kuitenkin tekemään erilliset määritykset, jolloin sovellustiedostot tallentuvat selaimen välimuistiin. Natiivi- ja hybridisovellukset toimivat oletuksena verkottomasti.

Käyttökokemuksen osalta natiivisovellukset toimivat parhaiten. Niillä tehdyt käyttöliittymät toimivat sulavasti ja puhelinten erilaiset pyyhkäisy- ja muut ominaisuudet toimivat oletuksena. HTML5:llä toteutettu käyttöliittymä ei usein vastaa puhelimen käyttöjärjestelmän yleistä käyttökokemusta. Vaikka sovellus näyttää natiivisovellukselta, ei se toimikkaan vastaavasti, mikä voi aiheuttaa käyttäjässä ihmetystä. Myös selaimella pitkien listojen vierittäminen alaspäin voi tuntua tahmaiselta, kun taas natiivisti ne toimivat sulavasti.

Sovelluskehitysmenetelmän valintaan vaikuttaa sovelluksen tarkoitus ja julkaisija. HTML5-sovelluksen myyminen ja mainostaminen yleiseen tietoon voi olla vaikeaa. Yksittäisen sovelluskehittäjän kannattaakin tällöin suosia hybridisovellusta, koska se voidaan laittaa kauppapaikkaan. Sitä kautta sovellukselle saadaan helposti näkyvyyttä ja sitä voidaan myydä ilman erillisiä kirjautumisominaisuuksia. Jos julkaisija on kuitenkin joku valmiiksi tunnettu sivusto, kuten esimerkkinä Iltalehti, voitaisiin sovellus tehdä suoraan HTML5:llä. Tällöin toteutukseen voitaisiin tehdä erillinen optimointi mobiililaitteille ja tietokoneilla sivusto näkyisi täytenä versiona. Sivusto ei näin tarvitsisi erillistä mainostusta, koska sillä on jo valmiiksi laajaa mobiilikäyttöä.

Natiivisovelluksien käyttö on usein sulavampaa kuin HTML5-tekniikoihin pohjautuvissa. Mikäli resurssit ja osaaminen riittävät, olisi niiden toteuttaminen suositeltavaa. Koska Applen osuus on niin merkittävä mobiilimarkkinoilla, sille olisi hyvä tehdä myös sovellus, jos se halutaan laajaan jakoon. Tällöin natiivisovelluksia tehdessä joudutaan panostamaan enemmän laitteistoon. HTML5-sovelluskehitys onnistuu myös Windows- ja Linux-ympäristöissä.

On tilanteita, jossa sovelluksen toimiminen riittää yhdellä alustalla. Esimerkiksi yritysmaailmassa voidaan käyttää työpaikoilla vain tietyn alustan puhelimia. Näin ollen sovellukset voidaan toteuttaa yhdelle alustalle juuri tiettyä tarkoitusta varten. Tällöin sovelluksen sulavuus on usein tärkeässä asemassa ja sovelluskehittäjälle testaaminen helpompaa.

Sovellusten ylläpito ja päivittäminen on selkeästi helpompaa HTML5-sovelluksissa. Natiivi- ja hybridisovelluksien päivitysten yhteydessä joudutaan päivittämään aina koko sovelluspaketti kauppapaikkaan. Näissä tapauksissa sovellukselle tehdään joka kerta tarkistus, joka vie oman aikansa. Tarkistus voi viedä kauppapaikasta riippuen tunneista useisiin päiviin. HTML5:llä päivitys tapahtuu palvelimen sovellustiedostoja päivittämällä. Muokkaukset ovat heti näkyvissä kaikkien alustojen käyttäjille.

7 Yhteenveto

Lopputyössä käymme läpi älypuhelinten historiaa, nykytilannetta ja tulevaisuutta. Nykyään markkinoilla on kolme merkittävää alustaa, jotka ovat iOS, Windows Phone ja Android. Työssä käydään läpi näiden valmistajien ekosysteemejä.

Sovelluskehityksestä selitetään tarkemmin kolme tapaa, jotka ovat natiivi-, hybridi- ja HTML5-sovelluskehitys. Natiivisovelluskehityksessä käydään läpi kolmen merkittävimmän alustan sovelluskehitys. Hybridisovelluskehityksestä kerrotaan yleisluontaisesti, eikä keskitytä alustakohtaiseen sovelluskehitykseen.

Omissa toteutuksissa on tehty kolme sovellusta. Android-alustalle toteutetulla natiivisovelluksella nähdään lähimmät pankkiautomaatit.

Vastaava sovellus on toteutettu myös hybridi- ja HTML5-versioina. Nämä vastaavat käyttötavoiltaan hyvin pitkälle natiivisovellusta. Hybrid-sovellus on toteutettu ainoastaan Android-alustalle. HTML5-versio toimii uusimmilla Windows Phoneilla, Androideilla ja vanhemmilla iOS-laitteilla.

Vertailemme sovelluskehitystapoja keskenään ja sitä, mitkä ovat minkäkin tavan hyviä ja huonoja puolia. HTML5-tekniikoilla toteutus on suositeltavaa, mikäli sovellus on mahdollista niillä toteuttaa. HTML5:n rajoitukset voivat kuitenkin estää sen.

Projektina lopputyö oli mielenkiintoinen ja jokseenkin haastava, vaikka työtä emme varsinaiselle asiakkaalle tehneetkään. Kummallakaan meistä ei ollut mobiiliohjelmoinnista vahvaa kokemusta. Jouduimme siis tutkimaan meille hieman vierasta asiaa. Pääosaamisemme on lähinnä palvelinsovelluksissa, mutta client-side palvelinsovellusosaamisesta oli hyötyä tehdessä HTML5-pohjaisia sovelluksia.

Haastavinta työssä oli tiedon etsiminen. Kirjallisuutta aiheesta on vielä niukalti tarjolla, puhumattakaan suomenkielisestä kirjallisuudesta. Verkosta sovelluskehityksestä puolestaan tietoa löytyy, mutta luotettavat lähteiden kanssa täytyy olla erittäin tarkkana.

Lopputyön parissa pääsimme kehittämään omaa osaamistamme alati kehittyvissä mobiilipalveluissa. Vaikka kummatkin meistä työskentelevät palvelinsovellusten parissa, uskomme tästä työstä olevan meille hyötyä myös työelämässä. Organisaation muutoksen jälkeen osasto, jossa työskentelemme ottaa vastuun yhtiömme mobiilipalveluiden kehittämisestä, joten toivomme saavamme tulevaisuudessa uusia haasteita mobiilikehityksen puolelta.

Ryhmädynamiikkamme toimi hyvin. Teimme työtä itsenäisesti sekä yhdessä. Wordin apuna käytimme Microsoft OneDrive -pilvipalvelua, jonka avulla pystyimme samanaikaisesti työskentelemään lopputyön parissa. Työn edetessä oli selkeää, miten jaamme keskenämme insinöörityössämme sovelluksien ja tekstien valmistelun. Timo Kannalan vastuulle jäi natiivisovelluksen toteuttaminen ja Rolle Swanljungille taas hybridi- sekä HTML5-sovellukset. Yleinen osio ja vertailu on tehty yhdessä.

Lähteet

1 3. Distribute. Verkkodokumentti. https://developer.apple.com/programs/ios/distribute.html. Luettu 23.10.2014.

