HAAGA-HELIA
ammattikorkeakoulu

Windows Phone 8.1-sovellus: ShakesBeat

Mikko Niskanen

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2015

HAAGA-HELIA
ammattikorkeakoulu

Tiivistelmä

07.05.2015

Tekijä(t)
Mikko Ilmari Niskanen

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
Windows Phone 8.1- sovellus: ShakesBeat.

Sivu- ja liitesivumäärä
65 + 14

Opinnäytetyön otsikko englanniksi
Windows Phone 8.1 application: ShakesBeat.

Tämän opinnäytetyön aiheena on Windows Phone 8.1- sovelluksen kehitys. Työn tuloksena julkaistaan Windows Phone- kaupassa ilmainen sovellus nimeltään ShakesBeat.

Työssä tutustutaan Microsoftin mobiilikäyttöjärjestelmien historiaan sekä viimeisimmän version uusin ominaisuuksiin perustuen Microsoftin visioon käyttöjärjestelmien tulevaisuudesta. Julkaistavan sovelluksen kehitystyössä tarvittavia luokkia ja niiden ominaisuuksia käydään läpi huomioiden sovellusten elinkaari asennuksesta sovelluksen poistoon. Sen lisäksi tutustutaan sovelluksessa tarvittavien kuvien käyttöön ja niiden ominaisuuksiin. Työn teoriapohjan viimeisessä luvussa selvitetään julkaistavien sovellusten rahoitusmalleja, maksullisesta sovelluksesta mainosrahoitteiseen.

Työn tavoitteena on sovelluksen julkaisu kaikkien ladattavaksi. Sovelluksen avulla voidaan toistaa musiikkia ja muita äänilähteitä puhelinta heiluttamalla. Mitä nopeammin puhelinta heilutetaan edestakaisin, sitä nopeammin musiikkia toistetaan. Vastaavasti hitaasti heiluttamalla musiikkia toistetaan hidastettuna. Sovelluksen kohdeyleisö on lapset ja nuoret. Sen graafisessa ilmeessä keskitytään selkeyteen ja värikkyyteen.

Opinnäytetyön tuloksena sovellus julkaistiin Windows Phone- kaupassa suunnitellusti. Sovelluksen kehityksessä käytettiin teoriapohjan mukaisia taitoja ja tietoja, joiden avulla kehitystyö sujui vauhdikkaasti. Johtopäätöksenä todetaan, että Windows Phone 8.1:n C#- ja XAML-ympäristöt sopivat hyvin vähän resursseja syövien fyysiseen liikkeeseen reagoivien sovellusten ja pelien kehittämiseen. Sen lisäksi kehittäjällä on useita joustavia tapoja ansaita sovelluksensa julkaisun avulla.

Asiasanat
Windows Phone 8.1, Visual Studio, XAML, C#, mainonta

HAAGA-HELIA
University of Applied Sciences

Abstract

07.05.2015

Author(s)
Mikko Ilmari Niskanen

Degree programme
Degree Programme in Information Technology

Report/thesis title
Windows Phone 8.1 application: ShakesBeat.

Number of pages and appendix pages
65 + 14

The topic of this thesis is the development of a Windows Phone 8.1 application. The outcome is an application called ShakesBeat that will be published in Windows Phone Store.

The theoretical part dives into the history of Microsoft's mobile operating systems, continuing with the new features of the latest release reflecting the future vision of Microsoft's operating systems. The classes and their properties needed in the actual development work are studied next taking care of the application lifecycle starting from installation until the uninstallation of the piece of software. The images and their properties added to any given application are also explored. The monetization aspect of applications being published is studied in the last section of the theoretical part covering both paid and ad-based applications.

The main goal of this thesis was to publish an application for everyone to download. The application enables you to play music or other sounds by shaking your device. The faster you shake, the higher the playback rate is. Also if you shake the device slowly, the sound is played very slowly. The main audience will be children of all ages. Hence the layout of the application is clear and colourful.

The outcome of this thesis is an application published in Windows Phone Store as planned. The skills used in the development work are based on the theoretical section. The conclusion is that the C# and XAML environments for Windows Phone 8.1 are suitable for developing applications and games, which are non-resource intensive and react to physical user movement. There are also plenty of flexible ways for a developer to earn money along with publishing an application in Windows Phone Store.

Keywords
Windows Phone 8.1, Visual Studio, XAML, C#, advertising

Sisällys

Termit ja lyhenteet

C#	.NET-arkkitehtuuriin perustuva hallinnoitu (managed code) ohjelmointikieli, joka on kehitetty tarjoamaan C++:n monipuolisuus, mutta samalla Java-kielen helppokäyttöisyys.
XAML	Extensible Application Markup Language. Deklaratiivinen XML-pohjainen merkintäkieli. Käytetään .NET-ympäristöissä.
Managed code	Hallittu koodikieli, jota voidaan suorittaa vain erillisissä kielelle tarkoitetuissa ympäristöissä kuten .NET. Vastakohtana natiivi koodi.
Native code	Natiivi koodikieli, esimerkiksi C++, joka käännetään suoraan konekielelle ilman muita vaatimuksia ympäristöltä. Usein käytetään kuitenkin ristiin konekielen ja ”unmanaged code” merkitysten kanssa.
Visual Studio 2013	Microsoftin IDE (Integrated Development Environment) kehitysympäristö .NET:lle.
Code behind	Esimerkiksi XAML-tiedostojen taustalla olevat cs-tiedostot, jotka sisältävät C#-koodia.
Windows Phone Store	Windows Phone- kauppa.
WP	Windows Phone.
OS	Operating System. Käyttöjärjestelmä.
UX	User experience. Käyttökokemus.

NFC	Near Field Communication.
URI	Universal Resource Identifier.
URL	Universal Resource Locator.
IAP	In-app purchase. Sovelluksen sisäinen ostotapahtuma.
PDLC	Premium downloadable content.
API	Application programmable interface. Ohjelmointi-rajapinta.
SDK	Software development kit.

1 Johdanto

Tämän lopputyön tarkoitus on tuottaa musiikin toistoon hauskalla tavalla tarkoitettu mobiilisovellus Windows Phone 8.1- alustalle. Työ käsittelee sovelluksen luontiin vaadittavia taitoja ja tietoja Microsoft Visual Studio 2013:n avulla C# ja XAML- kielillä ohjelmoituna. Produktina työ tuottaa sovelluksen Windows Phone- kauppaan ladattavaksi. Työllä ei ole tilaaja ja julkaisu tehdään henkilökohtaisella kehittäjätunnuksella. Sovelluksella voidaan toistaa haluttua musiikki- tai muuta äänilähdettä heiluttamalla puhelinta edestakaisin. Heilutuksen nopeudesta riippuen musiikkia toistetaan eri nopeudella; nopeasti heiluttamalla äänitiedostoa toistetaan nopeutetusti ja vastaavasti hitaasti heiluttamalla toisto hidastuu.

