

Opinnäytetyö (AMK)

Tietotekniikan koulutusohjelma

Ohjelmistotuotanto

2014

Mikko Uusitalo

WINDOWS-PHONE SOVELLUSKEHITYS



TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

Mikko Uusitalo

WINDOWS PHONE SOVELLUSKEHITYS

Tämän opinnäytetyön tarkoituksena on tutustua mobiilisovelluskehitykseen ja Windows Phone mobiilikäyttöjärjestelmään. Työssä käydään läpi käyttöjärjestelmän perusteita ja tutustutaan sovelluskehitykseen liittyviin työkaluihin ja tekniikoihin. Lopuksi tutustutaan kurssityönä toteutettuun sovellukseen ja käydään läpi kuinka sovellus saatetaan myyntiin.

Windows Phone on Microsoftin kehittämä mobiilikäyttöjärjestelmä, joka julkaistiin vuonna 2010. Käyttöjärjestelmä on toteutettu käyttäen Silverlight-tekniikkaa, jonka avulla ohjelmistokehittäjät voivat myös toteuttaa omia sovelluksiaan kyseiselle alustalle. Lisäksi sovelluskehitystä voidaan tehdä käyttäen XNA-tekniikkaa, joka on suunniteltu erityisesti pelien tekemiseen.

Kaikki Windows Phone käyttöjärjestelmälle toteutetut sovellukset voidaan saattaa myyntiin Microsoftin mobiilisovelluskauppaan, Windows Phone Storeen. Opinnäytetyössä käydään läpi tarvittavat rekisteröitymisprosessit ja sovelluksen sertifiointi joiden avulla sovelluskehittäjä saa oman sovelluksensa jaettavaksi kuluttajille.

Lopuksi tutustutaan opinnäytetyön osana toteutettuun Frisbee Golf pistekorttisovellukseen ja erityisesti sen osana toteutettuun kartta-sivuun. Opinnäytetyössä esitellään vaiheittain kuinka kartta-sivu on toteutettu ja miten sovelluskehittäjä voi hyödyntää Microsoftin Bing karttarajapintoja.

Tätä opinnäytetyötä voidaan käyttää lähteenä, kun lähdetään tutustumaan mobiilisovelluskehitykseen ja Windows Phone mobiilikäyttöjärjestelmään kehitysalustana. Lisäksi sitä voidaan käyttää referenssinä kun ollaan päättämässä mille alustalle oma mobiilisovellus halutaan kehittää. Opinnäytetyö antaa myös yleiskatsauksen prosessiin, miten mobiilisovellus saatetaan ideavaiheesta myyntiin asti.

ASIASANAT:

Windows Phone, Visual Studio, Expression Blend, puhelinsovellus, Microsoft, Silverlight, XNA

Mikko Uusitalo

SOFTWARE DEVELOPMENT FOR WINDOWS PHONE

The purpose of this thesis is to look into mobile application development and Windows Phone mobile operating system. The thesis examines the basics for the operating system and introduces the tools and techniques related to application development. In addition, the study goes through the development of application which was made for it and discusses how it was put up for sale.

Windows Phone is a mobile operating system developed by Microsoft and it was released in late 2010. The operating system has been developed by using Silverlight technology which software developers can also use to develop their own applications to the platform. Apps can also be produced by using XNA technology, which is specially designed for game development.

All the Windows Phone applications can be put up for sale in Microsoft's mobile application store the Windows Phone Store. The thesis handles the necessary registration procedures and application certification that enables application distribution to consumers.

Moreover, the Frisbee Golf scorecard application and especially the Map Page are introduced. The design of Map Page is dealt with step by step and discussed how the application developer can take advantage of Microsoft's Bing Map interfaces.

This thesis can be used as a reference when getting familiar with mobile application development and Windows Phone mobile operating system as a development platform. Moreover, it can be used as a reference when deciding which platform to choose, when developing a mobile application. The thesis also gives a good overview of mobile application development, from design stage to selling the application.

KEYWORDS:

Windows Phone, Visual Studio, Expression Blend, mobile application, Microsoft, Silverlight, XNA

SISÄLTÖ

KÄYTETYT LYHENTEET (TAI SANASTO)	6
1 JOHDANTO	7
2 WINDOWS PHONE	9
2.1 Windows Mobile	9
2.2 Windows Phone 7	10
2.3 Windows Phone 8	12
2.4 Ominaisuudet	13
3 KEHITYSTYÖKALUT	17
3.1 Silverlight –tekniikka	17
3.1.1 XAML	18
3.1.2 C#	19
3.2 XNA-tekniikka	20
3.3 Kehitysvälineet	21
3.3.1 Visual Studio 2010	22
3.3.2 Expression Blend	24
4 MARKKINOINTI	26
4.1 Windows Phone Store	26
4.2 Ansaintamallit	28
4.3 Sovellusten saattaminen myyntiin	29
5 SOVELLUSPROJEKTI	32
5.1 Ongelma	32
5.2 Ratkaisu	33
5.3 Karttasivu	34
5.3.1 Bing Maps	34
5.3.2 A-GPS	35
5.3.3 Bing Routeservice	35
5.4 Suunnittelu	36
5.5 Käyttöliittymä	37
5.5.1 Sisältö-paneeli	39
5.5.2 Kontrolli-Paneeli	42

5.6 Toimintalogiikka	43
5.6.1 Kartta-kontrolli	44
5.6.2 Paikannuspalvelu	46
5.6.3 Reittipalvelupyynnön lähettäminen	49
5.6.4 Reittipalveluvastauksen käsittely	52
6 JATKOKEHITYS	55
7 YHTEENVETO	56
LÄHTEET	57

Kuvat

Kuva 1. Windows Phone 7-käyttöjärjestelmä (Wikipedia, 2011)	11
Kuva 2. People-hub (Segan, 2010)	14
Kuva 3. Windows Phone emulaattori	23
Kuva 4. Sovelluksen saattaminen myyntiin	30
Kuva 5. Frisbeegolf Pro	33
Kuva 6. Karttasivu	37
Kuva 7. Bing-karttanimiavaruuden lisääminen projektiin	39
Kuva 8. Käytettävien ominaisuuksien määrittely	44
Kuva 9. Routeservice-palvelun lisäys projektiin	49
Kuva 10. Reitti käyttäjän sijainnista valitulle radalle	54

KÄYTETYT LYHENTEET (TAI SANASTO)

Silverlight	Adobe Flashin tapainen Web-ohjelmointiympäristö
XNA	Microsoftin kehittämä oliopohjainen peliohjelmointikirjasto
HPC	Määritelmä, jolla kuvataan tietokoneita jotka ovat pienempiä kuin kannettava tietokone
NFC	Lyhyen kantaman tiedonsiirtoon tarkoitettu tekniikka
People-hub	Windows Phone toimintakokonaisuus, jonka avulla voidaan kommunikoida usealla eri tavalla
Metro-tyylikieli	Käytetään kuvaamaan Windows Phone käyttöjärjestelmässä käytettävää käyttöliittymän tyyliä
IL	Välikieli mihin korkean tason ohjelmointikieli käännetään ennen ajonaikaista käännöstä
JIT	Ajonaikainen käännös, välikieli käännetään lennossa konekieleksi
CLR	Virtuaalikone-komponentti, joka vastaa välikielen käännöksestä konekieleksi.
XAP	Pakattu tiedosto, mikä sisältää Silverlight ohjelman asennustiedostot
REST	HTTP-protokollaa käyttävä rajapintaratkaisu

1 JOHDANTO

Tänä päivänä jokainen ihminen maailmassa voi julkaista oman mobiilisovelluksen mille alustalle hyvänsä. Sovelluksilla voi jopa ansaita rahaa. Ongelma on kuitenkin siinä, mistä lähteä liikkeelle, kuinka toteuttaa oma sovellus, miten se julkaistaan ja tärkeimpänä, miten siitä tehdään kannattavaa liiketoimintaa. Tässä opinnäytetyössä aiheeseen tutustutaan Microsoftin toteuttaman Windows Phone käyttöjärjestelmän näkökulmasta.

Opinnäytetyössä tutustutaan ensin yleisesti Windows Phone mobiilikäyttöjärjestelmään, mikä se on ja mistä se on tullut. Lisäksi käydään läpi käyttöjärjestelmän tärkeimmät ominaisuudet ja käyttöfilosofia, joita kehittäjä tarvitsee toteuttaakseen mahdollisimman käyttöjärjestelmän näköisen sovelluksen. Näin saavutetaan yhtenäisyyttä eri sovellusten välillä ja sovellusta on helppo lähteä käyttämään.

Seuraavaksi käydään läpi sovellusten kehittämistapoja, kuten tekniikoita ja työkaluja. Windows Phone-käyttöjärjestelmän tapauksessa tekniikoita on kaksi: Silverlight ja XNA. Silverlight-tekniikalla tehdään sovelluksia ja XNA-tekniikalla pelejä. Työkaluissa tutustutaan syvällisemmin Microsoftin tarjoamiin työkaluihin, Visual Studioon ja Expression Blendiin.

Kun tiedetään, miten sovelluksia tehdään, selvitetään kuinka niitä voidaan jakaa kuluttajille. Opinnäytetyössä käydään läpi sovelluksen saattaminen omalta kotikoneelta myyntiin asti. Ensin selvitetään, kuinka Microsoftin kehittäjäpalveluun rekisteröidytään ja mitä tietoja kehittäjän tulee täyttää ennen sovelluksen myyntiä. Seuraavaksi tutustutaan erilaisiin ansaintamalleihin eli kuinka sovelluksilla voidaan ansaita rahaa. Lopuksi saatetaan sovellus myyntiin kauppaan.

Viimeisenä esitellään opinnäytetyön osaksi tehty sovellus, jossa hyödynnetään jo opittuja taitoja. Sovelluksessa tutustutaan lisäksi muutamiin Microsoftin tarjoamiin sovelluskomponentteihin ja rajapintoihin, joiden avulla voidaan osoittaa kuinka rikkaita sovelluksia voidaan toteuttaa käyttämällä jo olemassa olevia työ-

kaluja. Lisäksi sovellus toteutetaan valmiiden sovelluspohjien avulla, jotta käyttöösi liittymästä saadaan mahdollisimman käyttäjäystävällinen ja helposti omaksuttava.

2 WINDOWS PHONE

Windows Phone on Microsoftin kehittämä käyttöjärjestelmä, joka julkaistiin Barcelonan mobiilikehitysmessuilla keväällä 2010. Se perustuu Windows CE-käyttöjärjestelmäperheeseen ja on Microsoftin aiemmin kehittämän Windows mobile käyttöjärjestelmän seuraaja. Ensimmäiset Windowsit Phone–matkapuhelimet tuotiin markkinoille loppuvuodesta 2010 (Saltzman, 2012).

Windows Phone käyttöjärjestelmä julkaistiin kuitenkin vain muutamalle puhelinmallille ja vain muutamassa maassa, joten se jäi hyvin vähäiselle huomiolle. Vasta vuonna 2011 helmikuussa julkaistu yhteistyö Nokian kanssa sai aikaan mediamyrskyn, mikä toi käyttöjärjestelmän laajemman yleisön tietoisuuteen. (Microsoft 2011) Yhteistyö on muutenkin kantanut hedelmää ja Nokian lanseeraama Lumia-tuoteperhe on tuonut Windows Phone – käyttöjärjestelmän suosiolle sen kaipaamaa nostetta (O'Brien, 2012).

2.1 Windows Mobile

Windows Phone – mobiilikäyttöjärjestelmän juuret ulottuvat aina vuoden 1996 marraskuulle jolloin ensimmäisen Microsoftin kehittämä, sulautetuille järjestelmille tarkoitettu käyttöjärjestelmä, Windows CE julkaistiin. Se oli suunnattu niin sanotuille käsikäyttöisille tietokoneille (HPC) ja tarvitsi toimiakseen ainoastaan yhden megatavun keskusmuistia (HPC:Factor 2011).

Varsinainen kehitys matkapuhelinkäyttöjärjestelmäksi voidaan sanoa alkaneen jo vuonna 2000, jolloin julkaistiin Windows CE 3.0 versioon perustuva Pocket PC 2000 käyttöjärjestelmä. Se oli alun perin tarkoitettu muistikirjatyypisille Pocket PC nimisille laitteille, mutta siihen perustuvia matkapuhelimia julkaistiin myös. Niitä kutsuttiin nimellä SmartPhone 2002 ja niitä voidaan pitää ensimmäisinä älypuhelimina, jotka perustuvat Microsoftin julkaisemaan käyttöjärjestelmään (Amy 2010).

Vuonna 2003 Microsoft julkaisi seuraavan version Pocket PC 2000 – käyttöjärjestelmästään ja nimeksi muutettiin Windows Mobile. Se kuvasi paremmin sitä käytettäviä kohdelaitteita, joita olivat esimerkiksi Pocket PC ja älypuhelimet. Käyttöjärjestelmästä julkaistiin neljä eri versioista, joista ”Windows Mobile 2003 for SmartPhone” oli suunnattu älypuhelimille (Amy 2010).

Windows Mobile – käyttöjärjestelmää kehitettiin aina versioon 6.5, joka julkaistiin vuoden 2009 helmikuussa. (Amy 2010) Vaikka Windows mobile – käyttöjärjestelmä oli graafisesti hieno ja ominaisuuksiltaan rikas, jäi se muiden isojen käyttöjärjestelmän kehittäjien, kuten Googlen Androidin, Applen iOS:n ja Nokian Symbianin jalkoihin. Parhaimmillaankin sen osuus markkinoista jäi vuonna 2009 ensimmäisellä neljänneksellä 10.2 prosenttiin, jolloin esimerkiksi Nokian Symbianin osuus oli lähes 50 prosenttia koko maailman matkapuhelinmarkkinoista (Gartner, 2010).

Kehittäjän kannalta tärkeä tieto on, että ohjelmien tekeminen Windows mobile 6.5 -alustalle on edelleen mahdollista. Windows Mobile -ohjelmia voidaan tehdä joko C++, C# tai Visual Basic -ohjelmointikielellä. (Microsoft 2012) Kuitenkin on hyvä muistaa, että käyttöjärjestelmän kehitys on lopetettu ja sille tehdyt ohjelmat eivät ole suoraan siirrettävissä Windows Phone alustalle (Hardy, 2010)

2.2 Windows Phone 7

Ensimmäinen versio Windows Phone 7 – käyttöjärjestelmästä julkaistiin keväällä 2010. Se oli myös ensimmäinen, Microsoftin kehittämä, pelkästään matkapuhelimille tarkoitettu käyttöjärjestelmä. Alun perin sen piti jatkaa Windows Mobile – nimellä, mutta hitaan kehitystyön aiheuttamien viivästysten vuoksi Microsoft päätti rakentaa koko konseptin uusiksi ja aloitti kehitystyön täysin alusta. Sen tuloksena syntyi kosketusnäyttöjä varten suunniteltua metro-käyttöliittymää käyttävä Windows Phone 7 (Kuva 1.) (Brushan, 2013).



Kuva 1. Windows Phone 7-käyttöjärjestelmä (Oram, 2010)

Seuraava versio, Windows Phone 7.5, mikä tunnetaan työnimikkeellä ”Mango”, julkaistiin heinäkuussa 2011. Se toi mukanaan paljon uudistuksia jo sinänsä toimivaan kokonaisuuteen. Sen mukana käyttöjärjestelmä sai muun muassa tuen ohjelmien moniajolle, ääniohjeistetulle navigoinnille, sosiaalisten medioiden, kuten Facebookin, integroinnin puhelimeen People-hub nimisen toiminnon kautta, SkyDrive -pilvipalvelun ja paljon muuta. (Molen, 2011) Sinänsä päivitys ei sisältänyt mitään mullistavaa ja useat siinä esitellyistä ominaisuuksista löytyivät jo muilta alustoilta, mutta sen myötä Windows Phone pystyi jo uskottavasti kilpailemaan muiden alustojen kanssa.

Kehittäjälle Windows Phone 7 toi tärkeitä uudistuksia verrattuna Windows Mobile ohjelmistokehitykseen. Merkittävimpänä muutoksena voidaan pitää ohjelmistojen pääohjelmointikielen muutosta Silverlightiin ja XNA:han. Silverlightista tuli hyöty-

ohjelmien kieli ja XNA:sta peliohjelmoinnin kieli. (Microsoft 2012a) Toisena Lisäksi C++ tuki poistui, mikä käytännössä tarkoitti etteivät vanhat Windows Mobile sovellukset toimisi enää uudessa järjestelmässä (Kumar, 2010).