2 A short history of the BlackBerry. Verkkodokumentti. http://www.bbscnw.com/a-short-history-of-the-blackberry.php. Luettu 21.10.2014.

3 About the TSG (Technical Steering Group). Verkkodokumentti. https://www.tizen.org/about/about-tsg-technical-steering-group. Luettu 23.10.2014.

4 Account types, locations, and fees. Verkkodokumentti. http://msdn.microsoft.com/en-us/library/windows/apps/jj863494.aspx. Luettu 24.10.2014.

5 Activity. Verkkodokumentti. http://developer.android.com/reference/android/app/Activity.html. Luettu 4.11.2014.

6 Android 2.1 Platform. Verkkodokumentti. http://developer.android.com/about/versions/android-2.1.html. Luettu 21.10.2014.

7 Android Developers. Verkkodokumentti. https://plus.google.com/+AndroidDevelopers/posts/J1A5hc1ZnS1. Luettu 4.11.2014.

8 Android Lollipop. Verkkodokumentti. http://developer.android.com/about/versions/lollipop.html. Luettu 24.10.2014.

9 Android NDK. Verkkodokumentti. https://developer.android.com/tools/sdk/ndk/index.html. Luettu 24.10.2014.

10 Android Security Overview. Verkkodokumentti. https://source.android.com/devices/tech/security. Luettu 24.10.2014.

11 Android Studio BETA. Verkkodokumentti. http://developer.android.com/sdk/installing/studio.html. Luettu 24.10.2014.

12 Android Studio unveiled at Google I/O keynote. Verkkodokumentti. http://www.androidcentral.com/android-studio-unveiled-google-io-keynote. Luettu 24.10.2014.

13 Android/Fire OS. Verkkodokumentti. https://developer.amazon.com/public/solutions/platforms/android-fireos. Luettu 4.11.2014.

14 Android: Be together. Not the same. Verkkodokumentti. http://googleblog.blogspot.fi/2014/10/android-be-together-not-same.html. Luettu 21.10.2014.

15 Apache Cordova. Verkkodokumentti. http://cordova.apache.org. Luettu 21.10.2014.

16 App activation and deactivation for Windows Phone 8. Verkkodokumentti. http://msdn.microsoft.com/en-us/library/windows/apps/ff817008(v=vs.105).aspx. Luettu 11.11.2014.

17 App Brain Stats. Verkkodokumentti. http://www.appbrain.com/stats/number-of-android-apps. Luettu 21.10.2014.

18 App certification requirements for Windows Phone. Verkkodokumentti. http://msdn.microsoft.com/library/windows/apps/hh184843.aspx. Luettu 24.10.2014.

19 App Manifest. Verkkodokumentti. http://developer.android.com/guide/topics/manifest/manifest-intro.html. Luettu 5.11.2014.

20 App Store (iOS). Verkkodokumentti. http://en.wikipedia.org/wiki/App_Store_(iOS). Luettu 21.10.2014.

21 Apple unveils iPhone. Verkkodokumentti. http://www.macworld.com/article/1054769/iphone.html. Luettu 21.10.2014.

22 Apple's App Store: An economy for 1 percent of developers. Verkkodokumentti. http://www.cnet.com/news/apples-app-store-an-economy-for-1-percent-of-developers/. Luettu 21.10.2014.

23 Background Execution. Verkkodokumentti. https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/BackgroundExecution/BackgroundExecution.html#//apple_ref/doc/uid/TP40007072-CH4. Luettu 4.11.2014.

24 Basic Setup. Verkkodokumentti. https://developer.apple.com/library/ios/documentation/Swift/Conceptual/BuildingCocoaApps/index.html. Luettu 4.11.2014.

25 Beta test your app. Verkkodokumentti. http://msdn.microsoft.com/en-us/library/windows/apps/dn832612.aspx. Luettu 24.10.2014.

26 Beta Testing Your iOS App. Verkkodokumentti. https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/TestingYouriOSApp/TestingYouriOSApp.html. Luettu 10.11.2014

27 Certified 'Powered by Firefox OS' devices require Firefox Marketplace, minimum hardware specs. Verkkodokumentti. http://www.theverge.com/2013/2/28/4039320/certified-powered-by-firefox-os-devices-require-firefox-marketplace. Luettu 24.10.2014.

28 Common Language Runtime (CLR). Verkkodokumentti. http://msdn.microsoft.com/en-us/library/8bs2ecf4(v=vs.110).aspx. Luettu 11.11.2014.

29 Cross-Origin Resource Sharing. Verkkodokumentti. http://www.w3.org/TR/cors/#resource-requests. Luettu 5.11.2014.

30 Customize and sell. Verkkodokumentti. http://msdn.microsoft.com/library/windows/apps/jj206718.aspx. Luettu 24.10.2014.

31 Dell Venue Pro on sale at Microsoft stores, but good luck getting one. Verkko-dokumentti. http://www.engadget.com/2010/11/08/dell-venue-pros-on-sale-at-microsoft-stores-but-good-luck-getti/. Luettu 21.10.2014.

32 Develop custom apps for business. Verkkodokumentti. https://developer.apple.com/programs/volume/b2b/. Luettu 10.11.2014.

33 Devices. Verkkodokumentti. https://wiki.tizen.org/wiki/Devices. Luettu 23.10.2014.

34 Eclipse downloads. Verkkodokumentti. https://www.eclipse.org/downloads/index-developer.php. Luettu 24.10.2014.

35 Find Firefox OS near you. Verkkodokumentti. https://www.mozilla.org/en-US/firefox/os/devices/. Luettu 24.10.2014.

36 Firefox OS. Verkkodokumentti. https://developer.mozilla.org/en-US/Firefox_OS. Luettu 24.10.2014.

37 Gartner Worldwide Smartphone Sales 2008 Q1. Verkkodokumentti. http://www.gartner.com/newsroom/id/688116. Luettu 08.11.2014.

38 Gartner Worldwide Smartphone Sales 2008 Q2. Verkkodokumentti. http://www.gartner.com/newsroom/id/754112. Luettu 08.11.2014.

39 Gartner Worldwide Smartphone Sales 2008 Q3. Verkkodokumentti. http://www.gartner.com/newsroom/id/827912. Luettu 08.11.2014.

40 Gartner Worldwide Smartphone Sales 2008 Q4. Verkkodokumentti. http://www.gartner.com/newsroom/id/910112. Luettu 08.11.2014.

41 Gartner Worldwide Smartphone Sales 2009 Q1. Verkkodokumentti. http://www.gartner.com/newsroom/id/985912. Luettu 08.11.2014.

42 Gartner Worldwide Smartphone Sales 2009 Q2. Verkkodokumentti. http://www.gartner.com/newsroom/id/1126812. Luettu 08.11.2014.

43 Gartner Worldwide Smartphone Sales 2009 Q3. Verkkodokumentti. http://www.gartner.com/newsroom/id/1224645. Luettu 08.11.2014.

44 Gartner Worldwide Smartphone Sales 2009 Q4. Verkkodokumentti. http://www.gartner.com/newsroom/id/1306513. Luettu 08.11.2014.

