Windows 8 Store- ja Windows Phone 8 -sovelluskehitys

C#-ohjelmointikielellä ja XAML-merkintäkielellä

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulun opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Visamäki, syksy 2013

Niko Kuusinen

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

HAMK Visamäki
Tietojenkäsittely
Systeemityö

Tekijä Niko Kuusinen **Vuosi** 2013

Työn nimi Windows 8 Store- ja Windows Phone 8 -sovelluskehitys

TIIVISTELMÄ

Työn toimeksiantajana on Hämeen ammattikorkeakoulun tietojenkäsittelyn koulutusohjelma. Tavoitteena on luoda oppimateriaaliksi Hämeen ammattikorkeakoululle toimivat ristinolla-pelisovellukset Windows 8 sekä Windows Phone 8 -käyttöjärjestelmiin. Pelisovellus on verkon yli pelattava kaksinpeli ja sisältää keskustelumahdollisuuden toisten pelaajien kanssa. Sovellus toteutettiin kolmen hengen projektiryhmässä.

Käytännön osuudessa jokainen ryhmän jäsen teki sovelluskehitystä jaetuilla alueella. Tämä työ on kytköksissä Toni Ilomäen opinnäytetyöhön Windows Phone 8- ja Windows 8 Store -sovellusten koodipohjan jaosta ja Jarno Niemen opinnäytetyöhön, joka käsittelee sovelluksen tiedonsiirron toteutusta *Windows Communication Foundation* -tekniikalla.

Tässä työssä teoriaosuus koostuu uusille Windows 8 ja Windows Phone 8 -alustoille tehtävän sovelluksen C#- ja XAML-ohjelmoinnin perusteista ja sovelluksen julkaisusta sovelluskaupassa. Työssä esitellään myös käyttöjärjestelmiin kuuluvia ominaisuuksia ja rajoituksia.

Materiaalina on ollut saatavilla olevaa Windows 8- ja Windows Phone 8 -ohjelmistokehitystä käsittelevää kirjallisuutta, sekä MSDN:n materiaaleja. Aikaisempaa kokemusta Windows 8- tai Windows Phone 8 -ohjelmoinnista ei ollut. Aikaisempaa .NET-pohjaista ohjelmointikokemusta kuitenkin löytyi.

Toteutettu pelisovellus on opetuskäyttöön toteutettu runko, johon on helppo tehdä jatkokehitystä muun muassa ulkoasuun ja toimintoihin. Sovelluksen toteutuksen aikana huomasi, että kun kehitetään sovellusta kannettavalle laitteelle, tulee huomioida monia pieniä kohtia, joita normaalissa työpöytäsovelluksen kehityksessä ei ole. Sovelluksen julkaisu on tehty Windows-alustoilla yksinkertaiseksi ja vaivattomaksi prosessiksi. Microsoft on panostanut paljon omiin materiaaleihinsa ja MSDN sisältää paljon hyviä artikkeleja ja esimerkkikoodeja.

Avainsanat Windows 8, Windows Phone 8, sovelluskehitys, sovelluskauppa

Sivut 42 s. + liitteet 3 s.

HAMK
UNIVERSITY OF APPLIED SCIENCES

ABSTRACT

HAMK Visamäki
Degree Programme in Business Information Technology
System engineering

Author Niko Kuusinen **Year** 2013

Subject of Bachelor's thesis Windows 8 Store and Windows Phone 8 software development

ABSTRACT

The client in this thesis is HAMK University of Applied Sciences, Degree Programme in Business Information Technology. The goal is to create a HAMK University of Applied Sciences teaching material, a tic-tac-toe game application for Windows 8 and Windows Phone 8 operating systems. The game application runs on the network and is a two-player, and includes the ability to chat with other players. The application was made for three-person groups.

In the practical part, each member of the group made his own thesis on the application. The work is linked to Toni Ilomäki's thesis Windows Phone 8 and Windows 8 Store applications code base distribution and Jarno Niemi's thesis, which deals with the implementation of the application data transfer for Windows Communication Foundation technology.

The theoretical part of the work consists of the basics of new C# and XAML programming on Windows 8 and Windows Phone 8 platforms and the application release in the application store. The thesis also presents operating systems features and limitations.

The material has been Windows 8 and Windows Phone 8 software development literature, as well as the MSDN materials. There was no previous experience of Windows 8 and Windows Phone 8 programming. There is some experience of other NET-based programming.

The implemented game application is only a frame for educational use. It is easy to develop the application further, concerning, among other things, the layout and functions. During the implementation of the application the author noticed that in developing applications for portable devices, a number of small points that do not exist in normal desktop applications must be taken into account. The software release has been made simple and convenient process for Windows platforms. Microsoft has invested a lot in their own materials and MSDN is full of good articles and code samples.

Keywords Windows 8, Windows Phone 8, software development, application store

Pages 42 p. + appendices 3 p.

KÄSITELUETTELO

.NET Framework	Kehitysalusta, jonka Microsoft on kehittänyt Windowsille ja jota käytetään useiden ohjelmointikielien kanssa, muun muassa C# ja Visual Basic. Puhekielessä puhutaan usein vain .NET-sanalla.
Merkintäkieli	Merkintäkieli tai kuvauskieli, on kieli, jolla kuvataan tekstin rakennetta tai esitystapaa metatiedoilla. Merkintäkielissä tekstin looginen rakenne pyritään erottamaan sisällöstä. Tunnetuin ja käytetyin merkintäkieli on HTML.
Store	Sanalla on useita merkityksiä. Se voi tarkoittaa esimerkiksi sovelluskauppaa, kuten Windows 8 Store tai Windows Phone Store. Windows 8 Modern UI -sovelluksista käytetään yleisesti myös nimestystä Store-sovellus.
UI	On lyhenne sanoista *user interface* ja se tarkoittaa käyttöliittymää.
Windows 8	Uusin Microsoftin luoma käyttöjärjestelmä, joka on suunnattu pöytätietokoneille sekä tablet-laitteille ja kuuluu Microsoftin Windows-käyttöjärjestelmäsarjaan.
Windows Phone 8	Uusin Microsoftin kehittämä mobiililaitteisiin suunnattu käyttöjärjestelmä.
WCF	Lyhenne koostuu sanoista *Windows Communication Foundation*. WCF on Microsoftin ohjelmistokomponenttikirjasto / ohjelmistorajapinta ja -viitekehys, jonka avulla voi rakentaa verkkototeutuksia sovelluksille.
Visual Studio 2012	Microsoftin ohjelmistokehitystyökalu, jota käytetään ohjelmistokehityksessä kaikilla Microsoftin luomilla alustoilla, kuten .NET Framework ja Silverlight. Visual Studiosta käytetään usein lyhennettä VS tai VS2012.

SISÄLLYS

1 JOHDANTO

Tarkoitus on luoda opetuskäyttöön materiaali Windows 8- ja Windows Phone 8 -käyttöjärjestelmille tehtävästä C#- ja XAML-pohjaisesta ohjelmoinnista, jota voitaisiin Hämeen ammattikorkeakoulussa käyttää tukena opetuksessa. Materiaali koostuu sovellusrungosta ja kolmesta opinnäytetyöstä. Oma opinnäytetyöni on vahvasti kytköksissä Toni Ilomäen ja Jarno Niemen opinnäytetöihin. Työ koostuu kokonaisuudesta, johon kuuluvat Windows 8- ja Windows Phone 8 -käyttöjärjestelmille luodut ristinolla-pelisovellukset, jotka sisältävät keskustelumahdollisuuden. Pelisovellus toimii verkon yli. Sovellusten välillä tapahtuva tiedonsiirto toteutetaan WCF-tekniikalla (*Windows Communication Foundation*). Tarkoitus on, että kyseistä peliä ja sen ohjelmistokoodia voidaan käyttää tietojenkäsittelyn koulutusohjelman opetusmateriaalina tulevaisuudessa. Tavoitteisiin omalta kohdalta kuuluu tietenkin oppia kehittämään Windows 8 Store- sekä Windows Phone 8 -sovelluksia. Tavoitteena on myös saada hyvä ja selkokielinen opinnäytetyödokumentti, jossa on perusteet ohjelmistokehityksestä kyseessä oleville alustoille.

Ristinolla-sovelluksessa tiedonsiirron toteutus on Jarno Niemen vastuulla, Phone toteutus Toni Ilomäen ja itse vastaan Windows 8 Store -sovelluksen toteutuksesta. Ilomäen työn teoriaosuudessa käsitellään saman ohjelmistokoodin jakamista molemmilla alustoilla. Niemen teoriaosuus koostuu WCF-tiedonsiirrosta sekä Microsoftin Azure-pilvipalvelun käytöstä. Oman työni teoriaosuus sisältää esittelyn uusista Windows 8- ja Windows Phone 8 -käyttöjärjestelmistä sekä perusteet ohjelmistokehityksestä käyttöjärjestelmille. Ohjelmointi suoritetaan XAML-merkintäkielellä ja .NET C#-ohjelmointikielellä.

Aiheen valintaan johti kiinnostukseni Microsoft .NET-ohjelmointia kohtaan. Itsellä oli suuri kiinnostus tutustua Microsoftin luomiin uusiin alustoihin Windows 8 ja Windows Phone 8. Aihe on varsin ajankohtainen, koska molemmat alustat ovat hyvin tuoreita ja oman näkemyksen mukaan myös nousevassa trendissä.

Windows Phone -käyttöjärjestelmän osalta aihe on rajattu niin, että se koskee vain versiota 8 ja aikaisempiin versioihin tai niiden ominaisuuksiin ei oteta kantaa. Sovelluksen julkaisuprosessi esitetään vain toisen käyttöjärjestelmän osalta, koska se on hyvin samantapainen molemmilla alustoilla. Sovelluksia voi kyseessä oleville alustoille kehittää useilla ohjelmointikielillä, työssä keskitytään tarkemmin vain C#- ja XAML-yhdistelmään. C#-ohjelmointikielen käsittely jätetään sivummalle, koska työn on tarkoitus olla oppimateriaalina oppilaille, joilta jo löytyy perustaidot C#-ohjelmoinnista. Myöskään XAML-kieltä ei käydä syvällisesti läpi, vaan otetaan vain perusteet esille. Työssä käydään läpi asiat, jotka ovat oppilaitokselle kehitetyn sovelluksen kannalta oleelliset ymmärtää. Tämän työn ulkopuolelle jätetään myös sovellusten tiedonsiirto, joka esitellään Jarno Niemen opinnäytetyössä. Myös saman koodin hyödyntäminen molemmilla alustoilla rajataan pois, koska se tulee Toni Ilomäen opinnäytetyössä. (Sovellukset pystyvät hyödyntämään todella paljon samaa koodia.)

Työn tilaajana on Hämeen ammattikorkeakoulu ja tietojenkäsittelyn koulutusohjelma. Varsinaisesti opinnäytetyö ei ole suoraan kytköksissä työelämään, kuten monet yrityksille tehdyt opinnäytetyöt, mutta toisaalta taas, jos julkaisee Store-sovelluksen Windows 8- tai Phone 8 -alustalle, niin matka siitä oman toiminimen tai yrityksen perustamiseen ei ole kovin suuri.

2 WINDOWS 8 JA WINDOWS PHONE 8

2.1 Uusi Windows 8 ja uudistettu käyttöliittymä

Microsoft julkaisi virallisesti uuden Windows 8 -käyttöjärjestelmän lokakuussa 2012. Windows 8:n mukana tuli uusi ulkoasu, joka nimettiin aluksi nimellä Metro UI, mutta nimen oikeuksien (saksalaisen lehden) takia siitä luovuttiin ja nyt nimi on virallisesti vain Windows 8 UI tai Modern UI. Metro-nimen poistuminen on hieman aiheuttanut sekaannusta ja nyt saakin olla hyvin tarkkana välttääkseen sekaannukset, kun puhutaan Windows-käyttöjärjestelmistä. Windows 8 sisältää periaatteessa kaksi käyttöliittymää, koska niistä löytyy perinteinen työpöytänäkymä, sekä Modern UI -käyttöliittymä. Windowsista on tarjolla neljä eri versiota. Perusversio Windows 8, Windows 8 Pro, joka sisältää hieman enemmän ominaisuuksia (kuten Hyper-V virtualisointimahdollisuuden) ja vastaa ominaisuuksiltaan Windows 7 Professional- ja Ultimate-versioita. Windows 8 Enterprise vastaa sisällöltään Pro versiota, mutta sisältää vielä hieman lisää yrityksille suunnattuja ominaisuuksia. Windows RT -versio on tarkoitettu kannettaville vähävirtaiseen ARM-prosessoriarkkitehtuurin perustuville tablet-laitteille. RT-versiossa on käytössä vain Storen kautta ladattavissa olevat sovellukset. Perinteisiä työpöytäsovelluksia ei siis RT-versioon pysty asentamaan, jos niistä ei ole julkaistu versiota sovelluskaupassa.

Microsoftin tavoitteena on ollut luoda yhtenäinen käyttöjärjestelmä, joka toimii puhelimissa, tablet-laitteissa sekä perinteisissä pöytätietokoneissa. Windows 8 ja Windows Phone 8 ovat jo hyvin lähellä toisiaan. Windows 8 -käyttöjärjestelmissä kirjautumisessa voi käyttää Microsoft-tiliä tai Windows 7 -tyylistä perinteistä kirjautumista. Microsoft-tili on uusi yhtenäinen nimitys aikaisemmille Live ID-, Hotmail- ja Outlook-tileille. Microsoft-tilillä kirjauduttaessa etuna on, että voi käyttää Microsoftin synkronointipalvelua. Sen avulla käyttäjä saa pidettyä useita asetuksia ja ominaisuuksia mukana, vaikka käyttäjä vaihtaisikin tietokoneelta toiselle. Synkronointi onnistuu rajallisin mahdollisuuksin myös Xbox- ja Windows Phone -laitteiden kanssa. Suunta on kuitenkin se, että laitteita pyritään tuomaan lähemmäksi toisiaan ja tulevaisuudessa synkronointimahdollisuudet luultavasti laajenevat.

Windows 8:n käyttöliittymäsuunnittelussa on kosketusnäytöt asetettu selvästi etusijalle. Tämä on ollut varmasti viisas päätös, koska aikaisemmissa Windows-versiossa kosketusnäyttöjen hyödyntäminen on ollut varsin vaatimatonta. Kosketusnäytöt ovat kuitenkin vahvasti nykypäivää. Microsoftilla on ymmärretty, että aikaisemmat Windows-versiot eivät olisi enää

sellaisenaan soveltuneet lainkaan ilman näppäimistöä ja hiirtä toimiville tablet-laitteille.

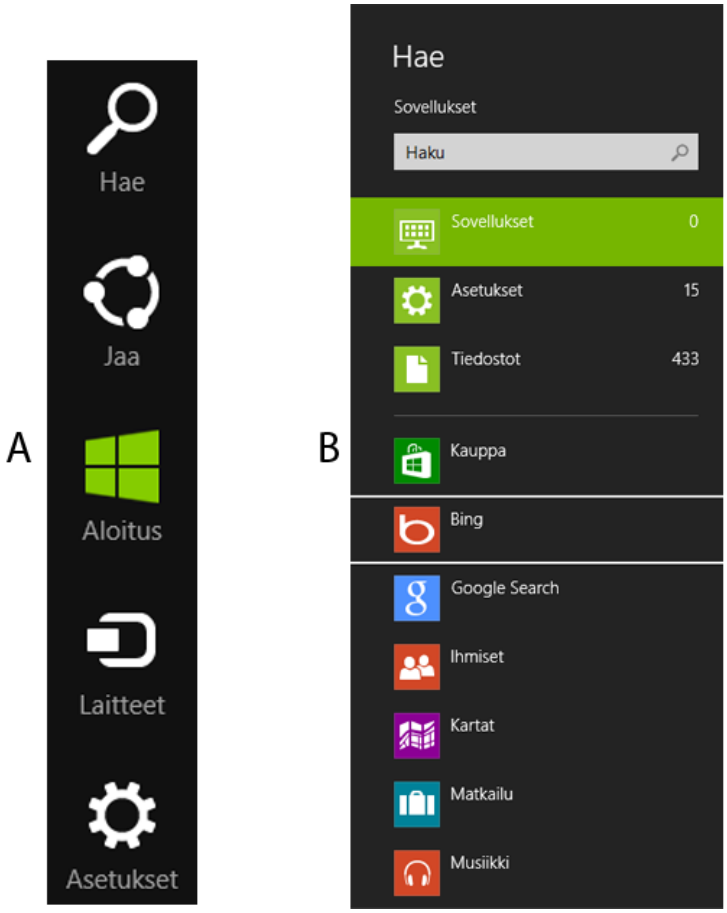
Uusittu käyttöliittymä eli Modern UI perustuu suorakulmion muotoisiin laatikkoihin, joita on Suomessa kutsuttu Live-tiiliksi tai tapahtumaruuduiksi. Ohjelmat ja toiminnot on myös pyritty käyttöliittymässä pitämään hyvin yksinkertaisina, pelkistettyinä ja selkeinä. Tällä on pyritty siihen, että käyttö on helppoa ja mutkatonta. Sovellukset ovat aina koko näytöllä, pois lukien käyttäessäsi toista sovellusta niin sanotusta Snipped-tilassa. Modern UI -käyttöliittymästä siis puuttuvat perinteiset Windowsin ikkunat, joissa sovellukset olivat. Työpöytäpuolella uuden käyttöliittymän tehtäväksi jää lähinnä poistuneen Käynnistä-valikon korvaaminen. Perinteisen Käynnistä-valikon pystyy palauttamaan erilaisilla sovelluksilla, mutta en suosittele tekemään tätä ainakaan heti, koska Modern UI -valikko on todellisuudessa toimiva ratkaisu, joka kuitenkin vaatii hieman totuttelua.

Windows 8 -käyttöliittymästä voidaan käyttää myös termiä Natural UI. Natural UI:lla tarkoitetaan käyttöliittymän tyyliä, joka toimii ilman hiirtä ja näppäimistöä. Natural UI sanaa käytetään siis yleisesti kaikista käyttöliittymistä, jotka toimivat esimerkiksi kosketuksella, kädenliikkeillä tai äänellä, esimerkiksi Xbox Kinect.

Kuva 1. Modern UI -käyttöliittymä

Kun hiiren vie vasempaan näytön reunaan näyttää käyttöliittymä taustalla päällä olevat sovellukset ja käyttäjän on helppo palata aikaisempiin sovelluksiinsa. Uudistettuun käyttöliittymään kuuluu myös Charm Bar, joka aukeaa viemällä hiiri näytön oikeanpuoleiseen ylä- tai alakulmaan tai painamalla Win + C näppäinyhdistelmää. Tällöin näytön oikeaan reunaan aukeaa valikko, josta löytyvät hae, jaa, aloitus, laitteet ja asetukset. Charm Bar -valikosta käytetään suomeksi nimitystä Oikopolut. Nimi Oikopolut viittaakin tähän, että valikosta löytyy nopea kulku useisiin hyödyllisiin ominaisuuksiin. Charm Barin tuomiin etuihin voidaan lisätä se, että esimerkiksi Store-sovellusten asetukset löytyvät nyt aina samasta paikasta, eikä niitä tarvitse erikseen hakea ohjelman valikoista.

Charm Barin hakuominaisuus on myös hyvin kätevä, pystyy helposti hakemaan samalla hakusanalla useista lähteistä, esimerkiksi internethakupalvelusta, Store-sovelluksista, musiikeista ja asetuksista. Hakukohdetta voi vaihtaa valitsemalla halutun hakukohteen listasta.