Sovellus ohjelmoidaan XAML- ja C#- ohjelmointikielillä. Lopputyö rajataan kehitettävän sovelluksen vaatimiin keskeisiin ohjelmointikielten osiin, joten kieliin ei uppouduta sen syvällisemmin. Myös kehitysympäristön oletetaan olevan lukijalle tuttu, sen vuoksi siihen ei paneuduta yleiskuvausta enempää.

Lopputyössä tutustutaan aluksi Windows Phone 8.1- käyttöjärjestelmän historiaan, toimintoihin ja niiden ohjelmointiin C#/ XAML- kielillä.

Seuraavaksi keskitytään sovellusten elinkaaren ajatteluun sekä käyttöliittymän suunnitteluun ja ulkonäköön.

Tietoperustan viimeisen kappaleen muodostaa sovellusten rahoitus ja markkinointi, jossa käsitellään eri tapoja ansaita rahaa julkaisemalla sovelluksia Windows Phone- kaupassa.

Produkti-osiossa tuotetaan sovellus ja julkaistaan se Windows Phone- kaupassa. Sovelluksen kehitys perustuu tietoperustassa vastattuihin tutkimuskysymyksiin, joita ovat kuinka laitteiden liikeantureita voidaan hyödyntää, kuinka laitteeseen tallennettua mediaa voidaan hyödyntää sovelluksissa, kuinka sovellus voidaan kohdentaa erikokoisille

laitteilla käyttäjäkokemuksen muuttumatta ja kuinka Windows Phone- kaupan sovellusten mainospohjainen rahoitus toimii.

2 Windows Phone 8.x

Tässä kappaleessa esitellään tiivistetysti Windows Phone 8.1- käyttöjärjestelmän ominaisuuksia. Aluksi esitellään käyttöjärjestelmän yleisiä kehityspolkuja, jonka jälkeen keskitytään valittuihin käyttöjärjestelmän käyttötapoihin ja niiden ohjelmointiin kehittäjän näkökannalta katsottuna. Valitut ominaisuudet keskittyvät lopputyön tuloksena tuotettavan sovelluksen vaatimuksiin, joten muut ominaisuudet sivutetaan. Kehitysympäristönä käytetään Visual Studio 2013 Ultimate- alustaa, jolla sovelluksia kehitetään XAML- ja C#- kielillä.

2.1 Yleistä Windows Phone 8.x- käyttöjärjestelmästä

2.1.1 Microsoftin puhelinkäyttöjärjestelmät

Microsoftin puhelinkäyttöjärjestelmien historia on värikäs, käsittäen useita eri versioita erilaisille laitteille ja käyttökohteille. Yritysmaailmaan kohdennetut toiminnot ja yleinen käytettävyyden heikkous on pitänyt ne hieman marginaaliyleisön mielenkiinnon kohteena. Myös Microsoftin sulautettujen laitteiden käyttöjärjestelmä, Windows CE, on saattanut haitata yrityksen mobiilikäyttöjärjestelmien suosiota samankaltaisella ulkonäöllään ja hankalalla käytettävyydellään. Microsoftin korjatessa nämä käytettävyys- ja imago-ongelmat, suosio kääntyi kasvuun ja sen ennustetaan vain kasvavan. Vuonna 2014 Windows Phonen maailmanlaajuinen markkinaosuus oli vain 2,7 % johtavan Androidin osuuden ollessa 82,3 %. Ennustus vuoden 2018 osuuksista lupaa Windows Phonelle 5,6 % markkinaosuutta, Androidin ja iOS:n laskiessa prosentilla parilla. (IDC 2014).

Seuraavassa lyhyesti Microsoftin mobiilikäyttöjärjestelmien historiaa.

Windows Mobile kehitettiin Pocket PC:lle jo vuonna 2000 nimellä Pocket PC 2000. (Brushan, C. 2013). Seuraavien versioiden kehitys oli hajanaista ja niiden nimet olivat toisistaan eriäviä, mutta ominaisuuksissa oli laajalti yhteneväisyyksiä. Viimeisin versio oli Windows Mobile 6.5, joka julkaistiin 2009. (Brushan, C. 2013).

Windows Phone- käyttöjärjestelmän ensimmäinen versio oli Windows Phone 7 ja se julkaistiin vuonna 2010. (Nevala, J. 2013. s. 6). Käyttöjärjestelmän kehitys ei perustunut edelliseen raskaaseen versioon, vaan se aloitettiin täysin alusta. Sovellusten yhteensopivuus katkaistiin myös versioiden välillä. (Nevala, J. 2013. s. 6). Versiossa esiteltiin myös suuren suosion saavuttanut aloitusruutu tiileineen ja yksinkertaisine värimaailmoineen. Version myötä käyttöjärjestelmän käytettävyys hyppäsi aivan uudelle tasolle.

Windows Phone 7 päivitysten, Mangon ja Tangon, jälkeen julkaistiin vielä versio 7.8, joka oli sekoitus Windows Phone 7:ää ja 8:aa. Osa ominaisuuksista oli jo Windows Phone 8:n tulevia ominaisuuksia. Sitä ei kuitenkaan voinut päivittää Windows Phone 8:ksi laitteiston yhteensopivuusongelmien vuoksi. (Brushan, C. 2013).

Windows Phone 8 julkaistiin vuonna 2012 koodinimellä Apollo. Silmäänpistävin ominaisuus oli aloitusnäytön muunneltavan kokoiset kuvakkeet, eli ”live tiles”. Versio 8 esitteli Internet Explorer 10- selaimen, NFC- kosketus- sekä ohjelmien moniajo-ominaisuudet. (Brushan, C. 2013).

2.1.2 Windows Phone 8.1 uudet ominaisuudet

Windows Phone 8.1 julkaistiin kuluttajille heinäkuussa 2014. (Microsoft 2015 A). Päivitys sisälsi uusia odotettuja ominaisuuksia, jotka paransivat hyvää käyttöjärjestelmää entisestään.

Edistyksellisin uusi ominaisuus oli Cortana, henkilökohtainen avustaja, joka auttaa puheluiden, tekstiviestien ja kalentereiden tapaamisten sekä monien muiden asioiden kanssa puheen välityksellä. (Microsoft 2015 B). Se oli saatavilla aluksi vain englantia puhuvissa maissa, mutta hiljalleen levisi myös muihin maihin.

Windows Phone 8.1 aloitusnäytön taustan voi vaihtaa haluamakseen kuvaksi, niin että kuva näkyy ”palapelinä” kuvakkeiden läpi koko ruudun kokoisena. (Microsoft 2015 B). Aloitusnäytön ruutuja voi myös muokata sisältämään lisäsarakkeen, jolloin pieniä ruutuja mahtuu ruudulle enemmän.

Uusi Word Flow- kirjoitustapa esiteltiin Windows Phone 8.1:n myötä. Sen avulla voi kirjoittaa tekstiä nopeasti pyyhkäisemällä sormella sitä välillä nostamatta kirjaimelta toiselle. (Microsoft 2015 B). Kirjoitustapa vaatii totuttelemista, mutta on opittaessa nopea.