45 Gartner Worldwide Smartphone Sales 2010 Q1. Verkkodokumentti. http://www.gartner.com/newsroom/id/1372013. Luettu 08.11.2014.

46 Gartner Worldwide Smartphone Sales 2010 Q2. Verkkodokumentti. http://www.gartner.com/newsroom/id/1421013. Luettu 08.11.2014.

47 Gartner Worldwide Smartphone Sales 2010 Q3. Verkkodokumentti. http://www.gartner.com/newsroom/id/1466313. Luettu 08.11.2014.

48 Gartner Worldwide Smartphone Sales 2010 Q4. Verkkodokumentti. http://www.gartner.com/newsroom/id/1543014. Luettu 08.11.2014.

49 Gartner Worldwide Smartphone Sales 2011 Q1. Verkkodokumentti. http://www.gartner.com/newsroom/id/1689814. Luettu 08.11.2014.

50 Gartner Worldwide Smartphone Sales 2011 Q2. Verkkodokumentti. http://www.gartner.com/newsroom/id/1764714. Luettu 08.11.2014.

51 Gartner Worldwide Smartphone Sales 2011 Q3. Verkkodokumentti. http://www.gartner.com/newsroom/id/1848514. Luettu 08.11.2014.

52 Gartner Worldwide Smartphone Sales 2011 Q4. Verkkodokumentti. http://www.gartner.com/newsroom/id/1924314. Luettu 08.11.2014.

53 Gartner Worldwide Smartphone Sales 2012 Q1. Verkkodokumentti. http://www.gartner.com/newsroom/id/2017015. Luettu 08.11.2014.

54 Gartner Worldwide Smartphone Sales 2012 Q2. Verkkodokumentti. http://www.gartner.com/newsroom/id/2120015. Luettu 08.11.2014.

55 Gartner Worldwide Smartphone Sales 2012 Q3. Verkkodokumentti. http://www.gartner.com/newsroom/id/2237315. Luettu 08.11.2014.

56 Gartner Worldwide Smartphone Sales 2012 Q4. Verkkodokumentti. http://www.gartner.com/newsroom/id/2335616. Luettu 08.11.2014.

57 Gartner Worldwide Smartphone Sales 2013 Q1. Verkkodokumentti. http://www.gartner.com/newsroom/id/2482816. Luettu 08.11.2014.

58 Gartner Worldwide Smartphone Sales 2013 Q2. Verkkodokumentti. http://www.gartner.com/newsroom/id/2573415. Luettu 08.11.2014.

59 Gartner Worldwide Smartphone Sales 2013 Q3. Verkkodokumentti. http://www.gartner.com/newsroom/id/2623415. Luettu 08.11.2014.

60 Gartner Worldwide Smartphone Sales 2013 Q4. Verkkodokumentti. http://www.gartner.com/newsroom/id/2665715. Luettu 08.11.2014.

61 Get the Android SDK. Verkkodokumentti. https://developer.android.com/sdk/index.html. Luettu 24.10.2014.

62 Getting Started with MIT App Inventor 2. Verkkodokumentti. http://appinventor.mit.edu/explore/get-started. Luettu 24.10.2014.

63 Google Play Services. Verkkodokumentti. http://developer.android.com/google/play-services/index.html. Luettu 4.11.2014.

64 Google Services. Verkkodokumentti. http://developer.android.com/google/index.html. Luettu 4.11.2014.

65 Google's iron grip on Android: Controlling open source by any means necessary. Verkkodokumentti. http://arstechnica.com/gadgets/2013/10/googles-iron-grip-on-android-controlling-open-source-by-any-means-necessary/. Luettu 4.11.2014.

66 HTC Dream (T-Mobile G1). Verkkodokumentti. http://www.engadget.com/products/htc/dream/specs/. Luettu 21.10.2014.

67 HTML5 Application Cache. Verkkodokumentti. http://www.w3schools.com/html/html5_app_cache.asp. Luettu 21.10.2014.

68 HTML5 Canvas. Verkkodokumentti. http://www.w3schools.com/html/html5_canvas.asp. Luettu 21.10.2014.

69 HTML5 compatibility on mobile and tablet browsers with testing on real devices. Verkkodokumentti. http://mobilehtml5.org/. Luettu 21.10.2014.

70 HTML5 Features on Tizen. Verkkodokumentti. https://developer.tizen.org/documentation/articles/html5-features-on-tizen. Luettu 23.10.2014.

71 HTML5: A vocabulary and associated APIs for HTML and XHTML. Verkkodokumentti. http://www.w3.org/TR/html5/. Luettu 21.10.2014.

72 Implementing GCM Client. Verkkodokumentti. https://developer.android.com/google/gcm/client.html. Luettu 23.10.2014.

73 Installing the Eclipse ADT Bundle. Verkkodokumentti. https://developer.android.com/sdk/installing/index.html?pkg=adt. Luettu 24.10.2014.

74 Introducing Google Drive… yes, really. Verkkodokumentti. http://googleblog.blogspot.fi/2012/04/introducing-google-drive-yes-really.html. Luettu 21.10.2014.

75 iTunes App Store Now Has 1.2 Million Apps, Has Seen 75 Billion Downloads To Date. Verkkodokumentti. http://techcrunch.com/2014/06/02/itunes-app-store-now-has-1-2-million-apps-has-seen-75-billion-downloads-to-date/. Luettu 21.10.2014.

76 JQuery write less, do more. Verkkodokumentti. http://jquery.com/. Luettu 10.11.2014.

77 Marketplace-esittely. Verkkodokumentti. http://www.windowsphone.com/fi-FI/how-to/wp7/apps/marketplace-hub. Luettu 21.10.2014.

78 Mobile World Congress 2010 – day one overview. Verkkodokumentti. http://www.techcentral.co.za/mobile-world-congress-2010-day-one-overview/12826/. Luettu 21.10.2014.

79 Multi-resolution apps for Windows Phone 8. Verkkodokumentti. http://msdn.microsoft.com/en-us/library/windows/apps/jj206974(v=vs.105).aspx. Luettu 7.11.2014.

80 Nokia Confirms The PureView Was Officially The Last Symbian Phone. Verkkodokumentti. http://techcrunch.com/2013/01/24/nokia-confirms-the-pure-view-was-officially-the-last-symbian-phone/. Luettu 10.11.2014.

81 Nokia Corporation Q4 and full year 2012 Interim Report. Verkkodokumentti. http://company.nokia.com/sites/default/files/www.results.nokia.com/results/Nokia_results2012Q4e.pdf. Luettu 10.11.2014.

82 Nokia transitions Symbian source to non-open license. Verkkodokumentti. http://arstechnica.com/gadgets/2011/04/nokia-transitions-symbian-source-to-non-open-license/. Luettu 10.11.2014.

83 Nokia X Platform overview. Verkkodokumentti. http://developer.nokia.com/nokia-x/platform-overview. Luettu 4.11.2014.

84 OneDrive käytön aloittaminen. Verkkodokumentti. http://windows.microsoft.com/fi-fi/windows-8/getting-started-onedrive-tutorial. Lu-ettu 21.10.2014.

85 Partner program. Verkkodokumentti. https://www.tizenassociation.org/tizen-association-partner-program/. Luettu 23.10.2014.