Kuva 2. A) Windows 8 Charm Bar- eli Oikopolut-valikko. B) Charm Bar -haku onnistuu useista lähteistä. Hakukohteen voi vaihtaa selaamalla luetteloa.

2.2 Uusi Windows Phone 8

Windows Phone -käyttöjärjestelmät ovat älypuhelimille suunnattu käyttöjärjestelmäperhe. Käyttöjärjestelmää ei ole lukittu yhteen valmistajaan, vaan se on kaikkien valmistajien saatavilla, jotka tekevät sopimuksen Microsoftin kanssa. Tällä hetkellä Windows Phone käyttöjärjestelmällä varustettuja puhelimia valmistavat Nokia, HTC, Samsung, LG sekä DELL. Windows Phone 7 tuotiin markkinoille loppuvuodesta 2010, se tuli korvaamaan Microsoftin heikosti menestynyttä Windows Mobile -alustaa. Phone 7 oli ensimmäinen Microsoftin käyttöjärjestelmä, joka perustui Metro-suunnittelukehykseen eli nykyiseen Modern-suunnittelukehykseen. Kesäkuussa 2012 julkaistiin uusi versio, Phone 8.

Phone 8 on ensimmäinen Microsoftin mobiilikäyttöjärjestelmä, joka käyttää Windows NT -käyttöjärjestelmäydintä (kernel), joka on sama kuin Windows 8 -käyttöjärjestelmässä. Käyttöjärjestelmäytimen vaihdoksen myötä Windows Phone 8 toi mukanaan useita uusia laitteistovaatimuksia ja samalla tietenkin useita uusia mahdollisuuksia ja ominaisuuksia. Omi-

naisuuksia on muun muassa tuki moniydinsuorittimille ja 1280 x 720 sekä 1280 x 768 resoluutiolle. Phone 8 mukana tuli myös paremmat mahdollisuudet suorittaa sovellusten tausta-ajoa. (Windows Phone 8 2013; Whitechapel & McKenna 2013, 13–14). Muutoksista johtuen sovellukset Phone 7- ja Phone 8 -käyttöjärjestelmien välillä eivät ole kaikissa tapauksissa suoraan yhteensopivia.

2.3 Windows 8 ja Phone 8 Store - sovelluskauppa

Windows 8:n myötä tuli Store-kauppa myös Windows käyttöjärjestelmille, kaupasta käyttäjät voivat ladata niin ilmaisia kuin maksullisiakin sovelluksia. Sovelluskaupan käyttämiseen vaaditaan Microsoft-tili, Phone- ja Windows 8 RT -käyttöjärjestelmän sisältö jääkin hyvin kapeaksi ilman Microsoft-tiliä. Windows 8- ja Windows Phone -käyttöjärjestelmillä on molemmilla omat sovelluskauppansa. Windows 8 -kauppaan voi kehittää myös perinteisiä työpöytäkäyttöliittymän sovelluksia, mutta niiden osuus kaupan sovelluksista oli murto-osa ainakin työn kirjoitushetkellä. Microsoftin Store -sovelluskaupat toimivat siis kuten vastaavat Android-sovellusten Google Play -kauppa ja Applen App Store.

Windows Store -sovelluskauppoihin sovelluksia voi kehittää kuka tahansa. Kehitystyökalut on saatavilla ilmaiseksi. Sovelluskaupan kehittäjätili kuitenkin on vuosimaksullinen. Sovelluksen hinnan voi sovelluskehittäjä asettaa itse, maksullisissa sovelluksissa sovelluskauppa ottaa myynnistä provisiota, joka on 30 %, kun sovelluksen myynti on alle 25 000 dollaria ja jos summa on yli, niin 20 %. Pienin summa, jonka Microsoft maksaa sovelluksen voitosta, on 200 euroa. Sovelluksen on siis kerättävä 200 euron edestä ostoja, ennen kuin saa euroakaan sovelluksesta. Maksettavat voitot ovat veronalaista tuloa, joten tässä tapauksessa täytyy hoitaa myös asianmukaiset verolomakkeet.

Päästäkseen sovelluskauppaan, täytyy sovelluksen läpäistä Microsoftin sertifiointi. Sovelluksien sertifiointi on Microsoftin tarkka prosessi. Sovelluksille on asetettu laatuvaatimuksia ja ne esimerkiksi tarkastetaan haittaohjelmien varalta, ennen kuin ne pääsevät sovelluskauppaan. Toiminnolla Microsoft pyrkii varmistamaan sovelluskaupassa olevien sovellusten turvallisuuden ja riittävän laadun.

2.4 Windows 8 Store -sovellusten kehitys

Windows 8 Store -sovellusten kehittämiseen tarvitaan 64-bittinen Windows 8 -käyttöjärjestelmä. Store-kehitystä varten voi käyttää ilmaiseksi saatavilla olevaa Visual Studio 2012 Express for Windows 8 -ohjelmistokehitystyökalua. Jos käytössä on jo Windows 8 ja Visual Studio 2012 Professional tai sitä korkeampi versio, ei Store-kehitystä varten tarvita mitään lisää, vaan tarvittavat ovat jo saatavilla.

Windows Store -sovelluksia voi rakentaa useilla ohjelmointikielillä. Tarjolla on seuraavanlaisia yhdistelmiä: JavaScript ja HTML5, varsinainen ohjelmointi tapahtuu siis JavaScript-kielellä ja HTML-kielellä rakennetaan

ulkoasu. Tämän yhdistelmän avulla pystyy helposti esimerkiksi tuomaan aikaisemmin kehitettyjä JavaScript-pohjaisia pelejä Windows Store -alustalle. C#, Visual Basic tai C++ ja XAML on toinen yhdistelmä, jolla ohjelmointi Windows Store -alustalle onnistuu. Tässä vaihtoehdossa XAML:lla rakennetaan graafinen ulkoasu ja varsinainen ohjelmointi suoritetaan C#-, C++-, tai Visual Basic-kielellä. Lisäksi on mahdollisuus C++ ja DirectX -kokonaisuuden käyttöön. DirectX ja C++ on paras valinta tapauksissa, joissa tarvitaan 2D- ja 3D-grafiikkaa ja laskentatehoa. Tässä työssä keskitytään vain C#- ja XAML-yhdistelmään.

Kehittääksesi Windows 8 Store -sovellusta tarvitset työkalujen lisäksi kehittäjäkäyttöoikeuden. Käyttöoikeus on konekohtainen ja ilman kehittäjäkäyttöoikeutta ei koneella voi ajaa Windows Store -sovellusta, jota Windows-kauppa ei ole sertifioinut. Sertifiointi selitetään tarkemmin sovelluksen julkaisua käsittelevässä luvussa 4. Toiminnolla estetään loppukäyttäjiä asentamasta sovelluksia, joita Microsoft ei ole varmistanut luotettaviksi. Käyttöoikeuden hankkiminen on ilmaista, mutta vaatii Microsoft-tilin. Normaali käyttöoikeus on uudistettava 30 päivän välein. Jos käytössä on Windows-kaupan tili, saa kehittäjäkäyttöoikeuden, joka kestää 90 päivää. Kehittäjäkäyttöoikeuteen ei käytännössä tarvitse kiinnittää huomiota, koska Visual Studio huomauttaa, kun käyttöoikeus vaaditaan tai kun se on vanhentunut ja käyttöoikeuden hankkiminen on hyvin nopea ja helppo toimenpide. (Kehittäjäkäyttöoikeuden hankkiminen 2013.)

Store-sovellukset julkaistaan, jaetaan tai myydään Windows-kaupan kautta. Sovelluksen testausta varten voi tehdä sovelluksesta paketin, jonka voi jakaa haluamallaan tavalla, mutta kuten aikaisemmin totesin, paketin ajaminen vaatii kehittäjäkäyttöoikeuden.

Laitteistovaatimukset Windows 8 -kehityksessä

Windows 8 vaatii resoluutioksi 1024 x 768, mutta on suositeltavaa käyttää ainakin resoluutiota 1366 x 768, jotta pystyy käyttämään uuden Windows 8 -käyttöliittymän kaikkia uusia ominaisuuksia. Keskusmuistin minimi vaatimus on 2 Gt 64-bittisellä Windowsilla. Suorittimelle vaatimuksena on PAE-, NX- ja SSE2-tuki. Kyseiset ominaisuudet sisältyvät kaikkiin nykypäivän pöytätietokoneisiin suunnattuihin suorittimiin, mutta vanhempien mallien osalta kannattaa tarkastaa oman suorittimen sovelluttavuus ennen Windows 8 -asennusta. (Windows 8:n järjestelmävaatimukset 2013.) Tarkemmat laitteistovaatimukset löytyvät osoitteesta: http://windows.microsoft.com/fi-fi/windows-8/system-requirements.

2.5 Windows Phone 8 -sovellusten kehitys

Windows Phone 8 -sovellusten kehitykseen tarvitaan 64-bittinen Windows 8 -käyttöjärjestelmä. Emulaattorin käyttäminen kuitenkin vaatii Windows 8 Pro -version, koska se käyttää Pro-versioon sisällytettyä Hyper-V-ominaisuutta. Phone-kehitykseen tarvitsee myös ilmaisen Windows Phone SDK (*software development kit*) 8.0, joka sisältää Visual Studio Express 2012 for Windows Phone -kehitystyökalun. Kehitystä varten voi myös ladata Visual Studio Express 2012 for Windows Phone -kehitystyökalun,

joka sisältää Phone SKD:n. Jos käytössä on jo Windows 8 Pro ja Visual Studio 2012 Professional tai sitä korkeampi versio, tarvitsee Phone-sovellusten kehitystä varten vielä ladata Phone SDK.

Windows Phone 8 -sovelluksia pystyy rakentamaan muutamalla eri ohjelmointi-merkintäkieli yhdistelmällä. Tarjolla on kuten Windows 8 Store puolellakin C# tai Visual Basic ja XAML sekä C++ ja DirectX -yhdistelmät. Lisäksi pystyt rakentamaan HTML5-sovelluksia, mutta HTML-sovellukset toimivat todellisuudessa XAML-alustan päällä, joka vain käynnistää sovelluksen selaimen. Tässä työssä keskitytään vain C#- ja XAML-yhdistelmään.

Normaalissa Windows Phone -kehityksessä kaikki Phone-sovellukset julkaistaan Phone Store -sovelluskaupan kautta. Sovelluskaupassa voi tehdä myös niin sanotusti kohdennetun julkaisun, jolloin sovellus ei näy Store-kaupassa kaikille käyttäjille, vaan sitä jaetaan linkin avulla (Targeted app distribution 2013). Phone 8 toi mahdollisuuden käyttää myös muita kanavia sovellusten jakamiseen, esimerkiksi yritykset pystyvät jakamaan sovelluksia työntekijöilleen myös internetin, intranetin, sähköpostin tai microSD-muistikortin avulla (Whitechapel & McKenna 2013, 18). Phone-sovelluksesta voi myös julkaista beta-testaus version, jonka voi jakaa haluamilleen henkilöille testausta varten. Phone beta-testaus on rajoitettu 90 päivän mittaiseksi. (Beta testing your app and in-app products 2013.)

Laitteistovaatimukset Windows Phone 8 -kehityksessä

Kuten Store-sovelluksissa myös Phone vaatii 64-bittisen Windows 8 -version. Sovellusten testauksessa käytettävä Windows Phone 8 -emulaattori toimii Hyper-V:n kautta virtuaalikoneena, joten emulaattorin käyttöön tarvitsee Windows 8:sta Pro-version. (Windows Phone Emulator 2013). Hyper-V tarvitsee toimiakseen tuen Hardware-assisted virtualization-, Second Level Address Translation (SLAT)- ja Hardware-based Data Execution Prevention (DEP) -ominaisuuksille. Toinen vaihtoehto on käyttää testauksessa Windows Phone 8 -laitetta, tämä vaatii kuitenkin puhelimen niin sanotun lukituksen poistamisen, johon tarvitsee Phone-sovelluskaupan kehittäjätilin. Omassa sovelluskehitystyössä käytössä oli Nokia Lumia 920-, 720- ja 620-puhelimet.

SLAT-tuki puuttuu useista vanhemmista suorittimista. Esimerkiksi Hämeen ammattikorkeakoululla ei ollut opinnäytetyön rakentamisvaiheessa SLAT-tuella varustettuja koneita kuin kourallinen. Intelin prosessoreissa SLAT-tuki löytyy i-sarjan prosessoreista, AMD:llä ensimmäiset SLAT-tuella varustelut suorittimet kuuluvat Opteron-sarjaan, mutta AMD:llä ei ole yhtä selvää rajaa, mistä suorittimista tuki löytyy. Oman suorittimen tuen voi tarkastaa Coreinfo-nimisellä sovelluksella, joka löytyy osoitteesta: http://technet.microsoft.com/en-us/sysinternals/cc835722.aspx. Jos tilanne on se, että ei pysty käyttämään Phone-emulaattoria tai laitetta testauksessa, on sovelluksen kehitys mahdotonta.

3 OHJELMISTOKEHITYS WINDOWS 8

Kuten lähes aina, aikaisemmasta sovelluskehityskokemuksesta on hyötyä ja Windows 8 -alustojen kanssa varsinkin aikaisempi WPF-kokemus tai muun XAML-pohjaisen alustan käyttö helpottaa Windows 8 Store- ja Windows Phone 8 -sovelluksen kehityksen aloittamista, koska tällöin hallussa on jo periaatteessa tietotaito rakentaa sovelluksen käyttöliittymä. Aikaisempi kokemus auttaa tietenkin myös Store-alustan JavaScript ja HTML5 yhdistelmän kanssa. Kokemusta näistä löytyy monilta, koska ne ovat kieliä, jotka ovat käytössä useilla muillakin alustoilla. Luku 3 sisältään lyhyen opastuksen XAML-kielen maailmaan ja esittää tärkeimmät kohdat Windows 8- ja Windows Phone 8 -sovelluskehityksessä.

3.1 Mitä ovat C# ja XAML?

3.1.1 C#

Microsoftin .NET-alusta ja siihen liittyvä C#-ohjelmointikieli julkaistiin virallisesti vuonna 2002. Julkaisunsa jälkeen C# nousi nopeasti tärkeäksi osaksi Windows-alustojen ohjelmistokehitystä. (Troelsen 2012, 3.) C#-kieli on yksinkertainen, moderni ja oliopohjainen kieli. C#-ohjelmointikielen syntaksi vastaa hyvin paljon Java-ohjelmointikielen syntaksia (Troelsen 2012, 5–6). Kielen kehityksestä vastasi Anders Hejlsberg, nykyinen versio on C# 5.0, joka julkaistiin .NET Framework 4.5 yhteydessä elokuussa 2012.

3.1.2 XAML

XAML on Microsoftin kehittämä XML-pohjainen merkintäkieli, jota käytetään .NET Framework -tuotteissa. Lyhenne XAML tulee sanoista *Extensible Application Markup Language*. XAML julkaistiin lokakuussa 2006. XAML on tekniikka, jota voidaan käyttää useissa eri käyttökohteissa, mutta alun perin se on suunnattu WPF (*Windows Presentation Foundation*) käyttöliittymien rakentamiseen eli sovelluksen graafisen näkymän toteutukseen. Nykypäivänä XAML:lilla on tärkeä osuus myös Windows Phone- ja Store -sovellusten rakennuksessa. XAML:lia voi käyttää tällä hetkellä WPF:n, Phonen ja Storen lisäksi Silverlight- ja Windows Workflow Foundation -sovelluksissa. XAML on esitetty tässä kappaleessa pääpiirteittäin, XAML on itsessään jo niin laaja-alainen käsite, että siitä pystyisi kirjoittamaan jo yksinään opinnäytetyön.

WPF on nykyaikainen graafisen käyttöliittymän rakennustyökalu Windowsissa, se eroaa suuresti aikaisemmista teknologioista, kuten Windows Formeista. WPF:än suurimmat edut on sisäänrakennettu laitteistokiihdytys sekä resoluutioriippumattomuus, joka parantaa sovellusten skaalautuvuutta. Voidaankin sanoa, että WPF on paras työkalu rakentaa tyylikäs työpöytäsovellus Windows Vistalle, 7 tai 8. Huomaa kuitenkin, että Windows Store -sovelluksissa ei käytetä WPF:ää vaan Store-sovellusten omaa XAML-kirjastoa, joka vastaa kuitenkin monelta osin WPF:ää. (Mac-

Donald 2012, 3.) Myös Phone-sovelluksilla on oma XAML-kirjasto. XAML-syntaksi on luettavissa ja kirjoitettavissa ilman työkaluja, laajoissa kokonaisuuksissa se on usein kuitenkin työlästä. Yksi XAML:lin suurimmista eduista on se, että saadaan koodi ja graafinen näkymä erotettua toisistaan, joka on tehokkain tapa käsitellä varsinkin graafisesti monimutkaisia sovelluksia. Tällä mahdollistetaan myös se, että graafisen puolen ja ohjelmistollisen osuuden voivat helposti tehdä eri henkilöt ja sovelluksen molempia osia voidaan suunnitella ja kehittää erikseen. (MacDonald 2012, 21.) XAML-merkintäkielen kanssa sovelluksen graafisen ja ohjelmallisen osuuden jaossa hyödynnetään usein MVVM-suunnittelumallia, joka on esitelty luvussa 3.4.6.

Visual Studio 2012 sisältää editorin ohjelmistokehitykseen sekä käyttöliittymäsuunnitteluun, jonka avulla elementtien sijoittelu onnistuu helposti raahaa-pudota-menetelmällä. Ohjelmistokehityksessä käytetään perinteisesti Visual Studiota ja käyttöliittymänsuunnitteluun Blend for Visual Studio 2012 -editoria. Perinteisestä ohjelmistokehitykseen suunnatusta Visual Studio -editorista löytyy perustyökalut käyttöliittymän rakentamiseen, mutta jos on tarvetta tyylikkäämmälle tai monimutkaisemmalle ulkoasulle, on suositeltavaa käyttää Blend-versiota. Aikaisemmin Microsoft julkaisi erikseen Expression Blend -editoria, joka oli tarkoitettu ulkoasujen suunnitteluun, mutta nyt siitä on luovuttu ja uusimassa versiossa, Visual Studio 2012:sta, Blend on osa Visual Studiota. (Microsoft Expression Changes 2013.) Käyttöliittymien graafisesta kehityksestä kiinnostuneiden kannattaa käydä design.windows.com osoitteessa.