Automaattinen WLAN- seuranta auttaa käyttäjää vähentämään dataverkkojen kustannuksia etsimällä avoimia verkkoja ja muodostamalla yhteyden niihin. Se myös hyväksyy automaattisesti mahdolliset käyttöehdot ja syöttää käyttäjän puolesta tarvittavat tiedot niitä vaadittaessa. (Microsoft 2015 B).

Windows Phonesta puuttui tiedostojen käsittelyyn tarkoitettu ohjelma, mutta versio 8.1 tarjosi Tiedostot-sovelluksen uutena ominaisuutena. (Microsoft 2015 B). Sen avulla puhelimen ja muistikorttien tiedostojärjestelmää pystyy selaamaan kuten työpöytä-Windowsin Resurssienhallinnalla.

Windows Phone 8.1:n helppokäyttöisyyttä lisää uusi toimintokeskus, jonka voi avata pyyhkäisemällä sormella ruudun yläreunasta alaspäin puhelimen ruudulla. Oletuksena muokattavassa valikossa ovat linkit WLAN-verkkojen, Bluetooth-yhteyksien sekä lentotilan hallintaan. (Microsoft 2015 B).

Uudistetussa kalenterissa on käytettävissä vuorovaikutteisia näkymiä, joiden avulla tietojen selaaminen on helpompaa. Myös säätiedot löytyvät kalenterin tiedoista. (Microsoft 2015 B). Uusi versio toi tullessaan myös kaivatun viikkonäkymän viikkonumeroineen kalenteriin.

VPN- yhteyden muodostaminen oman yrityksen verkkoon on mahdollista Windows Phone 8.1:n omilla työkaluilla. S/MIME:n avulla Exchange ActiveSync (EAS)- tilien suojaus on korkeammalla tasolla ja työpaikkatilien lisäämisen kautta yritysten yksityiset sovellukset ovat käyttäjän saatavilla helpolla tavalla. (Microsoft 2015 B).

2.1.3 Windows Phone 8.x:n visio

Windows Phone 7:n julkaisun myötä Microsoft teki pesäeron muihin mobiilikäyttöjärjestelmävalmistajiin ulkonäöllään ja käytettävyydellään. Käyttöliittymässä keskityttiin

enemmän sisältöön, kuin kiiltäviin yksityiskohtiin. (Whitechapel, A., McKenna, S. 2013. S.3). Julkaisun aikaan kilpailijoiden käyttöjärjestelmät eivät sallineet tai sallivat vain vähän aloitusnäkymien henkilökohtaista muokkaamista. Uusien Live Tile- kuvakkeiden avulla voitiin käynnistää joko ohjelma tai suorana jokin sen osa. (Whitechapel, A., McKenna, S. 2013. S.3).

Windows Phone 8:n myötä ominaisuudet ovat vain lisääntyneet ja kehittäjille on tarjolla yhä enemmän mahdollisuuksia luoda uusia ohjelmia tai kehittää edelleen jo olemassa olevia Windows Phonen perusominaisuuksia. Microsoftin filosofia käyttöjärjestelmän olemuksesta ei ole uuden version kautta muuttunut. Päinvastoin samaa filosofiaa sovelletaan myös muiden Windows-käyttöjärjestelmien, Office-tuotteiden ja Xbox-pelikoneiden suunnittelussa. (Whitechapel, A., McKenna, S. 2013. S.3).

Käyttöjärjestelmän suunnittelussa haluttiin keskittyä olennaiseen, eli itse suoritettaan tehtävään. Se oli pystyttävä suorittamaan helposti ja keskeytyksettä alusta loppuun saakka ilman muita häiritseviä tekijöitä. Tässä Microsoft käytti vertauksen maanalaisten metrojen toimintaa; satoja ihmisiä käyttää niitä sujuvasti ilman suurempia ongelmia niiden suurten ja selkeiden merkkien ja kylttien ansioista. Suurikokoisissa kylteissä on suurilla ja selkeillä kirjainkooilla ja kuvioilla kerrottu kaikki olennainen, jonka jokainen niistä pystyy helposti tulkitsemaan. (Whitechapel, A., McKenna, S. 2013. S.4). Microsoft haki vaikutteita myös Bauhaus-taide- ja arkkitehtuurikoulun modernista esinesuunnittelusta. Siinä pyritään riisumaan kohteesta pois kaikki epäolennainen ja keskittymään pelkästään olevaisen ytimeen. (Clayton, C. 2014). Samalla tavalla käyttäjän tulee pystyä käyttämään laitettaan helposti, vaikka siinä tapahtuisikin pinnan alla useita eri asioita samaan aikaan.

Windows Phone 8:n typografia muodostuu ainoastaan yhdestä fontista, Segoe WP:stä. Sitä käytetään kaikkialla käyttöjärjestelmässä. Eri asioita painotettaessa eri tavoin, kirjasimen kokoa muunnellaan tarpeen mukaan. (Whitechapel, A., McKenna, S. 2013. S.4). Myös kirjasimien käyttöön sovelletaan samaa vertausta metrojen ja muiden massakuljetusvälineiden merkistöstä; selkeän tasaisen värin ja lihavoidun fontin yhdistelmä on tyylikeino, jonka ihmiset ovat jo omaksuneet, nyt sitä vain käytetään myös asian yh-

teydessä, jota ihmiset haluavat tehdä. (Clayton, C. 2014). Kehitystyökalujen mukana tulee useita eri tyylipohjia, joiden avulla sovellukset saavat tyylikkään ja yhtenäisen ulkonäön helposti.

Liike on myös yksi suunnittelun peruspilareista. Hyvin suunniteltu liike ei pelkästään miellytä silmää, vaan auttaa käyttäjää saavuttamaan tavoitteensa helpommin. (Microsoft 2015 C). Microsoft on luottanut aikaisten edelläkävijöiden, kuten Saul Bass:in, esimerkkeihin. Hän käytti graafista suunnittelua, erilaisia typografioita ja liikettä luoden tyylipuhtaita ja tunteellisia aloitustekstejä elokuvilleen. (Clayton, C. 2014). Käyttöjärjestelmää käytettäessä huomataan, kuinka sulavasti liike sitoo eri sivut ja kokemukset yhteen. Sivujen välillä siirryttäessä, sormella näyttöä pyyhkäisemällä puolelta toiselle, käyttäjä huomaa sulavan animaation, joka näytetään aikana, jolloin käyttäjä ei voisi laitetta käyttää sen ladatessa jo seuraavaa sivua. Sen lisäksi animaatio toimii luonnollisena siltana kahden mahdollisesti erilaisen sivun ja käyttäjäkokemuksen välillä. (Whitechapel, A., McKenna, S. 2013. S.4).

2.2 Windows Phone 8.1 sovelluskehitys

2.2.1 Median toisto

Median toisto Windows Phone 8.1- käyttöjärjestelmässä tapahtuu käyttämällä MediaElement- luokkaa. Se periytyy FrameworkElement- luokasta, joka puolestaan periytyy UIElement- luokasta. (Microsoft 2015 D). Se tarjoaa useita metodeja ja ominaisuuksia ääni- ja kuvalähteiden toistoon. Tuettuja tiedostomuotoja listalta ovat MPEG-4, MPEG-2, ASF, ADTS, MP3, WAV, AVI ja AC-3 eri tiedostopäätteineen. (Microsoft 2015 E). Taulukossa 1 on listattu yleisimpiä MediaElement- luokan ominaisuuksia sekä taulukossa 2 luokan metodeja.