Mango-päivitys tarjosi kehittäjille uudet ohjelmistorajapinnat sekä mahdollisuuden yhdistää Silverlightia ja XNA:ta keskenään. Mango-päivityksen jälkeen julkaistiin vielä ”Tango” –työnimikkeellä kulkeva päivitys, mikä käytännössä ei näkynyt asiakkaalle mitenkään, mutta toi kehittäjille uusia mahdollisuuksia luoda ohjelmia pienempitehoisille laitteille (Molen 2011).

2.3 Windows Phone 8

Vaikka Windows Phone 7-käyttöjärjestelmä herätti paljon mielenkiintoa, odotukset kohdistettiin jo kohti syksyllä 2012 julkaistavaa käyttöjärjestelmäperheen seuraavaa versiota. Tämä ”Apollo” työnimikkeellä kulkeva versio sai nimen Windows Phone 8 (Thurrott 2012). Uusi versio toi mukanaan lisää merkittäviä ominaisuuksia, joita käyttöjärjestelmältä oli siihen asti puuttunut verrattuna kilpailijoihin, kuten tuen SD-muistikorteille, moniydin-prosessoreille, korkeatarkkuusnäytöille ja NFC-tekniikalle (Near Field Communication). Lisäksi uusina ominaisuuksina julkaistiin Kid’s Corner-toiminto eli lasten rajoitettu puhelimen käyttö, Rooms-ominaisuus, mikä mahdollistaa ryhmien muodostamisen People-Hub-toiminnossa, Wallet-toiminto ostosten tekoa varten, Nokian Here-kartat ja vahvasti integroitu Skype-palvelu (Warren 2012a).

Kehittäjille Windows Phone 8 toi myös paljon uudistuksia. Esimerkkeinä voidaan mainita mahdollisuus jälleen kehittää ohjelmia natiivi-ohjelmointikielillä kuten C++, mahdollisuus tehdä pelejä käyttäen Direct3D-tekniikkaa, Bluetooth-rajapinnan avautuminen kehittäjille ja mahdollisuus tehdä luokkakirjastoja usealla alustalle (Portable Class Library). Lisäksi, kehittäjien helpotukseksi, Microsoftin johtajistoon kuuluva Larry Lieberman kertoi huhtikuussa 2012 pidetyssä lehdistötilaisuudessa, että Windows Phone 7.1 ohjelmat toimivat myös käyttöjärjestelmän seu-

raavassakin versiossa. Tämä tarkoittaa, ettei vanhoja sovelluksia tarvitse kääntää Windows Phone 8:lle erikseen ja kehittämistä voi jatkaa molemmille alustoille (Microsoft 2012; Lieberman 2012).

Tärkeimpänä uudistuksena voidaan kuitenkin pitää Windows Phonen 8 ja Windows 8 vahvaa integraatiota. Microsoftin markkinointisuunnitelma voidaan kiteyttää kolmen näytön taktiikkaan, missä tarkoituksena on tiedon liikkuminen PC-tietokoneen, tabletin ja älypuhelimien välillä mahdollisimman huomaamattomasti ja sulavasti. Esimerkiksi, SkyDrive pilvipalvelun kautta samat tiedostot ovat käytävissä kaikissa laitteissa, joihin käyttäjä on kirjautunut Live-tunnuksen avulla. Lisäksi tarkoituksena on, että näille laitteille tehtävää ohjelmien kehitystyötä tiivistetään, jolloin ohjelman siirtäminen toiselle alustalle tehdään mahdollisimman helpoksi. Käytännössä tämä tarkoittaa sitä, että Windows Phone ohjelmien kehittäjä voi tehdä kehitystyötä kolmelle alustalle samanaikaisesti, mitä voidaan pitää merkittävänä etuna verrattuna esimerkiksi Applen käyttöjärjestelmiin, joissa iOS-mobiilikäyttöjärjestelmä ja MAC-tietokonekäyttöjärjestelmä kehitykset tehdään selvästi erikseen. (Clayton, 2011).

2.4 Ominaisuudet

Julkaistessaan Windows Phonen, Microsoft ei pelkästään tuonut vain uutta käyttöjärjestelmää, vaan myös täysin uuden käyttäjäkokemuksen. Windows Phone on ihmiskeskeinen käyttöjärjestelmä, missä tärkeintä on sosiaalisuus ja käyttäjäkeskeisyys (Järvinen 2012, 50).

Markkinoiden muut käyttöjärjestelmät kuten Googlen Android- ja Applen iOS-käyttöjärjestelmä ovat sovelluskeskeisiä. Kaikki käyttöjärjestelmän ominaisuudet on selkeästi jaoteltu eri sovelluksiin. Hyvänä esimerkkinä voidaan pitää sosiaalisia medioita. Mikäli haluat kommunikoida kavereidesi kanssa Facebookissa, käynnistät Facebook-sovelluksen. Mikäli taas haluat lähettää viestin Twitterin kautta, käynnistät Twitter-sovelluksen (Järvinen 2012, 50, 54).

Windows Phone ottaa täysin uuden lähestymistavan. Käyttöliittymä on sovellusten sijasta jaettu toimintakokonaisuuksiin. Tärkeää on kuitenkin mainita, että

käyttöjärjestelmässä on myös sovelluksia, mutta pääpaino on toimintakokonaisuuksilla. Esimerkiksi kommunikointi ja sosiaaliset mediat on yhdistetty yhdeksi toimintakokonaisuudeksi nimeltään People-hub (Kuva 2.). Idea on se, että ihminen haluaa ensisijaisesti kommunikoida toisen ihmisen tai ihmisryhmän kanssa riippumatta siitä, minkä kautta ja missä muodossa kommunikointi kulkee ihmiseltä toiselle (Järvinen 2012, 50, 54).



Kuva 2. People-hub (Segan, 2010)

Esimerkiksi käyttäjä aloittaa keskustelun toisen ihmisen kanssa People-hub-toiminnossa ja viestit kulkevat Facebook Messenger-ohjelman kautta. Käyttäjälle tulee tilanne, ettei data-yhteys ole käytössä. Tällöin keskustelu automaattisesti vaihtuu tekstiviestien puolelle ilman, että käyttäjän tarvitsee siirtyä tekstiviestisovellukseen. Saadessaan jälleen data-yhteyden päälle, käyttäjä voi jälleen siirtyä keskustelemaan Facebook Messenger-viestien kautta. Tärkeintä on siis, että

käyttäjät voivat saumattomasti keskustella keskenään käyttäen samaa käyttöliittymää vaikka rajapinta niiden välillä vaihtuisikin.

Toinen merkittävä ero muihin markkinoilla oleviin käyttöliittymiin on ulkoasu, joka perustuu Microsoftin kehittämään Metro-tyylikieleen. Tyyli perustuu aloitusruudussa oleviin niin sanottuihin informatiivisiin tiiliin (tiles) eli tapahtumaruutuihin. Tiilet voivat olla aktiivisia ja ohjelmat voivat päivittää niitä tarpeen tullen. Esimerkiksi Viestit-tiili kertoo milloin käyttäjä on vastaanottanut viestin ja kuinka paljon lukemattomia viestejä ohjelma sisältää. Lisäksi voidaan käynnistää sovelluksia ja toimintakokonaisuuksia koskettamalla tiiliä (Järvinen 2012, 54).

Kolmas, ei ehkä käyttäjälle niin näkyvä, ero muihin käyttöjärjestelmiin verrattuna on järjestelmän homogeenisuus. Microsoft on määritellyt, että järjestelmän tulee toimia samalla tavalla kaikissa älypuhelin-laitteissa (Järvinen 2012, 16). Lisäksi Microsoft on kuitenkin määritellyt, että valmistajakohtaiset erottumiset ovat sallittuja kunhan puhelin täyttää tekniset minimivaatimukset. Lisäksi eroavaisuuksia voi olla eri näyttökokojen muodossa, kuten kolmas tiilirivi (Pratap 2013).

Kehittäjälle näistä ominaisuuksista tärkeimmät ovat järjestelmän homogeenisuus ja Metro-tyylikieli. Toimintakokonaisuudet eivät sinällään ole merkittäviä, koska kehittäjän ei ole mahdollista rakentaa sovellusta, joka integroituisi toimintakokonaisuuteen, vaan ne toimivat ainoastaan erillisinä toiminnallisuuksina (Järvinen 2012, 50).

Homogeenisuutta voidaan pitää tärkeimpänä, koska puhelinlaitteiden samankaltaisuus nopeuttaa sovellusten kehitystyötä ja niiden testaamista. Esimerkiksi vaikka näyttöresoluutioita on olemassa useampia, ei tarvitse miettiä, miten kuvan venyminen tai kutistuminen vaikuttaa näyttöelementtien sijoitteluun, koska käyttöjärjestelmä hoitaa sen automaattisesti. Lisäksi kun voidaan olettaa, että kun tietyt sensorit, kuten GPS-sensori, ovat minimivaatimuksia, ei tarvitse erikseen pohtia onko paikkatieto aina saatavilla vai ei (Järvinen 2012, 50-51).

Metro-tyylin käyttö on tärkeää sovellusten integroimiseksi osaksi Windows Phone kokemusta. Microsoft ei pakota kuitenkaan tyylin käyttämistä sovelluksissa ja täten se ei vaikuta sovelluksen julkaisuun. Kuitenkin Metro-tyylin noudattamisesta

on selvä etu: Käyttäjä pystyy omaksumaan sovelluksen käytön helposti (Järvinen 2012, 54).

3 KEHITYSTYÖKALUT

Ohjelmistokehitys Windows Phone-käyttöjärjestelmälle perustuu pääasiassa Silverlight ja XNA tekniikoihin. Jako näiden kahden tekniikan välillä on selkeä, Silverlight-tekniikalla tehdään hyötysovelluksia ja XNA-tekniikalla pelejä. Kuitenkaan mikään ei estä esimerkiksi tekemästä pelejä Silverlight tekniikalla, mutta pääsääntöisesti jako tapahtuu edellä mainitulla tavalla. Windows Phone 7.5 päivityksen myötä tuli näiden tekniikoiden yhdistäminen mahdolliseksi. Ohjelmat yleensä perustuvat puhtaasti jompaan kumpaan tekniikkaan, mutta yksittäisten ohjelmistokomponenttien käyttö toisesta tekniikasta on mahdollista. Itse kehitystyö tehdään pääasiassa käyttäen Microsoftin kehittämiä sovelluskehitystyökaluja Visual Studiota ja Expression Blendiä (Montonen 2011).

3.1 Silverlight-tekniikka

Silverlight-tekniikka pohjautuu vuonna 2006 esiteltyyn Windows Presentation Foundation-tekniikkaan osana .NET 3.0-ohjelmistokehystä. .NET on Oraclen kehittämän Java-ympäristön kaltainen kokonaisuus ja perustuu Javan lailla virtuaalikonetekniikkaan. Virtuaalikonetekniikalla tarkoitetaan sitä, että ohjelmointiin käytetyn kielen, esimerkiksi C#, kääntäjä ei käännä suoraan konekieleksi vaan niin sanotuksi välikoodiksi (intermediate language, IL). Välikoodi puolestaan käännetään sovelluksen suorituksen aikana konekieleksi JIT-käännöksenä (Just In Time) ajonaikaisen CLR-moottorin (Common Language Runtime) avulla (Järvinen 2012, 17, 96).

Silverlight luotiin alun perin Web-sovellusten tekemiseen ja kilpailemaan Adoben kehittämän Flash-teknologian kanssa. Nykyään sillä kuitenkin kehitetään myös normaaleja työpöytä-, tablet- ja puhelinsovelluksia. Silverlight on täysin alustariippumaton ja se voidaan ottaa käyttöön niin Windows PC-koneissa kuin Applen kehittämässä Mac käyttöjärjestelmässä ja avoimen koodin Linux järjestelmissä asentamalla Silverlight-ajoympäristö. Linux ympäristössä Silverlight-ajoympäristö kulkee nimellä Moonlight. Ajonaikainen ympäristön alhaisen muistin käytön

vuoksi, se oli myös luonnollinen valinta Windows Phone – mobiilikäyttöjärjestelmän käyttöliittymäteknikaksi. Windows Phone 7.1 käyttää Silverlight versiota 3.0, mihin on lisätty muutamia Silverlight 4.0 version ominaisuuksia. Windows Phone 8 perustuu Silverlight versioon 5.0 (Järvinen 2012, 96).

3.1.1 XAML

Windows Phone Silverlight-sovellusten pääkomponentti on sivu (Page) ja niitä on jokaisessa sovelluksessa vähintään yksi. Sivun rakentuu puolestaan kahdesta osasta, jotka on jaettu omiksi kokonaisuuksiksi: käyttöliittymä (Markup) ja toimintalogiikka (Code-behind). Käyttöliittymä rakennetaan deklarativista XAML-ohjelmointikieltä (eXtensible Application Markup Language) käyttäen. Deklaratiivisuus tarkoittaa sitä, että kääntäjälle kuvataan, mitä sen tulee tehdä. Tällöin painopiste siirtyy koodista visuaalisuuteen. Ei siis tarvitse välittää kuinka ympäristö luo objektin vaan riittää, että kertoo mihin kohtaan sivua se tulee ja millä tavalla se pitää esittää. Tämä on yleensä haasteellista kehittäjille, jotka ovat tottuneet imperatiivisiin ohjelmointikieliin, kuten C#, missä kerrotaan tarkasti, mitä koneen halutaan tekevän (Järvinen 2012, 96, 101; Montonen 2011).

XAML-ohjelmointikieli on XML-pohjainen (eXtensible Markup Language) ja XAML-sivu koostuu XML-syntaksin mukaisista elementeistä ja niiden attribuuteista eli ominaisuuksista. Elementit ladotaan sivulle päällekkäin ja sisäkkäin. Sisäkkäiset elementit muodostavat äiti-lapsi-pareja (Parent-Child) ja kaikki elementit muodostavat yhdessä visuaaliseksi puuksi (Visual Tree) kutsutun kokonaisuuden, jonka avulla Silverlight tietää, missä järjestyksessä elementit tulee piirtää. Itse elementit kuvataan niiden sisällä määritetyillä attribuuteilla. Attribuuteilla voidaan kuvata muun muassa kokoa, paikkaa, näkyvyyttä ja sisältöä. (Järvinen 2012, 96, 97).

XAML-sivujen teko onnistuu sekä Microsoftin Visual Studiolla tai Expression Blendillä ja sitä voidaan työstää kahdella eri tavalla. Koodia voidaan perinteiseen tapaan kirjoittaa käsin tai komponentteja voidaan visuaalisesti vetää käyttöliittymä-

mään suoraan ohjelmien työkalupaletista. Tällöin ohjelmat kirjoittavat XAML-koodin kehittäjän puolesta. Kehittäjä voi luoda vaikka koko käyttöliittymän kirjoittamatta riviäkään koodia. Tämä on merkittävä etu, koska esimerkiksi käyttöliittymäsuunnittelija voi tällöin itsenäisesti rakentaa käyttöliittymän ulkoasun käyttäen tähän tarkoitukseen kehitettyä Expression Blendiä. Myöhemmin ohjelmoija voi lisätä toimintalogiikan suunnittelijan toiveiden mukaisesti käyttäen Visual Studiota. Tämä helpottaa ja nopeuttaa molempien työtä, kun voidaan työskennellä rinnakkain toisistaan riippumatta. Lopuksi käyttöliittymä-sivu tallennetaan xaml päätteiseksi tiedostoksi (Microsoft 2012c).