XAML-syntaksi on hyvin samankaltaista kuin kaikissa muissakin XML-pohjaisissa kielissä. XAML-elementti alkaa aina < -merkillä, jonka jälkeen tulee elementin nimi, esimerkiksi Button. Tämän jälkeen voidaan asettaa erilaisia arvoja ja ominaisuuksia eli attribuutteja elementille, lopulta elementti suljetaan > -merkillä, eli kokonaisuudeksi tulee <Button>. Lopulta elementti suljetaan </Button> -merkinnällä. Sulkemiseen voidaan käyttää myös niin sanottua itsestään sulkevaa muotoa, asettamalla /> -merkit elementin nimen perään. Kuten missä tahansa XML-tiedostossa voi elementin sisälle laittaa toisen elementin. Attribuutteihin asetetaan myös mahdolliset tapahtumat, esimerkiksi mitä tapahtuu, kun painiketta painetaan. Katso esimerkkikoodi (1) alta. Jokainen XAML-elementti käännetään .NET-luokan instanssiksi. Jokaisen elementin takana on siis .NET-luokka, jonka nimi vastaa täysin elementin nimeä. XAML-tiedostolla voi olla vain yksi ylimmän tason elementti, ja kun tämän elementin sulkee, ei sen jälkeen voi olla enää muuta sisältöä. Ylimmän tason elementit vaihtelevat hieman käytettävän alustan mukaan, mutta Windows Store alustalla on esimerkiksi Application, Page ja UserControl. (MacDonald 2012, 21-25.)

```
<Grid>
<Button Content="Click Me" Click="ClickHandlerMethod />
</Grid>
```

Esimerkki 1. XAML-syntaksi, Grid- ja Button-elementit

XAML-koodi kulkee aina niin sanotun parserin eli jäsentelijän läpi, joka etsii .NET-nimiavaruuksista todelliset elementtien luokat, jotka pitää to-

teuttaa. Jäsentelijää tarvitaan myös tulkitsemaan XAML-koodia, koska sitä pystyy rakentamaan muutamalla erilaisella rakenteella. XAML-jäsentelijä tarvitsee tiedon, mihin .NET-nimiavaruuteen luokka kuuluu, jotta se osaa hakea elementtejä oikeasta paikasta. Tieto annetaan jäsentelijälle ylimmän tason elementin attribuutteina. Nimiavaruuksien asettamiset on esitetty alla olevassa esimerkissä (2) kursivoituna.

```
<Page
x:Class="App1.MainPage"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:local="using:App1"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
mc:Ignorable="d">
```

Esimerkki 2. XAML-luokan viittaukset nimiavaruuksiin

XAML:illa rakennetaan käyttöliittymä, mutta saadakseen sovellukseen toiminnollisuuksia täytyy yhdistää tapahtumankäsittelijät ja varsinainen sovelluksen koodi. Yksinkertaisin tapa XAML:lin kanssa on käyttää Code-Behind-luokkaa eli niin sanottua koodin taustaluokkaa. Taustaluokan määritys tapahtuu yksinkertaisesti käyttämällä x:Class="App1.MainPage" määrettä, kuten ylläolevassa esimerkissä (2). Visual Studio helpottaa työtä luomalla kyseisen taustaluokan valmiiksi.

Taustaluokka on oletuksena hyvin tyhjä. Se sisältää vain oletusrakentajan, joka sisältää kutsun InitializeComponent()-metodille ja OnNavigatedTo-metodin. XAML-taustaluokkien sisältö vaihtelee hieman valitun XAML-alustan mukaan, alla on koodiesimerkki Windows Storen taustaluokasta.

```
namespace App1
{
    public sealed partial class MainPage : Page
    {
        public MainPage()
        {
            this.InitializeComponent();
        }

        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
        }
    }
}
```

Esimerkki 3. XAML-taustaluokka

InitializeComponent()-metodi on avainroolissa XAML-sovelluksissa ja sitä ei koskaan tule poistaa. Metodi kutsuu sivun rakentajaa, jolla luodaan itse sivu ja siihen lisätyt muut elementit.

XAML-kielessä elementtien nimeäminen on hyvin yksinkertaista ja nimet asetetaankin elementeille, kuten kaikki muutkin tiedot. Kun elementillä on nimi, se on integroitu taustaluokkaan ja pystyy sitä helposti hallitsemaan koodilla, alla esimerkki (4) jossa painikkeen näkyvyysmääre asetetaan.

XAML-koodi: `<Button Name="Painike"/>`
C#-koodi: `Painike.Visibility = Visibility.Visible;`

Esimerkki 4. XAML-elementin käyttäminen nimellä C#-taustaluokasta

3.2 Erilaiset mahdollisuudet pelikehityksessä

Pelialustaksi on tarjolla useita hyviä vaihtoehtoja, työn suunnitteluvaiheessa oli tarkoitus lähteä Microsoftin XNA-peliohjelmointikirjastolla. XNA oli vaihtoehtona numero yksi, koska siitä oli aikaisempaa kokemusta koko projektiryhmällä. XNA:ta pystyy käyttämään useille Microsoftin omille alustoille, kuten Windows, Xbox 360 ja Windows Phone. Asiaa tarkemmin tutkittua ilmeni kuitenkin hyvin nopeasti, että Microsoft on lopettanut XNA-alustan tukemisen uusilla Windows-käyttöjärjestelmillä. XNA-pelikirjaston käyttämisestä siis luovuttiin.

XNA:sta siirryttiin MonoGame-kirjastoon, joka on avoimen lähdekoodin toteutus Microsoftin XNA:sta. MonoGamen tarkoitus on tuoda XNA-kirjasto saataville kaikille alustoille. MonoGamen ongelmaksi tuli, että toteutus Windows 8 Store -alustalle oli vielä puutteellinen. Tilanne tulee kuitenkin luultavasti muuttumaan pian ja MonoGame on jatkossa varteenotettava vaihtoehto.

Nyt oli jäljellä vielä kolme varteenotettavaa vaihtoehtoa, tarkempi tutkailu kohdistui Microsoftin DirectX-ohjelmointirajapintaan. DirectX suljettiin lopulta pois, koska DirectX-ohjelmointi tapahtuu C++ -ohjelmointikielellä, josta kellään ryhmän jäsenistä ei ollut kokemusta ja täysin uuden ohjelmointikielen haltuun ottaminen olisi ollut työlästä ja aikaa vievää. Lisäksi todettiin, että kun kyseessä on vain ristinolla-peli, ei välttämättä tarvita, eikä ole järkevää käyttää näin kehittynyttä pelirajapintaa tai -kirjastoa, joten päätettiin luopua kokonaan pelialustoista.

HTML5 + JavaScript ja XAML + C# yhdistelmien välillä harkinta oli lopulta lyhyt, kun huomattiin, että JavaScript ei ole suoranaisesti tuettu Windows Phone 8 -alustalle. Työn alkuvaiheessa kartoitettiin vain Store-alustan rakennusvaihtoehdot ja oletettiin, että samat mahdollisuudet ovat tarjolla myös Phonelle. Näin ei kuitenkaan todellisuudessa ollut. Valinta kääntyi siis XAML + C# -yhdistelmän puolelle, koska kaikilla kehitysprojektin ryhmän jäsenillä oli jo aikaisempaa kokemusta huomattavasti enemmän C#-ohjelmointikielestä kuin muista vaihtoehtoisista kielistä, oli tämä luonnollinen valinta. Myös kiinnostus tutustua XAML-merkintäkieleen oli suurempi kuin HTML5-merkintäkieleen. XAML:lin valintaa puolsi myös se, että muu ohjelmarunko tulisi olemaan XAML-pohjaista.

Vaihtoehtoja on siis monia ja muutamia jätettiin kokonaan harkinnan ulkopuolelle, kuten XAML + Visual Basic ja XAML + C++. Lopulta projektiryhmässä päätimme siis luopua pelikirjastojen ja rajapintojen käytöstä, koska totesimme, että suunnitteilla oleva peli niitä ei vaadi. Lisäksi työn aikataulutuksen sekä laajuuden kannalta oli järkevää rajata ne pois työstä.

Pelikehitystä varten on tarjolla myös useita framework-alustoja, joiden avulla pelin tekeminen voisi olla helpompaa ja nopeampaa. Työstä nämä rajattiin ulkopuolelle, koska haluttiin keskittyä varsinaiseen Store- ja Phone-sovellusten luomiseen. Framework-alustoissa tekninen koodi usein sivuutetaan ja keskitytään graafiseen ulkoasuun. Esimerkkejä framework alustoista löytyy osoitteesta http://build.windowsstore.com/using-frameworks. Googlesta voi hakea myös hakusanalla ”*Windows 8 game starter kit*”, sillä löytyy muutamia mielenkiintoisia pelikehitykseen tarkoitettuja pohjia, joita voi hyödyntää vapaasti.

3.3 Windows 8 -alustoille tarjolla olevat SDK:t ja App Toolkit:it

Microsoft tarjoaa kehittäjille useita valmiita ja maksuttomia työkaluja ja SDK:ta. Työkalujen avulla saa otettua käyttöön monia käyttöjärjestelmiin sisältyviä ominaisuuksia. Työssä ei kuitenkaan perehdytä niiden toimintaan tarkemmin, vaan vain esitellään, mitä on tarjolla. SDK on lyhenne sanoista *Software development kit* ja se tarjoaa yleensä rajapinnat ja ohjausobjektit jonkin komponentin käyttöön. Suomen kielessä SDK:sta käytetään yleensä nimitystä ohjelmankehityspaketti.

Saatavilla molemmille alustoille on Live SDK, joka sisältää valmiita rajapintoja ja ohjausobjekteja, joiden avulla sovelluksen voi esimerkiksi integroida Microsoft-tiliin. Tämän SDK:n avulla voi siis luoda helposti sovellukseesi esimerkiksi kirjautumisen, joka käyttää Microsoft-tiliä.

Multilingual App Toolkit for Visual Studio 2012 -työkalu tarjoaa mahdollisuuden lokalisoida sovellus, lisätä käännöstuki, käännöstiedostojen hallinta. Eli se sisältää kaikki tarpeelliset työkalut tehdä sovelluksesta kansainvälinen ja monikielinen.

Windows Azure SDK (for Windows 8 and Windows Phone 8) tarjoaa mahdollisuuden käyttää Azure-mobiilipalveluratkaisua. Azuren mobiilipalvelu tarjoaa mahdollisuuden tallentaa pienen määrän tietoa sovelluksesta pilvipalveluun.

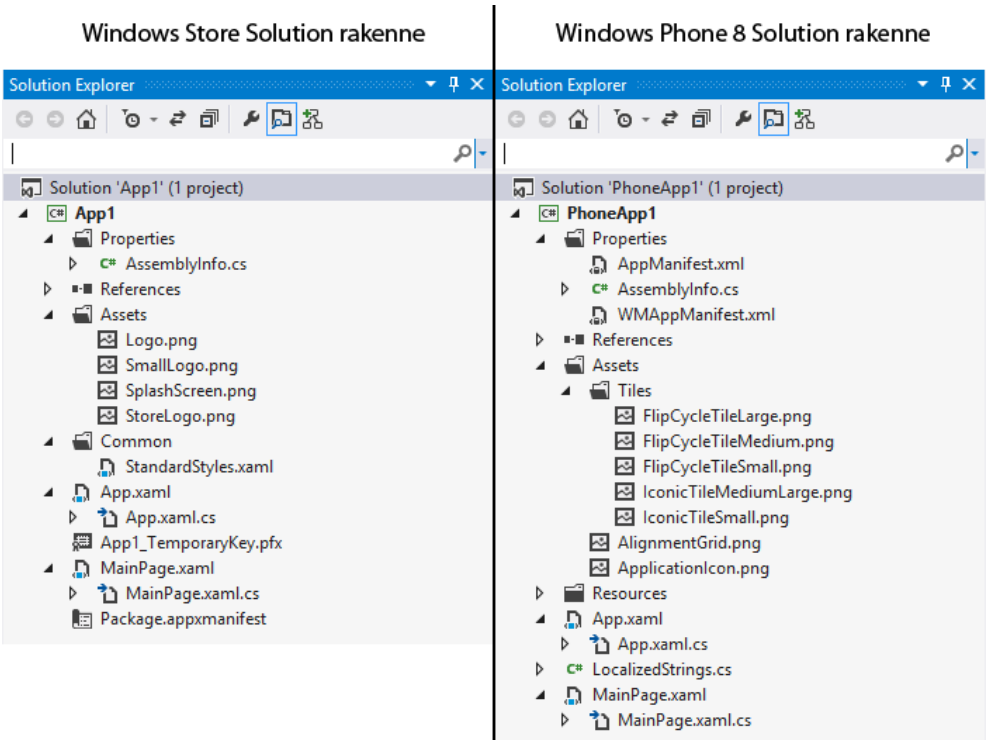
Store-sovelluksille on tarjolla myös Bing Maps SDK ja Ads in Apps SDK. Maps tarjoaa tarvittavat rajapinnat ja ohjausobjektit Bing-karttojen käyttämiseen osana sovellusta. Ads in Apps SDK:lla pystyt helposti sijoittamaan ja käyttämään Microsoftin mainossisältöä, jonka avulla voit tienata sovelluksellasi pientä korvausta sen tekemisestä, vaikka olisit asettanut sovelluksesi maksuttomaksi sovelluskauppaan.

3.4 Store- ja Phone-sovellusten yhteiset ominaisuudet

Store- ja Phone-käyttöjärjestelmien perustuessa samaan järjestelmäytimeen, ei ole yllättävää, että käyttöjärjestelmillä on paljon yhteisiä ominaisuuksia. Tässä luvussa listataan yhtäläisyydet ja yhteiset ominaisuudet ja luvuissa 3.5 ja 3.6 poimitaan molempien järjestelmien erityispiirteitä ja ominaisuuksia.

3.4.1 Aloitus - Perusteet

Kappaleessa käsitellään lyhyesti sovelluksiin kuuluvat tiedostot ja niiden tarkoitus. Tutkitaan siis Windows Phone 8- ja Windows 8 Store -sovellusten rakennetta Visual Studiolla. Aloita sovelluksen luominen valitsemalla Visual Studion -valikosta uusi projekti ja Windows Store -valikosta valitse *Blank App (XAML)* eli tyhjä XAML-ohjelmapohja tai Windows Phone -valikosta Windows Phone App. Solution Explorer -valikosta voi huomata, että Visual Studio generoi valmiiksi useita tiedostoja ja kansioita.



Kuva 3. Windows Store- ja Windows Phone -sovelluksen solution oletusrakenne

Windows Store -sovelluksessa Assets-kansiosta löytyy neljä kuvaa, joilla jokaisella on oma tarkoituksensa. Logo.png on kuva, joka näkyy ohjelman kuvakkeena Windows 8 UI -valikoissa. SmallLogo.png kuva näkyy esimerkiksi, kun valitset näytä kaikki sovellukset -toiminnon tai haet sovelluksia Windows 8:n valikossa. SplashScreen.png tulee näkyviin, kun käynnistät sovelluksen ja sovellus on latausvaiheessa. StoreLogo.png on kuva, joka näkyy Store-sovelluskaupassa, kun käyttäjät selaavat sovelluksia. Windows Phone -sovelluksessa kansiosta löytyy ApplicationIcon.png, joka näkyy sovelluksen kuvakkeena sovellusvalikossa. Phone-sovelluksessa on myös Tiles-alikansio, joka sisältää erikokoiset kuvakkeet erilaisiin tilanteisiin tiilien eli tapahtumaruutujen kanssa. (Whitechapel & McKenna 2013, 280–286.) Tarkemmat käyttökohteet Phone-sovelluksen kuvista löytyvät luvusta 3.4.4. Kuvien merkitystä ei kannata vähätellä, sillä jos kuvat on tehty tyylikkäästi, antaa se sovelluksesta ammattimaisen kuvan, varsinkin siinä vaiheessa, kun käyttäjät ovat lataamassa sovellusta sovelluskaupasta.

App.xaml-tiedosto löytyy molemmista alustoista samasta sijainnista, ja se on ensimmäinen tiedosto, joka ladataan, kun sovellus käynnistyy. Tiedosto sisältää kaikki sovellustasolla asetetut viittaukset ja resurssit. Oletuksena Store-sovelluksessa tiedosto sisältää vain viittauksen StandardStyles.xaml-tiedostoon. App.xaml-tiedoston avulla StandardStyles.xaml-tiedoston sisältö on käytettävissä kaikissa sovelluksen osissa tai sivuissa. Phone-puolella on viittaus sovelluksen elinkaaren hallintametodeihin.

App.xaml.cs on niin sanottu *code-behind*-tiedosto eli se on App.xaml-tiedoston taustalla (löytyy Visual Studiossa, kun avaat App.xaml-tiedoston rakenteen nuolella) ja sisältää sen toiminnot ja varsinaisen koodin. Tiedostolla hallitaan sovellusten elinkaarta. Tiedoston sisältä löytyy muun muassa Store-puolella OnLaunced()-metodi ja Phone-puolella Application_Launching()-metodi, joka ajetaan sovelluksen käynnistyessä.

Properties-kansiosta löytyvä Assemblyinfo.cs-tiedosto sisältää tietoja sovelluksesta, kuten versionumeron ja kuvauksen. Tiedosto on kuitenkin vain kehittäjälle näkyvillä ja varsinaiset viralliset nimet ja vastaavat julkiset tiedot löytyvät Store-sovelluksessa Package.appmanifest-tiedostosta ja Phone sovelluksessa WMAppManifest.xml-tiedostosta.

Package.appxmanifest- ja WMAppManifest-tiedostot sisältävät kaikki oleelliset asetukset. Ne ovatkin ehkä sovellusten tärkeimmät tiedostot, sillä niistä hallitaan sovelluksen yleistietoja ja myös esimerkiksi sitä, mitä laitteen ominaisuuksia sovellus tarvitsee, kuten esimerkiksi GPS-paikannus tai kamera. Tiedostojen sisältöön kannattaa tutustua huolella, viimeistään ennen sovelluksen siirtämistä sovelluskauppaan.

Store-sovelluksen Common-kansiosta löytyvä StandardStyles.xaml on lähes 2000 riviä pitkä tiedosto, joka sisältää oletustyylejä, esimerkiksi tekstille, painikkeille ja sovelluksen kuvakkeille. Tiedosto on niin kattava, että normaalisti jokaiselle löytyy tarvitut tyylit, ainakin pienellä muokkauksella ja ylimääräisten tyylitiedostojen lataamiselle ei ole tarvetta. Tiedoston tyylejä voi muokata oman tahtonsa mukaan ja voi myös lisätä omia muotoilujasi. Suurin osa tyyleistä on oletuksena kommentoitu pois käytöstä, mutta poistamalla kommentit tyylin ympäriltä, saa sen otettua käyttöön. Phone-sovelluksissa on oletuksena käytössä vain niin sanottuja kovakoodattuja tyylejä, jotka tulevat käyttöjärjestelmän uumenista. Tarkemmin Phone-käyttöjärjestelmän tyyleistä ja teemoista MSDN: http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff402557%28v=vs.105%29.aspx.

MainPage.xaml on oletuksena sovelluksen aloitussivu. Oletuksena sivu on käytännössä tyhjä ja sisältää vain tyhjän Grid-ruudukon. Grid-ruudukko on XAML:lin peruselementti Store- ja Phone -sovelluksissa. Ruudukon kokoa, sarakkeiden ja rivien määrää säätelemällä muodostetaan käyttöliittymän rakennetta, sisälle voidaan asettaa XAML-elementtejä, myös toinen Grid-ruudukko. Tähän tiedostoon voi siis rakentaa sovellusta käyttämällä Visual Studion Toolbox -valikosta löytyviä elementtejä ja komponentteja.

MainPage.xaml.cs on tiedosto MainPage.xaml:lin taustalla ja kuten MainPage.xaml on sekin oletuksena käytännössä tyhjä, sisältäen vain rakentajan ja OnNavigatedTo()-metodin. OnNavigatedTo() suoritetaan, kun ohjelma avaa MainPage-sivun.

Jokainen Windows Store -sovellus on allekirjoitettu varmenteella. Kun Visual Studiolla luo projektin, luo se samalla testivarmenteen, joka löytyy SovelluksenNimi_TemporaryKey.pfx-tiedostosta. Varmenteita pystyy hallitsemaan Package.appxmanifest-tiedoston kautta, mutta niihin ei yleensä tarvitse kiinnittää huomiota sovelluksen kehitysvaiheessa.