Taulukko 1. MediaElement-luokan ominaisuuksia. (Microsoft 2015 F).

Ominaisuus	**Kuvaus**
AutoPlay	Toiston automaattinen aloitus. Oletus on True.
Volume	Äänen voimakkuus välillä 0- 1.
Balance	Äänen tasapaino välillä -1 – 1. Oletus on 0.

IsLooping	Määrittää toiston alkamaan uudelleen alusta, kun toisto päättyy.
Source	Lähde URI ääni- tai kuvalähteelle.
Stretch	Määrittää MediaElementin käyttämän tilan sille määritetyn sijainnin kokonaistilasta.
DefaultPlaybackRate	Määrittää median toiston nopeuden. Oletus on 1.0, eli normaali toiston nopeus. Arvo voi olla myös negatiivinen, jolloin toisto tapahtuu takaperin

Taulukko 2. MediaElement-luokan metodeja. (Microsoft 2015 D).

Metodi	**Kuvaus**
Play	Toistaa mediaa sen nykyisestä toistokohdasta alkaen. Jos olion tila on ”paused”, toisto jatkuu sen hetkisestä kohdasta. Jos ääni- tai kuvalähde on juuri avattu tai toisto on lopetettu, toisto alkaa alusta.
Pause	Pysäyttää toiston säilyttäen sen hetkisen tilan tiedon.
Stop	Pysäyttää toiston ja asettaa toiston tilan alkuun (0).
SetSource	Asettaa Source- ominaisuuden arvon annettuun parametriin.

MediaElement-olion ääni- tai kuvalähde lähde asetetaan joko Source-ominaisuuden tai SetSource metodin avulla. Source-ominaisuuden tyyppi on Uri, jossa määritetään polku toistettavaan tiedostoon. SetSource-metodi hyväksyy IRandomAccessStream-lähteen, joka on medialähde. Kohdassa 2.2.3 käsiteltävän FileOpenPicker-luokan avulla saadaan helposti stream-olio tiedostolle paikallisessa tiedostojärjestelmässä tai Microsoftin OneDrivella. (Microsoft 2015 D).

Akun kestoa ja muuta toiminnallisuutta parantaa äänen laitteistopohjainen purkaminen (hardware audio offloading). Se voidaan määrittää käytettäväksi AudioCategory-ominaisuuden avulla, asettamalla arvoksi joko ForegroundOnlyMedia tai BackGroundCapableMedia. (Microsoft 2015 G). Hyvänä tapana voidaan pitää myös median

varaamien resurssien vapauttamista aina median käytön päätyttyä, jolloin sovelluksen muistijalanjälki pienenee huomattavasti.

Windows Phone- sovelluksessa MediaElement- olioita voi olla olemassa vain yksi kerrallaan, joten jos tarkoitus on toistaa useita ääniä, kuten esimerkiksi peleissä tehdään, MediaElement-luokan kanssa se ei onnistu. (Microsoft 2015 F). Kun sovellus vaatii useita eri äänilähteitä tai äänilähteen nopeaa uudelleen toistamista, johon MediaElement-luokka ei aina kykene, tulee median toistoon käyttää SharpDX-kirjastoa ja sen XAudio2-pakettia. (Hoekstra, M. 2013). Se koostuu C++:lla kirjoitetusta DirectX-koodista, joka on paketoitu C#:ia varten. Käyttö onnistuu helposti asentamalla SharpDX.XAudio2-paketti Visual Studion NuGet- paketinhallintakonsolilla käskyllä 'Install-Package SharpDX.XAudio2'. Tämän jälkeen omaan projektiin lisätään XAudio2-instanssi, jolle kerrotaan toistettavan WAV-tiedoston polku. Kirjasto sopii parhaiten pelien kehitykseen, mutta se toimii hyvin myös äänitehosteiden ja muiden huomioäänien kanssa. SharpDX.XAudio2-nimiavaruuden SourceVoice-luokan metodeista Start() ja Stop() soveltuvat parhaiten niiden toistoon. (SharpDX.org. 2012).

2.2.2 Asentosensorit

Sensorit ilmaisevat käyttöjärjestelmälle muutoksia laitteen suhteessa ympäröivään maailmaan. Windows Phone pystyy hyödyntämään kompassia, kiihtyvyysanturia ja gyroskooppia. (Falafel Software. 2013. S. 266). Näitä sensoreita voidaan hyödyntää sovelluksissa tai peleissä mielekkäillä tavoilla. Sensoreita edustavat luokat Compass, Accelerometer ja Gyroscope periytyvät SensorBase<T>-luokasta. (Whitechapel, A., McKenna, S. 2013. S. 271).

Kiihtyvyysanturi raportoi laitteen fyysisen kiihtyvyyden muutoksista käyttäen X, Y ja Z orientaatiota. (Whitechapel, A., McKenna, S. 2013. S. 238). Aina muutoksen tapahtuessa, se ilmoittaa tapahtumasta nostaen eventin. Kiihtyvyysanturia vastaa Accelerometer-luokka, joka löytyy Microsoft.Devices.Sensors-nimiavaruudesta. Tärkein event on ReadingChanged, joka tapahtuu aina kiihtyvyysanturin raportoidessa uuden lukeman. (Microsoft 2015 H).

Accelerometer-luokan metodeja kuvataan taulukossa 3.

Taulukko 3. Accelerometer-luokan metodeja. (Microsoft 2015 H).

Metodi	Kuvaus
GetCurrentReading	Lukee anturin sen hetkisen arvon.
GetDefault	Lukee laitteen oletus-kiihtyvyysanturin.

Accelerometer-luokan ominaisuuksia luetellaan taulukossa 4.

Taulukko 4. Accelerometer-luokan ominaisuuksia. (Microsoft 2015 H).

Ominaisuus	Kuvaus
DeviceID	Laitteen id-numero.
MinimumReportInterval	Pienin laitteen tukema raportointiväli.
ReadingTransform	Laitteeseen sovellettavan muunnoksen asetus.
ReportInterval	Kiihtyvyysanturin arvon raportointiväli. Tämän ominaisuuden avulla voidaan säätää sensorin herkkyyttä. Kaikki sovellukset eivät tarvitse nopeinta näytteenottotaajuutta.