3.1.2 C#

Silverlight-sovellusten toimintalogiikka, Windows Phone ympäristössä, ohjelmoidaan pääasiassa kahdella eri kielellä: C#-ohjelmointikielellä tai Visual Basic-ohjelmointikielellä (VB). Alun perin Windows Phone 7.0 versiossa ainut tuettu kieli oli C#, mutta Visual Basic lisättiin mukaan Windows Phone 7.5-päivityksen yhteydessä. Koska työosuudessa toimintalogiikka on toteutettu käyttäen C#-ohjelmointikieltä, tullaan työssä keskittymään ainoastaan sen esittelemiseen. Todettakoon kuitenkin, että kaikki asiat mitkä voidaan tehdä C#-kielellä, on mahdollista toteuttaa käyttäen Visual Basic-kieltä. Lisäksi apuna voidaan käyttää C++ kieltä. (Järvinen 2012, 17)

C#-kieli on Microsoftin kehittämä oliopohjainen kieli joka on suunniteltu yhdistämällä muun muassa C-kielen, C++-kielen ja Javan parhaita puolia. Se kehitettiin alun perin .NET-ohjelmistokehityksen tarpeisiin. C#-kielessä haluttiin yhdistää C++-kielen tehokkuus ja Javan helppokäyttöisyys. C# on, kuten edellä mainittu, imperatiivinen ohjelmointikieli, mihin on yhdistetty ominaisuuksia muun muassa funktionaalisista ohjelmointikielistä. Funktionaalinen ohjelmointi on ohjelmointiparadigma, joka perustuu matemaattisten funktioiden suorittamiseen. C#-kielen pääkomponentit ovat nimiavaruudet (Namespace), luokat (Class), tietueet (Struct), funktiot (Method) ja muuttujat (Variable). (Microsoft 2013b)

C#-kielellä kehitetään pääasiallisesti käyttäen Visual Studiota. Visual Studio on Microsoftin kehittämä maksullinen ohjelmistokehitysympäristö. Tätä C#-kielen implementaatiota kutsutaan nimellä Visual C#, jossa visual osa viittaa Visual Studio-työkaluun (Microsoft 2013c). Muita implementaatiota ovat muun muassa Mono ja DotGNU, jotka puolestaan ovat ilmaisia avoimen koodin työkaluja käytettäväksi esimerkiksi Linux ympäristössä (Bollow 2013; Xamarin 2013).

Toimintalogiikka-osuudesta luodaan, käyttöliittymä-osuuden tavoin, oma C#-sivu, mikä on rakenteellisesti yhdistetty käyttöliittymään. Tällöin voidaan käyttöliittymän tapahtumat yhdistää toimintalogiikan puolella luotuihin metodeihin eli funktioihin. Esimerkiksi käyttöliittymään voidaan tuoda nappi, jonka painamis-tapahtumaan (Click) voidaan yhdistää "Hello world!"-merkkijonon tulostaminen näytölle. Lisäksi toimintalogiikka yhdistää käyttöliittymän kaikkiin puhelimen ja sovelluksen rajapintoihin kuten tietokantaan ja sensoreihin. Lopuksi, kun ollaan tyytyväisiä, voidaan toimintalogiikka sivu tallentaa xaml.cs päätteiseksi tiedostoksi (Microsoft 2013d).

3.2 XNA-tekniikka

XNA-tekniikalla tarkoitetaan Microsoftin kehittämää .NET-pohjaista pelikehikkoa ja -moottoria. Ensimmäinen versio XNA-tekniikkaa käyttävistä työkaluista julkaistiin maaliskuussa 2004. Alun perin XNA-tekniikka kehitettiin Windows-pelien tekemiseen, mutta myöhemmin tukea laajennettiin koskemaan muitakin Microsoftin kehittämää alustoja kuten XBOX-konsolikäyttöjärjestelmää ja Windows Phone-mobiilikäyttöjärjestelmää. Syy on ollut ilmeinen, koska käytettävissä olevat ohjelmointikirjastot ovat lähellä toisiaan, voidaan pelikehitystä tehdä periaatteessa kaikille kolmelle alustalle samanaikaisesti. Lisäksi kyseinen yhteensopivuus avaa uusia mahdollisuuksia kehittäjille. Pelejä voidaan kehittää hyödyntämään useaa alustaa samanaikaisesti, jolloin pelikokemusta voidaan monipuolistaa. Esimerkiksi Windows-alustalle voidaan kehittää peli, jossa ohjaimena käytetään Windows Phone-alustaa käyttävää puhelinta. (Järvinen 2012, 188)

XNA-tekniikkaa käyttävät pelit ovat pohjimmiltaan .NET pohjaisia sovelluksia ja täten yhteensopivia Silverlight-tekniikan kanssa. Lisäksi, kuten edelläkin mainittiin, Silverlight-tekniikalla on myös mahdollista tehdä pelejä. Kuitenkin, toisin kuin Silverlight-sovellukset, XNA-sovellukset toimivat lähellä laitteistoa ja voivat käyttää laitteistokiihdytystä hyväkseen. Tämä tarkoittaa sitä, että XNA-tekniikka soveltuu huomattavasti paremmin monimutkaisten ja paljon tehoa vaativien pelien tekemiseen. Yhteensopivuus kuitenkin mahdollistaa niin sanottujen hybridi-sovellusten tekemisen, missä hyödynnetään molempien tekniikoiden parhaita puolia. Esimerkiksi kaikki valikot voidaan toteuttaa käyttäen Silverlight-tekniikkaa ja taas itse peli voidaan tehdä käyttäen XNA-tekniikkaa. (Järvinen 2012, 188)

XNA 4.0 julkaistiin syyskuussa 2010, minkä mukana julkaistiin myös työkalut Windows Phone-alustalle. Versio jäänee myös viimeiseksi, koska tammikuussa 2013 vuotaneessa Microsoftin sisäisestä sähköpostista ilmenee, että tuki XNA-tekniikalle ollaan vähitellen lopettamassa. Muun muassa Microsoftin uusimmat käyttöjärjestelmät Windows 8, Windows RT ja Windows Phone 8 eivät enää tue suoraan XNA-tekniikkaa. (Crossley 2012; Microsoft 2012d)

3.3 Kehitysvälineet

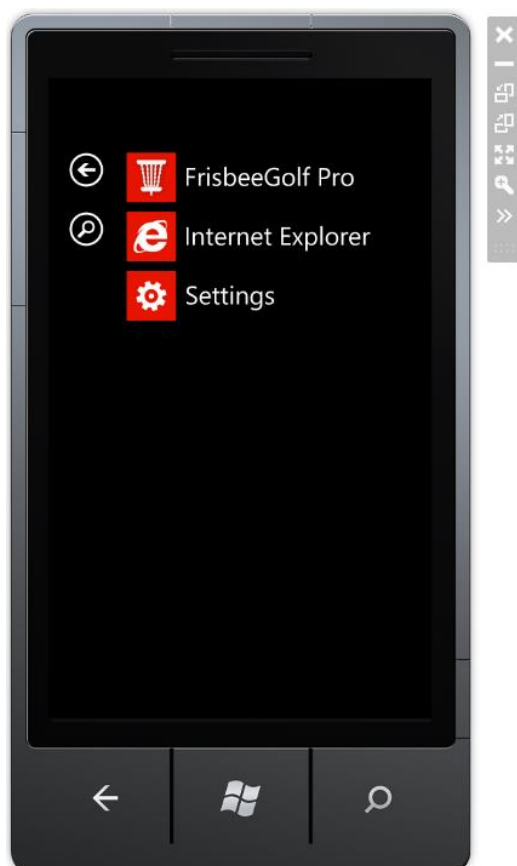
Microsoftin Windows Phone-käyttöjärjestelmälle tehtävä kehitystyö voidaan tehdä käyttäen Microsoftin omia työkaluja, Visual studiota ja Expression Blendiä. Käytännön ohjelmointi tehdään käyttäen Visual studiota ja graafinen toteutus puolestaan käyttäen Expression Blendiä. Käytännön ohjelmoinnin voi toki toteuttaa käyttäen mitä tahansa kehitystyökaluja, mutta edellä mainitut työkalut soveltuvat tähän parhaiten. Lisäksi graafinen toteutus on mahdollista tehdä myös käyttäen Visual Studiota, mutta Expression Blend kehitettiin tätä tarkoitusta varten (Järvinen 2012, 68).

3.3.1 Visual Studio 2010

Visual Studio on Microsoftin kehittämä ohjelmistokehitysympäristö, jossa voi tehdä kehitystyötä useilla eri ohjelmointikieliä kuten C# ja Visual-Basic. Sillä voi kehittää ohjelmia Windows, Windows-Phone- ja XBOX-alustalle ja tehdä vaikka internet-sivuja. Visual Studio esiteltiin ensimmäistä kertaa jo vuonna 1997 nimellä Visual Studio 97. Tässä ja seuraavassa versiossa, Visual Studio 6.0, käyttö painottui pääasiassa natiivi-ohjelmointikielten, kuten C++, kirjoittamiseen. Vuonna 2002 esitellyssä versiossa, Visual Studio .NET, tapahtui merkittävä muutos. Ohjelmistoaalusta sai tuen hallitun koodin (Managed Code) kirjoittamiselle. Tämä mahdollisti koodin kirjoittamisen useilla eri kielillä mistä ne tulkataan yhteiselle IL-kielelle, jonka CLR-ympäristö tulkaa ajon aikana koneen ymmärtämälle kielelle (Mariani 2009a; Mariani 2009b). Vuonna 2010 julkaistussa versiossa, Visual Studio 2010, mukana tuli tuki sovelluskehitykselle Windows Phone-alustalle, mikä perustuu edellä mainittuun CLR-ympäristöön (Microsoft 2013e).

Visual studio 2010 on monipuolinen kokonaisuus ja se sisältää kaiken tarvittavan ohjelmien tekoon suunnittelusta toteutukseen ja Windows Phone-sovellusten tapauksessa julkaisuun asti. Tärkeimpinä Windows Phone-sovellusten tekoon tarkoitettuja ominaisuuksina voidaan mainita itse koodieditori (Code Editor), virheidenjäljittäjä (Debugger), käyttöliittymän suunnittelutyökalu (Designer) ja Windows Phone-emulaattori (Kuva 3.). Koodieditorin puolella kirjoitetaan kaikki ohjelmassa käytettävä koodi. Virheidenjäljittäjää käytetään esimerkiksi asettamalla koodiin pysäytyspisteitä (breakpoint). Tällöin koodin suoritus pysähtyy, kun asetettu piste saavutetaan suorituksen aikana. Näin esimerkiksi seurataan, asetetaanko oikeita arvoja oikeisiin muuttujiin, mikäli koodi ei toimi halutulla tavalla. Lisäksi virheenjäljittäjä pysäyttää koodin paikkoihin, missä se havaitsee virheellisen toiminnon. Käyttöliittymän suunnittelutyökalua käytetään graafisen käyttöliittymän (Graphical User Interface, GUI) suunnitteluun ja toteutukseen. Työkalun avulla muun muassa asetetaan haluttuja graafisia objekteja hiiren avulla sille osoitettuun paikkaan ja Visual Studio kirjoittaa koodit automaattisesti. Windows Phone-emulaattori on virtuaalikone, jossa sovellukset ajetaan ja testataan ennen

julkaisua. Sovelluksia voidaan ajaa myös suoraan puhelimessa, mikäli se on rekisteröity kehittäjäpuhelimeksi (Microsoft 2012e; Järvinen 2012, 78-81).



Kuva 3. Windows Phone emulaattori

Muita hyödyllisiä Visual Studion 2010 ominaisuuksia Windows Phone-sovelluskehitykseen ovat muun muassa luokkien suunnittelutyökalu (Class Designer), tietokantojen suunnittelutyökalu (Database Schema Designer), suoritusanalysointityökalu (Performance Analysis) ja ryhmäkatselmointityökalu (Team Explorer). Lisäksi Visual Studioon on mahdollista asentaa laajennuspaketteja, makroja ja tukea uusille kielille. Makroilla tarkoitetaan käskypinoja, joilla ohjataan Visual Studiota tekemään tietynlaisia toimintoja automaattisesti. (Microsoft 2012f)

Microsoft tarjoaa Visual Studio 2010 ohjelmistosta viisi eri versiota: Express, Professional, Test-Professional, Premium ja Ultimate. Näistä ainoa ilmaiseksi jaossa

oleva versio on Express, joka sisältää vain välttämättömimmät työkalut sovellusten tekemiseen. Se ei itsessään rajoita mitä ja millaisia sovelluksia voi tehdä, mutta sillä tekeminen on luonnollisesti hitaampaa kuin enemmän ominaisuuksia sisältävät versiot. Lisäksi Expressissä voi kehittää sovelluksia ainoastaan yhdelle alustalle kerrallaan. Tämä tarkoittaa, jokaiselle alustalle on olemassa oma Express versionsa esimerkiksi tässä tapauksessa Windows Phone Express. Se tulee Windows Phone SDK-kehityspaketin (Software Development Kit) mukana, joka sisältää myös tarvittavat kirjastot ja emulaattorin Windows Phone-sovellusten kehittämiseksi. (Microsoft 2013; Järvinen 2012, 69)

Maksulliset versiot, Visual Studio Professional, Premium ja Ultimate, sisältävät mahdollisuuden kehittää mille tahansa alustalle eli kaikki tarvittava löytyy yhden sovelluksen alta. Lisäksi niillä on paljon ylimääräisiä ominaisuuksia muun muassa testaussovelluksen muistikäytön tarkkailuun. Näiden maksullisten versioiden erot ovat hyvin pienet. Ultimate on kallein ja ominaisuuksiltaan rikkain versio näistä kolmesta. Halvemmissa versioista puuttuu joitakin koodaustyötä helpottavia ominaisuuksia. Selkeästi muista tuotteista eroaa ainoastaan Test-Professional. Se on tarkoitettu ainoastaan, nimensä mukaisesti, ohjelmien testaukseen ja näin ollen siitä puuttuvat kaikki ohjelmointiin tarvittavat työkalut (Microsoft 2013a).

3.3.2 Expression Blend

Vaikka Visual Studio onkin sellaisenaan hyvä kokonaisuus ja sen avulla voidaan toteuttaa koko sovellus alusta loppuun, on sen graafiseen suunnitteluun ja toteutukseen tarkoitettu työkalu hieman puutteellinen. Tähän avuksi soveltuu toinen Microsoftin kehittämä mainio työkalu Expression Blend. Sovelluksen tarkoituksena on antaa graafikoille työkalu, jolla voidaan suunnitella ja toteuttaa graafinen käyttöliittymä ilman, että koodiin tarvitsee edes koskea. Tarkoituksena on, että graafikot ja ohjelmoijat voivat työskennellä ohjelman parissa toisistaan riippumatta. Graafikko toteuttaa käyttöliittymän ja ohjelmoija lisää siihen tarvittavat toiminnallisuudet. Expression Blendissä voidaan toki myös toteuttaa sovellusten toiminnallisuuksia, mutta hyvin rajallisesti. Huomionarvoista on se, että Expression

Blend kirjoittaa kaikki tehtävät muutokset pelkästään XAML-kielellä käyttöliittymä puolelle, joten sillä ei voida vaikuttaa toimintalogiikka puolella tehtävään koodiin. Ensimmäinen kokeiluversio Expression Blendistä julkaistiin yleisölle tammi-kuussa 2007 osana Expression tuoteperhettä ja uusia versioita on tullut tasaiseen tahtiin. Ainoa merkittävä muutos koettiin 2012 kun Expression tuoteperheen kehittäminen lopetettiin ja Expression Blend sulautettiin osaksi Visual studiota. (Järvinen 2012, 82-87)

Expression Blend käyttöliittymässä on kaksi työtilaa: Design ja Animation. Design työtila on tarkoitettu käyttöliittymän suunnitteluun. Se muistuttaa olemukseltaan Visual Studion käyttöliittymän suunnitteluun tarkoitettua tilaa, mutta on paljon monipuolisempi. Esimerkiksi kun työtilassa käytetään piirtotyökalua, voidaan käyttöliittymään lisätä monikulmioita ja soikioita. Piirtämisen jälkeen Expression Blend kirjoittaa tarvittavan koodin automaattisesti XAML-ohjelmointikielellä. Piirtäminen on mahdollista myös Visual Studiassa, mutta kaikki piirrokset on toteutettava kirjoittamalla itse sitä vastaava koodi, mikä on hankalaa ja työlästä. Lisäksi Expression Blendin vahvuuksiin kuuluu myös, että käyttöliittymäkomponenttien toiminnallisuuksia voidaan tehdä valmiiksi, koskematta itse koodiin, käyttäytymis- (behaviors) ja liipaisin-toimintojen (trigger) kautta. Kuitenkin toimintoja on vähän ja monimutkaisemmat toiminnallisuudet jäävät koodaajien toteutettaviksi (Järvinen 2012, 82-87).