86 See all iPhone models. Verkkodokumentti. https://www.apple.com/iphone/compare/. Luettu 7.11.2014.

87 Smartphone OS Market Share, Q2 2014. Verkkodokumentti. http://www.idc.com/prodserv/smartphone-os-market-share.jsp. Luettu 21.10.2014.

88 Smartphone. Verkkodokumentti. http://en.wikipedia.org/wiki/Smartphone. Luettu 21.10.2014.

89 Suomi on Nokia-maa: Windows Phonen markkinaosuus 39%. Verkkodokumentti. http://www.talouselama.fi/uutiset/suomi+on+nokiamaa+windows+phonen+markkinaosuus+39/a2213832. Luettu 21.10.2014.

90 Symbian OS. Verkkodokumentti. http://developer.nokia.com/community/wiki/Symbian_OS. Luettu 10.11.2014.

91 The ahead-of-time compiler. Verkkodokumentti. http://publib.boulder.ibm.com/infocenter/realtime/v1r0/index.jsp?topic=%2Fcom.ibm.rt.doc.10%2Frealtime%2Faot.html. Luettu 24.10.2014.

92 The App Life Cycle. Verkkodokumentti. https://developer.apple.com/library/ios/documentation/iphone/conceptual/iphoneosprogrammingguide/TheAppLifeCycle/TheAppLifeCycle.html. Luettu 4.11.2014.

93 The WebKit Open Source Project. Verkkolähde. https://www.webkit.org. Luettu 21.10.2014.

94 T-Mobile G1: Full Details of the HTC Dream Android Phone. Verkkodokumentti. http://gizmodo.com/5053264/t-mobile-g1-full-details-of-the-htc-dream-androidphone. Luettu 21.10.2014.

95 Ubuntu Overview. Verkkodokumentti. http://www.ubuntu.com/phone. Luettu 21.10.2014.

96 What Is Managed Code? Verkkodokumentti. http://msdn.microsoft.com/en-us/library/windows/desktop/bb318664(v=vs.85).aspx. Luettu 11.11.2014.

97 What is Visual Studio Online? Verkkodokumentti. http://www.visualstudio.com/products/what-is-visual-studio-online-vs. Luettu 24.10.2014.

98 Windows App Studio. Your app in 4 steps. Verkkodokumentti. https://appstudio.windows.com/en-us. Luettu 23.10.2014.

99 Windows Phone API reference. Verkkodokumentti. http://msdn.microsoft.com/en-us/library/windows/apps/ff626516(v=vs.105).aspx. Luettu 11.11.2014.

100 Windows Phone architecture overview. Verkkodokumentti. https://dev.windowsphone.com/en-us/OEM/docs/Getting_Started/Windows_Phone_architecture_overview. Luettu 23.10.2014.