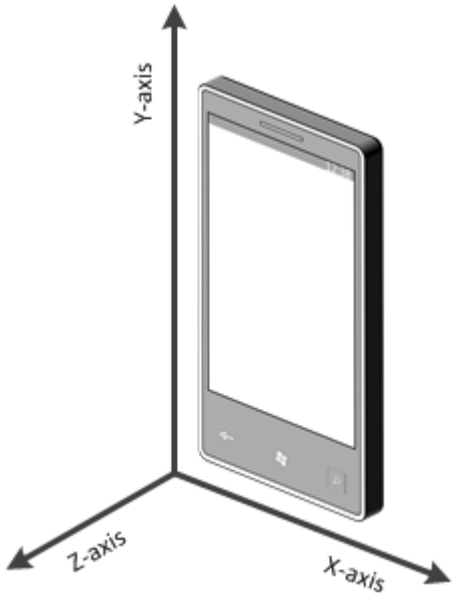
Accelerometer-luokan tarjoama arvo voidaan lukea AcclerometerReading-luokan olioon esimerkiksi tapahtuman AccelerometerReadingChanged yhteydessä AccelerometerReadingChangedEventArgs-olion Reading-ominaisuuden kautta. (Microsoft 2015 I).

AccelerometerReading-luokan ominaisuuksia listataan taulukossa 5.

Taulukko 5. AccelerometerReading-luokan ominaisuuksia. (Microsoft 2015 I).

Ominaisuus	Kuvaus
AccelerometerX	Sisältää X-akselin g-voiman kiihtyvyyden
AccelerometerY	Sisältää Y-akselin g-voiman kiihtyvyyden
AccelerometerZ	Sisältää Z-akselin g-voiman kiihtyvyyden
Timestamp	Aikaleima raportoidulle lukemalle

X-, Y- ja Z- akselit ovat esitettynä kuvassa 1.



Kuva 1. Laitteen koordinaatiojärjestelmä. (Kuva Windows Phone Dev Center. 2015)

2.2.3 Tiedostojen avaaminen

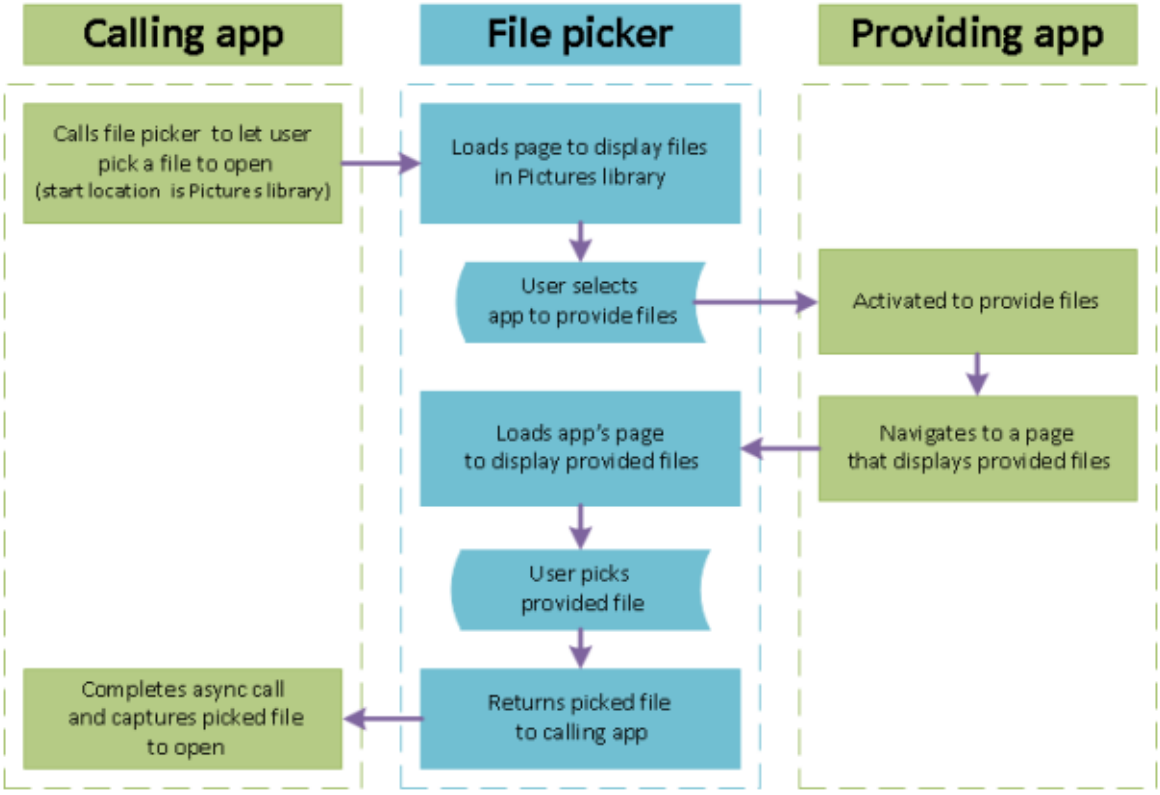
Tiedostojen avaaminen sovelluksesta käsin onnistuu käyttämällä tiedoston valitsijaa (file picker). Se toteutetaan FileOpenPicker-luokan avulla. (Microsoft 2015 L). Luokan avulla saadaan suodatettua näkyviin halutut tiedostotyypit (ominaisuus FileTypeFilter) sekä valittua haluttu tiedostonäkymä, esimerkiksi Musiikki-kirjasto (ominaisuus SuggestedStartLocation).

Tiedostoa valittaessa sovelluksen suoritus keskeytyy ja käyttäjälle avautuu käyttöjärjestelmän standardi tiedostoselain. (Microsoft 2015 J). Muistinhallinnan toiminnan vuoksi laitteissa, joissa on vain vähän muistia käytettävissä, sovellus voidaan terminoida kokonaan. Käyttämällä seuraavia FileOpenPicker-luokan metodeja tämä voidaan välttää ja sovellus palautuu valinnan jälkeen normaalitilaan:

Taulukko 6. FileOpenPicker-luokan metodeja. (Microsoft 2015 J).

Tehtävä	**Metodi**
Tiedoston valinta	PickSingleFileAndContinue
Tiedostosijainnin valinta	PickSaveFileAndContinue
Kansion valinta	PickFolderAndContinue

Sovelluksessa voidaan haluta esimerkiksi avata musiikkitiedosto toistettavaksi. Sovellus itse toimii tiedoston valitsijaa kutsuvana sovelluksena (Kuva 2). Sovelluksen painiketta painettaessa sen suoritus keskeytyy ja ruudulle avautuu standardi tiedostoselain, jonka avulla musiikkitiedosto voidaan valita. Kun valinta tehdään, sovellukselle palautetaan StorageFile- olio. (Microsoft 2015 K).



Kuva 2. Tiedoston valitsijan toiminta (Kuva Microsoft 2015 K).

Tiedoston valinnan jälkeen sovelluksen on osattava jatkaa toimintaansa saamansa StorageFile-olion kanssa. Tässä auttavat SuspensionManager-luokka ja helper-luokka ContinuationManager. SuspensionManager tallentaa sovelluksen tilan, joka vallitsee ennen sovelluksen suorituksen keskeyttämistä ja tiedoston valinnan jälkeen palauttaa sen samaan tilaan. ContinuationManager muistaa, mitä käyttäjä oli tekemässä ja jatkaa SuspensionManagerin avulla sovelluksen suoritusta. (Microsoft 2015 M). SuspensionManager on kopioitavissa uuden Windows Phone 8.1- projektin Common-kansiosta omaan projektiin. Uuden projektin tulee pohjautua johonkin muuhun mallipohjaan,

kuin Blank App. ContinuationManager-luokan koodi on saatavissa MSDN:stä. (Microsoft 2015 M).