4 MARKKINOINTI

Kun Windows Phone sovellus on valmis, on mietittävä, kuinka sitä jaetaan kuluttajille. Halutaanko sovelluksella ansaita rahaa vai jakaa ilmaiseksi? Oli vastaus kumpi hyvänsä, voidaan sovelluksia jakaa yksityisille ainoastaan keskitetyn markkinapaikan kautta. Sovelluksia ei siis voida esimerkiksi ladata ja asentaa suoraan omalta internet-sivulta. Näin sovellukset pysyvät laadukkaina ja tietoturva puhelimissa paranee, koska haittaohjelmat eivät pääse tarkkailun läpi. Lisäksi, koska kolmas osapuoli hoitaa myymisen, rahastamisen ja rahojen tilittämisen kehittäjälle, voi sovelluskehittäjä keskittyä ainoastaan tuotteen parantamiseen (Järvinen 2012, 210-211)

Mikäli kyseessä on yrityksen sisäiseen käyttöön tarkoitettu sovellus, jakaminen onnistuu myös julkisen markkinapaikan ulkopuolelta. Mahdollisuuksia on kaksi. Sovelluksia voidaan jakaa Windows Intune-pilvipalvelun kautta tai yrityksen luoman oman Company Hub sovelluksen kautta. Windows Intune on pilvipohjainen päätelaitehallinta-palvelu, jonka kautta voidaan hallita yrityksen laitekantaa, laitteiden käyttöjärjestelmäpäivityksiä ja ohjelmistojen asennuksia (Microsoft 2013r). Company Hub on yrityksen itse luoma sovellus, jonka kautta voidaan jakaa ohjelmistoja ja sisäisiä tiedotteita. Koska opinnäytetyön osana tehtyä sovellusta halutaan myydä sovelluskaupassa julkisesti kuluttajille, en tule käsittelemään yrityskohtaisten sovellusten jakamista tämän enempää (Microsoft 2013q).

4.1 Windows Phone Store

Windows Phone sovellusten markkinapaikkoja on ainoastaan yksi, Microsoftin hallinnoima Windows Phone Store. Se julkaistiin Windows Phone 7 käyttöjärjestelmän yhteydessä lokakuussa 2010 nimellä Windows Phone Marketplace ja korvasi samalla Windows Mobile sovelluksia jakaneen Windows Marketplace for Mobile sovelluskaupan. Nykyisen nimensä, Windows Phone Store, se sai elokuussa 2012 (Warren 2010; Warren 2012b).

Julkaisunsa jälkeen sovelluskaupan kasvu on ollut nopeaa. Jo kesäkuussa 2012 Windows Phone Marketplace saavutti merkittävän rajapyykin kun myytävien sovellusten määrä ylitti 100,000 rajan vain 20 kuukautta sovelluskaupan julkistamisen jälkeen. Esimerkiksi Googlen Play-sovelluskauppa Androidille saavutti saman rajapyykin 24 kuukaudessa. Sovellusten määrän kasvua nostatti erityisesti Nokian Lumia-perheen julkaisu loppuvuodesta 2011. Ripeästä kasvutahdistaan huolimatta sovelluskaupan tarjonta kalpenee kilpailijoiden sovellusmääriin verrattaessa. Lokakuussa 2013 Windows Phone Storessa sovelluksia oli 175,000 kun taas esimerkiksi Googlen Play-sovelluskaupassa niitä oli jo yli miljoona (Brix 2013 & Graziano 2012)

Sovellusten jako tapahtuu Windows Phone-käyttöjärjestelmän mukana tulevan integroidun Store sovelluksen kautta. Sovelluksesta käyttäjä voi vapaasti selata sovelluksia ja pelejä. Pelit ja sovellukset on jaettu omiin luokkiinsa ja ryhmitelty kategorioihin tuotekuvauksen mukaisesti. Lisäksi käyttäjä voi hakutoiminnon kautta hakea sovelluksia nimellä. Kauppapaikan sisältö on myös saatavilla suomeksi osoitteessa www.windowsphone/fi-fi/store (Microsoft 2013i).

Kun päätös sovelluksen hankkimisesta on tehty, on navigoitava ensin ostettavan tuotteen sivulle. Mikäli sovellus on ilmainen, löytyy vasemmasta alanurkasta asenna-nappi, jonka painaminen käynnistää automaattisen lataus- ja asennustoiminnon. Asentamisen jälkeen sovellus on vapaasti käytettävissä. Mikäli kyseessä on maksullinen sovellus, löytyy vasemmasta alanurkasta osta-nappula. Osta-nappulan painaminen vie käyttäjän ostoksen hyväksymissivulle, josta valitaan ostotapa. Sovelluskaupassa ostokset voi tehdä käyttäen joko luottokorttia tai operaattorilaskutusta maasta riippuen. Kun ostos on hyväksytty, sovellus ladataan ja asennetaan samoin kuin ilmainen sovellus. Lisäksi joihinkin maksullisiin sovelluksiin on olemassa kokeilu-mahdollisuus jolloin sovellussivulta löytyy lisäksi kokeile-nappi. Se mahdollistaa sovelluksen kokeilun rajoitettuna versiona jolloin joko toiminnallisuuksia tai käyttöaikaa on rajoitettu (Microsoft 2013i).

4.2 Ansaintamallit

Windows Phone sovellus on mahdollista kaupallistaa useilla eri tavoilla. Yleisin on sovelluksen myyminen Windows Phone Storessa. Kehittäjä asettaa haluamansa hinnan, laittaa sovelluksen kauppapaikalle myyntiin ja odottaa, että rahaa tulee tilille. Sovelluksen myyminen ei kuitenkaan ole ilmaista ja sovelluskehittäjä maksaa sovelluksen pitämisestä kaupassa (Järvinen 2012, 215).

Rahaa kerätään kehittäjätilien vuosimaksuina ja sovelluksista saatavina provisio-maksuina. Vuosimaksu on 99 Yhdysvaltain dollaria eli noin 75 euroa per tili. Vuosimaksu on maksettava, vaikka sovelluksia jaettaisiin ilmaiseksi. Opiskelijat ovat toistaiseksi vapautettuja vuosimaksuista. Provisio-maksujen suhteen Microsoft noudattaa mobiilimaailmassa yleistä 70/30-periaatetta eli kehittäjä saa 70 prosenttia jokaisen myydyin sovelluksen hinnasta ja Microsoft 30 prosenttia. Käytännössä tämä tarkoittaisi siis yhden euron hintaisesta sovelluksesta jää kehittäjälle käteen 0,70 euroa ja arvonlisäveron (24 prosenttia) jälkeen 0,53 euroa Suomessa. Kehittäjän osuus on myös noustava yli Microsoftin määräämän minimimaksumäärän yli (200 yhdysvaltain dollaria). Lisäksi on hyvä muistaa, että kehittäjä on itse velvollinen selvittämään kotimaassaan maksettavat verot saamistaan tuloista. Suomessa, yksityisen kehittäjän tapauksessa, kannattaa saadut tulot ilmoittaa verotuspäätöksen yhteydessä saadulla muutoksenhakulomakkeella, jossa ne merkitään ulkomailta saaduksi tuloksi (Microsoft 2012b).

Toinen mahdollisuus ansaita rahaa ovat mainokset. Mainosrahoitteisuus mobiilisovelluksissa tarkoittaa sitä, että itse sovellus on yleensä ilmainen käyttäjille ja tuloa saadaan näyttämällä mainoksia. Kehittäjä saa rahaa kun mainos näytetään käyttäjälle ja kun sitä klikataan. Mainokset saadaan lisättyä ohjelmallisesti sovellukseen ja niiden sijoittelua kannattaa jo suunnitteluvaiheessa miettiä tulojen optimoimiseksi. Mainokset näytetään AdControl-käyttöliittymäkontrollin avulla. Lisäksi Microsoftin kanssa on tehtävä erillinen mainostussopimus. Sopimus tehdään rekisteröimällä sovellus Microsoftin Advertising PubCenter palveluun. Palveluun rekisteröityminen ja sen käyttäminen on ilmaista. Kun sovellus on rekiste-

röity ja mainokset on luotu palvelun kautta, alkavat mainokset näkyä sovelluksessa. Maksupolitiikka noudattaa maksullisten sovellusten tyyliä eli kehittäjä saa mainostuloista 70 prosenttia, mutta toisin kuin maksullisissa sovelluksissa, minimimaksumäärä on 50 euroa (Microsoft 2013k; Microsoft 2013l).

Kolmas tapa saada tuloa on myydä sovellukseen sisältöä sisäisten ostosten kautta. Sisältö voi olla esimerkiksi pelien tapauksessa uusi taso tai sovelluksen yhdistäminen pilvipalveluun, jonka kautta voidaan jakaa sovelluksesta saatavaa sisältöä muiden ihmisten kanssa. Yleinen markkinointistrategia on, että sovellus on ilmainen, mutta sisällöstä on maksettava erikseen. Tällöin käyttäjä maksaa tasan siitä sisällöstä mihin itse haluaa päästä käsiksi. Sisäisiä ostoksia tehdään suoraan itse sovelluksesta, johon sisältö tulee. Kehittäjä lisää sovellukseen linkin tuotteiden ostosivulle, josta ostoksia tehdään kuten sovelluskaupasta. Sisäisten ostosten tapauksessa myyntivoiton jako tapahtuu samoin kuin muissakin ansaintatavoissa, 70/30 periaatteella. Minimimaksumäärä on sama kuin sovelluksen myynnin tapauksessa eli 200 dollaria. (Microsoft 2012b; Microsoft 2013j)

4.3 Sovellusten saattaminen myyntiin

Jotta sovellus saadaan myyntiin Windows Phone Storeen, on se julkaistava Windows Phone Dev Center-palvelun kautta. Palvelu vaatii rekisteröitymisen ja 99 dollarin vuosimaksun. Rekisteröitymiseen vaaditaan Microsoft Live-tili, jonka voi hankkia ilmaiseksi esimerkiksi osoitteesta www.live.com. Saadakseen sovelluksen myynnistä tai sisäisistä ostoista tulleita myyntivoittoja, on yksityisen, ei Yhdysvaltain verovelvollisen, lisäksi lähetettävä Microsoftille Yhdysvaltojen veroviraston (IRS, Internal Revenue Service), W-8BEN-lomake, jolla todistetaan, että sovelluksen kehittäjä maksaa veronsa muualle kuin Yhdysvaltoihin. Mikäli haluaa rekisteröityä yrityksenä, on lisäksi todennettava yrityksen olemassa olo. Tämä tapahtuu Symatec ID vahvistusprosessilla (Microsoft 2013m & Microsoft 2013n).

Kun rekisteröityminen palveluun on tehty ja tarpeelliset tiedot täytetty, voidaan aloittaa itse sovelluksen saattaminen myyntiin. Uuden sovelluksen julkaisuprosessi alkaa täyttämällä uuden sovelluksen tiedot ja lähettämällä sovelluksen Xap-

tiedosto palvelun kautta Microsoftille (Kuva 4.). Xap-tiedosto sisältää sovelluksen asentamiseen tarvittavat tiedostot. Sovelluksen tiedot täydennetään kohdassa App info. Tiedoissa tulee vähintään käydä ilmi sovelluksen nimi, kategoria, hinta ja kohdemarkkinat. Kohdemarkkinoiden valinnassa kannattaa olla varovainen sillä joillakin mailla on erityisvaatimuksia sen kauppapaikassa myytyjen sovellusten suhteen. Esimerkiksi Kiina vaatii, että sovelluksen tulee estää pääsy sisältöihin, jotka ovat Kiinan lainsäädännön mukaan kiellettyjä. Kun sovelluksen tiedot ovat valmiina, siirrytään kohtaan Upload and describe your XAP(s). Tässä kohtaa prosessia palveluun ladataan sovelluksen Xap-tiedosto ja tarvittavat kuvatiedostot kuten sovelluskaupan ikonit ja täydennetään muita sovellukseen liittyviä tietoja kuten esimerkiksi sovelluksen versionumero. Tämän jälkeen sovellus on valmis lähetettäväksi Microsoftin testikeskukseen (Järvinen 2011, 218–221).

 Windows Phone | Dev Center

Design Develop Publish Community **Dashboard**

Submit app

You've spent hours developing and designing your app, and now it's time for the rest of the world to experience your masterpiece. In just two steps we'll gather the information we need to successfully launch your app in the Windows Phone Store. [Learn more](#) about the steps for successfully submitting your app.

Required

1

App info

Give your app a Dev Center alias, price it, and enter other relevant info

2

Upload and describe your XAP(s)

For each XAP in your app, this is where you'll enter descriptions and upload screenshots that will showcase your app in the Store.

Kuva 4. Sovelluksen saattaminen myyntiin

Kun Microsoft on vastaanottanut sovelluksen tarkastettavaksi, alkaa kolmivaiheinen prosessi, joista ensimmäisenä on sovelluksen-paketin digitaalinen allekirjoitus. Vaihe on automaattinen ja se toteutetaan Microsoftin luomalla digitaalisella avaimella. Avain luodaan käyttäen Authenticode-tekniikkaa ja se on kehittäjäkohtainen. Tässä kohdassa tarkistetaan lisäksi, että sovelluksen Xap-tiedosto on ehjä ja täten ajettavissa Windows Phone laitteella (Järvinen 2011, 221)

Mikäli sovellus läpäisee ensimmäisen vaiheen, on vuorossa sovelluksen sertifiointi eli soveltuvuustestaus. Se on prosessi, jossa tarkistetaan, että sovellus täyttää Microsoftin asettamat tekniset ja sisällölliset vaatimukset. Teknisillä vaatimuksilla tarkoitetaan esimerkiksi sitä, että sovellus ei saa varata käyttöönensä yli 90MB (megatavua) Ram-muistia (Random Access Memory) mikäli puhelimella on käytettävissä vain 256MB muistia. Sisällöllisillä vaatimuksilla puolestaan tarkoitetaan esimerkiksi sitä, ettei sovelluksessa saa esiintyä väkivaltaista sisältöä. Sertifiointivaatimukset ovat löydettävissä Windows Phone Dev Center palvelusta hakusanalla App certification requirements for Windows Phone (Microsoft 2013o).

Mikäli tässä vaiheessa prosessia tulee ongelmia ja sovellus ei täytä sertifiointivaatimuksia, lähettää Microsoft sähköpostilla raportin tuloksista ja tiedot mahdollisista muutoksista joilla sertifiointi prosessi saadaan saatettua loppuun. Kun tarvittavat muutokset on tehty, alkaa sovelluksen julkaisuprosessi käytännössä alusta lukuun ottamatta jo täytettyjä sovellustietoja, mikäli ongelma ei koskenut juuri niitä (Järvinen 2011, 221).

Viimeinen vaihe on itse sovelluksen julkaisu. Sovellus julkaistaan automaattisesti sovelluskauppaan, mikäli sovelluskehittäjä ei ole erikseen valinnut manuaalista julkaisuvaihtoehtoa jolloin sovellus voidaan julkaista haluttuna ajankohtana onnistuneen sertifiointiprosessin jälkeen. Koko julkaisuprosessi, Microsoftin mukaan, kestää noin 5-7 arkipäivää, mutta se voi kestää kaksikin viikkoa joten sovelluksen julkaisuun kannattaa varata runsaasti aikaa. Kun sovellus on saatettu myyntiin, voi kehittäjä jäädä odottelemaan sovellusmyynnistä saatavia tuloja (Microsoft 2013p).

5 SOVELLUSPROJEKTI

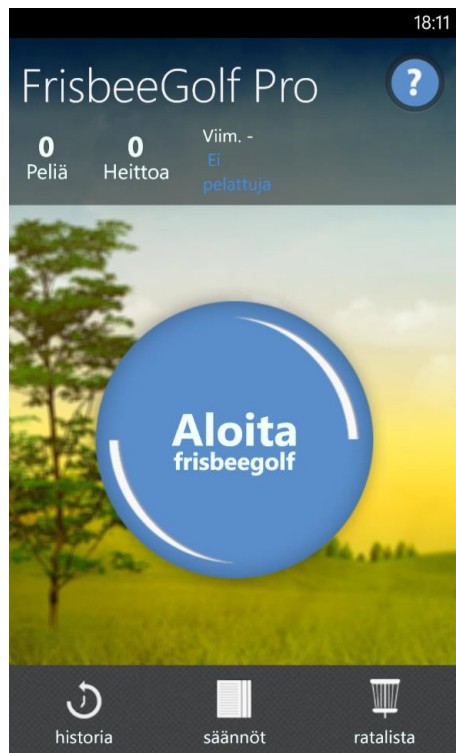
Tämän opinnäytetyön pohjaksi kehitelty sovellus on nimeltään FrisbeeGolf Pro for Windows Phone. Alun perin se tehtiin Windows Phone alkeiskurssilla kurssityönä 3 muun ohjelmistokehittäjän kanssa. Työn tarkoitus oli ensisijaisesti tutustua mobiilisovelluskehitykseen ja Windows Phone alustaan.