Android Java - MainActivity.java

```
package missa.otto.maatti;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Locale;

import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.protocol.BasicHttpContext;
import org.apache.http.protocol.HttpContext;

import com.google.android.gms.common.ConnectionResult;
import com.google.android.gms.common.GooglePlayServicesClient;
import com.google.android.gms.common.GooglePlayServicesUtil;
import com.google.android.gms.location.LocationClient;
import com.google.android.gms.location.LocationRequest;
import com.google.android.gms.maps.CameraUpdateFactory;
import com.google.android.gms.maps.GoogleMap;
import com.google.android.gms.maps.MapFragment;
import com.google.android.gms.maps.model.BitmapDescriptorFactory;
import com.google.android.gms.maps.model.LatLng;
import com.google.android.gms.maps.model.Marker;
import com.google.android.gms.maps.model.MarkerOptions;

import android.app.ActionBar;
import android.app.Activity;
import android.app.Dialog;
```

```
import android.app.DialogFragment;
import android.app.FragmentTransaction;
import android.content.Intent;
import android.content.IntentSender;
import android.location.Location;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentActivity;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.ListView;
import android.widget.ProgressBar;
import android.widget.Toast;

public class MainActivity extends FragmentActivity implements
ActionBar.TabListener,
GooglePlayServicesClient.ConnectionCallbacks,
GooglePlayServicesClient.OnConnectionFailedListener,
com.google.android.gms.location.LocationListener {

                private String csvUrl = "http://otto.fi/wp-admin/admin.php?tuo=1";

    SectionsPagerAdapter mSectionsPagerAdapter;
    Location currentLocation;
```

```
    Marker currentMarker;
    ViewPager mViewPager;
                ListView listView;
                LocationClient mLocationClient;
                LocationRequest mLocationRequest;

                private static final int MILLISECONDS_PER_SECOND = 1000;
    public static final int UPDATE_INTERVAL_IN_SECONDS = 1;
    private static final long UPDATE_INTERVAL = MILLISECONDS_PER_SECOND *
UPDATE_INTERVAL_IN_SECONDS;
    private static final int FASTEST_INTERVAL_IN_SECONDS = 5;
    private static final long FASTEST_INTERVAL = MILLISECONDS_PER_SECOND *
FASTEST_INTERVAL_IN_SECONDS;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final ActionBar actionBar = getActionBar();
        actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
        mSectionsPagerAdapter = new SectionsPagerAdapter(getSupportFragmentMan-
ager());
        mViewPager = (ViewPager) findViewById(R.id.pager);
        mViewPager.setAdapter(mSectionsPagerAdapter);
        mViewPager.setOnPageChangeListener(new  ViewPager.SimpleOnPageChange-
Listener() {
            @Override
            public void onPageSelected(int position) {
                actionBar.setSelectedNavigationItem(position);
            }
        });

        for (int i = 0; i < mSectionsPagerAdapter.getCount(); i++) {
            actionBar.addTab(actionBar.newTab()
                        .setText(mSectionsPagerAdapter.getPageTitle(i))
```

```
                    .setTabListener(this));
        }
        new DownloadAsyncTask().execute();
        servicesConnected();
        mLocationClient = new LocationClient(this, this, this);
        mLocationRequest = LocationRequest.create();
        mLocationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
        mLocationRequest.setInterval(UPDATE_INTERVAL);
        mLocationRequest.setFastestInterval(FASTEST_INTERVAL);

    }


    @Override
    protected void onStart() {
        super.onStart();
        mLocationClient.connect();
    }

    @Override
    protected void onResume() {
            super.onResume();
    }

    @Override
    protected void onPause() {
            super.onPause();
    }

    @Override
    protected void onStop() {
            mLocationClient.removeLocationUpdates(this);
        mLocationClient.disconnect();
        super.onStop();
    }
```

```
    @Override
    protected void onDestroy() {
            super.onDestroy();
    }

    private ArrayList<OttoMaattiDistance> refreshAtmData(ArrayList<OttoMaatti> atms) {
            ArrayList<OttoMaattiDistance> list = new ArrayList<OttoMaattiDistance>();
            ArrayList<OttoMaattiDistance> shortList = new ArrayList<OttoMaattiDis-
tance>();
            for (OttoMaatti otto : atms) {
                    Location atmLocation = new Location("GPS");
                    atmLocation.setLatitude(otto.getLatitude());
                    atmLocation.setLongitude(otto.getLongitude());
                    atmLocation.setAltitude(currentLocation.getAltitude());
                    double distance = currentLocation.distanceTo(atmLocation);
                    list.add(new OttoMaattiDistance(otto, distance));
                    }
            Collections.sort(list, new OttoMaattiDistanceComparator());
            for (int i = 0; i < 30; i++) {
                    shortList.add(list.get(i));
            }
            return shortList;
    }

            private void updateMylocation(Location location) {
                    MapFragment mapFragment = (MapFragment) getFragment-
Manager().findFragmentById(R.id.map);
                    GoogleMap googleMap = mapFragment.getMap();
                    if (currentMarker != null)
                    {
                            currentMarker.remove();
                    }
                    final LatLng pin = new LatLng(currentLocation.getLatitude(),
currentLocation.getLongitude());
```

```
                    currentMarker = googleMap.addMarker(new MarkerOp-
tions()
        .position(pin).title("Nykyinen sijaintini")
        .icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFac-
tory.HUE_BLUE)));
                }


    private void updateAtmDistList(ArrayList<OttoMaatti> atmDataList) {
                listView = (ListView) findViewById(R.id.nearByAtmList);
                    OttoMaattiListAdapter adapter = new OttoMaattiListA-
dapter(MainActivity.this, refreshAtmData(atmDataList));
                    listView.setAdapter(adapter);
                }

    private void initAtmList(ArrayList<OttoMaatti> atmDataList) {
                listView = (ListView) findViewById(R.id.nearByAtmList);
                listView.setOnItemClickListener(new OnItemClickListener() {
                                    @Override
                                    public void onItemClick(AdapterView<?> par-
ent, View view, int position, long id) {
                                                OttoMaattiDistance omd = (Ot-
toMaattiDistance)parent.getAdapter().getItem(position);
                                                MapFragment mapFragment =
(MapFragment) getFragmentManager().findFragmentById(R.id.map);
                                                GoogleMap googleMap = map-
Fragment.getMap();

                                googleMap.moveCamera(CameraUpdateFactory.new-
LatLngZoom(
                                                new
LatLng(omd.getOttoMaatti().getLatitude(), omd.getOttoMaatti().getLongitude()), 14));
                                mViewPager.setCurrentItem(0);
                                    }
        });
```

```
                updateAtmDistList(atmDataList);
    }

                private void initAtmMap(ArrayList<OttoMaatti> atmDataList) {
                                MapFragment mapFragment = (MapFragment) getFragment-
Manager().findFragmentById(R.id.map);
                                GoogleMap googleMap = mapFragment.getMap();
        if (currentLocation != null) {
                        googleMap.moveCamera(CameraUpdateFactory.new-
LatLngZoom(new LatLng(currentLocation.getLatitude(), currentLocation.getLongi-
tude()), 12));
        }
        for (OttoMaatti otto : atmDataList) {
                final LatLng pin = new LatLng(otto.getLatitude() , otto.getLongitude());
                                String info = "";
                                if (!otto.getInfo().equals("")) {
                                                info = "(" + otto.getInfo() + ")";
                                }
           googleMap.addMarker(new MarkerOptions()
           .position(pin).title(otto.getAddress() + ", " + otto.getCity() + " - " + otto.getLoca-
tion() + info));
        }
                }

                Handler handlerDownloadAsyncTask = new Handler() {
        @Override
        public void handleMessage(Message msg) {
                @SuppressWarnings("unchecked")
                                                ArrayList<OttoMaatti> atmList = (ArrayList<Ot-
toMaatti>) msg.obj;
                Toast.makeText(MainActivity.this, "Data päivitetty, " + atmList.size() + " ot-
toa löydetty.", Toast.LENGTH_LONG).show();
                initAtmList(atmList);
                initAtmMap(atmList);
                ProgressBar progresBar = (ProgressBar) findViewById(R.id.progressBar);
```

```
                progresBar.setVisibility(View.GONE);
        }
    };

    public class DownloadAsyncTask extends AsyncTask<Void, Integer, ArrayList<Ot-
toMaatti>> {
                int myProgress;
                @Override
                protected void onPostExecute(ArrayList<OttoMaatti> result) {
                        Message msg = new Message();
        msg.obj = result;
        handlerDownloadAsyncTask.sendMessage(msg);
                }
                @Override
                protected void onPreExecute() {
                        myProgress = 0;
                }
                @Override
                protected ArrayList<OttoMaatti> doInBackground(Void... params) {
                                        return downloadAtms(csvUrl);
                }
                @Override
                protected void onProgressUpdate(Integer... values) {
                }
    }


    private ArrayList<OttoMaatti> downloadAtms(String csvPath) {
                ArrayList<OttoMaatti> ottoMaatit = new ArrayList<OttoMaatti>();
                BufferedReader br = null;
                String line = "";
                String cvsSplitBy = ";";
                try {
                        HttpClient httpClient = new DefaultHttpClient();
        HttpContext localContext = new BasicHttpContext();
```

```
HttpGet httpGet = new HttpGet(csvPath);
HttpResponse response = httpClient.execute(httpGet, localContext);
br = new BufferedReader(new InputStreamReader(response.getEntity().getCon-
tent(), "ISO-8859-1"));
                while ((line = br.readLine()) != null) {
                        String[] atm = line.split(cvsSplitBy);
                        OttoMaatti tempAtm = null;
                        try {
                                String info = (String)atm[0].sub-
string(1, atm[0].length()-1);
                                String atmID = (String)atm[1].sub-
string(1, atm[1].length()-1);
                                String address = atm[2].sub-
string(1, atm[2].length()-1);
                                String postalCode = atm[3].sub-
string(1, atm[3].length()-1);
                                String city = atm[4].substring(1,
atm[4].length()-1);
                                String location = atm[5].sub-
string(1, atm[5].length()-1);
                                String temtLat = (String)
atm[6].substring(1, atm[6].length()-1);
                                double latitude = Dou-
ble.parseDouble(temtLat);
                                String temtLon = (String)
atm[7].substring(1, atm[7].length()-1);
                                double longitude = Dou-
ble.parseDouble(temtLon);
                                double altitude = 0;
                                tempAtm = new OttoMaatti(info,
atmID, address, postalCode, city, location, latitude, longitude, altitude);
                        }
                        catch (Exception e)
                        {
                                //e.printStackTrace();
```

```
                                        }
                                        if (tempAtm != null){
                                                ottoMaatit.add(tem-
pAtm);
                                                }
                                                else
                                                {

                                                }
                        }
                } catch (FileNotFoundException e) {
                        //e.printStackTrace();
                } catch (IOException e) {
                        //e.printStackTrace();
                } finally {
                        if (br != null) {
                                try {
                                        br.close();
                                } catch (IOException e) {
                                        //e.printStackTrace();
                                }
                        }
                }
                return ottoMaatit;
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {

        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
```

```
    public void onTabSelected(ActionBar.Tab tab, FragmentTransaction fragmentTrans-
action) {
        mViewPager.setCurrentItem(tab.getPosition());
    }

    @Override
    public void onTabUnselected(ActionBar.Tab tab, FragmentTransaction frag-
mentTransaction) {
    }

    @Override
    public void onTabReselected(ActionBar.Tab tab, FragmentTransaction frag-
mentTransaction) {
    }

    public class SectionsPagerAdapter extends FragmentPagerAdapter {

        public SectionsPagerAdapter(FragmentManager fm) {
            super(fm);
        }

        @Override
        public Fragment getItem(int position) {

                if (position == 0)
                {
                        return new MapSctionFragment();
                } else {
                        return new ListSctionFragment();
                }
        }

        @Override
        public int getCount() {
            return 2;
```

```
    }

    @Override
    public CharSequence getPageTitle(int position) {
        Locale l = Locale.getDefault();
        switch (position) {
            case 0:
                return getString(R.string.title_section1).toUpperCase(l);
            case 1:
                return getString(R.string.title_section2).toUpperCase(l);
        }
        return null;
    }
  }

  public static class MapSctionFragment extends Fragment {

              public MapSctionFragment() {

              }
              @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        View rootView = inflater.inflate(R.layout.map, container, false);
        ProgressBar progresBar = (ProgressBar) rootView.findViewById(R.id.progress-
Bar);
        progresBar.setVisibility(View.VISIBLE);
        return rootView;
    }
  }

  public static class ListSctionFragment extends Fragment {
    public ListSctionFragment() {

    }
```

```
        @Override
        public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
            View rootView = inflater.inflate(R.layout.list, container, false);
            return rootView;
        }
    }

    private final static int CONNECTION_FAILURE_RESOLUTION_REQUEST = 9000;
                public static class ErrorDialogFragment extends DialogFragment {
        // Global field to contain the error dialog
        private Dialog mDialog;
        // Default constructor. Sets the dialog field to null
        public ErrorDialogFragment() {
            super();
            mDialog = null;
        }
        // Set the dialog to display
        public void setDialog(Dialog dialog) {
            mDialog = dialog;
        }
        // Return a Dialog to the DialogFragment.
        @Override
        public Dialog onCreateDialog(Bundle savedInstanceState) {
            return mDialog;
        }
    }

                @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        // Decide what to do based on the original request code
        switch (requestCode) {
            case CONNECTION_FAILURE_RESOLUTION_REQUEST :
                switch (resultCode) {
                    case Activity.RESULT_OK :
```

```
                break;
            }
        }
    }

                private boolean servicesConnected() {
        int resultCode = GooglePlayServicesUtil.isGooglePlayServicesAvailable(this);
        if (ConnectionResult.SUCCESS == resultCode) {
            return true;

        } else {
            Dialog errorDialog = GooglePlayServicesUtil.getErrorDialog(resultCode, this,
CONNECTION_FAILURE_RESOLUTION_REQUEST);
            if (errorDialog != null) {
                ErrorDialogFragment errorFragment = new ErrorDialogFragment();
                errorFragment.setDialog(errorDialog);
                errorFragment.show(this.getFragmentManager(), "Location Updates");
            }
            return false;
        }
    }

                @Override
                public void onConnectionFailed(ConnectionResult connectionResult) {
                        if (connectionResult.hasResolution()) {
            try {
                connectionResult.startResolutionForResult(this, CONNECTION_FAIL-
URE_RESOLUTION_REQUEST);
            } catch (IntentSender.SendIntentException e) {
                e.printStackTrace();
            }
        } else {
            showDialog(connectionResult.getErrorCode());
        }
```

```
        }

        @Override
        public void onConnected(Bundle bundle) {
    mLocationClient.requestLocationUpdates(mLocationRequest, this);
    currentLocation = mLocationClient.getLastLocation();
        }

        @Override
        public void onDisconnected() {
        }

        @Override
        public void onLocationChanged(Location location) {
                currentLocation = location;
                updateMylocation(location);
        }
}
```