2.2.4 Sivujen välillä siirtyminen

Jos sovelluksessa on useita sivuja, niiden välillä tulee pystyä navigoida. Navigointia voi verrata websivujen välillä siirtymiseen. Visual Studiossa voidaan erityyppisten mallipohjien (Hub Page, Split Page jne.) avulla aloittaa kehitys valmiilta pohjalta, jossa on sivujen tai näkymien välillä siirtyminen jo valmiiksi koodattuna. Yksinkertaisimmillaan sivujen välillä siirtyminen tapahtuu luomalla ensimmäiselle sivulle XAML:lla uusi linkkipainike (kuva 3) ja liittämällä siihen tapahtuman, jonka Frame.Navigate-metodi siirtää näkymän toiselle sivulle (kuva 4). (Microsoft 2015 N).

XAML Copy

```
<StackPanel Grid.Row="1"
            Margin="120,0,120,60">
    <HyperlinkButton Content="Click to go to page 2" Click="HyperlinkButton_Click"/>
</StackPanel>
```

Kuva 3. Linkkipainikkeen luonti. (Kuva Microsoft 2015 N).

C# VB Copy

```
private void HyperlinkButton_Click(object sender, RoutedEventArgs e)
{
    this.Frame.Navigate(typeof(BasicPage2));
}
```

Kuva 4. Frame.Navigate-metodi tapahtumassa. (Kuva Microsoft 2015 N).

Frame-luokka on vastaa navigoinnista tarjoamalla käyttöön metodeja, kuten Navigate, GoBack ja GoForward. (Microsoft 2015 N). Niiden avulla sisältöä näytetään Frame-oliossa. Kun sovellus käynnistyy, App.OnLaunched-metodi luo uuden Frame-olion ja välittää sille Navigate-metodin avulla halutun aloitussivun, esimerkiksi BasicPage1:n. Sen jälkeen metodi määrittää sovelluksen sen hetkiseen ikkunaan näytettäväksi luodun

Frame-olion. Tämän vuoksi sovelluksen sen hetkinen ikkuna sisältää Frame-olion, johon on määritetty haluttu aloitussivu näytettäväksi. (Microsoft 2015 N):

Edellisessä kappaleessa mainittu SuspensionManager-luokka toimii avustajana myös sivun välillä siirryttäessä. Jos sovelluksesta halutaan sulavasti toimiva, eri sivujen tiloista täytyy tallentaa tietoa johonkin. Kun sovellus käynnistetään, uusi Frame-olio luodaan. Se tulee rekisteröidä heti ja rekisteröinnin suorittaa SuspensionManager. RegisterFrame- metodi. Tila tallennetaan SessionState-nimiseen dictionary-tietorakenteeseen, joka toimii avain-arvo-pohjaisesti. Täten jokaiselle Frame-oliolle luodaan oma yksilöllinen avain, jonka arvoksi määritetään FrameState, johon puolestaan tallennetaan Frame-olion ikkunoiden tila. (Microsoft 2015 N). Kun tila tulee tallentaa tietorakenteeseen, käytetään SaveAsync-metodia ja tilaa palautettaessa puolestaan RestoreAsync- metodia. SaveAsync-metodi tallentaa koko SessionStaten paikalliseen tiedostoon ApplicationData-määrityksen mukaisesti. Vastaavasti tiedosto voidaan lukea muistiin tilaa palautettaessa. (Microsoft 2015 N).

Sivujen välillä siirryttäessä tietoa tulee joskus säilyttää sivujen muistissa. Jos käyttäjä haluaa palata edelliseen ikkunaan muokkaamaan syöttämäänsä tietoa tekstikentässä, sivun tulee tarjota alkuperäinen tieto muuttumattomana sivujen välillä takaisin siirryttäessä. Jotta tekstikenttä ei olisi tyhjä takaisin palattaessa, sivun konstruktoriin määritetään arvo ominaisuudelle NavigationCacheMode, eli navigaation välimuistitus. Kuvassa 5 NavigationCacheMode otetaan käyttöön.

C# VB Copy

```
public BasicPage1()
{
    this.InitializeComponent();
    ...
    this.NavigationCacheMode =
        Windows.UI.Xaml.Navigation.NavigationCacheMode.Enabled;
}
```

Kuva 5. NavigationCacheMode.Enabled. (Kuva Microsoft 2015 N).

2.2.5 Ilmoitukset

Sovelluksen suorituksen aikana käyttäjä tekee valintoja, esimerkiksi haluaa lopettaa musiikkitiedoston toiston kesken sovelluksen suoritusta. Sovelluksen tulisi tässä tapauksessa pystyä esittämään dialogi, jossa käyttäjältä varmistetaan toiminne, eli halutaanko toisto varmasti lopettaa. Ilmoitusten esittämiseen Windows Phone 8.1:ssä on useita eri luokkia, esimerkiksi Flyout, MenuFlyout, ToolTip, MessageDialog ja Popup.

MessageDialog-luokan avulla voidaan esittää dialogeja käyttäjällä. Dialogi mahdollistaa maksimissaan kolmen eri käskyn näytön. Dialogi keskeyttää ohjelman suorituksen ja pimentää dialogin taustalla näkyvän näytön. Samalla kaikki kosketusnäytön toiminnot poistetaan käytöstä, kunnes käyttäjä tekee valinnan dialogissa. (Microsoft 2015 O).

MessageDialog-luokkaa tulisi käyttää harvakseltaan. Vain kriittisiä ja ohjelman normaalin toiminnan keskeyttäviä toimintoja tulee hyväksyttää luokan avulla. (Microsoft 2015 O). Ilmoitusten ja muun normaalin tiedon näyttöön kannattaa käyttää ToolTip- tai Popup-luokkia.

2.3 Windows Phone- sovelluksen elinkaari

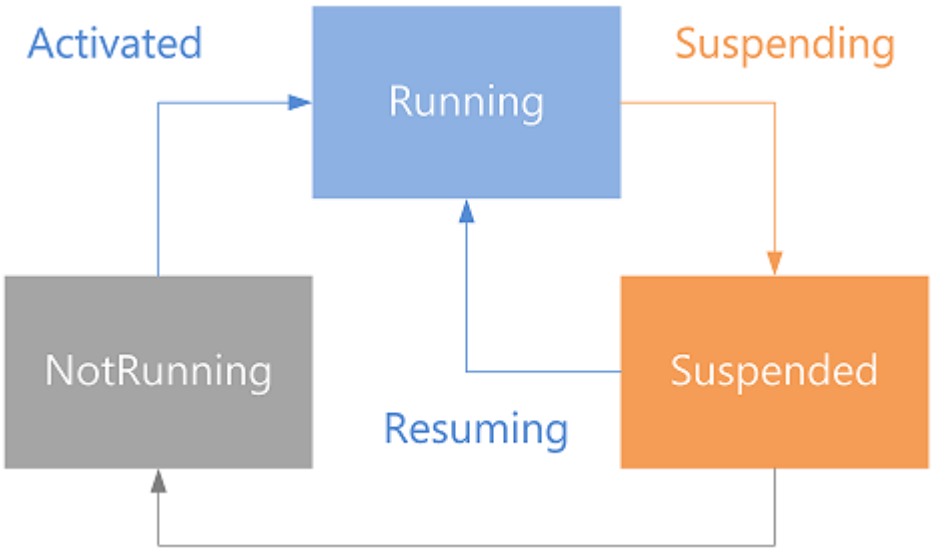
Sovelluksen elinkaari muodostuu useista tarkoin määritellyistä vaiheista. Sen tulee pystyä palautumaan edelliseen tilaansa esimerkiksi vapaan muistin loputtua, jolloin käyttöjärjestelmä lopettaa sovelluksen suorituksen pakotetusti. Edellä mainittua tiedostonvalitsijaa käytettäessä sovellus keskeyttää suorituksensa käyttöjärjestelmän tiedostoselaimen käytön ajaksi. Keskeytetystä tilasta palataan takaisin normaaliin sovelluksen suorituksen tilaan.