5.1 Ongelma

Sovelluksen teko lähti liikkeelle yksikertaisesta ajatuksesta luoda pistekorttisovellus frisbeegolf pelaajille. Frisbeegolf, eli liitokiekkogolf on peruseriaaltaan samanlainen kuin perinteinen golf. Pelivälineenä, golffista poiketen, toimii liitokiekko, joka yritetään saada koriin mahdollisimman vähin heitoin. Peli aloitetaan golfin tapaan tiiltä ja päättyy, kun kiekko on saatettu koriin asti. Pelaaminen tapahtuu ulkoilmaradalla, joka koostuu useista eri väylistä.

Pelaamisen hankalin puoli on tulosten muistaminen ja seuraaminen kauden aikana. Perinteisesti pisteiden keruu hoidetaan kirjoittamalla ne ylös paperille, josta muut pelaajat voivat tarkastella omia ja toisten tuloksia. Ongelmallista on kuitenkin, että täyttäminen on hankalaa, tulosten tarkastelu kymmenistä ellei sadoista papereista on työlästä ja tulosten jakaminen muille pelaajille ei onnistu kuin kopioiden tulokset käsin. Koska nykyään Suomessa lähes jokaisella on älypuhelin, päätimme, että se soveltuisi parhaiten pisteiden merkkailuun. Lisäksi tuloksia on helppo jakaa suoraan verkon yli. Myöhempi tulostarkastelukin onnistuu helposti, kun tuloksia voidaan tuoda samaan näkymään ja niin monta kuin halutaan.

5.2 Ratkaisu



Kuva 5. Frisbeegolf Pro

FrisbeeGolf Pro (Kuva 5.) on tulostenseuranta sovellus Windows Phone alustalle. Sovellus sisältää mahdollisuuden luoda pelaajia ja ratoja rajattomasti. Peli aloitetaan valitsemalla ensin pelaajat ja rata, jossa ollaan heittelemässä. Kun peli on pelattu loppuun, tallennetaan tulokset puhelimen tietokantaan. Myöhemmin niitä voi tarkastella historiasivun kautta joko yksittäin tai kymmenen parhaan tuloksen ryhmissä radoittain.

Pelkkä pisteenlaskusovellus olisi ollut kuitenkin liian yksinkertainen, joten sovellukseen päätettiin lisätä lisäksi tiedot kaikista Suomen frisbeegolfradoista. Myöhemmässä tarkastelussa kuitenkin huomasimme, että pelaajien on välillä erittäin hankalaa löytää uusia radoille vaikka osoite olisikin tiedossa. Tästä syystä sovellukseen tuotiin vielä karttasivu, josta käyttäjä näkee ratojen sijainnit ja pystyy navigoimaan radoilla mistä päin maailmaa tahansa.

5.3 Karttasivu

Karttasivulle käyttäjä pääsee navigoimalla ensimmäiseksi ratalistasivulle, jossa valitaan haluttu rata. Valinta käynnistää navigoinnin karttasivulle, jossa käyttäjälle näytetään kyseisen radan sijainti. Karttapohja, jossa radan sijainti näkyy, on toteutettu käyttäen Microsoftin Bing Maps karttarajapintaa. Seuraavaksi käyttäjä on vapaa selaamaan karttaa ja voi halutessaan pyytää reitin omasta sijainnistaan kohteeseen. Reitti radalle saadaan ensin selvittämällä käyttäjän sijainti käyttäen A-GPS paikannuspiiriä. Tämän jälkeen käyttäjän sijainnin ja kohteen koordinaatit lähetetään Microsoftin Bing Routeservice palveluun, joka laskee halutun reitin ja palauttaa valmiit reittitiedot. Lopuksi käyttäjälle näytetään etäisyys ja reitti kohteeseen.

5.3.1 Bing Maps

Bing Maps on Microsoftin kehittämä karttapalvelu, mikä toimii osana Microsoftin omaa Bing hakukonetta. Ohjelmistokehittäjät voivat hyödyntää palvelun tarjoamia karttapalveluja ohjelmistorajapinnan (API) kautta. Ohjelmistorajapintoja käytetään välikätenä eri ohjelmien ja palvelujen välisissä keskusteluissa. Etuna on, että sovelluskehittäjän ei tarvitse tietää millä menetelmillä keskustelu hoituu vaan riittää, että rajapinnalle esitetään pyyntö tarvittavista toimenpiteistä. Rajapinta välittää tiedon vastaanottavalle osapuolelle ja palauttaa pyytävälle osapuolelle mahdollisen palautusarvon (Microsoft 2013f).

Microsoft tarjoaa useita eri ohjelmistorajapintoja karttatietojen käyttötarkoituksen ja alustan mukaisesti. Windows Phone-käyttöjärjestelmä voi käyttää Ajax- (Asynchronous Javascript And XML), REST- (Representational State Transfer), Spatial Data- ja Silverlight-rajapintaa. Näistä viimeksi mainittu on yleisimmin käytössä, koska siihen löytyy valmis karttakontrolli (Map). Kartan käyttö on maksutonta, mikäli sitä käyttävä sovellus on myynnissä tai tarjolla ilmaiseksi Windows Phone sovelluskaupassa ja käyttö ei ylitä sallittuja rajoja. Esimerkiksi rajapinnalle saa

esittää vain 50000 kyselyä kuukaudessa ilmaiseksi per sovellus (Microsoft 2013f).

Karttojen käyttö edellyttää lisäksi rekisteröitymisen Bing Maps Portal-palveluun. Palvelun kautta kehittäjä saa kartta-avaimen, johon tulee olla viittaus, kun kartta-kontrollia käytetään. Mikäli viitatusta ei ole, kontrolli ei voi ladata karttadataa Bing-karttapalvelusta. Portaalin avulla Microsoft pystyy tarkkailemaan avaimen käyttöön liittyvää liikennettä. Lisäksi kehittäjän on mahdollista saada arvokasta tietoa siitä kuinka paljon sovelluksen karttaominaisuutta todellisuudessa käytetään (Microsoft 2013g).

5.3.2 A-GPS

Jokaisesta Windows Phone-käyttöjärjestelmää käyttävästä laitteesta löytyy A-GPS paikannuspiiri, mikä tarkoittaa, että puhelimen sijainti voidaan selvittää GPS-satelliittien lisäksi hyödyntäen WLAN-tukiasemia ja puhelintolppia. Tämä nopeuttaa varsinaista paikannusta, koska aluksi voidaan selvittää käyttäjän sijainti karkealla tasolla ja kun GPS-satelliittiin on saatu yhteys, voidaan sijainti tarkentaa metrien tarkkuudella. Paikkatiedon lisäksi puhelimesta on mahdollista saada tieto kuinka korkealla käyttäjä on merenpinnasta ja mahdollisen liikkeen suunta ja nopeus (Järvinen 2011, 152-153)

5.3.3 Bing Routeservice

Bing Routeservice on Microsoftin tarjoama navigaatiopalvelu, jonka avulla on mahdollista laskea reittitietoja kahden tai useamman pisteen välillä. Palvelun avulla voidaan laskea reitit autoilijoille ja jalankulkijoille. Jalankulkijoiden kohdalla palvelu voi myös hyödyntää joukkoliikenteen tarjoamia mahdollisuuksia. Näitä tietoja on kuitenkin hyvin rajallisesti saatavana ja vain isoimmista kaupungeista. Palvelu toimii REST rajapinnan kautta ja SOAP (Simple Object Access Protocol) palveluna (Microsoft 2013h).

Samoin kuin kartta-kontrolli, Routeservice-palvelun käyttö vaatii rekisteröitymisen Bing Maps portaaliin. Rekisteröitymisen jälkeen kyselyitä voi tehdä yhtä paljon kuin karttakontrollikin eli 50000 kappaletta kuukaudessa. Muita rajoituksia ei ole. Palvelun käyttö on helppoa ja riittää, kun määrittää lähtöpisteen, loppupisteen, siirtymistavan (autolla vai kävellen), mennäänkö lyhintä vai nopeinta reittiä ja käytetäänkö maateitä vai pikkuteitä. Tulosten laskennan jälkeen palvelu palauttaa listan koordinaatteja, joiden avulla muodostetaan reitti kohteeseen. Lisäksi palvelusta saa esimerkiksi tiedot matkan pituudesta ja arvioidusta matkustusajasta. Lopuksi reitti voidaan tulostaa kartalle esimerkiksi piirtämällä havainnoiva reitti- viiva lähtöpisteestä välietappien kautta kohteeseen (Microsoft 2013h).

5.4 Suunnittelu

Karttasivua ja sen käyttöliittymää suunniteltaessa haluttiin keskittyä tekemään siitä mahdollisimman yksikertainen ja helposti omaksuttava. Se noudattelee Microsoftin karttasivun mallirakennetta. Mallirakenteita noudattamalla saavutetaan yhtenäisyyttä eri sovellusten välillä. Kun käyttäjä on tottunut, että sovellukset toimivat tietynlaisella tavalla, on hänen helppo omaksua uusienkin sovellusten toiminta.



Kuva 6. Karttasivu

Karttasivu (Kuva 6.) on jaettu kahteen osaan, sisältöön ja kontrolleihin. Näytettävä sisältö on määritetty sisältöpaneelille. Paneeli sisältää itse kartan lisäksi piilotettuina etäisyyspaneelin, latauspaneeli ja ”pois käytöstä”-paneelin. Paneelit tuodaan esille tarvittaessa esimerkiksi informoimaan käyttäjää, että reitin lataus on käynnissä.

Toinen osa on kontrollipaneeli, josta käyttäjä ohjaa karttanäkymässä tapahtuvaa toimintaa. Se sisältää kolme painikekontrollia (Button), joista ulommaisista voidaan lähentää ja loitontaa valittuun kartan keskipisteeseen ja keskimmäisestä pyytää näyttämään reitti käyttäjän sijainnista kohderadalle.

5.5 Käyttöliittymä

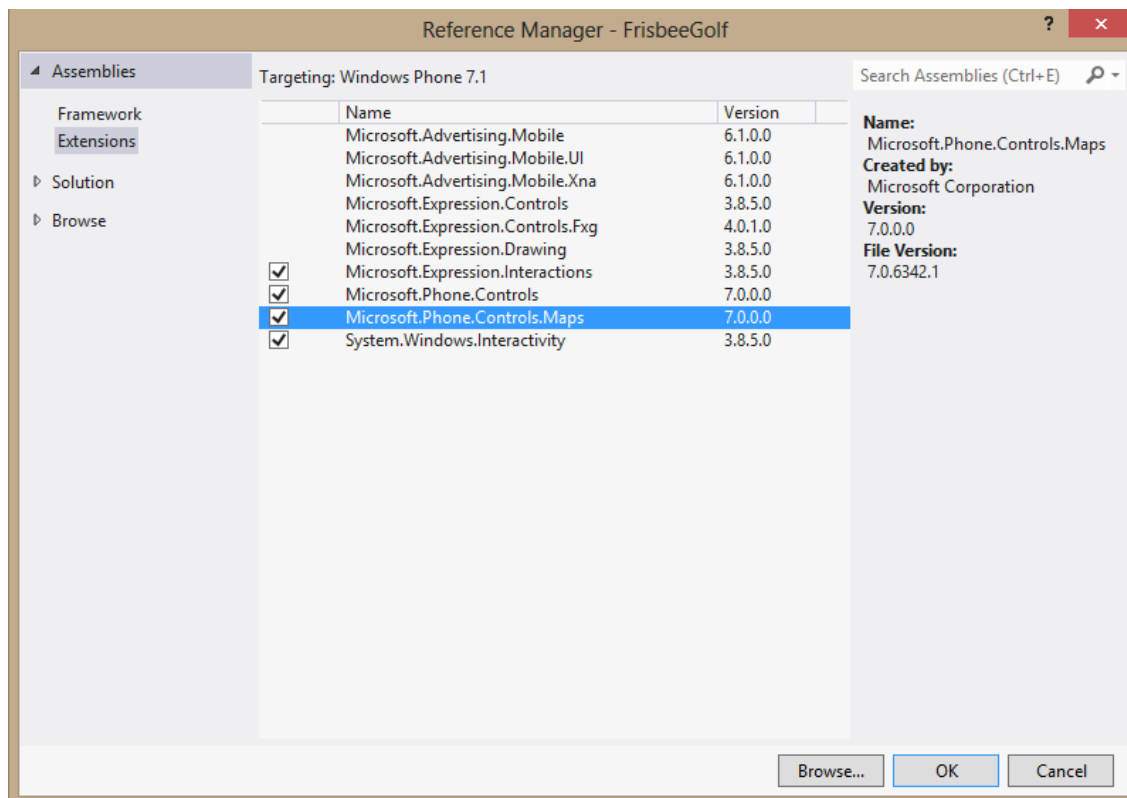
Karttasivun käyttöliittymä on sovelluksessa toteutettu käyttäen XAML-ohjelmointikieltä. Se on rakennettu oletus Windows Phone käyttöliittymäsivun pohjalle,

koska se sisältää valmiiksi suurimman osan tarvittavista määrittelyistä. Määrittelyt on tehty sivun pääelementin (PhoneApplicationPage) sisään. Elementti sisältää oman luokkamäärittelynsä, sivulla käytettävien nimiavaruuksien määrittelyt, sivun kokomäärittelyt ja mahdollisia muita UI-määrittelyksiä.

Mikäli sivulle halutaan tuoda uusia elementtejä muista kuin oletusnimiavaruuksista, pitää ne esitellä pääelementissä. Esimerkiksi karttakontrollin käyttämiseksi on tehtävä viittaus BING-karttanimiavaruuteen seuraavasti.

```
xmlns:m="clr-namespace:Microsoft.Phone.Controls.Maps;  
assembly=Microsoft.Phone.Controls.Maps"
```

Visual Studiota käytettäessä, käytettävään nimiavaruuteen on lisäksi tehtävä viittaus projektissa. Viittaus tehdään Windows Phone projektin viittauskansioon (References). Lisäksi viittauksia tulee tehdä kun halutaan käyttää COM (Component Object Model) DLL (Dynamic-Link Library) tyyppisiä elementtejä, itse tehtyjä luokkakirjastoja, toisia projekteja tai XML pohjaisia Web-palveluita. Viittaus voidaan tehdä Visual Studiossa Solution Explorerin kautta Add Reference toiminnolla (Kuva 7.).



Kuva 7. Bing-karttanimiavaruuden lisääminen projektiin

Pääelementin alle on luotu sivun varsinainen rakenne, jonka ensimmäinen komponentti on pohjapaneeli (LayoutRoot), joka on Grid tyyppinen paneeli. Grid tyyppiset paneelit toimivat alustoina mihin voidaan vapaasti sijoittaa elementtejä joko käyttämällä rivi- (Row) ja/tai jonoattribuutteja (Column) tai määrittelemällä suoraan marginaaliattribuuttiin (Margin) kuinka kappale sijoittuu suhteessa paneelin reunoihin. Pohjapaneeli sisältää koko sivun vapaan alueen lukuun ottamatta sovelluspalkkia (Application Bar). Sovelluspalkki mahdollistaa helpon pääsyn sovelluksen useasti käytettäviin toimintoihin ja tuo yhtenäisyyttä sovellusten välisiin eroihin. Karttasivulla sovelluspalkkia ei tosin käytetä.

5.5.1 Sisältö-paneeli

Pohjapaneelin alla nähdään sovelluksen varsinainen asettelu, jossa sivu on jaettu kahteen osaan, joista ensimmäinen on sisältöpaneeli (ContentPanel). Se on määritetty seuraavalla tavalla.

```

<Grid x:Name="ContentPanel" VerticalAlignment="Top">
    <Grid.Resources>
        <Storyboard x:Name="ZoomIn">
            <DoubleAnimation x:Name="ZoomLevel"
                Storyboard.TargetProperty="ZoomLevel"
                Storyboard.TargetName="RataMap"
                From="1.0"
                To="13.0"
                Duration="0:0:3" />
        </Storyboard>
    </Grid.Resources>

    <m:Map x:Name="RataMap" Height="768" Width="480"
        VerticalAlignment="Top" CredentialsProvider="..." />

    <TextBox x:Name="NoInternetTextBox" Height="116"
        Text="Ei internet-yhteyttä" Visibility="Visible" />

    <Grid x:Name="DistanceGrid" Height="90" Width="480"
        Visibility="Collapsed" Margin="0,7,0,671">
        <Grid.Background>
            <ImageBrush ImageSource="/Assets/Button/
                TopBottomBgRata.png" />
        </Grid.Background>

        <TextBlock x:Name="DistanceTextBlock"
            TextAlignment="Center" Margin="0,8,0,22" />
    </Grid>

    <Grid x:Name="LoadingGrid" Background="Black"

        Opacity="0.5" Margin="0,0,0,0" Visibility="Collapsed">

        <ProgressBar x:Name="customProgressBar"
            IsIndeterminate="true" Foreground="white" />

        <TextBlock x:Name="WaitTextBlock" Height="30"
            Width="110" Text="Ladataan..." />
    </Grid>
</Grid>

```

Sisältöpaneelin rakenne alkaa resurssiattribuutin (Resources) määrittelyllä. Tämän alle määritetään kaikki paneelin sisältämien elementtien tarvitsemat resurssit kuten tyylimäärittelyt ja animaatiot. Resurssit ovat objekteja, joita voidaan toistuvasti käyttää useiden eri elementtien toimesta. Tässä tapauksessa se sisältää karttakontrollin tarvitseman tarkennusanimaation (ZoomIn). Animaatioiden

toistamiseen tarvitaan käsikirjoituselementti (Storyboard). Sen tehtävä on sen sisältävien animaatioiden toistaminen synkronoidusti ja toiston pysäyttäminen pyydettäessä. Sen sisällä on puolestaan lineaarinen animaatio (DoubleAnimation), joka kertoo millainen animaatio halutaan suorittaa ja missä ajassa.