Sovellusta suunnitellessa ja rakentaessa on hyvä huomioida puhelinten ja tablet-laitteiden näytön kääntämisen mahdollisuus. Kyseinen ominaisuus saattaa usein hieman unohtua, kun sovelluskehitystä tehdään laitteilla, jotka eivät ominaisuutta tue, testaus onnistuu kuitenkin helposti Store-simulaattorin ja Phone-emulaattorin avulla. Kääntöominaisuuden käsittely tapahtuu molemmilla alustoilla eri tavalla. Phone-alustalla jokaiselle sivulle määritetään, mitkä ovat sen tuettuja näytön asentoja. Store-sovelluksessa asetetaan taas Package.appxmanifest-tiedostosta koko sovelluksen tuetut näytön asennot. Store-sovelluksessa on oletuksena asetettu, että sovellus tukee kaikkia näytön asentoja. Phone-sovelluksissa taas sivujen asetus on oletuksena *Portrait*, eli vain puhelimen näytön pystysuuntainen käyttö. Jos sallit sovelluksesi toimia kaikissa mahdollisissa näytön asennoissa, kannattaa myös suunnitella ja testata, miten näytät sovelluksen sisällön, esimerkiksi piilotatko jotain, mikä on vaakasuuntaisesti näkyvissä, jos laite käännetään pystysuuntaisesti.

3.4.2 Valmiit pohjat - Templates

Store-sovellukselle Visual Studio 2012 tarjoaa kolme ohjelmapohjaa, joista yksi on jo esitelty Blank, joka on siis nimensä mukaisesti käytännössä tyhjä. Tarjolla on myös Grid ja Split, joissa on esimerkkisovellus, johon on aseteltu valmiiksi esimerkkitietoja. Grid on kolmesivuinen ja ruudukkotyyppinen. Split on kaksisivuinen ja rakenteeltaan listatyyppinen. Suosittelen tutkimaan ainakin toisen toimintaa, koska näistä ohjelmapohjista voi hahmottaa, kuinka esimerkiksi ohjelman sisällä sivulta toiselle siirtyminen tapahtuu ja pohjissa näkyy myös kuinka tyylitiedostoa käytetään. Oman sovelluksen rakentamiseen pohjat eivät välttämättä ole aina sopivimpia, mutta joskus voi pystyä hyödyntämään ainakin jotain osaa omassa sovelluksessa.

Phone-sovellukselle on tarjolla Blank-pohjan lisäksi Databound-, Panorama- ja Pivot-ohjelmapohjat. Databound sisältää perusrungon MVVM-mallista, MVVM-malli on käsitelty kappaleessa 3.4.6. Panorama ja Pivot sisältävät saman MVVM-pohjan kuin Databound, mutta ne sisältävät lisäksi peruspohjan panorama-sovelluksesta ja pivot-sovelluksesta. Panorama-sovelluksessa ohjelman toisen sivun reuna näkyy oikeassa laidassa, pivot-sovelluksessa sivut taas on otsikoitu yläreunaan, sivuja selataan vierittämällä otsikoita. Kannattaa tutustua panorama- tai pivot-sovelluspohjan toimintaan, molemmat pohjat ovat oivallisia apuja oman sovelluksen rakentamisessa.

3.4.3 Sivulta toiselle siirtyminen - Navigation

Siirtymiset sivuilta toiselle tapahtuvat molemmilla alustoilla hieman eri tavalla. Molemmilla se on kuitenkin hyvin yksinkertaista. Pienimuotoista navigointia voi tehdä myös ilman sivulta toiselle siirtymistä käyttämällä elementtien näkyvyysmääreitä, mutta tässä on kuitenkin se huono puoli, että tällöin ei saa Storen tai Phonen takaisin nappeja käyttöön. Store-sovelluksessa nappi tulee näkyviin vasempaan yläkulmaan ja Phone-laitteissa on käytössä fyysinen nappi. Kun Store-sovellukseen lisää toisen sivun ja siirtyy sovelluksessa sinne, ilmestyy takaisin-painike automaattisesti näkyviin.

Store-sovelluksessa navigointi tapahtuu Frame.Navigate-metodilla, joka asetetaan esimerkiksi painikkeeseen. Sivujen välillä voi siirtää tietoa asettamalla sen attribuutiksi Navigate-metodiin. Lähetetty tieto pitää käsitellä vastaanottavassa sivussa. Siirtyessä toiselle sivulle ja sieltä palatessa tiedot kuitenkin katoavat, jos NavigationCacheMode ei ole asetettu päälle. Siirtymisissä voi käyttää myös GoHome-, GoBack- ja GoForward-metodeja. Metodit voi myös yli kirjoittaa ja tehdä niille oman toteutuksen. Esimerkissä (5) on esitetty navigoinnin toiminta.

```
public PageA()
{
this.InitializeComponent();
NavigationCacheMode = NavigationCacheMode.Enabled;
}

protected override void OnNavigatedTo(NavigationEventArgs e)
{
    base.OnNavigatedTo(e);
    string text = e.Parameter as string;
    if (!string.IsNullOrWhiteSpace(text))
    {
        TextBlockValue.Text = text;
    }

    else
    {
        TextBlockValue.Text = "You need to pass a string.";
    }
}

...

private void Button_Click(object sender, RoutedEventArgs e)
{
Frame.Navigate(typeof (PageB), TextBoxValue.Text);
}

private void HomeButton_Click(object sender, RoutedEventArgs e)
{
GoHome(sender, e);
}
```

Esimerkki 5. Navigointi-esimerkki, rakentajassa NavigationCacheMode asetettu päälle, OnNavigatedTo-metodissa on käsitelty saapuva teksti, Button_Click navigoi toiselle sivulle ja lähettää tekstiä mukana. HomeButton_Click navigoi aloitussivulle.

Phone-sovelluksissa navigointiin käytetään NavigationService.Navigate-metodia. Phonessa GoBack- ja GoForward-metodit löytyvät NavigationServicen takaa. Muuten sivulta siirtyminen tapahtuu samaan tapaan, esimerkissä (6) esitetty navigoinnin toiminta. Phonessa tulee muistaa myös, että puhelimissa on fyysinen takaisin-painike. Phonen kanssa ei tarvitse huolehtia tiedon säilymisestä sivulta toiselle selatessa, kuten Store-sovelluksissa, koska se on asetettu automaattisesti päälle.

```
private void Button_Click(object sender, RoutedEventArgs e)
{
NavigationService.Navigate(new Uri("/PageB.xaml", UriKind.Relative));
}

private void Button_Click_Data(object sender, RoutedEventArgs e)
{
NavigationService.Navigate(new Uri("/PageB.xaml?msg="+textBox1.Text,
UriKind.Relative));
}

protected override void OnNavigatedTo(NavigationEventArgs e)
{
base.OnNavigatedTo(e);
string msg = "";
if (NavigationContext.QueryString.TryGetValue("msg", out msg))
textBlock1.Text = msg;
}
```

Esimerkki 6. Phone-navigointi, Button_Click esittää perinteisin siirtymisen sivulta toiselle. Button_Click_Data esittää tiedon siirtämisen sivulta toiselle ja OnNavigated-metodissa käsitellään vastaanotettu tieto.

3.4.4 Live Tiles & Notifications

Tapahtumaruudut eli Metro-tiilet tai nykyiseltä nimeltään Live-tiilet ovat tärkeä osa käyttäjien käyttökokemusta Windows 8- ja Windows Phone 8 -käyttöjärjestelmillä. Live-tiilet ovat sovelluksien käynnistyspikakuvakkeita, mutta ne ovat huomattavasti monipuolisempia kuin perinteinen Windows-pikakuvake. Käyttäjä pystyy asettelemaan tiilet aloitusruutuun haluamaansa järjestykseen ja asettamaan haluamansa koon tarjolla olevista vaihtoehdoista. Windows 8 -käyttöjärjestelmässä tarjolla on kaksi eri kokoa, pieni ja suuri. Windows Phone 8:ssa käyttäjällä on mahdollisuus asettaa kolmenkokoisia tiiliä aloitusruutuunsa. Tiilien kokoa ei voi hallita sovelluksella vaan se on aina käyttäjän päätös, minkä kokoisia tiiliä haluaa käyttää. Tämän vuoksi onkin tärkeää asettaa projektiin kuvat kaikille mahdollisille kokovalinnoille. Phone 8:ssa on tarjolla kokojen lisäksi kolme erilaista tyyliä tiilille. Iconic- eli ikoni-tyyli perustuu Microsoftin valmiiksi kuvittamiin ikonikuviin ja tekstiin, jotka pohjautuvat Modern-käyttöliittymäkehykseen. Cycle on tarkoitettu kuvien esittelyyn ja tiilessä pyörivät tällöin vain valokuvat. Flip-tyylissä on kaksi näkymää, jotka vaihtelevat keskenään, eli etusivu ja takasivu, vaihto tapahtuu joka kuudes sekunti. Käyttöjärjestelmä kuitenkin estää kaikkia sovelluksia päivittämästä tiiliänsä samanaikaisesti, käyttöjärjestelmä asettaa siis pientä ajoitusta aloitusaikoihin. (Whitechapel & McKenna 2013, 280–283.) Windows Store -sovelluksille tyylejä on tarjolla huomattavasti enemmän, niitä ei tässä kannata luetella, koska Microsoftin artikkeli asiasta selventää paljon

enemmän: The tile template catalog (Windows Store apps) http://msdn.microsoft.com/en-us/library/windows/apps/hh761491.aspx.

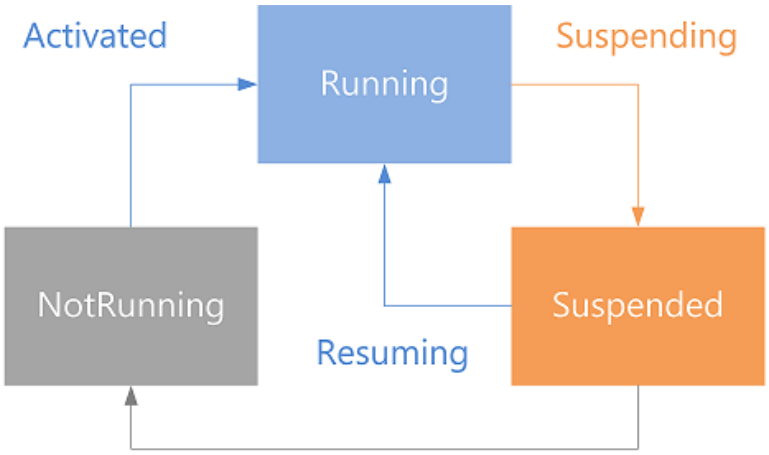
Monipuolisuutta tuo myös mahdollisuus asettaa ohjelmallisesti pakollisen ensisijaisen tiilen lisäksi toissijaisia tiiliä. Ominaisuutta on hyödynnetty esimerkiksi siis valokuva sovelluksessa niin, että sovellus valitsee kansioista satunnaisesti kuvia ja vaihtelee niitä tiileen.

Yksi todella kätevä ominaisuus on, että pystyy päivittämään tiilen sisältöä. Päivitys voidaan tehdä paikallisesti sovelluksesta tai käyttää ulkoista palvelinta ja Push Notification -toimintoa. Ominaisuus on käytössä esimerkiksi sähköposti-ohjelmassa, ohjelman nurkassa on numerona, kuinka monta lukematonta sähköpostiviestiä käyttäjällä on. Päivitys voi tapahtua sovelluksen ollessa päällä tai myös kun sovellus on suljettuna.

3.4.5 Elinkaari - App Lifecycle

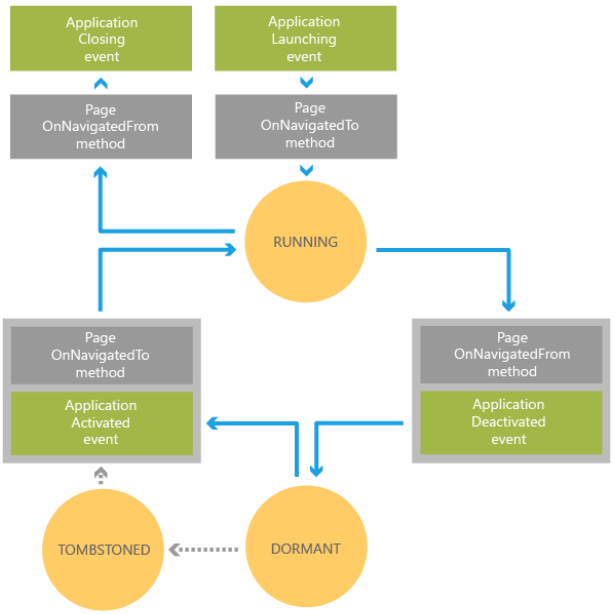
App Lifecycle eli sovelluksen elinkaari, molemmilla käyttöjärjestelmillä on hyvin tarkka elinkaari sovelluksillensa, jota jokainen sovellus noudattaa. Molemmilla käyttöjärjestelmillä on kuitenkin omat nimityksensä sovellusten eri tiloille.

Windows 8:ssa tilat ovat NotRunning, Running ja Suspended. Kaksi ensimmäistä ovat helposti ymmärrettäviä, sovellus on suljettuna ja sovellus on ajossa. Suspended-tila onkin hieman mielenkiintoisempi. Normaalisti, jos käyttäjä vaihtaa muutaman sovelluksen välillä ja palaa takaisin, jatkaa sovellus siitä mihin se jäi, mutta tässä tilanteessa sovellus ei vielä ole asettunut Suspended-tilaan vaan on ollut aktiivisena taustalla. Suspended-tilassa sovellus on asetettu taustalle ja sen tietoja voidaan tallentaa ohjelmallisesti Suspending-tapahtumankäsittelijän avulla, mutta sovellus ei tässäkään vaiheessa ole suljettu. Tila toteutuu esimerkiksi silloin, kun käyttäjä avaa useita sovelluksia sulkematta nykyisiä ja kone haluaa vapauttaa resursseja. Ohjelma pysyy Suspended-tilassa kunnes järjestelmä päättää sen resurssien puutteessa sulkea kokonaan tai kun käyttäjä avaa sovelluksen uudestaan. Windows Store -alustalla on tarjolla myös kätevät tapahtumat, jotka laukeavat tilan muuttuessa ja helpottavat sovelluksen hallintaa tilojen vaihtuessa. Sovelluksen palatessa asetukset voidaan ladata uudestaan Resuming-tapahtumankäsittelijän avulla. Windows 8 Store -sovelluksen elinkaaren kulku on kuvattu kuvassa (4).



Kuva 4. Store-sovelluksen elinkaari (Kuva MSDN)

Windows Phone 8:n mahdolliset tilat ovat Running, NotRunning, Tombstoned ja Dormant. Kuten Store-sovelluksissa tilan vaihtuessa sovellus laukaisee tapahtuman, jolla tilannetta voidaan käsitellä, tapahtumien nimet Phonen tapauksessa ovat Launching, Deactivated, Activated ja Closing. Windows Phone 8 -sovelluksen elinkaari on esitetty kuvassa (5). Kun käyttäjä vaihtaa puhelimessa sovelluksesta toiseen, siirtyy sovellus ensin Dormant-tilaan, jossa sen säikeet on pysäytettynä, mutta tiedot ovat tallessa muistissa ja kun käyttäjä palaa sovellukseen, pysyvät tiedot tallessa. Dormant-tilasta Tombstoned-tilaan siirtyminen tapahtuu, kun käyttäjä avaa uusia sovelluksia ja käyttöjärjestelmän täytyy vapauttaa muistia käyttöön. Tombstoned-tilassa sovellus on suljettu, mutta käyttöjärjestelmällä on tallessa pieniä määriä tietoja esimerkiksi sovelluksen tilasta, jossa se oli, kun se siirtyi Tombstoned-tilaan. Tietoja säilytetään enintään viiden sovelluksen osalta. (App activation and deactivation for Windows Phone 2013.)



Kuva 5. Phone-sovelluksen elämänkaari, pallot kuvaavat sovelluksen tiloja. (Kuva MSDN)

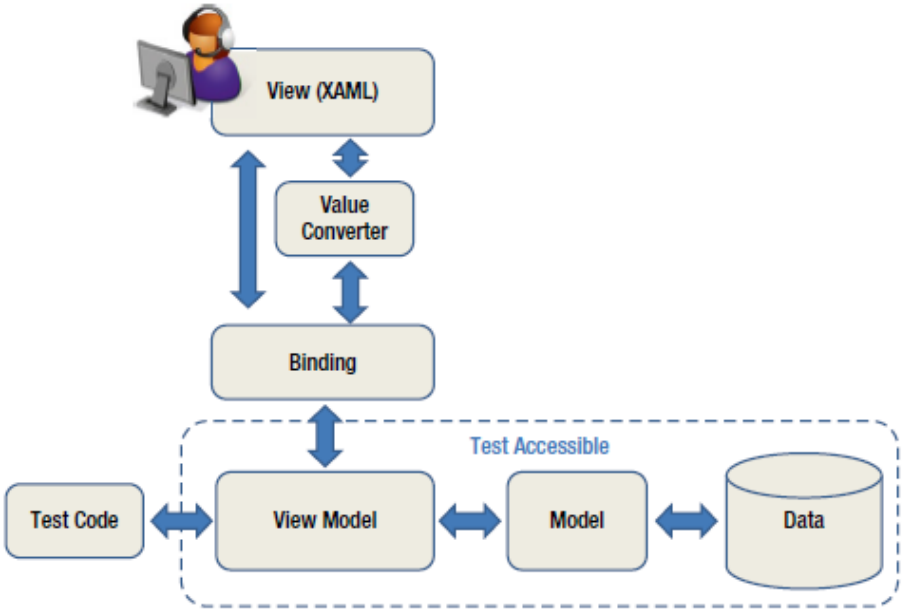
Tärkeää on myös huomata, että sovelluksessa pystyy hallitsemaan kahta tilaa, käytännössä on sivukohtainen tila ja koko sovellusta koskeva tila. Sovelluksen elinkaaren aikaisten tilojen käsittely on tärkeää käyttökokemuksen kannalta, esimerkiksi tilanne jossa käyttäjä poistuu sovelluksesta hetkeksi tekemään jotain muuta ja kun hän palaa sovellukseen, olettaa hän tietojensa olevan yhä tallella.

Erilaisten tilojen simuloiminen ei ole itse laitteilla välttämättä kovin helppoa, koska laitteet saattavat asettaa erilaisia tiloja hieman arvaamattomasti. Paras keino testata tapahtumankäsittelijöihin lisäämiään koodeja on käyttää Visual Studion virheiden jäljittäjästä eli debuggerista löytyviä tila-asetuksia, joilla voi pakottaa sovelluksen haluamaansa tilaan. (Freeman 2012, 99–100.)

3.4.6 MVVM – suunnittelumalli ja sovellusten rakenne

MVVM eli Model-View-ViewModel suunnittelumalli on suosittu tapa rakentaa XAML-pohjaisia sovelluksia. MVVM-malli on käytössä lähes kaikissa Microsoftin XAML-pohjaisissa ohjelmointiympäristöissä. Microsoft suosittelee vahvasti mallin käyttöä ja esimerkiksi lähes kaikki Microsoftin XAML-esimerkit toteuttavat MVVM-mallia. Mallissa koodi jaetaan kolmeen osaan Model- eli Malli-luokkaan, View- eli Näkymä-luokkaan ja ViewModel eli NäkymäMalli-luokkaan. MVVM-mallin käyttö tuo etuja varsinkin sovelluksen päivittämisen ja testaamisen yhteydessä. Kun varsinainen koodi ja käyttöliittymä on sidottu toisiinsa, esimerkiksi asettamalla koodi XAML-tiedoston *code-behind*-tiedostoon, tulee testauksessa se ongelma, että koko sovellus pitää ajaa ja käyttää sen käyttöliittymää, jotta saadaan testattua sovelluksen koodia. Tästä ei tietenkään yksinkertaisten sovellusten kanssa tule ongelmaa, mutta laajemman sovelluksen kanssa kyllä. (Burns 2012, 127–129.)