Windows Phone sovelluksen elinkaaren eri vaiheita ovat asennus, sovelluksen käynnistys, sovelluksen aktivointi, sovelluksen keskeytys, sovelluksen palautuminen, sovelluksen sulkeminen, sovelluksen kaatuminen sekä sovelluksen asennuksen poisto. (Microsoft 2015 P).

Sovelluksen elinkaari alkaa asennuksesta. Sovellus voidaan asentaa käyttäjän taholta tai sovelluskehittäjän toimesta Visual Studion kautta paikallisesti. (Microsoft 2015 P).

Asennuksen jälkeen sovellus voidaan käynnistää, jolloin se myös aktivoidaan. Käynnistyksen aikana käyttäjälle näytetään splash screen eli aloituskuva. Kuvan näytön aikana sovellus tarkastaa ympäristön ja varmistaa, että kaikki on kunnossa sen suoritusta varten. Samalla tarkastetaan syy, miksi sovellus aktivoidaan ja mistä tilasta sovellus siirtyy Running-tilaan. (Microsoft 2015 P). Näin saadaan selville myös oliko sovellus jo ennestään käynnissä taustalla. Aktivointi voi tapahtua myös muulla tavalla kuin kuvaketta klikkaamalla; tiedostotyyppiin sidotun tiedoston avauksen kautta, tulostuspyynnön kautta, protokollaan sidotun yhteyden kautta tai esimerkiksi kameran käytön kautta. (Microsoft 2015 P).

Sovelluksen suoritus voi keskeytyä monella eri tavalla. Jos käyttäjä siirtyy aloitusvalikkoon kesken sovelluksen suorituksen tai laite siirtyy virransäästö-tilaan, sovellus yleensä vaipuu suspended-tilaan. Tässä keskeytetyssä tilassa ollessaan, sovelluksen tulisi vapauttaa käyttämänsä resurssit muiden sovellusten käyttöön tallentamalla tilansa SuspensionManagerin avulla. Kun sovellukseen siirrytään uudelleen, sen tulisi palautua 1- 10 sekunnissa normaalitilaan. Jos näin ei käy, käyttöjärjestelmä tulkitsee sovelluksen kaatuneen ja terminoi sen. (Microsoft 2015 P). Käyttöjärjestelmä yrittää pitää mahdollisimman monta samanaikaista Suspended-tilaista sovellusta muistissaan. Jälleen kerran muistin uhatessa loppua, sovelluksia voidaan terminoida lopullisesti.



Kuva 6. Sovelluksen suorituksen tila. (Kuva Microsoft 2015 P).

Windows Phone- sovelluksiin ei voida lisätä perinteistä toimintoa, jolla sovellus voidaan sulkea. Windows App Certification- prosessi ei hyväksy sovellusta, jossa sellainen sovelluksessa on. Yleensä käyttäjien ei tarvitse erikseen sulkea käyttämiään sovelluksia, riittää että palataan aloitusvalikkoon ja annetaan käyttöjärjestelmän huolehtia muistinhallinnasta. (Microsoft 2015 P). Kun sovellus siirtyy NotRunning-tilaan, se ei enää näy näytöllä eikä Task Managerissa, mutta sitä ei kuitenkaan terminoida lopullisesti tässä vaiheessa. Sen vuoksi eri tapahtumien avulla sovellukselle voidaan määrittää toimintoja myös sulkemisen jälkeen. Tämä ei toimi, jos ohjelma sulkee itse itsensä; käyttöjärjestelmä käsittelee silloin ohjelmaa, kuin se olisi kaatunut. (Microsoft 2015 P).

Kaatuneiden sovellusten tulee soveltaa ”system crash experience”- proseduuria, joka palauttaa näkymän aina takaisin aloitusnäyttöön mahdollisimman nopeasti. (Microsoft 2015 P).

Sovelluksen elinkaari päättyy asennuksen poistoon. Jos sovelluksen avulla käsiteltiin käyttäjän tietoja, ne säilyvät koskemattomina jos tiedot sijaitsivat yleisissä tiedostosijainneissa, kuten Dokumentit-, Kuvat- tai Musiikki-kirjastoissa. (Microsoft 2015 P).

2.4 UX suunnittelu

2.4.1 Sovelluksen värit

Värien avulla voidaan korostaa jonkin tarkoitusta, tunnelmaa tai tunnetilaa. Niiden avulla voidaan myös havainnollistaa jonkin yksittäisen brändin läsnäoloa. Windows Phonen kanssa värejä käytetään luomaan graafisia rakenteita. (Microsoft 2015 Q).