Paneelin sisältö on jaettu kolmeen osaan. Ensimmäinen osa on kartta-kontrolli. Kontrollin käyttö vaatii edellä mainitun nimiavaruusmäärittelyjen lisäksi edellä mainitun kartta-avaimen (Bing Maps Key). Avain sijoitetaan suoraan valtuutus-attribuuttiin (CredentialsProvider), jonka jälkeen kartta on vapaasti käytettävissä. Kartta-avaimen lisäksi karttakontrolli vaatii toimiakseen internet-yhteyden. Mikäli internet-yhteyttä ei ole käytettävissä, näytetään käyttäjälle tekstilaatikko-elementti (TextBlock) johon tulostetaan teksti "Ei internet yhteyttä". Tekstilaatikko-elementti, niin kuin kaikki muutkin XAML-elementit, voidaan tuoda ja poistaa näkyvistä määrittämällä näkyvyysattribuutti (Visibility) joko näkyväksi (Visible) tai näkymättömäksi (Collapsed) tilaan. Oletusarvo on näkyvä.

Seuraava osa on etäisyyspaneeli (DistanceGrid). Sen tarkoituksena on näyttää käyttäjälle kartalle tuotavan reittitiedon pituus. Etäisyyspaneelin sisällä on määritetty ensimmäisenä tausta-attribuutti (Background). Sen avulla voidaan paneelin taustalle asettaa kuva tai esimerkiksi gradientti väri. Taustakuva määritellään kuvasivellinelementin (ImageBrush) avulla määrittämällä lähdeattribuuttiin (ImageSource) kuvan sijainti. Sijainti voi olla joko paikallinen tai internet-osoite. Mikäli taas haluttaisiin määrittää esimerkiksi ainoastaan yksivärinen tausta, määritettäisiin tausta-attribuutin sisälle värisivellinelementti (SolidColorBrush) ja sen väriattribuuttiin (Color) haluttu väri.

Viimeisenä määritellään latauspaneeli (LoadingGrid). Kun reittitiedot on lähetetty reittipalvelun laskettavaksi, tuodaan käyttäjälle näkyviin latauspaneeli indikoimaan tapahtuvaa laskentaa. Muista paneeleista poiketen, lataus-paneelin taustaväri on määritetty mustaksi ja sen läpinäkyvyyssattribuutti (Opacity) on määritetty 0.5. Tarkoituksena on, että paneeli peittää karttakontrollin latauksen ajaksi ja estää muut toiminnot, jotta laskenta voidaan rauhassa suorittaa loppuun. Paneeli sisältää edistymispalkkiolon (ProgressBar) ja tekstilaatikko-elementin indikoimaan tapahtuvaa latausta.

5.5.2 Kontrolli-Paneeli

Pohjapaneelin toinen osa on kontrollipaneeli (ControlPanel) ja se puolestaan on määritelty seuraavalla tavalla.

```
<Grid Name="ControlPanel" VerticalAlignment="Bottom" >
  <Grid.Background>
    <ImageBrush ImageSource="/Assets/Button/
      TopBottomBg.png"/>
  </Grid.Background>

  <Button Height="80" Width="80" HorizontalAlignment="Left"
    Click="minusButton_Click">
    <Button.Background>
      <ImageBrush ImageSource="/Assets/button/
        Zoomout.png"/>
    </Button.Background>
  </Button>

  <Button Height="80" Width="260" FontSize="24"
    Content="Näytä reitti" Foreground="Azure"
    Click="navigationButton_Click" BorderThickness="0"/>

  <Button Height="80" Width="80" HorizontalAlignment="Right"
    Click="plusButton_Click">
    <Button.Background>
      <ImageBrush ImageSource="/Assets/
        button/Zoomin.png"/>
    </Button.Background>
  </Button>
</Grid>
```

Paneelin rakenne alkaa taustakuvan määrittämisellä. Varsinainen sisältö koostuu kolmesta nappikontrollista (Button). Nappikontrollien avulla käyttäjä ohjaa sisäl-
töpaneelissa tapahtuvaa toimintaa. Ylin ja alin nappikontrolli on tarkoitettu kartan
keskipisteen lähentämiseen ja loitontamiseen. Itse toiminto tapahtuu toimintalo-
giikkasivulla ja viittaus tapahtumalaukaisiaan (EventTrigger) tehdään napsautus-
tapahtuman (Event) yhteyteen. Tapahtuman nimi kertoo, minkälainen käyttäjän
ele laukaisee (Triggers) tapahtumalaukaisijan, mikä tässä tapauksessa on napin
painaminen (Click).

Keskimmäisen nappikontrollin avulla käyttäjä voi halutessaan kysyä reittiä omasta sijainnistaan kohteeseen. Muista kontrolleista poiketen, sille ei ole määritetty taustakuvaa vaan ainoastaan tekstisisältö (Content). Lisäksi se määrittää, että kontrolli on täysin läpinäkyvä eli sen reunat on hävitetty määrittämällä reunan paksuus (BorderThickness) nollassi.

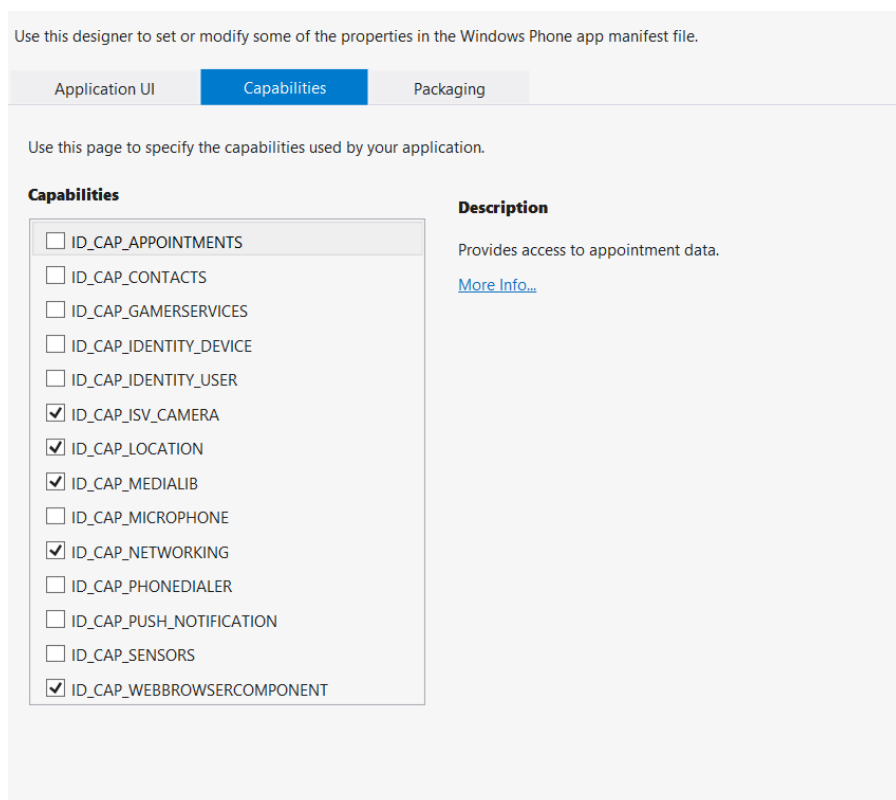
5.6 Toimintalogiikka

Karttasivun varsinaiset toiminnallisuudet on määritetty toimintalogiikkasivulle ja ne on toteutettu käyttäen C#-ohjelmointikieltä. Käyttämällä Visual Studiosta löytyvää perus Windows Phone-sivun pohjaa, on toimintalogiikka-sivu tullut valmiiksi luotuna käyttöliittymäsivun ohessa. Käyttöliittymä- ja toimintalogiikkasivut muodostavat yhdessä luokan, josta toimiva sovellussivu muodostuu. Tämä tapahtuu määrittämällä toimintalogiikkasivu käyttöliittymän osaluokaksi, jolloin kääntäjä osaa yhdistää sivut toisiinsa välikoodikäänösvaiheessa.

Toimintalogiikkasivu alkaa käytettävien nimiavaruuksien määrittelyllä. Kuten käyttöliittymäsivulla, nimiavaruuksiin tulee olla viittaus lisäksi viittauskansiossa. Toimintalogiikkasivulla määrittäminen tehdään käyttäen using-direktiiviä. Esimerkiksi kun halutaan käyttää paikannuspiiriä, tehdään viittaus seuraavasti.

```
using System.Device.Location;
```

Paikannuksen käyttämiseksi on siitä lisäksi kerrottava sovelluksen tiedostossa WMAppManifest.xml (Kuva 8.) määritteellä ID_CAP_LOCATION. Tiedostossa määritellään kaikki sovelluksen käyttämät puhelinominaisuudet.



Kuva 8. Käytettävien ominaisuuksien määrittely

Nimiavaruuksien esittelyn jälkeen määritellään sivun nimiavaruus, mikä tässä tapauksessa on sovelluksen nimiavaruus, FrisbeeGolf. Windows Phone projektissa oletuksena sovelluksen nimiavaruus on nimetty projektin nimen mukaisesti ja kaikki luodut luokat, sivut ja muut elementit sijoitetaan automaattisesti sen alle. Tämän jälkeen määritellään sivun luokka, luokan rakentajafunktio (Constructor) ja aloitetaan varsinaisten toimintojen määrittely.

5.6.1 Kartta-kontrolli

Saavuttaessa karttasivulle ensimmäinen asia, mitä käyttäjälle näytetään, on käyttäjän valitseman radan sijainti. Valinta tehdään ratasivulla. Radan valinta käynnistää navigoinnin ja sijoittaa radan tiedot globaaliin Radat tyyppiseen olioön. Seuraavaksi alustetaan karttasivu ja ladataan käyttöliittymän komponentit. Kun

alustus on suoritettu, tarkistetaan internet-yhteyden tila kysymällä sitä verkkorajapintaluokan (NetworkInterface) staattiselta (Static) verkkotila-funktiolta (GetIsNetworkAvailable).

```
NetworkInformation.NetworkInterface.GetIsNetworkAvailable();
```

Funktio palauttaa totuusarvomuuttuja tyyppisen (Boolean) tosi (True), mikäli yhteys on käytössä ja epätosi (False), mikäli verkkoa ei ole saatavilla. Jos yhteys on kunnossa, kutsutaan ShowLocation-funktiota. Funktio näyttää kohteen kartalla. Parametreiksi tuodaan valitun radan tiedot. Funktio alkaa pinni-olion (Pushpin) määrittelyllä.

```
Pushpin pin1 = new Pushpin();
pin1.Location = new GeoCoordinate(Latitude, Longitude);
pin1.Content = name + System.Environment.NewLine +
addressLine + System.Environment.NewLine + postalCode + ", " +
city;
```

Pinnioliota käytetään merkitsemään paikkaa, jossa valittu rata sijaitsee. Oliolle paikka-attribuuttiin (Location) määritetään ensin sijainti funktioon parametreina saatujen pituuspiirin koordinaattien (Longitude) ja leveyspiirin koordinaattien (Latitude) muodostamalla niistä uusi paikkaolio (GeoCoordinate). Viimeisenä määritellään sisältöattribuuttiin (Content) käyttäjälle näytettävä tuloste.

```
this.RataMap.Children.Add(pin1);
this.RataMap.SetView(pin1.Location, 1);
```

Pinni lisätään karttakontrollin lapsikokoelmaan (Children). Karttakontrolli lataa automaattisesti kokoelmaan lisätyt objektit kartalle. Lopuksi asetetaan kartan keskipiste pinnin sijaintiin, määritetään etäisyystaso pienimmäksi eli ykköseksi ja kutsutaan käyttöliittymässä määritellyn lähentymis-animaation (ZoomIn) aloitus-funktiota (Begin). Animaatio hoitaa tarkennuksen kohteeseen.

```
ZoomIn.Begin();
```

Kun animaatio on suoritettu loppuun, pääsee käyttäjä itse tutkimaan karttaa. Käyttäjä voi vapaasti sormieleillä (Gesture) liikutella karttaa ja lähentää ja loitontaa näkyvyyttä kohteeseen. Lähentäminen tapahtuu painamalla käyttöliittymässä pluskontrollia, mikä laukaisee kontrolliin liitetyn tapahtumakäsittelijäfunktion.

```
private void plusButton_Click(object sender, RoutedEventArgs e)
{
    RataMap.ZoomLevel += 1;
}
```

Funktio ottaa parametreiksi tapahtuman kutsujaolion (sender) ja tapahtumatiedot (RoutedEventArgs) sisältävän olion (e). Loitonnus tapahtuu painamalla miinus-kontrollia.

```
private void minusButton_Click(object sender, RoutedEventArgs e)
{
    RataMap.ZoomLevel -= 1;
}
```

Lopuksi käyttäjä voi myös navigoida valitulle radalle. Navigointi aloitetaan painamalla ”Näytä reitti”-kontrollia. Reitti saadaan selville lähettämällä Microsoftin Bing Routeservice palveluun halutun reitin alku- ja loppukoordinaatit ja tarvittavat parametrit. Loppukoordinaatit eli radan sijainti on jo tiedossa, joten seuraavaksi tulee selvittää alkukoordinaatit eli käyttäjän sijainti.

5.6.2 Paikannuspalvelu

Sovelluksessa paikkatieto saadaan selville käyttämällä paikanhakuluokkaa (GeoCoordinateWatcher). Luokkaa käytetään tapahtumapohjaisesti eli se ilmoittaa sovellukselle kun käyttäjän sijainti on muuttunut. Tieto sijainnin muutoksesta ja nykyisestä sijainnista saadaan kuuntelemalla luokkaan kuuluvaan paikanmuutostapahtumaa (PositionChanged). Luokan käyttö aloitetaan luomalla siitä käytettävä olio (watcher).

```
GeoCoordinateWatcher watcher = new GeoCoordinateWatcher(GeoPositionAccuracy.High);
```

Olion luonnin yhteydessä määritellään sijaintitiedon haluttu tarkkuus, mikä asetetaan tarkkuusparametrilla (`GeoPositionAccuracy`). Parametri voi ottaa arvon suuri (`High`) tai oletus (`Default`). Niiden ainoa ero on, että oletus käyttää pelkästään WLAN-tukiasemia ja puhelintolppia sijainnin määrittämiseen. Tällöin mitaustulos ei ole kovinkaan tarkka, mutta se säästää akkua, koska GPS-satelliittiin ei tarvitse ottaa yhteyttä.