Android Java - OttoMatti.java

```
package missa.otto.maatti;
public class OttoMaatti {
		private String info;
		private String atmID;
		private String address;
		private String postalCode;
		private String city;
		private String location;
		private double latitude;
		private double longitude;
		private double altitude;
		public OttoMaatti(String info, String atmID, String address,
					String postalCode, String city, String location,
					double latitude, double longitude, double alti-
tude) {
				this.info = info;
				this.atmID = atmID;
				this.address = address;
				this.postalCode = postalCode;
				this.city = city;
				this.location = location;
				this.latitude = latitude;
				this.longitude = longitude;
				this.altitude = altitude;
		}
		public String getInfo() {
				return info;
		}
		public void setInfo(String info) {
				this.info = info;
		}
		public String getAtmID() {
				return atmID;
```

```
}
public void setAtmID(String atmID) {
		this.atmID = atmID;
}
public String getAddress() {
		return address;
}
public void setAddress(String address) {
		this.address = address;
}
public String getPostalCode() {
		return postalCode;
}
public void setPostalCode(String postalCode) {
		this.postalCode = postalCode;
}
public String getCity() {
		return city;
}
public void setCity(String city) {
		this.city = city;
}
public String getLocation() {
		return location;
}
public void setLocation(String location) {
		this.location = location;
}
public double getLatitude() {
		return latitude;
}
public void setLatitude(double latitude) {
		this.latitude = latitude;
}
public double getLongitude() {
```

```
                return longitude;
        }
        public void setLongitude(double longitude) {
                this.longitude = longitude;
        }
        public double getAltitude() {
                return altitude;
        }
        public void setAltitude(double altitude) {
                this.altitude = altitude;
        }
}
```

Android Java - OttoMaattiDistance.java

```
package missa.otto.maatti;
public class OttoMaattiDistance implements Comparable<OttoMaattiDistance> {
		OttoMaatti ottoMaatti;
		double distance;

		public OttoMaattiDistance(OttoMaatti ottoMaatti, double distance) {
			this.ottoMaatti = ottoMaatti;
			this.distance = distance;
		}
		public OttoMaatti getOttoMaatti() {
			return ottoMaatti;
		}
		public void setOttoMaatti(OttoMaatti ottoMaatti) {
			this.ottoMaatti = ottoMaatti;
		}
		public double getDistance() {
			return distance;
		}
		public void setDistance(double distance) {
			this.distance = distance;
		}
		@Override
		public int compareTo(OttoMaattiDistance another) {
			if (this.distance > another.distance) {
				return 1;
			} else if (this.distance < another.distance) {
				return -1;
			} else {
				return 0;
			}
		}
}
```

Android Java - OttoMaattiDistanceComparator.java

```
package missa.otto.maatti;
import java.util.Comparator;
public class OttoMaattiDistanceComparator implements Comparator<OttoMaattiDis-
tance>  {
		@Override
		public int compare(OttoMaattiDistance lhs, OttoMaattiDistance rhs) {
				return lhs.compareTo(rhs);
		}
}
```