Mallissa on tarkoituksena, että Näkymämalli-luokka ei tiedä mitään Näkymä-luokan rakenteesta. Teoriassa (ja myös käytännössä) käytössä voi olla erilaisia versioita Näkymä-luokista erilaisille alustoille, kuten mobiililaitteelle, pöytätietokoneelle ja Xbox-pelikonsolille, jotka kuitenkin käyttävät samoja NäkymäMalli- ja Malli-luokkia. Alla on selventävä kuva (6) MVVM-mallin rakenteesta ja luokkien vuorovaikutuksista.



Kuva 6. MVVM-suunnittelumalli (Kuva K. Burns Appres Media)

MVVM-suunnittelumallista ja sen käytöstä on tarkemmin kerrottu Toni Ilomäen opinnäyteyössä. MVVM:stä löytyy paljon hyvää materiaalia myös MSDN:n sivuilta: http://social.msdn.microsoft.com/Search/en-US?query=MVVM&ac=4.

3.4.7 Tiedon sidonta - Data Binding

Data Binding eli suomeksi tiedon sidonta on merkittävässä osassa kun käytössä on MVVM-suunnittelumalli. Sidonnalla yhdistetään myös elementtejä eli saadaan yhdistettyä käyttöliittymän elementtejä suoraan toisiinsa niin, että ne osaavat päivittyä automaattisesti. Esimerkiksi vaikka ListBox ja sivun taustaväri, ListBoxiin on listattu värit ja kun käyttäjä valitsee listasta värin, vaihtuu sivun taustaväri. Tämä tapahtuu yleensä *DependencyProperty*-ominaisuuden avulla. (Data Binding Overview n.d.)

Sidonnalla yhdistetään käyttäjälle näkyvä käyttöliittymäluokka ja näkymää päivittävä NäkymäMalli-luokka. Tiedon sidonnan avulla saadaan näkymä päivittymään automaattisesti, kun tieto NäkymäMalli-luokassa muuttuu. Tämä on usein todella hyödyllinen ominaisuus, joka helpottaa käyttöliittymän päivittämistä. Tällöin käytetään *INotifyPropertyChanged* rajapintaa, jossa toteutettavana on yksi tapahtuma *PropertyChanged*. (Data Binding Overview n.d.)

Ajoittain tulee vastaan tapauksia joissa sidosten päissä olevat tietotyypit eivät vastaa toisiaan. Esimerkiksi tapaus, jossa on bool-lippumuuttujan true- tai false-arvo ja Visibility-tyypin Visible- tai Collapsed-arvo. Tällöin täytyy sidokseen asettaa konvertteri, joka hoitaa tiedon kääntämisen sopivaksi. (Data Binding Overview n.d.).

Tiedon sidonnasta ja sen käytöstä on kerrottu tarkemmin Toni Ilomäen opinnäytetyössä. Lisää tietoa löytyy myös MSDN:n artikkelista: http://msdn.microsoft.com/en-us/library/ms752347.aspx.

3.4.8 Tiedon tallentaminen – Windows Runtime Storage

Windows Runtime Storage on ohjelmistokirjasto, joka tarjoaa helpon mahdollisuuden tallentaa tietoja. Se löytyy Windows.Storage-nimiavaruudesta. Yleisin käyttökohde on sovelluksen asetusten tallentaminen. Windows Store -alustalla tietoa pystyy tallentamaan paikallisesti tiedostoon kirjoittamalla, sekä myös Microsoft-tilin mukana liikkuvaksi roaming-tiedoksi. Windows Phone 8 -alustalle ei ole ainakaan vielä tarjolla kuin paikalliseen tiedostoon kirjoittaminen. (Whitechapel & McKenna 2013, 244.) Jos haluaa käyttää tilin mukana synkronoituvaa roaming-mahdollisuutta, kannattaa miettiä tarkkaan, mitä tietoa siirretään mukana, koska kaikki ylimääräinen tiedonsiirto hidastaa sovelluksen käyttöä. Liitteessä 1 on esitelty yksinkertainen esimerkki tiedon tallentamisesta. Microsoft onkin julkaissut tarkat ohjeistukset siitä, kumpaa mahdollisuutta tulisi käyttää missäkin tilanteessa. Microsoftin ohjeistus löytyy osoitteesta: http://msdn.microsoft.com/en-us/library/windows/apps/hh465094.aspx.

Windows Runtime Storagea käyttäessä on hyvä muistaa, että kaikki mitä tallennat, pysyy niin sanotussa hiekkalaatikossa sovelluksen sisällä, eli kun sovellus poistetaan koneelta tai puhelimesta, poistuvat sen *Storageen* tallennetutkin tiedot. *Storage* antaa mahdollisuuden myös luoda kategorioita tallennettaville tiedoille, ja tämä on kätevä ominaisuus monissa tapauksissa.

Windows Runtime Storage -kirjasto on Store-sovelluksissa ainoa vaihtoehto, jolla pystyy toteuttamaan tietojen tallennusta. Phone-sovelluksissa tarjolla on myös muita vaihtoehtoja, mutta *Windows Runtime Storage* on näistä yksinkertaisin käyttää.

3.4.9 Verkkototeutukset

Verkkoa käyttöjärjestelmillä voi hyödyntää monella tapaa ja tässä on pieni listaus mahdollisuuksista. Käyttöjärjestelmien välillä on kuitenkin aika paljon eroja siinä, mitä tekniikoita on tuettuna.

Phonessa verkkoa voi käyttää WebClient- ja HttpWebRequest-luokkien kautta. Tarjolla on myös muokattu versio Microsoft Silverlight WebBrowser -tekniikasta HTML-sivujen esittämistä varten. Storessa mahdollisuudet ovat laajemmat ja käytettävissä on System.Net- ja System.Net.Http-nimiavaruuksista löytyviä luokkia. Mainittujen luokkien avulla pystyy käsittelemään internetissä olevia sivuja eri tavoilla, tiedonsiirron toteutukseen ne eivät kuitenkaan sovellu. (Whitechapel & McKenna 2013, 188–189; Accessing WCF Services with a Windows Store Client App 2012.) Molempien alustojen kanssa pystyy käyttämään myös perinteistä Socket-ratkaisua.

WCF on tuettuna molemmissa käyttöjärjestelmissä ja se onkin yleensä paras ratkaisu toteuttaa kahdensuuntaista tiedonsiirtoa. Store-alustalla WCF on tuettuna monipuolisesti ja sille on saatavilla seuraavat sidostyypit: BasicHttpBinding, NetTcpBinding, NetHttpBinding ja CustomBinding. Phone-puolella tuettuna on vain BasicHttpBinding, joka rajoittaa WCF:n käyttömahdollisuuksia. (Whitechapel & McKenna 2013, 201–202; Network Isolation for Windows Store Apps n.d.) WCF-verkkotekniikoita on tarkasteltu syvemmin Jarno Niemen opinnäytetyössä.

Azure Mobile Services -palvelua voi hyödyntää molemmilla alustoilla, ja sen avulla voi tallentaa pilvipalveluun pieniä määriä tietoa. Microsoft tarjoaa tällä hetkellä palvelua käyttöön ilmaiseksi rajoitetuilla ominaisuuksilla. *Mobile Services* -palvelusta voi tarkemmin lukea Azuren sivuilta: http://www.windowsazure.com/en-us/develop/mobile/.

3.4.10 Muita ominaisuuksia

Semantic Zoom on toiminto, jolla saa monessa tilanteessa helpotettua tilannetta, jossa on paljon tavaraa näytöllä. Yksinkertaisin esimerkki *Semantic Zoom* -toiminnon käytöstä on yhteystietoluettelo. Toiminto toimii siis niin, että kun käyttäjä tekee näytöllä ulospäin suurennus -liikkeen, tulee näkyviin vain aakkoset, josta käyttäjä valitsee haluamansa ja sovellus näyttää kyseisen kirjaimen kohdalta yhteystiedot. Toiminto kannattaa usein rakentaa omaan sovellukseen, jos tietoa ruudulla on runsaasti, sillä se on useissa tilanteissa todella hyödyllinen toiminto ja sitä voi myös käyttää paljon monipuolisemmin, kuin vain näyttämällä aakkoslistan.

Lukitusnäytössä näkyvät sovellukset (*Lock Screen Apps*), eli sovellukset joilla näkyy pieni ikoni näytössä, kun laite on lukittuna. Käyttäjä voi valita itse sovellukset, jotka hän haluaa lukitusnäytössä näkyvän. Käytännössä siis pienessä ikonissa näytetään käyttäjälle tärkeätä tai kiinnostavaa tietoa. Esimerkiksi sähköpostiohjelma voi näyttää lukemattomien viestien määrän myös lukitusnäytössä. Varsinkin puhelimessa tällä ominaisuudella on käyttöä, koska käyttäjä voi tarkastaa ilman näppäinlukon avaamista, onko hänelle saapunut viestejä. Tietokonemaailmassa hyöty on pienempi, koska kone on lukittuna huomattavasti pienempiä aikoja kuin puhelin.

Taustatehtävät (*Background tasks*) on sovelluksen koodia, joka pyörii vaikka sovellus olisi suljettu. Käyttökohteita ovat esimerkiksi lukitusnäytössä olevan sovelluksen päivitys, Live-tiilen päivitys ja Toast-ilmoituksen antaminen, kun jotain merkittävää tapahtuu. Windows 8:ssa taustatehtäviä suorittavat luokat toteutetaan IBackgroundTask-rajapinnasta. Windows Phone 8:ssa lisätään ScheduledAgent-projekti Visual Studion solution -rakenteeseen. Windows Phone 7:ssa taustatehtävien ajo on hyvin rajoitettua, mutta Phone 8:ssa kehittäjille on annettu vapaammat kädet.

Toast-ilmoitukset (*Notifications*) ovat ilmoituksia, jotka tulevat Windows 8:ssa näytön oikeaan yläreunaan, vaikka sovellus ei olisi päällä. Windows Phone 8:ssa ilmoitukset tulevat näytön yläreunaan. Push-ilmoitukset taas ovat ilmoituksia, jotka lähetetään palvelimelta laitteeseen. Push-ilmoitukset saa otettua käyttöön *Azure Mobile Services* -palvelulla.

Laitteen paikallistaminen onnistuu Location API -kirjaston tarjoamilla välineillä, paikannusta voidaan tehdä usealla tasolla. Paikannus onnistuu esimerkiksi GPS:n ja internet-yhteyden avulla.

Windows 8- ja Windows Phone 8 -käyttöjärjestelmien laitteista löytyy monenlaisia sensoreita. Sensoreita ei kuitenkaan välttämättä jokaisessa laitteessa ole, esimerkiksi pöytätiekoneissa, mutta kaikissa kannettavissa laitteissa usein löytyvät ainakin yksinkertaisimmat ja yleisimmät.

Taulukko 1. Puhelimissa olevia sensoreita ja niiden tarkoituksia tai tehtäviä.

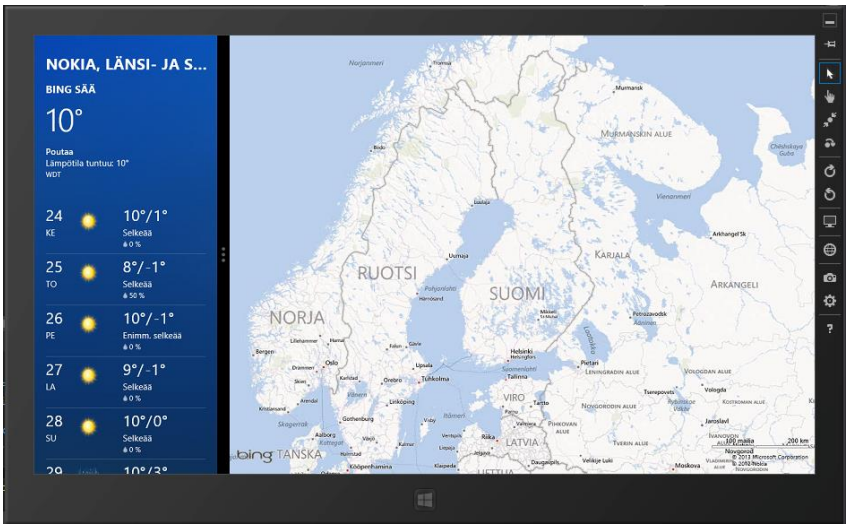
Sensori	**Tehtävä**
Nopeus sensori *The Gyrometer Sensor*	mittaa nopeutta kolmella akselilla
Kiihtyvyys sensori *The Accelerometer Sensor*	mittaa kiihtyvyyttä kolmella akselilla
Valoisuus sensori *The Light Sensor*	mittaa valon määrää
Kompassi sensori *The Compass Sensor*	kompassi
Kallistuskulma sensori *The Inclinometer Sensor*	mittaa kallistusta kolmella akselilla

3.5 Muut Store-sovelluksen ominaisuudet

Store-sovelluksilla on muutamia ominaisuuksia, joita ei Phone-alustasta löydy. Ne on esitelty tämän luvun aliluvuissa. Store-sovelluksilla on myös muutamia rajoituksia. Ohjelman täytyy tukea ainakin vaakasuuntaista käyttöä 1024 x 768 resoluutiolla. Ohjelman täytyy olla toimiva myös, kun käyttäjä käyttää Unsnapped- ja Snapped-toimintoja (resoluutio 320 x 768). (Windows 8 app certification requirements 2013, 3.6.)

3.5.1 Snapped- ja Filled-tilat

Windows Store -sovelluksen saa asetettua niin sanottuun Snapped- ja Filled-tilaan, jossa on kaksi sovellusta vierekkäin. Toinen sovellus on tällöin tilassa Snapped, jossa näytöstä on käytössä leveyssuunnassa vain 320 pikseliä. Toinen sovellus on Filled-tilassa ja ottaa lopun tilan näytöstä. Tilat ovat mahdollisia vain silloin, kun näyttö on vaakatasossa, eli jos näyttö käännetään pystysuuntaan, Snapped-tilassa ollut ohjelma menee taustalle piiloon ja Filled-tilassa ollut ohjelma asettuu koko näytön tilaan. Näytön kääntyessä takaisin palaavat ohjelmat omille paikoilleen.



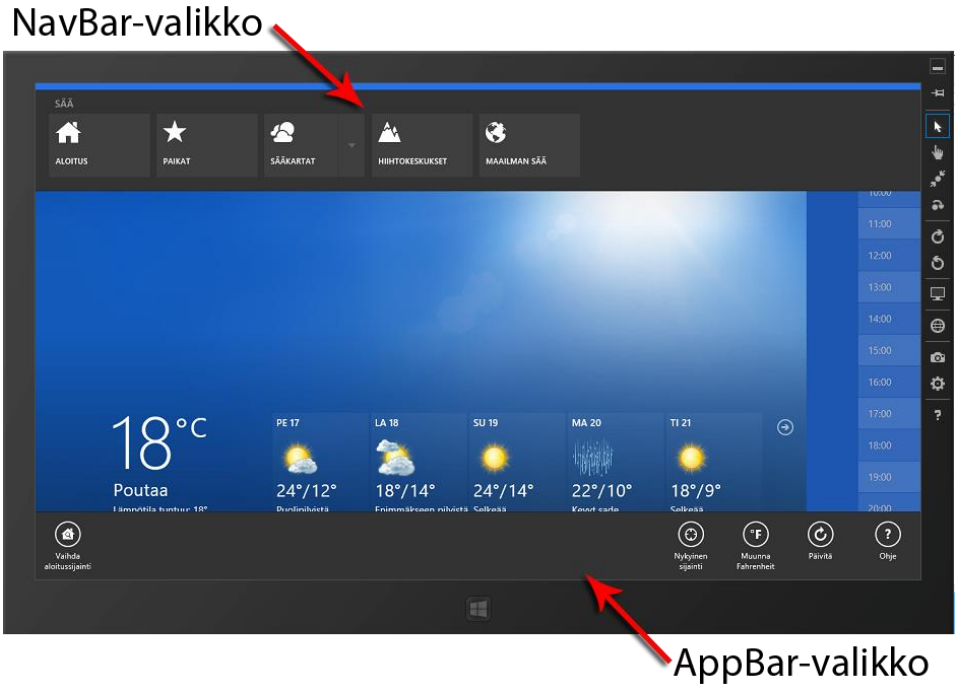
Kuva 7. Sääohjelma asetettu Snapped-tilaan ja karttaohjelma Filled-tilassa.

Ohjelmistokehityksen kannalta on tärkeää huomata, että tämä on ominaisuus, joka pitää Store-sovelluksissa toteuttaa aina. Filled-tila ei normaalisti tuota ongelmia, koska ohjelmat yleensä skaalautuvat tähän hyvin. Snapped-tila sen sijaan aiheuttaa usein ongelman, jos tilannetta ei käsitellä, koska alkuperäinen näytön leveys laskee alle kolmannekseen, samanaikaisesti korkeuden pysyessä samana.

Tilojen tunnistus onnistuu helposti esimerkiksi ApplicationView.Value-muuttujan avulla ja yksinkertainen tapa toimia, on muuttaa ohjelman osien järjestystä, ja mahdollisesti piiloittaa jotainkin ominaisuuksia. Omassa pelisovelluksessamme on esimerkiksi pohjana Grid-ruudukko, jossa on kaksi saraketta. Toisessa sarakkeessa on itse peli, jota pelataan ja toisessa keskustelualue. Kun sovellus asetetaan Snapped-tilaan, niin yksinkertaisesti piilotetaan keskustelualueen ruudukko kokonaan.

3.5.2 AppBar, NavBar ja Flyouts

AppBar-valikko on Store-sovelluksen valikko, joka aukeaa näytön alareunaan, kun käyttäjä painaa hiiren oikeanpuoleista painiketta tai tekee kosketusnäytöllä ylöspäin pyyhkäisyliikkeen. NavBar-valikko aukeaa samalla toiminnolla, mutta se sijaitsee näytön yläreunassa. NavBar-valikkoa on tarkoitus käyttää sovelluksen sisällä sivulta toiselle liikkumiseen, AppBar-valikkoon tarkoitus on lisätä toimintoja, joiden jatkuva näkyminen ei ole tarpeellista, usein valikossa on suosikkien asettelua, ohjeita ja niin edelleen. AppBar- ja NavBar-valikot saavat sisältää vain painikkeita ja usein samassa yhteydessä on tarvetta myös pienille toiminnallisuuksille, tässä käytetään usein apuna Flyout-ikkunoita. (Freeman 2012, 45–66).



Kuva 8. Sääohjelman NavBar- ja AppBar-valikot

Flyout-nimityksellä tarkoitetaan Store-sovelluksessa aukeavaa pop-up-ikkunaa. Flyout-ikkunat ovat yleensä käytössä AppBar-valikon yhteydessä, esimerkiksi lisää suosikki -toiminnosta voi aueta pieni ikkuna, johon syötetään halutut tiedot. Flyout-ikkunoiden tilalla käytetään usein myös perinteistä MessageBox-toimintoa. Molempia valikoita kannattaa hyödyntää, jos näkee sen lainkaan tarpeelliseksi. Koska valikoiden avulla saadaan sovelluksen, jossa on paljon sivuja tai toimintoja, käyttökokemus huomattavasti mukavammaksi. AppBar-valikon lisääminen Store-sovelluksen sivulle on esitelty lyhyesti liitteessä 2.