Seuraavaksi määritellään vähimmäismatka, mikä tulee kulkea kunnes järjestelmä voi jälleen kutsua paikanmuutostapahtumaa. Matkan pituus voidaan asettaa kynnysattribuutilla (`MovementThreshold`) ja se ilmoitetaan metreinä. Parametrin arvo kannattaa määritellä mahdollisimman suureksi sillä liian lyhyt matka kasvattaa paikannustiheyttä kuluttaen enemmän akkua.

```
watcher.MovementThreshold = 20;
```

Lopuksi tilataan tapahtumatiedot mahdollisista paikkamuutoksista paikanmuutostapahtumalta (`PositionChanged`). Kun paikkatieto muuttuu, kutsutaan funktiota `watcher_PositionChanged`.

```
watcher.PositionChanged += new EventHandler<GeoPosition-  
ChangedEventArgs<GeoCoordinate>>(watcher_PositionChanged);
```

Tämän lisäksi voidaan tilata tapahtumatietoja tilanmuutostapahtumalta (`StatusChanged`), mikä kertoo GPS-piirin tilasta. Se antaa tietoa muun muassa virhetilanteista, joista voidaan edelleen tiedottaa käyttäjälle ja pyytää tekemään tarpeellisia toimenpiteitä.

```
watcher.StatusChanged += new EventHandler<GeoPositionSta-  
tusChangedEventArgs>(watcher_StatusChanged);
```

Lopuksi käynnistetään palvelu kutsumalla paikanhakuolion käynnistysfunktioita (`Start`).

```
watcher.Start();
```

Kun GPS-piirin tila muuttuu, kutsutaan funktiota `watcher_StatusChanged`.

```
private void watcher_StatusChanged(object sender, GeoPosi-  
tionStatusChangedEventArgs e)
```

```

{
    switch (e.Status)
    {
        case GeoPositionStatus.Disabled:
            break;

        case GeoPositionStatus.Initializing:
            break;

        case GeoPositionStatus.NoData:
            break;

        case GeoPositionStatus.Ready:
            break;
    }
}

```

Funktio ottaa sisäänsä kutsujaolion ja paikannuspiirin tilaa (`GeoPositionStatusChangedEventArgs`) kuvaavan olion. Tietoja nykyisestä tilasta kysytään tilaparametrilta (`Status`) käyttämällä esimerkiksi switch-case rakennetta. Tiloja on neljä: Alustetaan (`Intializing`), Valmis (`Ready`), Ei dataa (`NoData`) ja pois käytöstä (`Disabled`), joista kaksi viimeisintä ovat kaikkein tärkeimpiä sillä ne kuvaavat paikannuspiirin virhetiloja.

Mikäli edellä mainittuja virhetilanteita ei synny ja paikannustiedot saadaan onnistuneesti, järjestelmä kutsuu funktiota `watcher_PositionChanged`.

```

private void watcher_PositionChanged(object sender, GeoPosition-
ChangedEventArgs<GeoCoordinate> e)
{
    GeoCoordinate UserLocation = new GeoCoordinate(e.
    Position.Location.Latitude, e.Position.Location.Longitude);
}

```

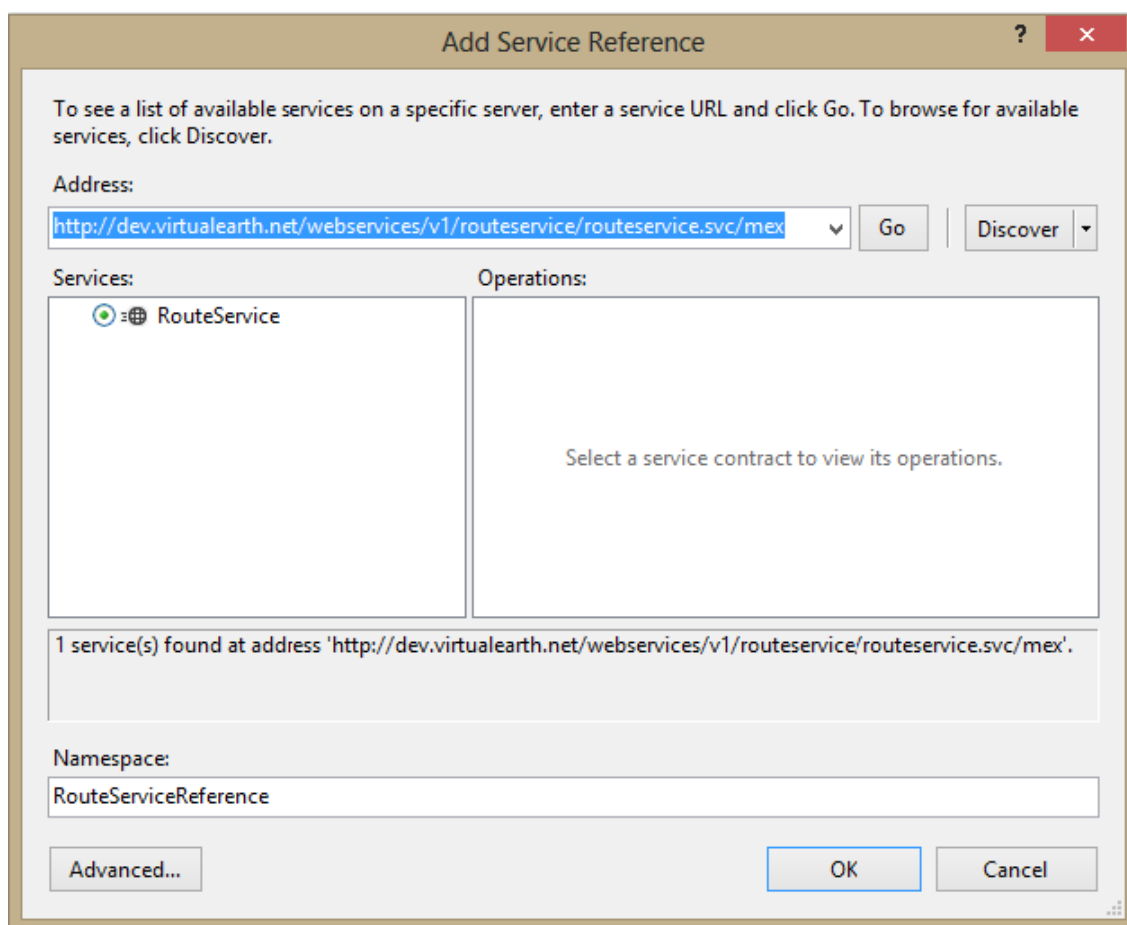
Kuten `watcher_StatusChanged` funktiossa, `watcher_PositionChanged` ottaa sisäänsä parametrina kutsujaolion, mutta poiketen paikkatietokokoelma (`GeoPositionChangedEventArgs<GeoCoordinate>`) olion. Nuolimerkkien sisään määritetään muoto, jossa paikkatiedot halutaan. Paikkatiedon tallentamiseksi luodaan paikkaolio (`UserLocation`), jonka rakentajafunktio ottaa parametriksi leveysas-

teen ja pituusasteen koordinaatit, mitkä saadaan parametrina tulleesta paikkatietokokoelmasta. Kun paikkatieto on saatu, eikä tietoja haluta enää kerätä, voidaan palvelu pysäyttää kutsumalla paikanhakuolion lopetusfunktiota (Stop).

```
watcher.Stop();
```

5.6.3 Reittipalvelupyynnön lähettäminen

Kun tiedot alku- ja loppukoordinaateista ovat selvillä, tiedot lähetetään Bing Routeservice-palveluun. Sovelluksessa hyödynnetään palvelun SOAP-rajapintaa, joten ensimmäiseksi on tehtävä viittaus kyseiseen palveluun (Service reference). Viittaus voidaan tehdä Visual Studiassa Solution Explorerin kautta Add service reference toiminnolla (Kuva 9.).



Kuva 9. Routeservice-palvelun lisäys projektiin

Lopuksi lisätään viittaus nimiavaruuteen mistä palvelua voi kutsua.

```
using FrisbeeGolf.RouteServiceReference;
```

Kun tarvittavat määritykset on suoritettu, voidaan palvelun käyttö aloittaa luomalla reittipalveluluokasta (RouteServiceClient) olio (routeservice), jonka kautta hoidetaan keskustelu itse palvelun kanssa. Kun palvelu on käsitellyt reittipalvelupyynnön, laukaisee se olioon liitetyn tapahtuman (CalculateRouteCompleted). Tapahtumaan lisätään tapahtumakäsittelijäfunktion routeservice_CalculateRouteCompleted, jota kutsutaan, kun tapahtuma laukeaa. Funktion parametreina tuodaan Laskentatulostiedot (CalculateRouteCompletedEventArgs), mitkä sisältävät laskennan tulokset.

```
RouteServiceClient routeService = new RouteServiceClient(
    "BasicHttpBinding_IRouteService");
```

```
routeService.CalculateRouteCompleted += new EventHandler<CalculateRouteCompletedEventArgs>(routeservice_CalculateRouteCompleted);
```

Seuraavaksi määritellään varsinainen palvelupyynnön sisältö. Pyyntö tehdään luomalla reittipyyntöluokasta (RouteRequest) olio (routerequest). Pyyntön mukana tulee toimittaa myös karttakontrollissa käytetty kartta-avain, jonka avulla tarkkaillaan palveluun tehtävien pyyntöjen määrää.

```
RouteRequest routerequest = new RouteRequest();
routerequest.Credentials = new Credentials();
routerequest.Credentials.ApplicationId = "...";
```

Ennen koordinaattitietojen lisäämistä tulee lisäksi määrittää, minkälaista reittiä halutaan matkustaa ja kuinka matkustus tapahtuu. Tässä tapauksessa haluamme matkustaa mahdollisimman nopeaa reittiä (RouteOptimization.MinimizeTime) autolla (TravelMode.Driving). Lisäksi määritämme, että haluamme reittitiedot pisteiden (RoutePathType.Points) kautta, mikä mahdollistaa reitin piirtämisen kartalle. Asetukset määritetään reittipyyntöolion asetus-attribuuttiin (Options).

```
routerequest.Options = new RouteOptions();
routerequest.Options.Mode = TravelMode.Driving;
routerequest.Options.RoutePathType = RoutePathType.Points;
```

```
routerrequest.Options.Optimization = RouteOptimization.
    MinimizeTime;
```

Koordinaattitiedot määritellään välipisteluokan (Waypoint) olioilla joille annetaan parametrina pituuspiirin ja leveyspiirin koordinaatit. Lisäksi voidaan antaa kuvaus kohdepisteestä. Lopuksi oliot lisätään pyynnön mukana lähetettävään kokoelmaan (Waypoints). Kokoelmaan määritetään tässä tapauksessa vain alku- ja loppupiste, mutta listalle voidaan lisätä yhteensä 25 pistettä, joista ensimmäinen on alkupiste ja viimeinen loppupiste. Reitti lasketaan alku- ja loppupisteiden välisten pisteiden kautta järjestyksessä.

```
routerrequest.Waypoints = new ObservableCollection<Waypoint>();

routerrequest.Waypoints.Add(new Waypoint
{
    Description = "startpoint",
    Location = new Location
    {
        Longitude = StartLocation.Longitude,
        Latitude = StartLocation.Latitude
    }
});
routerrequest.Waypoints.Add(new Waypoint
{
    Description = "endpoint",
    Location = new Location
    {
        Longitude = EndLocation.Longitude,
        Latitude = EndLocation.Latitude
    }
});
```

Lopuksi määritellään palveluun liittyvät aikarajat. Aikarajojen määrittäminen ei ole pakollista, mutta ne on hyvä määrittää, ettei palvelusta odoteta vastausta liian pitkään esimerkiksi silloin kun palvelu on alhaalla. Aikaraja voidaan määrittää luomalla aikaväliluokasta (TimeSpan) olio (timeout). Palvelua odotellaan 15 sekunnin ajan.

```
TimeSpan timeout = new TimeSpan(0, 0, 15);
```

```
routeService.Endpoint.Binding.ReceiveTimeout = timeout;
routeService.Endpoint.Binding.SendTimeout = timeout;
```

```
routeService.Endpoint.Binding.OpenTimeout = timeout;
```

Lopuksi pyyntö lähetetään palveluun ja jäädään odottamaan vastausta.

```
routeService.CalculateRouteAsync(routerequest);
```

5.6.4 Reittipalveluvastauksen käsittely

Kun reittipyyntö on käsitelty, palvelu kutsuu edellä määritellyn tapahtumakäsittelijäfunktion `routeservice_CalculateRouteCompleted`. Funktion alkuun on ensimmäiseksi määritetty virheen tarkastus.

```
if (e.Error != null)
{
    MessageBox.Show("Virhe: " + e.Error.Message +
        " Yritä myöhemmin uudelleen");
    return;
}
```

Mikäli reittilaskennan aikana on tapahtunut virhe, saadaan tieto parametreissa tuodulta `Laskentatulostiedot` (`CalculateRouteCompletedEventArgs`) parametrilta. Se sisältää kaikki palvelun palauttavat tiedot. Tarkistus tapahtuu null-tarkistuksena eli mikäli `e:n` virhe attribuutti (`Error`) ei ole alustettu, ei laskennan aikana ole tapahtunut virhettä. Mikäli virhe tapahtuu, tiedotetaan asiasta käyttäjälle erillisellä ponnahdusikkunalla (`MessageBox`) ja funktion suoritus lopetetaan.

Seuraavaksi tarkistetaan vielä onko vastauksen tila onnistunut, ja että vastauksen mukana on tullut navigointiohjeet. Vastauksen tila saadaan vastauksentila-luokalta (`ResponseStatusCode`) totuusarvoattribuutilta (`Success`). Navigointitietoja voi kysyä laskentatulostietojen mukana tulleet etappikokoelmalta (`Legs`). Mikäli kokoelmassa ei ole yhtään elementtiä, on laskenta epäonnistunut.

```
e.Result.ResponseSummary.StatusCode == ResponseStatusCode.
Success && (e.Result.Result.Legs.Count != 0)
```

Kun laskenta on todettu onnistuneeksi, aloitetaan tietojen sijoittaminen kartalle. Tietojen sijoittaminen aloitetaan Kartalle piirrettyjen lapsielementtien poistamisella.

```
this.RataMap.Children.Clear();
```

Kun kokoelma on tyhjennetty, luodaan karttaviivaluokasta (MapPolyline) olio (routeline), joka kuvaa vastaanotettujen pisteiden kautta kulkevaa reittiviivaa. Kun olio on alustettu, sille alustetaan paikkatietokokoelma (LocationCollection), Väri (SolidColorBrush), Läpinäkyvyys (Opacity) ja viivan paksuus (StrokeThickness).

```
Color routeColor = Colors.Red;
SolidColorBrush routeBrush = new SolidColorBrush(routeColor);
MapPolyline routeline = new MapPolyline();
routeline.Locations = new LocationCollection();
routeline.Stroke = routeBrush;
routeline.Opacity = 0.65;
routeline.StrokeThickness = 5.0;
```

Seuraavaksi lisätään kaikki vastaanotetut koordinaatit alustettuun paikkatietokoelmaan ja luodaan pinnioliot alku- ja loppupisteille.

```
foreach (Location p in e.Result.Result.RoutePath.Points)
{
    routeline.Locations.Add(new GeoCoordinate
        (p.Latitude, p.Longitude));
}
```

```
Pushpin startpin = new Pushpin();
startpin.Location = StartLocation;
startpin.Content = "Olet Tässä!";
```

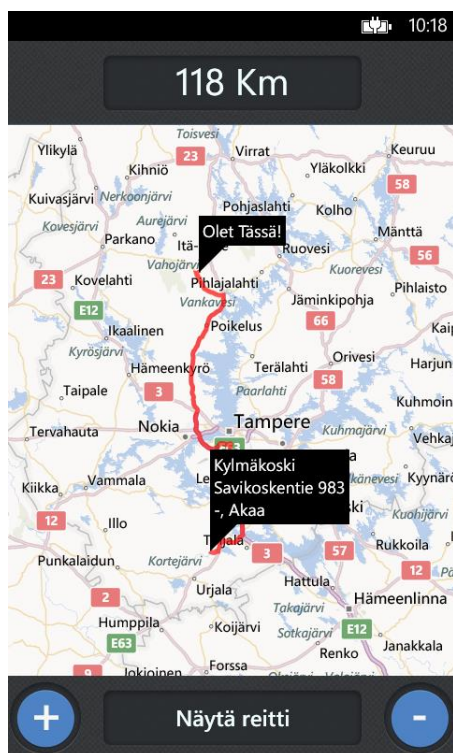
```
Pushpin endpin = new Pushpin();
endpin.Location = EndLocation;
endpin.Content = name + System.Environment.NewLine + addressLine
+ System.Environment.NewLine + postalCode + ", " + city;
```

Kun kaikki piirrettävät elementit on luotu, lisätään ne kartan lapsikokoelmaan ja asetetaan reitin alkupiste kartan keskipisteeksi.

```
this.RataMap.Children.Add(routeline);
this.RataMap.Children.Add(startpin);
this.RataMap.Children.Add(endpin);
this.RataMap.SetView(startpin.Location, 8);
```

Lopuksi näytetään käyttäjälle etäisyys kohteeseen tulostamalla se näkyväksi tuotavaan etäisyyspaneeliin. Etäisyys-paneeli saadaan näkyväksi asettamalla sen näkyvyys-attribuutti näkyväksi. Etäisyys kohteeseen saadaan valmiiksi laskettuna reittipalvelulta. Tulosta on kuitenkin syytä hiukan trimmata, koska palvelu palauttaa sen 14 desimaalin tarkkuudella. Tähän tehtävään on hyvä käyttää matematiikka kirjaston pyöristys funktiota (Round). Riittää, kun määritellään pyöristettävä arvo ja halutut desimaalit. Viimeisenä tulos käännetään teksti muotoon (String) ja tulostetaan käyttäjälle nähtäväksi (Kuva 10.).

```
double distance = Math.Round(e.Result.Result.Summary.Distance,
1);
this.DistanceGrid.Visibility = Visibility.Visible;
this.DistanceTextBlock.Text = Convert.ToString(distance) + "
Km";
```



Kuva 10. Reitti käyttäjän sijainnista valitulle radalle

6 JATKOKEHITYS

Puhelinsovelluksen ja varsinkin paikannuspalveluiden jatkokehityksen kannalta merkittävimpänä asiana voidaan pitää Windows Phone 8 käyttöjärjestelmän mukana tullut mahdollisuus käyttää Nokian kehittämiä Here-karttapalveluita. Varsinkin Here-karttapalvelun mahdollistama Here Drive-navigaattoripalvelun käyttö tuo täydellisen navigoinnin mahdolliseksi käyttäjälle ja näin ollen voidaan luopua omatekoisten navigaatiopalveluiden käytöstä kokonaan. Toinen tärkeä ominaisuus on rajoitusten poistuminen. Nokia Here-karttapalvelu ei rajoita palvelukyselyitä, joten ei tarvitse pelätä käytön ylittämisestä perittäviä maksuja.