Android Java - OttoMaattiListAdapter.java

```
package missa.otto.maatti;
import java.util.ArrayList;
import java.util.List;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.TextView;
public class OttoMaattiListAdapter extends ArrayAdapter<OttoMaattiDistance> {
                private List<OttoMaattiDistance> itemList;
                private Context context;
                public OttoMaattiListAdapter(Context context, List<OttoMaattiDistance>
itemList) {
                                super(context, android.R.layout.simple_list_item_1, item-
List);
                                this.itemList = itemList;
                                this.context = context;
                }
                public int getCount() {
                   if (itemList != null)
                      return itemList.size();
                   return 0;
                   }
                public OttoMaattiDistance getItem(int position) {
      if (itemList != null)
         return itemList.get(position);
      return null;
   }

   public long getItemId(int position) {
      if (itemList != null)
         return itemList.get(position).hashCode();
```

```
        return 0;
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View v = convertView;
        if (v == null) {
            LayoutInflater inflater = (LayoutInflater) context.getSystemService(Con-
text.LAYOUT_INFLATER_SERVICE);
            v = inflater.inflate(R.layout.ottomaatti, null);
        }
        OttoMaattiDistance od = itemList.get(position);
        TextView row1 = (TextView) v.findViewById(R.id.row1);
        TextView row2 = (TextView) v.findViewById(R.id.row2);
        String info = "";
        if (!od.getOttoMaatti().getInfo().equals("")) {
                info += " (" + od.getOttoMaatti().getInfo() + ")";
        }
        double distance = od.getDistance();
        distance = distance / 1000;
        distance = Math.round(distance * 1000);
        distance = distance/1000;
        String row1txt = "" + od.getOttoMaatti().getAddress() + ", " +
od.getOttoMaatti().getCity();
        String row2txt = "" + distance + "km - " + od.getOttoMaatti().getLocation() + info;
        row1.setText(row1txt);
        row2.setText(row2txt);
        return v;
    }
    public List<OttoMaattiDistance> getItemList() {
        return itemList;
    }
    public void setItemList(ArrayList<OttoMaattiDistance> itemList) {
        this.itemList = itemList;
    }
}
```

Andoid XML

activity_main.xml

```
<android.support.v4.view.ViewPager xmlns:android="http://schemas.an-
droid.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/pager"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" />
```

map.xml

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity$ListSctionFragment" >
                <fragment xmlns:android="http://schemas.android.com/apk/res/android"
                  android:id="@+id/map"
                  android:name="com.google.android.gms.maps.MapFragment"
                  android:layout_width="match_parent"
                  android:layout_height="match_parent"/>
                <ProgressBar
                   android:id="@+id/progressBar"
                   style="?android:attr/progressBarStyleLarge"
                   android:layout_width="wrap_content"
                   android:layout_height="wrap_content"
                   android:layout_centerHorizontal="true"
                   android:visibility="invisible" />
</RelativeLayout>
```

list.xml

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context=".MainActivity$ListSctionFragment" >
    <TextView
        android:id="@+id/info"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/infoTxt"
        android:paddingBottom="10dip" />
                <ListView
                    android:id="@+id/nearByAtmList"
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:layout_below="@+id/info"
                    android:fastScrollEnabled="true" />
</RelativeLayout>
```

ottomaatti.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <TextView
        android:id="@+id/row1"
        android:layout_width="match_parent"
```

```
                android:layout_height="wrap_content"
                android:textColor="#000"/>
    <TextView
        android:id="@+id/row2"
        android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:textColor="#000"/>
</LinearLayout>
```

Hybrid – Ottopisteethybrid.js

```
var map;
var pinLayer, pinInfo;
var store;
var assetURL = "http://otto.fi/wp-admin/admin.php?tuo=1";
var fileName = "ottopisteet.csv";

$(function(){
		document.addEventListener("deviceready", init, false);
})

function init() {
		store = cordova.file.dataDirectory;
		window.resolveLocalFileSystemURL(store + fileName, onDeviceReady,
downloadAsset);
}

function downloadAsset() {
		var fileTransfer = new FileTransfer();
		fileTransfer.download(assetURL, store + fileName,
		function(entry) {
		},
		function(err) {
			alert("Error");
		});
}

function onDeviceReady() {
		var options = { timeout: 10000, enableHighAccuracy: true };
		navigator.geolocation.getCurrentPosition(onBingGetLocationSuccess,
onBingGetLocationError, options);
}
```

```
function onBingGetLocationSuccess(location){
		var newHeight = $("#page1").height();
		newHeight -= $("#header").height();
		newHeight -= $("#tabbar").height();
		$("#movingBallG").hide();
		$("#mapDiv").height(newHeight);
		map = new Microsoft.Maps.Map(document.getElementById("mapDiv"), {
				credentials: "AhetmhJkFfYPVlVnINsZTPIf8-IRybLrS-PHj4wDJqBvpsD6ev93ZyCg2DcZRoCI4",
	center: new Microsoft.Maps.Location(location.coords.latitude,location.coords.longitude),
	mapTypeId: Microsoft.Maps.MapTypeId.road,
	disableBirdseye:true,
	showDashboard:false,
	showCopyright:false,
	zoom: 14
   });
		pinLayer = new Microsoft.Maps.EntityCollection();
		map.entities.push(pinLayer);
		var pushpin = new Microsoft.Maps.Pushpin(map.getCenter(), {icon: 'img/poi_custom.png', width: 50, height: 50, draggable: false});
		pushpin.Title = "Oma sijainti";
		pushpin.Description = "";
		pinLayer.push(pushpin);
		$.ajax({
		type: "GET",
		url: store + fileName,
		dataType: "text",
		contentType: 'Content-type: text/plain; charset=iso-8859-1',
		beforeSend: function(jqXHR) {
				jqXHR.overrideMimeType('application/x-www-form-urlencoded; charset=ISO-8859-1');
				},
		success: function(data) {processData(data);}
		});
```

```
}

function processData(allText) {
    var allTextLines = allText.split(/\r\n|\n/);
    var headers = allTextLines[0].split(';');
    var lines = [];

    for (var i=0; i < allTextLines.length; i++) {
        var data = allTextLines[i].split(';');
        if (data.length == headers.length) {

            var tarr = [];
            for (var j=0; j<headers.length; j++) {
                tarr.push(data[j].substring(1,data[j].length-1));
            }
            tarr.push(0);
            lines.push(tarr);
        }
    }
                    initOttoMap(lines);
                    refreshOttoList(lines);
}

function onBingGetLocationError(){
  alert("Did not work!");
}

function initOttoMap(lines){
                    for (i = 0; i < lines.length; i++) {
                                        var pinLocation = new Microsoft.Maps.Location(parse-
Float(lines[i][6].toString()),parseFloat(lines[i][7].toString()));
                                        var NewPin;
                                        var infoTxt, infoTxt2;
                                        var info = lines[i][0];
                                        infoTxt = lines[i][2] + ", " + lines[i][4];
```

```
                        if (info == "")
                        {
                                infoTxt2 = lines[i][5];
                        }
                        else {
                                infoTxt2 = lines[i][5] + " (" + info + ")";
                        }
                        NewPin = new Microsoft.Maps.Pushpin(pinLocation, { text: ""
});
                        NewPin.Title = infoTxt;
                        NewPin.Description = infoTxt2;
                Microsoft.Maps.Events.addHandler(NewPin, 'click', displayInfobox);
                        pinLayer.push(NewPin);


                }
    Microsoft.Maps.Events.addHandler(map, 'viewchange', hideInfobox);
    var infoboxLayer = new Microsoft.Maps.EntityCollection();
    map.entities.push(infoboxLayer);

    pinInfo = new Microsoft.Maps.Infobox(NewPin.getLocation(),
    {visible: false, offset: new Microsoft.Maps.Point(0,15)});
    infoboxLayer.push(pinInfo);
}

function displayInfobox(e)
{
                if (e.targetType == "pushpin") {
                        pinInfo.setOptions({
                                width: 150,
                                height: 75,
            title: e.target.Title,
            description: e.target.Description,
            visible: true,
            offset: new Microsoft.Maps.Point(0, 25)
        });
```