3.5.3 Store-sovelluksen asetukset - Settings contracts

Settings contracts on yksi osa niin sanottua contracts-toimintoa Windows 8 -käyttöjärjestelmässä ja sen avulla hallitaan asetuksien asettamista sovellukseen. Toiminnon tarkoitus on luoda käyttäjille yhdenmukainen käyttökokemus, kun asetukset on saatavilla aina samasta paikasta, eli Charm Bar -valikon kautta löytyvän Asetukset-toiminnon alta. Microsoft käytännössä määrääkin sovelluskehittäjät asettamaan sovellustensa asetukset toimivaan settings contract -toiminnon kautta. Toimintoa käytetään käytännössä siis niin, että luodaan sivu, jossa on asetusten säätö, eli mahdolliset tekstikentät tietojen syöttämistä varten ja niin edelleen. Tämän jälkeen otetaan sivu käyttöön App.xaml.cs-tiedostosta. (31 Days of Windows 8 2012, Day 5.) Liitteessä 3 on esitelty yksi tapa ottaa käyttöön sovelluksen asetukset.

3.5.4 Store-sovelluksen haku ja jako - Search & Share contracts

Käyttäjälle haku- ja jakotoiminnot löytyvät Charm Bar -valikon kautta. Search ja Share eli haku- ja jakotoiminnot ovat, kuten asetuksetkin, osa

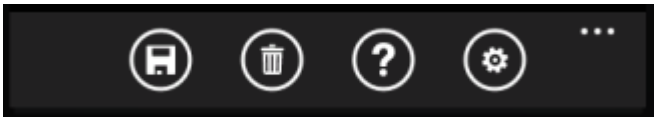
Windows Store -sovelluksen rakennetta ja niiden käyttöön ottaminen tapahtuu samalla periaatteella kuin asetustenkin.

3.6 Muut Phone-sovelluksen ominaisuudet

Windows Phone 8 -käyttöjärjestelmässä on muutamia eroja Windows 8 -käyttöjärjestelmään verrattaessa. Merkittävin ero on varmasti, että laitteiden näytöt ovat pieniä ja niitä on vain kolmella eri resoluutiolla. Toinen iso ero tulee siinä, että kyseessä on puhelin, joten siinä on puhelimen toimintoja ja niitä pystyy myös hyödyntämään sovelluksissa.

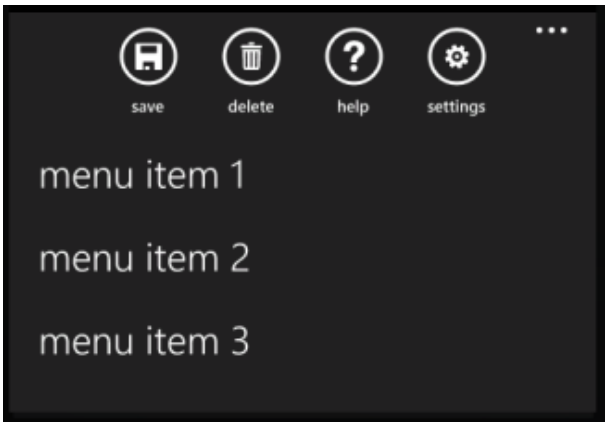
3.6.1 AppBar-valikko

Phone-sovelluksiin kuuluu myös näytön alareunassa sijaitseva AppBar-valikko, mutta Store-sovelluksista poiketen Phone-sovelluksissa ei ole käytettävissä navigointiin tarkoitettua NavBar-valikkoa tai Flyout-ikkunoita. Flyout-ikkunat korvataan ensisijaisesti MessageBox-ilmoituksilla tai kokonaan uusilla sivuilla. Navigointi sovelluksessa suoritetaan panorama-toiminnolla tai sivun yläreunan otsikoiden avulla.



Kuva 9. Phone-sovelluksissa käytettävä AppBar-valikko (Kuva MSDN)

Valikon käyttäminen ei ole pakollista, mutta usein se on suositeltavaa. Kuvassa (9) on esimerkki minkälaisia toimintoja voi sovelluksen AppBar-valikkoon asettaa. Valikkoa voi käyttää myös niin, että kuvakkeet on oletuksena piilotettuna ja valikosta näkyy vain kolme pistettä, joita painamalla valikko aukeaa. Jos kuvakkeet ovat jo valmiiksi esillä ja käyttäjä klikkaa pisteitä, ilmestyvät kuvakkeiden alle tekstit. Lisäksi mahdolliset vain tekstimuodossa valikossa olleet tiedot tulevat näkyviin, toiminto on esitetty kuvassa (10).



Kuva 10. AppBar-valikko aukaistuna (Kuva MSDN)

3.6.2 Phone and Media Services

Windows Phone -käyttöjärjestelmä tarjoaa mahdollisuuden käyttää sisäänrakennettuja puhelin- ja mediapalveluja. Phonessa on tarjolla joukko Launches (laukaisijat) ja Choosers (valitsijat) nimillä kulkevia toimintoja, joiden avulla voi hyödyntää integroituja sovelluksia ja ominaisuuksia omassa sovelluksessa. Kehittäjä pystyy hyödyntämään esimerkiksi puhelimen selainta, hakutoimintoa, kalenteria ja sähköpostia. Tarjolla on laaja tuki myös äänen ja videon toistoa varten, tarjottuna on kolme erilaista API-rajapintakirjastoa (*application programming interfaces*). (Whitechapel & McKenna 2013, 110–112.)

3.6.3 Puhetoiminnot – Speech

Windows Phone -käyttöjärjestelmä sisältää monipuolisesti puheeseen liittyviä toimintoja. Äänentallennus, äänikomennot, puheentunnistus ja tekstin muuntaminen puheeksi ovat ominaisuudet, jotka Phone-alusta tarjoaa käytettäväksi sovellusten kanssa. Puheeseen liittyvien toimintojen hyödyntämistä kannattaa harkita sovelluksesta, koska se voi olla nopeampaa, helpompaa tai hauskempaa kuin perinteinen käyttökokemus. (Whitechapel & McKenna 2013, 402,433.)

3.6.4 Puhelimen yhteystiedot ja kalenteri - Contacts and Calendar

Phone-sovelluksissa on mahdollisuus hyödyntää puhelimessa olevia yhteystietoja ja kalenterimerkintöjä, sekä lisätä myös omia. Yksi Windows Phone -käyttöjärjestelmän keskeisistä periaatteista on, että puhelin auttaa liittämään ja pitämään yhteyttä tärkeisiin ihmisiin. Microsoft on usein käyttänyt ”*Put People First*” -mottoa Phone-sovelluskehityksen ympärillä. Kalenterin ja yhteystietojen hyödyntäminen on yksi parhaista tavoista toteuttaa tätä mottoa. (Whitechapel & McKenna 2013, 328–330, 355, 360.)

4 WINDOWS 8 STORE -SOVELLUKSEN JULKAISU SOVELLUSKAUPASSA

Sovelluksen julkaisu Windows 8 Storessa ja Windows Phone Storessa on hyvin samankaltainen prosessi ja myös vaatimustaso sovellusten laadussa on hyvin samaa tasoa, ja siksi työssä onkin kuvattu vain Windows 8 Store -sovelluksen julkaisu. Muutamia eroavaisuuksia kuitenkin löytyy käyttöjärjestelmien eroista johtuen, kuten mainosten sisällyttäminen sovellukseen. Näkyvin ero on ehkä se, että Phone-sovellukselle ei voi itse suorittaa sertifiointitestiä ennen sovelluksen lisäystä kauppaan. Kuten jo aikaisemmin sovelluskauppaa käsittelevässä kappaleessa olen todennut, molemmilla on siis täysin toisistaan irralliset sovelluskaupat. Täten jos haluaa kehittää sovelluksia molemmille käyttöjärjestelmille, tarvitsee molempiin sovelluskauppoihin tilit ja myös täytyy maksaa molemmista tileistä.

Henkilökohtaisesti en pitänyt Microsoftin tavasta esittää muutamia kohtia Store-sovelluskaupan käyttöehdoista ja käytännöistä. Tietenkin Microsoftin kannalta tärkeät ehdot on selvästi ilmaistu, mutta muun tiedon saamisessa oli ajoittain hankaluuksia. Kaupan kehittäjätilin hankinnassa saa olla siis tarkkana, koska Microsoft ei kovin selvästi ilmaise esimerkiksi tilin maksukäytäntöjä. Esimerkiksi tiedon löytäminen jo siitä, että tili on vuosimaksullinen, oli hyvin haastavaa. Toinen tieto, mitä en löytänyt helpolla, oli se, voinko tienata sovelluksilla sen jälkeen, kun tilin käyttöoikeus päättyy tai mitä sovelluksille tapahtuu, kun tilin käyttöoikeus loppuu. Ehdot menevät siis niin, että tilistä täytyy maksaa vuosittain, jotta kaupan käyttöoikeudet säilyvät ja sovellukset pysyvät kaupassa. Tässä kuitenkin kuvaus kuinka sovelluksen julkaisuprosessi etenee sovelluskaupassa, tilin hankinnasta alkaen ja päättyen sovelluksen lisäämiseen kauppaan.

Store-sovelluksen sovelluskaupassa julkaisua varten täytyy hankkia Windows-kaupan kehittäjätunnukset (tili). Tunnuksien luontia varten tarvitsee luottokortin, jonka avulla henkilöllisyys todennetaan ja mahdolliset maksut peritään, sekä pitää omistaa Microsoft-tili. Vaikka ohjeistuksessa puhutaan vain luottokortista, kelpasi ainakin omassa testauksessani Nordea pankin Visa Electron -kortti. Sovelluksen lähettämisprosessi on Microsoftin sivuilla hyvin tarkasti ohjeistettu ja sen kanssa ei pitäisi tulla ongelmia. Sivut ja ohjeistukset ovat tarjolla useilla kielivaihtoehdoilla, mukaan lukien suomi.

4.1 Sovelluskauppatunnusten luominen

Tunnuksia on kahdentyyppisiä, Henkilö-tunnus ja Yritys-tunnus. Yritystunnus on nimensä mukaisesti suunnattu yrityksille ja se myös sisältää muutamia lisäominaisuuksia, joita ei yksityishenkilöille tarkoitetussa tunnuksessa ole. Yksi Yritys-tunnuksen ominaisuuksista on mahdollisuus kehittää ja lisätä kauppaan työpöytäsovelluksia. Henkilö-tunnus on tarkoitettu sovellusten kehitykseen yksityishenkilönä tai pienen, ei-yritysmuotoisen ryhmän kanssa. Henkilö-tunnuksen hinta on 37 euroa/vuosi ja Yritys-tunnuksen 75 euroa/vuosi. Korkeakouluopiskelijoille Microsoft DreamSpark kuitenkin tarjoaa yksityishenkilö-tyyppisen Windows 8- ja Windows Phone -kaupan kehittäjätunnuksen ilmaiseksi yhden

vuoden ajaksi. DreamSpark vaatii rekisteröitymisen ja tunnustautumisen opiskelijaksi, esimerkiksi koulun sähköpostiosoitteen avulla.

Tunnusten luonti aloitetaan osoitteesta dev.windows.com valitsemalla *Dashboard* (koontinäyttö). Jotta pääsee aloittamaan rekisteröinnin, täytyy käyttää Microsoft-tilin suojakoodia. Toiminnolla Microsoft varmistaa, että käytössä on varmasti oma tili.

Ensimmäisenä pyydetään valitsemaan kotimaa ja tilin tyyppi. Seuraavassa kohdassa täytetään tilin tiedot, joihin kuuluvat siis perushenkilötiedot. Kolmannessa kohdassa luettavaksi tulee Windows-kaupan sovelluskehittäjäsopimus, joka kannattaa lukea, jotta tietää mihin sitoutuu. Seuraavassa kohdassa ilmoitetaan hinta ja on mahdollisuus syöttää koodi, jolla palvelun saa ilmaiseksi. Korkeakouluopiskelijat saavat kyseisen koodin noudettua siis DreamSpark-sivuston kautta. Seuraavaksi edessä on Maksu-kohta, jossa syötetään luottokortin tiedot ja myös veloitetaan summa. Tämän jälkeen on vuorossa tilauksen vahvistus, jossa näkyy vielä yhteenveto tilauksesta. Vahvistuksen jälkeen tili on lähes valmis, tilin voi viimeistellä kaupan ohjeiden mukaan. Viimeistelyä varten täytyy vielä vahvistaa tilin luottokortti, jolla tilin maksut suoritetaan, sekä tili, jolle sovelluksilla mahdollisesti ansaidut rahat siirretään. Kun sovellukset ovat maksullisia ja saat niistä voittoa, täytyy myös täyttää tilin veroprofiili. Verotietojen täyttöä varten sovelluskaupan sivuilta löytyy selkokieliset ohjeet.

4.2 Varmistukset ennen sovelluksen lisäämistä

Tärkeää on tarkistaa, että on asettanut sovellukseen tarvittavat kuvat ja sovellus sisältää kaikki halutut ominaisuudet. Tärkeintä on tietenkin, että sovellus ylipäätänsä toimii toivotusti. Sovellusta kannattaa myös testata ennen sen lisäämisen aloittamista. Testausta voi suorittaa tekemällä sovelluksesta paketin ja antamalla sen vaikka parille ystävälle testattavaksi. Ohjelmistopaketin voi luoda valitsemalla Visual Studio 2012 Store -valikosta *Create App Packages*. Kohdassa jossa kysytään, haluatko lisätä sovelluksen Store-kauppaan, vastaa ei, niin Visual Studio luo paketin, jonka voi lähettää haluamilleen henkilöille.

Hyödyllisimpiä tehtäviä on tarkistaa, että sovellus läpäisee Windowsin sovellusten sertifiointipaketin testit. Testin voi suorittaa samalla, kun luo sovelluskauppaan lähetettävän ohjelmistopaketin. Jos haluaa kuitenkin testata sovelluksen jo ennen lisäysvaihetta, sen voi tehdä seuraavasti.

Käynnistä Windows App Cert Kit -ohjelma. Kun ohjelma on käynnissä, valitse *Validate a Windows Store app*. Ohjelman pitäisi löytyä koneelta, koska se on osa Windows 8 -sovelluskehityspakettia, joka asentuu Visual Studion mukana. Jos sovellusta ei kuitenkaan löydy, se on ladattavissa Microsoftin sivuilta. Sertifiointipaketti hakee kaikki Windows Store -sovellukset, jotka on käynnistetty tietokoneessa. Valitse testattava ohjelma listasta ja aloita testaus jatkamalla eteenpäin.

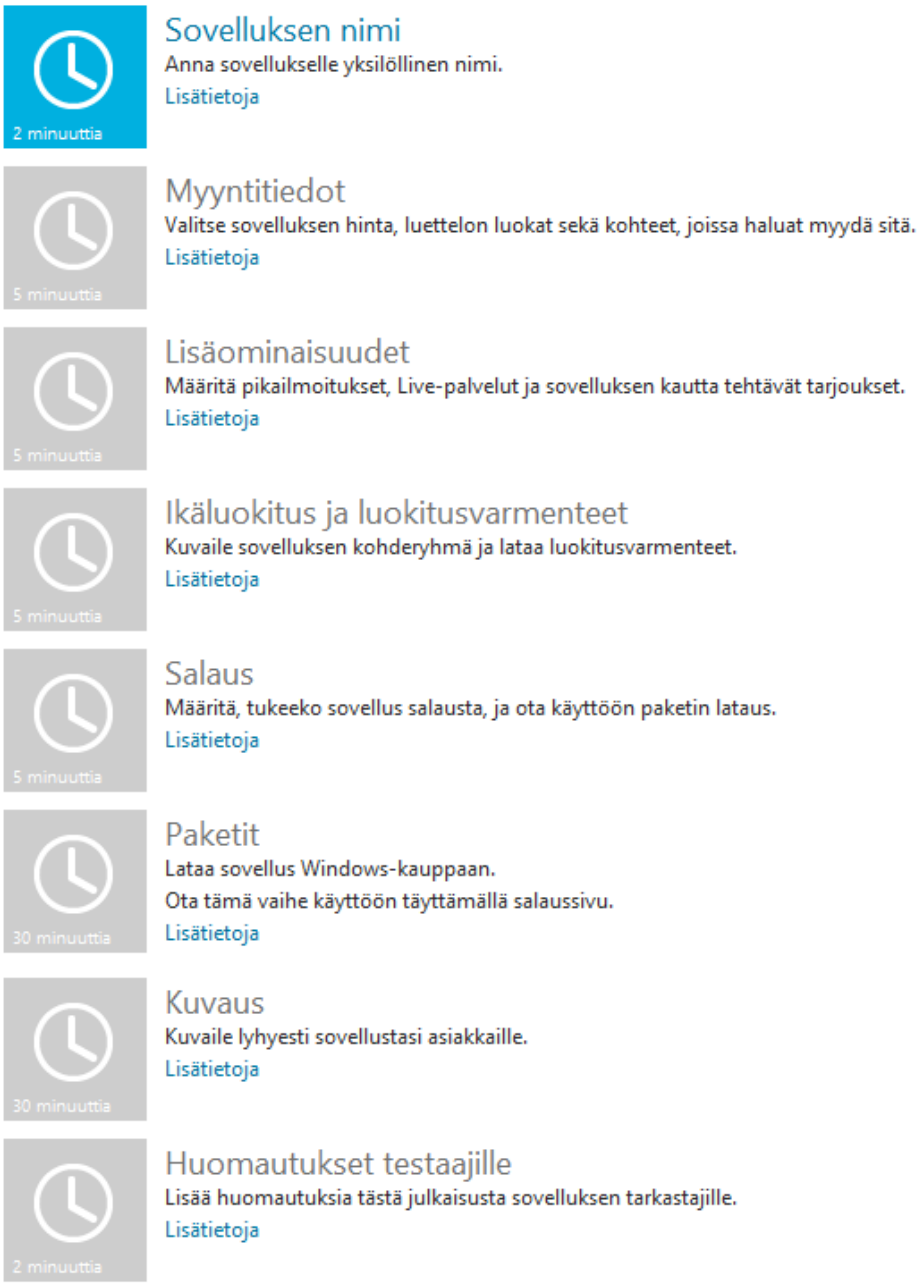
Testien aikana testattava sovellus käynnistyy useita kertoja ja sovelluksen kannattaa antaa käynnistyä rauhassa, jotta ei sotke testauksen tuloksia.

Esimerkiksi sain oman sovelluksemme käynnistystestin tulokseksi *failed*, kun selasin samalla internet-sivuja. Kun testit on saatu valmiiksi, kysyy tarkistusohjelma raportin tallennuspaikan. Raportti tallentuu xml-muotoon ja sen saa selaimella avattua helposti luettavaan muotoon. Lopuksi näkee suoraan testin lopputuloksen ja voi tarkastella tarkempia tietoja raportista.

Testi käy lävitse todella monia kohtia, joihin kuuluvat muun muassa suorituskykytestit, kaatumistestit, tietoturvatestit ja sovelluksen manifest-tiedoston tarkistus, joka sisältää esimerkiksi tarkistuksen, että sovelluksen oletuslogokuvat on vaihdettu. Testi ei tietenkään pysty huomaamaan kaiken tyyppisiä virheitä, vaan se tarkistaa lähinnä, että sovellus on teknisesti rakennettu oikein. Suosittelenkin suorittamaan myös sovellustestausta ihmistestaajilla.

4.3 Sovelluksen lisääminen sovelluskauppaan

Sovelluksen lisääminen aloitetaan kirjautumalla sovelluskaupan sivuille. Sivuilta löytyy Lähetä sovellus -linkki. Sovelluksen lisäys on jaettu kahdeksaan osaan. Ensimmäisenä on Sovelluksen nimi, sovellukselle tulee asettaa uniikki nimi, tätä nimeä tulee myös käyttää sovelluksen Display name -kohdan nimenä Package.appxmanifest-tiedostossa. Nimen asetus on varaus, joka raukeaa, jos sovellusta ei lähetetä vuoden sisällä.