Huomion arvoinen seikka on Bing-karttapalvelun vanhentuminen. Bing-karttapalvelu on Windows Phone 8 käyttöjärjestelmässä ”Depregaded” tilassa eli palvelua tuetaan toistaiseksi, mutta sen käytöstä ollaan luopumassa. Windows Phone 8.1 ei enää tue kyseistä palvelua. Siirtymä on kuitenkin toteutettava hiljalleen, sillä Here-karttapalvelu ei puolestaan ole tuettuna edellisessä käyttöjärjestelmän versiossa, Windows Phone 7.1:ssä, jossa on käytettävä Bing-karttapalvelua. On myös mahdollista tehdä erikseen versio sovelluksesta Windows Phone 8.1 käyttöjärjestelmälle.

Muita kehitysmahdollisuuksia voisi olla esimerkiksi kaikkien ratojen paikkatietojen tuominen suoraan kartalle. Tällöin kaikkien ratojen karttapisteet olisivat valmiiksi kartalla, josta käyttäjä voisi helposti tarkastella lähistöllä olevien ratojen sijainteja. Navigointi voisi tapahtua esimerkiksi suoraan napauttamalla karttapistettä, johon halutaan navigoida. Näin voitaisiin kokonaan luopua kontrolli-paneelin käytöstä, koska loitonnuks ja lähennys toimivat myös pelkillä sormieleillä. Lisäksi voitaisiin mahdollistaa, pilvipalvelun avulla, ratojen reaaliaikainen pelaajatilanne eli kuinka ruuhkainen rata tällä hetkellä on ja näyttää se suoraan kartalla radan informaatiotuudessa.

7 YHTEENVETO

Opinnäytetyön tarkoituksena oli tutustua mobiilisovelluskehitykseen ja Windows Phone alustaan. Lisäksi tutustuttiin Microsoftin tarjoamiin työkaluihin ja teknisiin rajapintoihin, joiden avulla luotiin työssä toteutettu sovellus. Työn avulla nähtiin käytännössä kuinka vahva ekosysteemi palvelee hyvin ohjelmistokehittäjää ja mahdollistaa suhteellisen vaivattoman sovelluskehitysprosessin.

Työn osana toteutettu sovellusprojektin karttasivu hyödyntää Microsoftin valmista karttakontrollia, karttarajapintaa ja reittipalvelurajapintaa. Se osoittaa, miten helposti Windows Phone alustalle voidaan luoda hyvinkin monimutkaisia sovelluksia pienellä vaivalla ja käyttämällä pelkästään jo olemassa olevia komponentteja. Lisäksi Microsoftin valmiit sivupohjat ja tyylidokumentit helpottavat kehittäjää sovellusten teossa muodostamaan käyttäjäystävällisiä sovelluksia.

Sovellus ja kartta-sivu saatiin toteutettua projektin aikana suhteellisen vaivattomasti halutulla tavalla. Hankkeen onnistuminen ja Microsoftin tarjoama ilmainen rekisteröityminen Windows Phone Dev Center palveluun, sai meidät julkaisemaan sovelluksen Windows Phone Store palvelussa. Haasteita julkaisemisessa toi Microsoftin sertifiointivaatimusten tulkitseminen ja pitkäksi venyneet tarkastusajat, jotka olivat pahimmillaan yli 2 viikkoa luvatus 5-7 päivän sijaan. Lopputuloksena kuitenkin syntyi onnistunut sovellus, joka julkaistiin Microsoftin sovelluskaupassa.

LÄHTEET

Amy 2010. A Brief history of Windows Mobile. Viitattu 27.04.2012 <http://no-tebooks.com/2010/04/12/a-brief-history-of-windows-mobile/>.

Bollow, Norton 2013. DotGNU Project. Viitattu 28.10.2013 <http://www.gnu.org/software/dotgnu/>.

Brix, Todd 2013. Windows Phone developer updates & offers from Mobile World Congress. Viitattu 10.11.2013 http://blogs.windows.com/windows_phone/b/wpdev/archive/2013/02/25/windows-phone-developer-updates-amp-offers-from-mobile-world-congress.aspx.

Clayton, Steve 2011. Three Screens and a Cloud. Viitattu 26.05.2013 <http://blogs.msdn.com/b/stevecla01/archive/2009/05/21/three-screens-and-a-cloud.aspx>.

Crossley, Rob 2012. Microsoft email confirms plan to cease XNA support. Viitattu 19.05.2013 <http://www.computerandvideogames.com/389018/microsoft-email-confirms-plan-to-cess-xna-support/>.

Gartner, 2010. Gartner Says Worldwide Mobile Phone Sales Grew 17 Per Cent in First Quarter 2010. Viitattu 07.05.2012 <http://www.gartner.com/newsroom/id/1372013>.

Graziano, Dan 2012. Windows Phone Marketplace tops 100,000 apps. Viitattu 10.11.2013 <http://bgr.com/2012/06/05/windows-phone-marketplace-100000-apps/>.

Hardy, Ed 2010. It's Official: Windows Phone 7 Will Not Run Any Windows Mobile Apps. Viitattu 26.05.2013 <http://www.brighthand.com/default.asp?newsID=16295&news=Microsoft+Windows+Phone+7+Series+Windows+Mobile+7+Software+Backward+Compatible>.

HPC: factor 2001. The History of Windows CE: Windows CE 1. Viitattu 07.5.2010 <http://www.hpcfactor.com/support/windowsce/wce1.asp>.

Järvinen, Jani 2012. Windows Phone sovelluskehitys. Helsinki: Docendo Oy.

Kumar, Prabhu 2012. Nokia and Microsoft Announce Plans for a Broad Strategic Partnership to Build a New Global Mobile Ecosystem. Viitattu 05.05.2012 <http://social.msdn.microsoft.com/Forums/en/windowsmobiledev/thread/1f425c1b-b72a-4203-b594-6672b21ec695>.

Lieberman, Larry 2012. Windows 8 and the Windows Phone SDK, pt. 2. Viitattu 07.05.2012 http://windowsteamblog.com/windows_phone/b/wpdev/archive/2012/04/05/windows-8-and-the-windows-phone-sdk-pt-2.aspx.

Mariani, Rico 2009a. My History of Visual Studio (Part 5). Viitattu 28.10.2013 <http://blogs.msdn.com/b/ricom/archive/2009/10/10/my-history-of-visual-studio-part-5.aspx>.

Mariani, Rico 2009b. My History of Visual Studio (Part 6). Viitattu 28.10.2013 <http://blogs.msdn.com/b/ricom/archive/2009/10/12/my-history-of-visual-studio-part-6.aspx>.

Microsoft 2011. Nokia and Microsoft Announce Plans for a Broad Strategic Partnership to Build a New Global Mobile Ecosystem. Viitattu 07.05.2012 <http://www.microsoft.com/en-us/news/press/2011/feb11/02-11partnership.aspx>.

Microsoft 2012a. Windows Mobile Developer Center. Viitattu 07.05.2012 <http://msdn.microsoft.com/en-us/windowsmobile/bb264318>.

Microsoft 2012b. WINDOWS PHONE STORE APPLICATION PROVIDER AGREEMENT. Viitattu 10.11.2013 <http://cmsresources.windowsphone.com/devcenter/en-us/legal/Windows-Phone-Store-Application-Provider-Agreement.pdf>.

Microsoft 2012c. The MVVM Pattern. Viitattu 13.05.2012 <http://msdn.microsoft.com/en-us/library/hh848246.aspx>.

Microsoft 2012d. Xna game studio 4.0 available for download. Viitattu 19.05.2013 <http://blogs.msdn.com/b/xna/archive/2010/09/16/xna-game-studio-4-0-available-for-download.aspx>.

Microsoft 2012e. Quick tour to of the Integratet Development Environment. Viitattu 19.05.2013 [http://msdn.microsoft.com/en-us/library/ms165088\(v=vs.100\).aspx](http://msdn.microsoft.com/en-us/library/ms165088(v=vs.100).aspx).

Microsoft 2012f. Visual Studio 2010 product highlights. Viitattu 19.5.2013 [http://msdn.microsoft.com/en-us/library/dd547188\(v=vs.100\).aspx](http://msdn.microsoft.com/en-us/library/dd547188(v=vs.100).aspx).

Microsoft 2013a. Visual Studio, compare products. Viitattu 19.5.2013 <http://www.microsoft.com/visualstudio/eng/products/compare>.

Microsoft 2013b. Introduction to the C# Language and the .NET Framework. Viitattu 28.10.2013 [http://msdn.microsoft.com/en-us/library/z1zx9t92\(v=vs.90\).aspx](http://msdn.microsoft.com/en-us/library/z1zx9t92(v=vs.90).aspx).

Microsoft 2013c. Visual C#. Viitattu 28.10.2013 <http://msdn.microsoft.com/en-us/library/vstudio/kx37x362.aspx>.

Microsoft 2013d. What is XAML? Viitattu 28.10.2013 <http://msdn.microsoft.com/en-us/library/cc295302.aspx>.

Microsoft 2013e. Smart Device Development. Viitattu 28.10.2013 [http://msdn.microsoft.com/en-us/library/vstudio/sa69he4t\(v=vs.100\).aspx](http://msdn.microsoft.com/en-us/library/vstudio/sa69he4t(v=vs.100).aspx).

Microsoft 2013f. Getting started with Bing Maps. Viitattu 10.11.2013 <http://msdn.microsoft.com/en-us/library/ff428643.aspx>.

Microsoft 2013g. Create a Bing Maps Key. Viitattu 10.11.2013 <http://www.microsoft.com/maps/create-a-bing-maps-key.aspx>

Microsoft 2013h. Developing a Silverlight Application Using Bing Maps SOAP Services. <http://msdn.microsoft.com/en-us/library/cc879136.aspx>.

Microsoft 2013i. Get apps from Windows Phone Store. Viitattu 10.11.2013 <http://www.windowsphone.com/en-us/how-to/wp8/apps/get-apps-from-the-store>.

Microsoft 2013j. In-app product properties. Viitattu 10.11.2013 [http://msdn.microsoft.com/en-us/library/windowsphone/help/jj206721\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/help/jj206721(v=vs.105).aspx).

Microsoft 2013k. Add in-app advertising. Viitattu 10.11.2013 [http://msdn.microsoft.com/library/windowsphone/help/jj206730\(v=vs.105\).aspx](http://msdn.microsoft.com/library/windowsphone/help/jj206730(v=vs.105).aspx)

Microsoft 2013l. Payment policies. Viitattu 10.11.2013 http://help.bingads.microsoft.com/Help.aspx?market=en-US&project=adCenter_Pub_RTW_ss&query-type=topic&query=PUB_CONC_Payment_Policy.htm.

Microsoft 2013m. Registration info. Viitattu 10.11.2013 [http://msdn.microsoft.com/library/windowsphone/help/jj206719\(v=vs.105\).aspx](http://msdn.microsoft.com/library/windowsphone/help/jj206719(v=vs.105).aspx).

Microsoft 2013n. Getting paid for Windows Phone. Viitattu 10.11.2013 [http://msdn.microsoft.com/en-US/library/windowsphone/help/jj206722\(v=vs.105\).aspx](http://msdn.microsoft.com/en-US/library/windowsphone/help/jj206722(v=vs.105).aspx)

Microsoft 2013o. Technical certification requirements for Windows Phone. Viitattu 10.11.2013 [http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh184840\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh184840(v=vs.105).aspx).

Microsoft 2013p. Understanding app and in-app product submission. Viitattu 10.11.2013 [http://msdn.microsoft.com/library/windowsphone/help/jj206729\(v=vs.105\).aspx](http://msdn.microsoft.com/library/windowsphone/help/jj206729(v=vs.105).aspx).

Microsoft 2013q. Company app distribution for Windows Phone. Viitattu 23.11.2013 [http://msdn.microsoft.com/en-us/library/windowsphone/develop/jj206943\(v=vs.105\).aspx](http://msdn.microsoft.com/en-us/library/windowsphone/develop/jj206943(v=vs.105).aspx).

Microsoft 2013r. Windows Intune. Viitattu 23.11.2013 <http://www.microsoft.com/en-us/server-cloud/products/windows-intune/default.aspx#fbid=K6vAUxgvG0B>.

Molen, Brad 2011. Windows Phone 7.5 Mango review. Viitattu 27.09.2011 <http://www.engadget.com/2011/09/27/windows-phone-7-5-mango-review/>.

Montonen, Jarno 2011. Windows Phone 7 sovelluskehitys. Viitattu 12.05.2012 <http://www.mit.jyu.fi/opiskelu/seminaarit/tiesem2011/WP7-sovelluskehitys.html>.

Obrien, Kevin J. 2012. A Fruitful Year for Microsoft and Nokia. Viitattu 07.05.2012 http://www.sci-tech-today.com/story.xhtml?story_id=0020008SU4X2.

Oram, John 2010. Microsoft Launches Catchup with Windows Phone 7. Viitattu 26.12.2013 <http://www.brightsideofnews.com/news/2010/10/12/microsoft-launches-catchup-with-windows-phone-7.aspx>.

Pratap, Kretan 2013. Windows Phone 8 update's third Live Tile column only for full-HD phones: Microsoft. Viitattu 24.10.2013 <http://gadgets.ndtv.com/mobiles/news/windows-phone-8-updates-third-live-tile-column-only-for-full-hd-phones-microsoft-432930>.

Segan, Sascha 2010. First Windows Phone 7 Devices Go On Sale: Full Reviews. Viitattu 26.05.2013 <http://www.pcmag.com/article2/0,2817,2372286,00.asp>.

Saltzman, Mark 2012. What's a Windows Phone Away? Viitattu 07.05.2012 <http://www.connectedmagazine.ca/gear/17869/windows-phone/>.

Thurrott, Paul 2012. Windows Phone 8 Preview. Viitattu 07.05.2012 <http://www.winsuper-site.com/article/windows8/windows-phone-8-preview-142154>.

Warren, Tom 2010. Windows Phone 7 Marketplace hits 777 apps on launch day. Viitattu 10.11.2013 <http://www.neowin.net/news/windows-phone-7-marketplace-hits-777-apps-on-launch-day>.

Warren, Tom 2012a. Windows Phone 8 in detail: new Start Screen, multi-core support, VoIP integration, and NFC. Viitattu 21.11.2013 <http://www.theverge.com/2012/6/20/3096667/windows-phone-8-screenshots-features-nfc-start-screen-dual-core>.

Warren, Tom 2012b. Windows Phone Store named as replacement for Marketplace, includes design and search tweaks. Viitattu 10.11.2013 <http://www.theverge.com/2012/9/12/3321902/windows-phone-store-rebrand-marketplace-features>.

Xamarin 2013. Mono. Viitattu 28.10.2013 http://www.mono-project.com/Main_Page.