```
pinInfo.setLocation(e.target.getLocation());
var buffer = 25;
                var infoboxOffset = pinInfo.getOffset();
                var infoboxAnchor = pinInfo.getAnchor();
                var infoboxLocation = map.tryLocationToPixel(e.target.get-
Location(), Microsoft.Maps.PixelReference.control);
                var dx = infoboxLocation.x;
                dx = dx + infoboxOffset.x;
                dx = dx - infoboxAnchor.x;
                var dy = infoboxLocation.y
                dy -= buffer;
                dy -= infoboxAnchor.y;
                if (dy < buffer) {
                        dy *= -1;
                        dy += buffer;
                }else{
                        dy = 0;
                }
                if(dx < buffer){
                        dx *= -1;
                        dx += buffer;
                }else{
                        dx = map.getWidth() - infoboxLocation.x + in-
foboxAnchor.x - pinInfo.getWidth();
                        if (dx > buffer) {
                                dx = 0;
                        }else{
                                dx -= buffer;
                        }
                }
                if (dx != 0 || dy != 0)
                {
                        map.setView({ centerOffset: new Mi-
crosoft.Maps.Point(dx, dy), center: map.getCenter() });
                }
```

```
		}
}

function hideInfobox(e)
{
    pinInfo.setOptions({ visible: false });
}

function refreshOttoList(lines){
		var loc = map.getCenter();
		var lat = loc.latitude;
		var lng = loc.longitude;
		for (i = 0; i < lines.length; i++) {
			var toLat = parseFloat(lines[i][6].toString());
			var toLng = parseFloat(lines[i][7].toString());
			var dist = calcDistance(lat, lng, toLat, toLng);

			lines[i][8] = Math.round(dist * 1000) / 1000;
		}
		lines.sort(sortByDistance);
		for (i = 0; i < 30; i++) {
			info = lines[i][0];
			if (info == "")
			{
				$("#ottoList").append("<li data-theme='c'><a href='#page1' onClick='moveToMap("+lines[i][6]+","+lines[i][7]+")' data-transition='slide'>"+lines[i][2]+", "+lines[i][4]+"</br>"+lines[i][8]+"km - " + lines[i][5] + "</a></li>");
			}
			else{
				$("#ottoList").append("<li data-theme='c'><a href='#page1' onClick='moveToMap("+lines[i][6]+","+lines[i][7]+")' data-transition='slide'>"+lines[i][2]+", "+lines[i][4]+"</br>"+lines[i][8]+"km - " + lines[i][5] + " (" + info + ")</a></li>");
			}
```

```
                }
}

function moveToMap(lat, lng){
                map.setView({zoom:14, center:new Microsoft.Maps.Location(lat, lng)});
}

function sortByDistance(a, b) {
   if (a[8] === b[8]) {
      return 0;
   }
   else {
      return (a[8] < b[8]) ? -1 : 1;
   }
}

function calcDistance(lat1, lon1, lat2, lon2) {
                var R = 6371;
                var dLat = toRad(lat2-lat1);
                  var dLon = toRad(lon2-lon1);
                  var lat1 = toRad(lat1);
                  var lat2 = toRad(lat2);
                  var a = Math.sin(dLat/2) * Math.sin(dLat/2) +
                  Math.sin(dLon/2) * Math.sin(dLon/2) * Math.cos(lat1) * Math.cos(lat2);
                  var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
                  var d = R * c;
                return d;
}
function toRad(Value)
{
   return Value * Math.PI / 180;
}
```

Hybrid – index.html

```
<html>
<head>
  <meta charset="ISO-8859-1">
  <title>Missä Otto.</title>
  <meta charset="utf-8" />
  <meta name="format-detection" content="telephone=no" />
  <meta name="msapplication-tap-highlight" content="no" />
  <!-- WARNING: for iOS 7, remove the width=device-width and height=device-height attributes. See https://issues.apache.org/jira/browse/CB-4323 -->
  <meta name="viewport" content="user-scalable=no, initial-scale=1, maximum-scale=1, minimum-scale=1, width=device-width, height=device-height, target-densitydpi=device-dpi" />

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="apple-mobile-web-app-capable" content="yes">
  <meta name="apple-mobile-web-app-status-bar-style" content="black">

  <link rel="stylesheet" type="text/css" href="css/index.css" />
  <link href="css/codiqa.ext.min.css" rel="stylesheet">
  <link href="css/jquery.mobile-1.3.1.min.css" rel="stylesheet">
</head>
<body>
  <div data-role="page" data-control-title="OttoKartalla" id="page1">
     <div id="header" data-theme="a" data-role="header">
        <h3>
           Ottomaatit
        </h3>
     </div>
     <div id="content" data-role="content">
        <!--<div data-role="fieldcontain" data-controltype="searchinput">
           <input name="" id="searchinput1" placeholder="" value="" type="search">
        </div>-->
        <div id="loadingScreen">
```

```
                    <div id="movingBallG">
                        <div class="movingBallLineG">
                        </div>
                        <div id="movingBallG_1" class="movingBallG">
                        </div>
                    </div>
                </div>
        <div id="mapDiv" style="position:relative; width:auto; height:400px"></div>
        <div id="tabbar" data-role="tabbar" data-iconpos="left" data-theme="a">
            <ul>
                <li>
                    <a href="#page1" data-transition="none" data-theme="" data-icon="">
                        Otto.Kartalla
                    </a>
                </li>
                <li>
                    <a href="#page2" data-transition="none" data-theme="" data-icon="">
                        Otto.Osoitteet
                    </a>
                </li>
            </ul>
        </div>
    </div>
</div>
<div data-role="page" data-control-title="OttoOsoitteet" id="page2">
    <div data-theme="a" data-role="header">
        <h3>
            Ottomaatit
        </h3>
    </div>
    <div data-role="content">
        <!--<div data-role="fieldcontain" data-controltype="searchinput">
            <input name="" id="searchinput2" placeholder="" value="" type="search">
        </div>-->
        <ul id="ottoList" data-role="listview" data-divider-theme="b" data-inset="true">
```

```
                <li data-role="list-divider" role="heading">
              EP = esteetön ja puhetuettu Otto (matala asiointikorkeus), P = puhetuettu
Otto (normaali korkeus), T = TalletusOtto
            </li>
          </ul>
        </div>
        <div data-role="tabbar" data-iconpos="left" data-theme="a">
          <ul>
            <li>
              <a href="#page1" data-transition="none" data-theme="" data-icon="">
                Otto.Kartalla
              </a>
            </li>
            <li>
              <a href="#page2" data-transition="none" data-theme="" data-icon="">
                Otto.Osoitteet
              </a>
            </li>
          </ul>
        </div>
    </div>
    <script src="js/jquery-1.9.1.min.js"></script>
    <script src="js/jquery.mobile-1.3.1.min.js"></script>
    <script src="js/codiqa.ext.min.js"></script>
    <script type="text/javascript" src="cordova.js"></script>
    <script type="text/javascript" src="http://ecn.dev.virtualearth.net/mapcontrol/mapcon-
trol.ashx?v=7.0&mkt=fi-FI,en-GB"></script>
    <script src="js/ottopisteethybrid.js"></script>
</body>
</html>
```