Kuva 11. Sovelluksen lähetyksen kahdeksan vaihetta ja Microsoftin asettamat aika-arviot niiden kestosta.

Toisena kohtana on Myyntitiedot, ja myyntiedoissa asetetaan sovelluksen hinta ja markkina-alueet, eli minkä maiden kaupoissa sovellus on näkyvissä. Jos sovellus on maksullinen, voi asettaa sovellukselle ilmaisen kokeilujakson. Sovellukselle asetetaan myös julkaisupäivä ja luokkatiedot. Lisäksi asetetaan vielä mahdolliset laitteistovaatimukset, joilla saadaan esimerkiksi rajattua tehoja vaativa sovellus pois ARM-arkkitehtuurin alustalta eli Windowsin RT -versiosta. Viimeisenä kohtana on helppokäyttöisyys, jota ei voi vain rastittaa, jos on itse sitä mieltä, että sovellus on helppo käyttää, vaan siihen kuuluu useita ehtoja ja testauksia.

Eteneminen seuraa yllä esitetyn kuvan kaaviota, seuraavaksi käsittelyyn Lisäominaisuudet, joissa määritetään mahdollisia Push Notification- ja Live Connect -palveluiden käyttöä. Tässä kohdassa voi myös asettaa tarjouksia sovelluksen kautta, jolla voi siis esimerkiksi myydä lisäominaisuuksia sovellukseen. Jos ei käytä kyseisiä palveluita tai ei halua asettaa tarjouksia, tämä kohta on hyvin nopea.

Seuraavana tulee Ikäluokitus ja luokitusvarmenteet. Kohdassa asetetaan siis sovelluksen ikäluokitus ja saatavilla on selvät ohjeet, miten luokitus asetetaan. Kohdassa määritetään myös luokitusvarmenteet, jotka tarkoittavat esimerkiksi Suomessa käytössä olevaa vapaaehtoista PEGI-luokitusjärjestelmää. Muutamissa maissa peliluokitus on pakollinen, eli peliä ei voi myydä kyseisten maiden kaupoissa, jos sillä ei ole vaadittavaa luokitusta. Peliluokituksen hankintaan ei tässä työssä oteta kantaa, mutta se tulee hoitaa luokitusjärjestön kautta.

Viidentenä kohtana on Salaus. Tässä määritetään salausta käsittelevät kohdat, koska esimerkiksi salasanojen suojauksesta on olemassa tietosuojalakeja. Kohdassa siis varmistetaan, että salaus on otettu huomioon tarvittavilta osin. Jos sovellus ei sisällä, lähetä tai vastaanota mitään salausta kaipaavaa, on tämä kohta muutamalla kuittauksella ohi.

Salauksen jälkeen käsiteltäväksi tulee Paketit-kohta, jossa siis ladataan Visual Studiolla tehty paketti sovelluskauppaan. Ohjelmistopaketin luominen aloitetaan valitsemalla Visual Studion Store -valikosta *Create App Packages*. Kun tarkoituksena on lisätä sovellus kauppaan, valitse *Yes*. Tällöin aukeaa kirjautumisikkuna, johon syötetään sovelluskaupan tilin tunnukset. *Select an app name* -kohtaan pitäisi tulla sovelluskauppaan asetettu nimi, valitaan oikea sovellus ja jatketaan painamalla *Next*. Seuraavaksi valitaan paketin sijainti, versionumero ja mahdollinen ohjelmistoarkkitehtuurin (kaikki, x86, 64, ARM) rajaus. Versionumerointiin voi asettaa automaattisen numeroinnin, eli Visual Studio nostaa automaattisesti numeroa aina, kun luo paketin. Lopuksi painetaan *Create* ja Visual Studio alkaa luoda pakettia. Kun paketti on luotu, ilmoittaa Visual Studio siitä output-ikkunassa. Visual Studio myös kysyy, halutaanko ajaa sertifiointitesti, joka on sama testi, jota käsittelin jo edellisessä luvussa. Testin aikana sovellus käynnistyy muutaman kerran ja *Certification kit* testaa sovellusta erilaisissa tilanteissa. Jos sovellusta ei aikaisemmin ole testattu, on se suositeltavaa, koska jos ohjelma ei läpäise testiä tuskin se läpäisee varsinaista Microsoftinkaan testiä. Kun sovellus ei läpäise Microsoftin sertifiointia, ei se tule julkaistuksi sovelluskauppaan, ennen kuin virheet on korjattu. Kun testi on onnistuneesti suoritettu voi ladata appxupload-tiedoston sovelluskauppaan. Kun paketti on ladattu onnistuneesti, voi painaa tallenna edetäkseen julkaisuprosessissa.

Sovelluksen Kuvaus on seitsemäs kohta, kuvaukseen kuuluvat tekstikuvaus sovelluksesta, lisäksi voi lisätä sovelluksen ominaisuuksia listatyyppisesti. Tarvitaan myös näyttökuvat eli kuvat, jotka näkyvät kaupassa asiakkaalle sovelluskuvauksessa, kuvia on oltava vähintään yksi, mutta tietenkin asiakas saa paremman kuvan, mitä monipuolisemmin kuvia on. Visual Studion simulaattori sisältää kuvaruutukaappaustoiminnon, jonka

avulla on helppo ottaa kuvat sovelluksesta. Päivityksen kuvaus -kohdassa voi asettaa kuvauksen päivityksen sisältöön. Päivityskuvaus sijaitsee sen takia tässä kohdassa, että kun sovellusta päivitetään, käydään koko julkaisuprosessi läpi taas kohta kohdalta, mutta tietenkin tässä vaiheessa suurin osa tiedoista on valmiiksi täytettynä.

Seuraavassa määritellään suositeltu laitteisto, johon on varsinkin silloin hyvä lisätä ominaisuuksia, kun ominaisuudet ovat sellaisia, että niitä ei löydy jokaisesta laitteesta, kuten GPS. Avainsanat kohtaan lisätään sanoja, jotka kuvaavat sovellusta. Tekijänoikeus- ja tavaramerkkitiedot kohtaan merkitään otsikon mukaiset tiedot, jos sovelluksen oikeudet ovat kehittäjällä itsellään, eli sovellusta ei ole tehty yritykselle, tällöin merkitään tähän kohtaan oma nimi. Muut käyttöehdot -kohtaan lisätään muita mahdollisia ehtoja, joita sovelluksen käyttöön kuuluu. Seuraavana vastaan tulevat esittelykuvat. Ne ovat kuvia, joita Microsoft voi käyttää sovellusten esittelyyn sovelluskaupassa. Kuvien lisäys kannattaa koska näillä kuvilla voi saada lisähuomioita sovellukselle kaupassa. Sivusto-kohtaan lisätään tieto esimerkiksi yrityksen tai sovelluksen kotisivusta. Tuen yhteystiedot -kohtaan lisätään esimerkiksi tukisivuston osoite tai tuen sähköpostiosoite. Viimeisessä kohdassa, tietosuojakäytäntö, lisätään perinteinen tietosuojakäytäntöohje siitä, kuinka käyttäjiltä kerättyjä tietoja käytetään.

Kahdeksas eli viimeinen kohta on Huomautukset testaajille. Kohtaan vain lisätään testausohjeita ja tunnuksia. Esimerkiksi jos sovellus vaatii kirjautumisen, lähetään testitunnukset. Jos sovelluksessa ei kuitenkaan ole mitään mikä, vaatisi testaajilta erityishuomioita, voi kentän jättää tyhjäksi ja painaa tallenna.

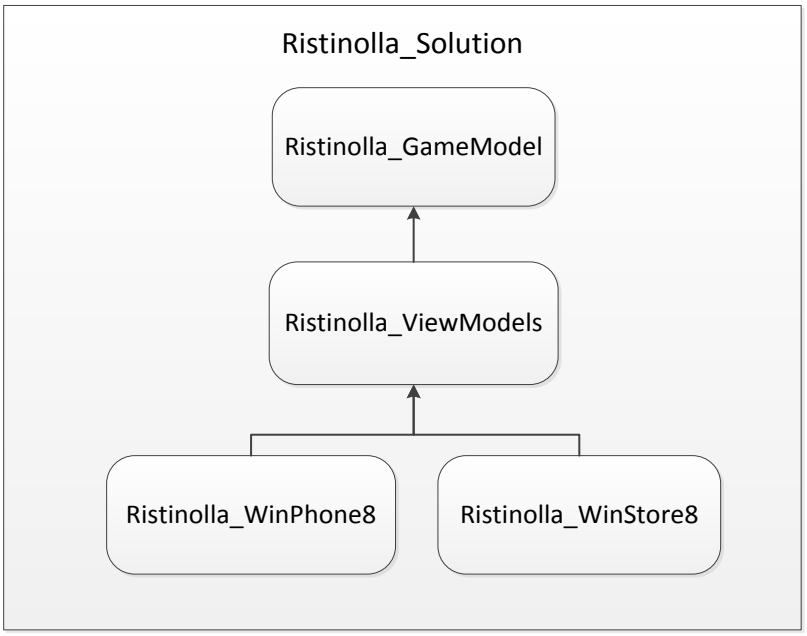
Nyt sovellus on valmis lähettäväksi Microsoftin sertifiointiin, tässä vaiheessa saa otettua yhteenvedon, jolla saa tarkastettua kaikki antamansa tiedot. Tiedot kannattaa lukea vielä kerran ajatuksella läpi, jotta mahdolliset virheet saadaan vielä poimittua pois ennen julkaisua. Kun tiedot on luettu läpi, viimeistellään sovelluksen lisäys painamalla Lähetä sertifioitavaksi ja sovellus siirtyy Microsoftin sertifiointijonoon.

Microsoftin sertifiointi kestää muutamasta päivästä viikkoihin, riippuen sen hetkisestä sertifiointijonosta ja jos sovellus läpäisee sertifioinnin, se julkaistaan valittujen maiden sovelluskaupoissa valitulla päivämäärällä. Jos sovellus ei läpäise sertifiointia, saa tästä selkeän tiedon, minkä takia sovellusta ei hyväksytty, sekä saa myös ohjeita, kuinka voi korjata tilanteen. Kun ilmenneet ongelmat on korjattu, voi sovelluksen lähettää uudestaan sertifioitavaksi.

5 TOTEUTETTU SOVELLUS JA JATKONÄKYMÄT

Varsinainen Windows Storelle ja Windows Phone 8:lle toteutettu sovellus oli siis internet-yhteyden yli toimiva ristinollan kaksinpeli, joka sisältää mahdollisuuden keskustella toisten pelaajien kanssa. Pelissä on mahdollisuus valita ruudukon koko 3 x 3 ja 10 x 10 välillä, voittorivin pituus skaalautuu ruudukon koon mukaan. Peli loppuu jommankumman pelaajan saadessa voittoon tarvittavan rivin tai ruudukon tullessa täyteen. Visual Stu-

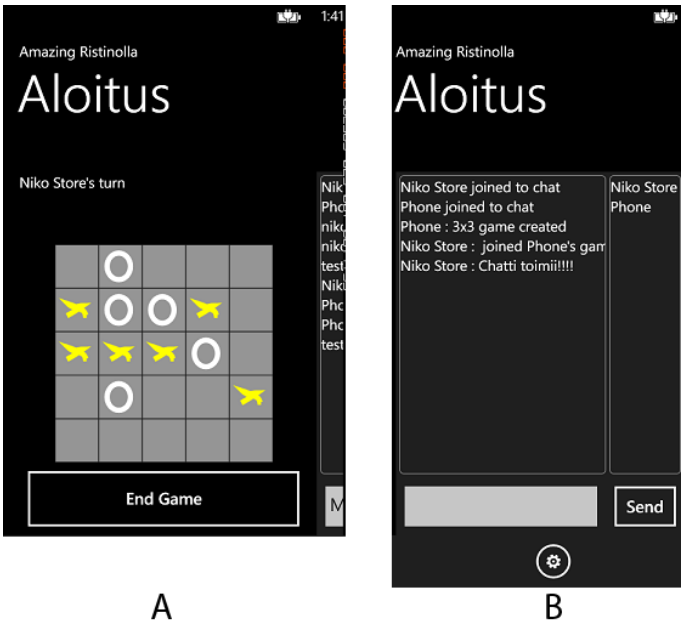
dion 2012 solution -rakenteessa asetettiin kaikki työn projektit saman solutionin-rakenteen sisälle, siis Windows 8 Store -sovelluksen projekti sekä Windows Phone 8 -sovelluksen projekti ja sovellusten yhteisesti käyttämät Model- eli Malli-projekti ja ViewModel- eli NäkymäMalli-projekti. Rakenteessa siis noudatettiin MVVM-suunnittelumallia. Store- sekä Phone-projektit eivät sisällä varsinaisia toiminnollisuuksia, vaan toimivat View- eli Näkymä-luokkina ja sisältävät vain käyttöliittymän.



Kuva 12. Kuva toteutetun sovelluksen projektirakenteesta

Verkkototeutus rakennettiin *Windows Communication Foundation* -tekniikalla ja pelin käyttämät palvelut on sijoitettu Microsoftin Azure-pilvipalveluun. WCF:n ja Azuren toiminnasta voi lukea tarkemmin Jarno Niemen opinnäytetyöstä. Alustojen eroista johtuen rakennettiin sovellusten verkkototeutus eri tekniikoilla, Storen on toteutettu NetHTTP-tekniikalla ja Phonen WebSocket-tekniikalla.

Rakennetussa sovelluskokonaisuudessa ei käytetty kovinkaan paljoa Store- ja Phone-alustan tarjoamia erityisominaisuuksia. Kaikkia ominaisuuksia ei käytetty aikataulu syistä ja osa jätettiin myös tarkoituksella pois mahdollista jatkokehitystä varten. Tarkoituksenahan oli luoda sovellusrunko, jota voidaan hyödyntää tulevaisuudessa opetuskäytössä muun muassa erikoistumisprojektien yhteydessä.



Kuva 13. Windows Phone -ristinolla-pelisovelluksen ulkoasu. A-kohdassa kuvattu pelitilanne, reunassa näkyy panorama-toiminnon toiminta, eli viereisen sivun reuna näkyy. B-kohdassa keskustelualue ja sivun alareunassa AppBar-valikko, josta pääsee pelin asetuksiin.

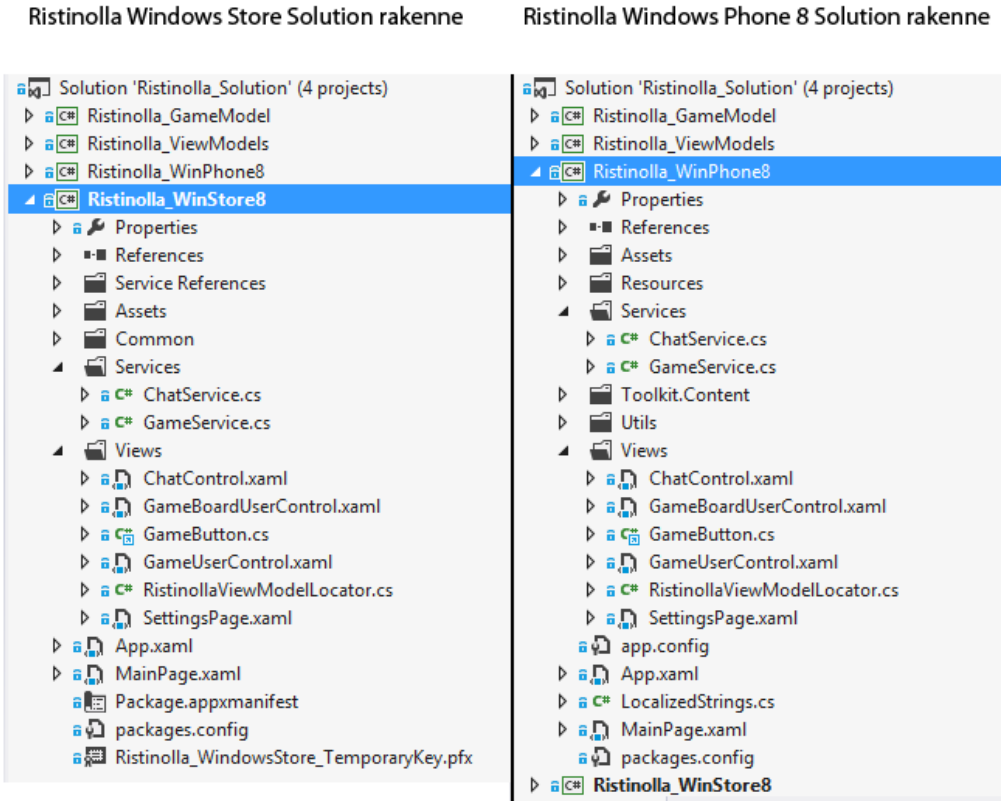
Sovellusrungosta tuli myös hieman teknisempi ja laajempi kokonaisuus kuin työn alussa oli kuviteltu. Tämä aiheuttaa sen, että työn koodi ei ole tulkittavissa lyhyessä ajassa, ja tästä syystä taas sovellusrunkoa voi olla vaikeampi hyödyntää pienemmissä kouluprojekteissa, mutta opettajat tulevat hyödyntämään työtä parhaaksi katsomallaan tavalla. Toteutuksessa siis käytettiin monipuolisesti uusia tekniikoita esimerkiksi koodipohjan jakamisessa molemmille alustoille. Koodipohjan jakamisvaihtoehdoista ja niiden käytöstä voi lukea lisää Toni Ilomäen opinnäytetyöstä. Työssä on esitelty mahdolliset tekniikat ja se, miten niitä on ristinolla-pelin toteutuksessa hyödynnetty.

Kehityksen aikana sovellusta testattiin useilla laitteilla ja erilaisissa tilanteissa. Fyysisiä laitteita oli Lumia 920-, 720- ja 620-puhelimet sekä kehityskäytössä olleet tietokoneet. Lisäksi sovellusta testattiin Phone-emulaattorilla ja Windows 8 -simulaattorilla. Testauksen aikana ilmeni, että Phonen verkkototeutuksessa käytetty WebSocket oli hyvin herkkä kaatumaan. Ongelma ilmeni kuitenkin vain fyysisillä laitteilla ja emulaattorilla testatessa ongelmaa ei saatu toistumaan. Ongelmat olivat lopulta pieniä ja niiden ilmetessä ei enää ollut aikaa korjata.

Kuva 14. Windows Store -ristinolla-pelisovelluksen ulkoasu.

5.1 Sovelluksen tarkempi rakenne

Näkymä-luokkina toimivissa Phone- ja Store-projekteissa on hyvin yhtäläinen rakenne ja nimeämiskäytännön avulla on pyritty pitämään rakenne myös mahdollisimman selvänä. Molemmissa on siis käytetty yhtäläisiä nimiä. Kuvassa (15) näkyy Store- sekä Phone-sovellusten rakenne, kuvassa (16) on avattuna Malli- ja NäkymäMalli-luokkien rakenteet.

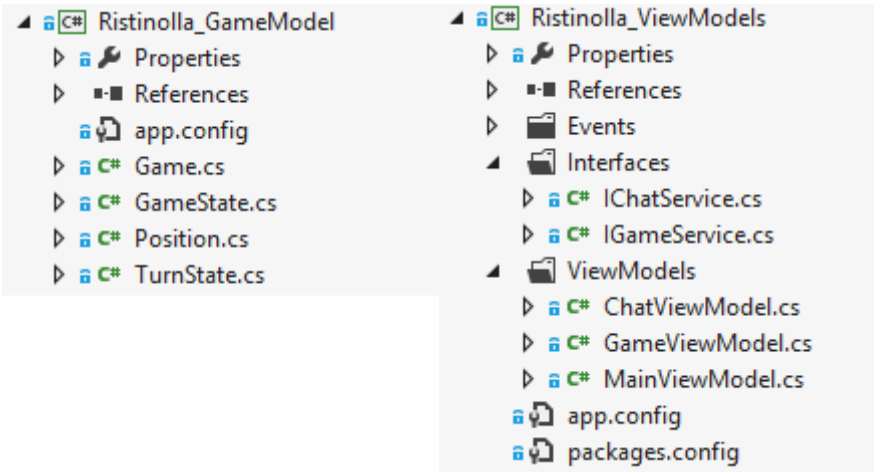


Kuva 15. Windows Store- ja Windows Phone 8 -projektien rakenne

Service-luokkien tehtävä on käsitellä palvelimelta tulevat tiedot, eli esimerkiksi pelissä tapahtuvat siirrot. Service-luokat on toteutettu ViewModel-projektissa olevista rajapinnoista. Store-projektin Common- ja Phone-projektin Utils-kansiot sisältävät tiedonsidonnassa käytetyt konvertterit. Common-kansiossa on myös StandardStyles-tyylitiedosto. App.xaml-

tiedostossa on käsitelty sovelluksen erilaisten tilojen hallinta ja käynnistyksen yhteydessä ohjaus aloitussivulle eli MainPage-sivulle.

Views-kansion alla on näkymän rakennuksessa käytetyt luokat. MainPage on sivu kaiken pohjalla, johon muut näkymän osat on aseteltu. ChatControl on sovelluksen keskustelualueen näkymä ja SettingsPage sisältää asetukset sivun sisällön. GameBoardUserControl on luokka, jossa on näkyvissä varsinainen peliruudukko, GameUserControl taas on luokka, jossa käyttäjä valitsee ruudukon koon ja suorittaa pelin tekemisen tai peliin liittymisen. GameButton on Button-luokasta periytetty ja se hoitaa pelin yksittäisen ruudun rakennuksen. Kuvassa 15 näkyvä pieni sininen merkki GameButton-luokan kohdalla tarkoittaa, että tiedosto on linkki jaettuun kooditiedostoon, eli molemmissa projekteissa käytetään samaa jaettua tiedostoa. Sovelluksen MVVM-mallin toteutuksen yhteydessä otettiin käyttöön ViewModelLocator-toiminto, joka helpottaa tiedon välitystä näkymien ja näkymämallien välillä. Locatorin ja jaetun koodin toiminnasta on kerrottu tarkemmin Toni Ilomäen opinnäytetyössä.



Kuva 16. ViewModel- ja Model-projektien rakenne

ViewModel-projektissa on Service-rajapinnat sekä erilliset NäkymäMalli-luokat etusivulle, pelille ja keskustelualueelle. Model-projekti sisältää itse Game-luokan, jossa suoritetaan pelin käsittely. Game-luokassa on siis esimerkiksi voittorivin tarkistus ja siirtojen laillisuuden tarkistus. GameState sisältää enum-listan pelin tiloista. TurnState sisältää enum-listan vuoroista. Position-luokkaa käytetään ruudukon koordinaattien muodostuksessa ja niiden käsittelyssä.

5.2 Jatkokehitys

Sovelluksen ympärillä on hyvin laajat mahdollisuudet tehdä jatkokehitystä. Sovelluksen ulkoasu jätettiin esimerkiksi hyvin pienelle huomiolle ja sen parissa voi kehittää Store- ja Phone-käyttöliittymän suunnittelutaitoja. Peliin voisi sisällyttää esimerkiksi pientä visualisointia voittoriville sekä uuden merkin asettamiseen. Sovelluksesta voi kääntää versioita muille alustoille eli siis esimerkiksi suosituille Googlen Android- ja Applen iOS -käyttöjärjestelmille.

Sovellukseen voisi tuoda lokalisoinnin ja monikielisyyden Microsoftin työkalujen avulla tai lisätä linkityksen Microsoft-tiliin. Peliin voi rakentaa myös tietokonevastustajan, jonka rakennus rajattiin tässä työssä pois. Peliin voisi lisätä erilaisia pelimuotoja, esimerkiksi loputon ruudukko tai mahdollisuuden käyttää myös ruudukkoa, jossa korkeus ja leveys eivät ole samat.

Monet alustoihin liittyvät erityisominaisuudet jätettiin kokonaan käyttämättä, sovellukseen voisi lisätä esimerkiksi Toast Notification -ilmoitukset tai mahdollisuuden jakaa omia pelituloksiaan yhteisöpalveluissa. Phone-sovelluksen verkkototeutuksen ongelmat olisi tietenkin hyvä ratkaista, mutta ne eivät sovelluksen selvän jaon ja rakenteen, eli MVVM-suunnittelumallin toteutuksen takia, haittaa muuta jatkokehitystä. Mahdollisuudet ovat siis monipuoliset.

5.3 Tulevaisuus - Windows 8.1 ja Windows Phone 8.1

Microsoft julkaisee tulevaisuudessa niin sanotun Blue-päivityksen, joka tunnetaan myös nimellä Windows 8.1. Päivitys tulee molemmille käyttöjärjestelmille. Sen sisältönä on useita korjauksia ja parannuksia, jotka tulevat parantamaan käyttökokemusta ja antavat paremmat mahdollisuudet kustomoida käyttöjärjestelmää haluamakseen. Päivityksen voi sanoa vastaavan aikaisemmissa Windows-versiossa julkaistuja Service Pack -päivityksiä, mutta ilmeisesti se sisältää kuitenkin enemmän myös käyttäjille näkyviä visuaalisia muutoksia. Tiedossa tulevista muutoksista on ainakin jo Käynnistä-valikon kuvakkeen palauttaminen vasempaan alakulmaan sekä lisää erilaisia kokovaihtoehtoja Live-tiiliin. Opinnäytetyön kirjoitusvaiheessa tietokoneiden puolella oli jo vahvasti puhetta 8.1-päivityksestä, joten luultavasti myös sen julkaisu tapahtuu hyvin pian, puhelimien osalta ilmeisesti saadaan odottaa hieman pidempään. Päivityksien ja sen muutoksien sisältöä ei tähän työhön ehditty saamaan.

Microsoft on kiinnostunut aloittamaan omien puhelimien valmistuksen ja siirtymään ohjelmistotalosta enemmän laitteistotaloksi. Tulevat muutokset voivat myös vaikuttaa Windows Phone -alustan tulevaisuuteen hyvinkin radikaalisti.

6 YHTEENVETO JA JOHTOPÄÄTÖKSET

Windows 8 Store- ja Windows Phone 8 -sovellusten rakentaminen on tehty suhteellisen yksinkertaiseksi, verrattuna useisiin muihin alustoihin ja lisäksi tarjolla on Microsoftin MSDN-sivuton kautta todella paljon hyviä artikkeleita, erimerkkikoodeja ja ohjeita. Varsinkin sivustot dev.windows.com, design.windows.com, dev.windowsphone.com ja microsoft.com/finland/geekout/ sisältävät todella paljon hyödyllistä materiaalia. Osa näistä on tarjolla jopa suomenkielellä ja kaikki löytyvät ainakin englanninkielisinä. Myös Microsoft Virtual Academy- ja Channel 9 -sivustot (channel9.msdn.com) tarjoavat useita kursseja ja materiaalia koskien Windows Phone- ja Windows Store -sovellusten kehitystä. Materiaa-

lin paljoudesta tosin tulee ajoittain ongelma sen suhteen, että osaa arvioida, mikä on itselle hyödyllisintä materiaalia. Microsoft tarjoaa myös työkalut kehitykseen ilmaiseksi. Jos kiinnostusta tarkempaan tutustumiseen on, suosittelen lukemaan myös Toni Ilomäen opinnäytetyön, joka opastaa käyttämään samaa ohjelmistokoodia molemmilla alustoilla ja Jarno Niemen työn, jossa on tutustuttu Microsoftin pilvipalvelun Azuren ja Windows Communication Foundationin käyttämiseen verkkototeutusta luodessa Windows Store- ja Phone-sovelluksille.

Yleisesti katsoen työ on kuitenkin hyvin pintaraapaisu siitä kaikesta sisällöstä, jota Windows Phone 8- ja Windows Store -sovelluskehitykseen kuuluu. Onnistuin kuitenkin mielestäni hyvin poimimaan paljon tärkeitä asioita, joita tulee huomioida eri sovelluskehityksenvaiheissa. Mutta todella kiinni sovellusten rakentamiseen pääsee vain itse kokeilemalla. Jos kokee kiinnostusta sovelluskehitykseen Store- tai Phone-alustalle ja mielessä on jokin idea sovelluksesta, joka voisi kiinnostaa muitakin käyttäjiä, niin antaa DreamSparkin kautta saatavilla oleva vuoden ilmainen sovelluskaupankehittäjätili hyvän mahdollisuuden kokeilla.

Lähdemateriaalia etsiessä ja tutkiessa huomasin, että Windows Storesta kertovaa materiaalia oli huomattavasti enemmän saatavilla kuin Windows Phone 8:sta. Omana veikkauksenani syyksi on Windows 8 Store -alustan tuoreus, se julkaistiin nyt ensimmäistä kertaa, kun Windows Phone 8:lla on jo edeltäjä Phone 7, josta kuitenkin oli jo materiaalit kirjoitettuna. Ja vaikka Phone 8 ja 7 välillä tuli paljon muutoksia, on niillä kuitenkin vielä enemmän yhtäläisyyksiä.

Työn sisällöstä joutui rajaamaan monia mielenkiintoisia ja ehkä hyödyllisiäkin asioita pois, tai esittelemään ne pintapuolisesti, jotta työ pysyi kohtuullisissa mitoissa. Työssä sivuutettiin esimerkiksi ominaisuudet, jotka liittyvät sijaintitietoihin, sensoreihin, puheeseen ja videon toistoon.

LÄHTEET

31 Days of Windows 8. 2012. Jeff Blankenburg & Clark Sell. Viitattu 15.4.2013. http://31daysofwindows8.com

App activation and deactivation for Windows Phone. 2013. Microsoft Oy. Viitattu 12.5.2013. http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff817008%28v=vs.105%29.aspx

Accessing WCF Services with a Windows Store Client App. 2012. Microsoft Oy. Viitattu 19.5.2013. http://msdn.microsoft.com/en-us/library/hh556233.aspx

Beta testing your app and in-app products. 2013. Microsoft Oy. Viitattu 26.4.2013. http://msdn.microsoft.com/en-us/library/windowsphone/help/jj215598%28v=vs.105%29.aspx

Burns, K. 2012. Beginning Windows 8 Application Development: XAML Edition. New York: Apress Media.

Data Binding Overview. n.d. Microsoft Oy. Viitattu 12.5.2013. http://msdn.microsoft.com/en-us/library/ms752347.aspx

Freeman, A. 2012. Windows 8 Apps Revealed using with XAML and C#. New York: Apress Media.

Kehittäjäkäyttöoikeuden hankkiminen. 2013. Microsoft Oy. Viitattu 22.4.2013. http://msdn.microsoft.com/fi-fi/library/windows/apps/hh974578.aspx

MacDonald, M. 2012. Pro WPF in C#, Fourth Edition. New York: Apress Media.

Network Isolation for Windows Store Apps. n.d. Microsoft Oy Viitattu 18.5.2013 http://msdn.microsoft.com/en-us/library/hh768193.aspx

Microsoft Expression Changes. 2013. Microsoft Oy. Viitattu 28.4.2013. http://www.microsoft.com/expression/eng/

Whitechapel, A. & McKenna, S. 2013. Windows® Phone 8 Development Internals. 2. Preview. Sebastopol: Microsoft Press.

Troelsen, A. 2012 Pro C# and the .NET 4.5 Framework, Sixth Edition. New York: Apress Media.

Targeted app distribution. 2013. Microsoft Oy. Viitattu 26.4.2013. http://msdn.microsoft.com/en-us/library/windowsphone/help/jj619160%28v=vs.105%29.aspx

Windows Phone 8. 2013. Wikipedia. Viitattu 26.4.2013. http://en.wikipedia.org/wiki/Windows_Phone_8

Windows 8 app certification requirements. 2013. Microsoft Oy. Viitattu 22.4.2013. http://msdn.microsoft.com/en-us/library/windows/apps/hh694083.aspx

Windows 8:n järjestelmävaatimukset. 2013. Microsoft Oy. Viitattu 13.4.2013. http://windows.microsoft.com/fi-fi/windows-8/system-requirements

Windows Phone Emulator. 2013, Microsoft Oy. Viitattu 13.4.2013. http://msdn.microsoft.com/en-us/library/windowsphone/develop/ff402563%28v=vs.105%29.aspx

Liite 1

WINDOWS RUNTIME STORAGE – ESIMERKKIKOODI

Storage-esimerkki luokka, jossa on käytetty tiedon tallennukseen sekä paikallista tiedontallennusta että roaming-tallennusta. Aseta breakpoint-pisteet esimerkiksi AddFile()-, ReadFile()- ja DeleteFile()-metodien kohdille.

```
public sealed partial class MainPage
    {
        StorageFolder folderLocal;
        StorageFolder folderRoaming;
        string fileName = "demotext.txt";
        string fileContents = "demo";

        public MainPage()
        {
            this.InitializeComponent();
            folderLocal = ApplicationData.Current.LocalFolder;
            folderRoaming = ApplicationData.Current.RoamingFolder;
            AddFile();
        }

        private async void AddFile()
        {
            // Lisätään tiedostot paikalliseen ja roaming-kansioon
             // Tiedoston polku löytyy Locals-ikkunasta folderLocal / folderRoaming
             arvojen Path-kohdasta
             StorageFile fileRoaming = await folderRoaming.CreateFileAsync(fileName,
             CreationCollisionOption.ReplaceExisting);
             StorageFile fileLocal = await folderLocal.CreateFileAsync(fileName, Crea-
             tionCollisionOption.ReplaceExisting);

            await FileIO.WriteTextAsync(fileLocal, fileContents + "LOCAL");
            await FileIO.WriteTextAsync(fileRoaming, fileContents + "ROAMING");

            ReadFile();
        }

        private async void ReadFile()
        {
            // Luetaan tietoja paikallisesta ja roaming-tiedostosta
            StorageFile fileLocal = await folderLocal.GetFileAsync(fileName);
            string textLocal = await FileIO.ReadTextAsync(fileLocal);
            fileContents = textLocal;

            StorageFile fileRoaming = await folderLocal.GetFileAsync(fileName);
            string textRoaming = await FileIO.ReadTextAsync(fileRoaming);
            fileContents = textRoaming;

            DeleteFile();
        }

        private async void DeleteFile()
        {    // Poistetaan tiedostot
            StorageFile fileRoaming = await folderRoaming.GetFileAsync(fileName);
            StorageFile fileLocal = await folderLocal.GetFileAsync(fileName);
            fileLocal.DeleteAsync();
            fileRoaming.DeleteAsync();
        }
    }
```

Alkuperäinen esimerkin koodi (laajempi): http://www.jeffblankenburg.com/2012/11/08/31-days-of-windows-8-day-8-local-and-roaming-data

Liite 2

APPBAR-VALIKON LISÄÄMINEN WINDOWS STORE SOVELLUKSEEN

Lisää seuraava koodi XAML-sivun loppuun jolle haluat lisätä AppBar-valikon. Valikossa on käytetty StandardStyles-tiedostosta löytyviä painiketyylejä, käy poistamassa kommentoinnit tarvittavista kohdista.

```
<Page ...

<Page.BottomAppBar>
  <AppBar>
      <Grid>
         <Grid.ColumnDefinitions>
              <ColumnDefinition />
              <ColumnDefinition />
         </Grid.ColumnDefinitions>

             <StackPanel Orientation="Horizontal" Grid.Column="0" HorizontalAlign-
             ment="Left">
             <Button x:Name="AppBarDoneButton" Style="{StaticResource DoneAppBar-
             ButtonStyle}" IsEnabled="false" Click="AppBarButtonClick" />
             </StackPanel>

             <StackPanel Orientation="Horizontal" Grid.Column="1" HorizontalAlign-
             ment="Right">
              <Button x:Name="AppBarAddButton" Style="{StaticResource AddAppBar-
             ButtonStyle}" AutomationProperties.Name="New Item"
             Click="AppBarButtonClick" />
            <Button x:Name="AppBarStoresButton" Style="{StaticResource StoresAppBar-
            Button}" Click="AppBarButtonClick" />
             <Button x:Name="AppBarZipButton" Style="{StaticResource HomeAppBar-
            ButtonStyle}" AutomationProperties.Name="Zip Code"
            Click="AppBarButtonClick" />
           </StackPanel>
      </Grid>
  </AppBar>
</Page.BottomAppBar>
</Page>
```

`AppBarButtonClick` toteutetaan XAML-tiedoston C#-taustaluokassa.

```
private void AppBarButtonClick(object sender, RoutedEventArgs e)
{
      //Toiminnallisuudet tähän
}
```

Liite 3

STORE-SOVELLUKSEN ASETUSTEN KÄYTTÖÖN OTTAMINEN

Koodi jonka asetat App.xaml.cs-tiedostoon. Koodilla käsitellään asetukset-sivun toiminnot, kyseisten koodien lisäksi tarvitset vielä varsinaisen sivun.

```
protected override void OnLaunched(LaunchActivatedEventArgs args)
{
   // OnLaunched-metodiin lisätään vain yksi rivi koodia, joka on tapahtuman käsitte-
   lijä
   SettingsPane.GetForCurrentView().CommandsRequested += App_CommandsRequested;

   ...
}

// Varsinainen SettingsCommand-objektin lisääminen
private void App_CommandsRequested(SettingsPane sender, SettingsPaneCommandsRe-
questedEventArgs args)
{
   SettingsCommand command = new SettingsCommand("settings", "This app settings",
   (handler) =>
   {
      Popup popup = BuildSettingsItem(new SettingsPage(), 346);
      popup.IsOpen = true;
   });
   args.Request.ApplicationCommands.Add(command);
}

private void App_CommandsRequested(SettingsPane sender, SettingsPaneCommandsRe-
questedEventArgs args)
{
SettingsCommand command = new SettingsCommand("settings", "This app settings", (han-
dler) =>
            {
                Popup popup = BuildSettingsItem(new SettingsPage(), 346);
                popup.IsOpen = true;
            });
            args.Request.ApplicationCommands.Add(command);
        }

// Popup-ikkunan hallintakoodi (Ei tarvitse välittää voi vain kopioida)
private Popup BuildSettingsItem(UserControl u, int w)
{
   Popup p = new Popup();
   p.IsLightDismissEnabled = true;
   p.ChildTransitions = new TransitionCollection();
   p.ChildTransitions.Add(new PaneThemeTransition()

   {
       Edge = (SettingsPane.Edge == SettingsEdgeLocation.Right) ?
                           EdgeTransitionLocation.Right :
                           EdgeTransitionLocation.Left
   });
   u.Width = w;
   u.Height = Window.Current.Bounds.Height;
   p.Child = u;
   p.SetValue(Canvas.LeftProperty, SettingsPane.Edge == SettingsEdgeLocation.Right ?
   (Window.Current.Bounds.Width - w) : 0);
   p.SetValue(Canvas.TopProperty, 0);
   return p;
}
```

Esimerkki kokonaisuudessaan: http://www.jeffblankenburg.com/2012/11/05/31-days-of-windows-8-day-5-settings-contract