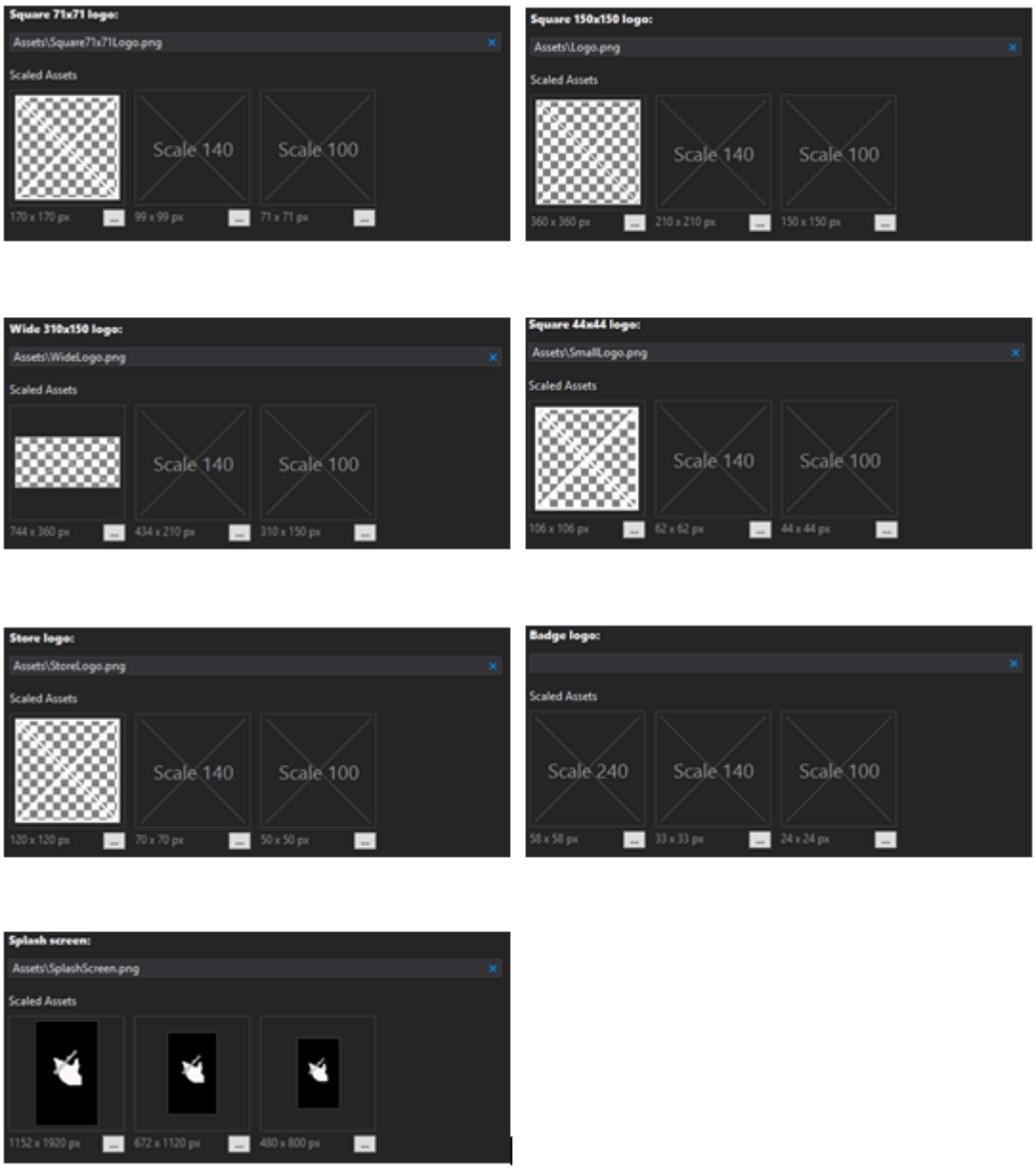
Windowsissa käytetään tasaisia ja puhtaita värejä ilman liukuvärjäyksiä. Syvyyttä sovellusten ikkunoihin saadaan käyttämällä läpinäkyvyyttä, kerroksia ja sekoituksia. Windowsin käyttöjärjestelmän värimaailma eroaa sovellusten värimaailmasta, joihin sovelluskehittäjät voivat luoda täysin omansa korostaakseen sovelluksen käyttötarkoitusta. Käyttöjärjestelmän värit ovat käyttäjän valittavissa ennalta määrätystä valikoimasta. (Microsoft 2015 Q).

Visuaalinen yksinkertaisuus, joka seuraa puhtaiden värien rohkeasta käytöstä, auttaa käyttäjää sovelluksien sisällä navigoinnissa. Myös aloitusnäkymän ruutujen mehukkaat värit kutsuvat käyttäjää kokeilemaan niiden toimintoja. (Microsoft 2015 Q).

2.4.2 Sovellusten kuvat

Windows Phone- sovelluksissa on useita erillisiä kuvia, osa pakollisia ja osa valinnaisia. Kuvien avulla sovelluksen identiteetti pääsee hyvin esille. Niiden avulla sovellusta voidaan markkinoida tehokkaasti. (Microsoft 2015 R). Hyvin suunniteltujen kuvien avulla sovellus erottuu massasta.

Visual Studio auttaa tarvittavien kuvien luonnissa kertomalla niiden eri koot ja tarkoitukset. Vaadituista kuvista löytyvät esimerkkikuvat jo valmiina suoraan Windows Phone 8.1- sovelluksen mallipohjasta.



Kuva 7. Visual Studion eri kuvat. (Kuva Visual Studio 2013).

Myös Windows App Certification Kit huolehtii ennen julkaisua vaadittavien kuvien tarkastuksesta. Microsoftin määrittelemät, julkaistavan sovelluksen vähimmäisvaatimuksena tarvittavat kuvat ovat on lueteltu taulukossa 7.

Taulukko 7. Sovelluksen julkaisun vähimmäisvaatimuksia kuvien osalta. (Microsoft 2015 S).

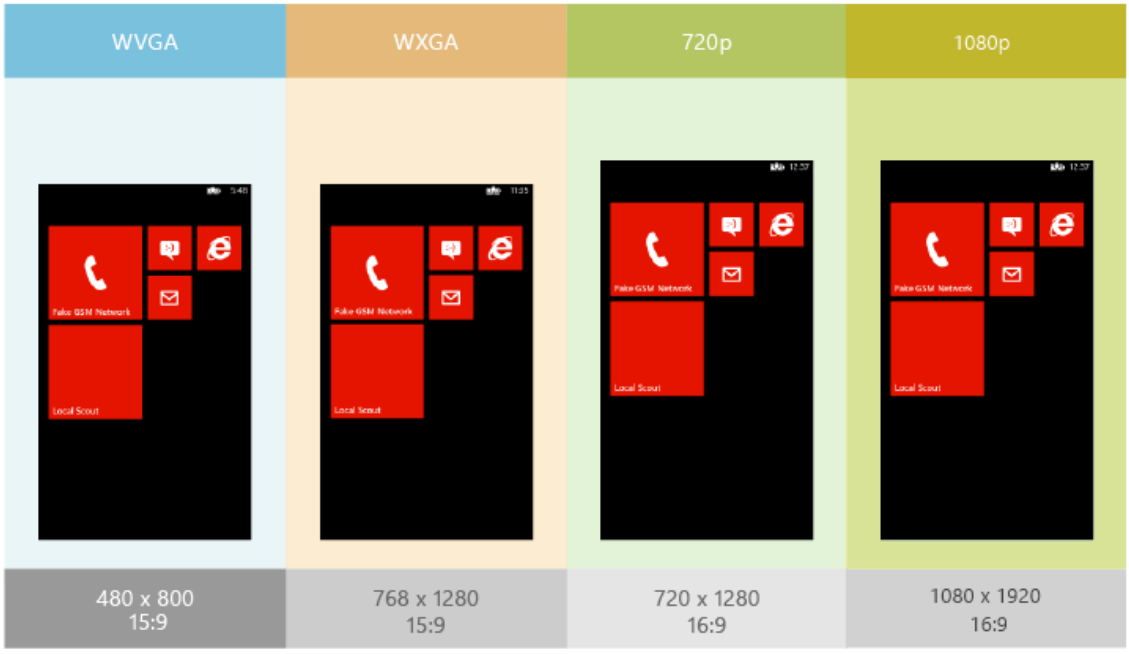
Kuva	**Vaatimukset**
Sovelluksen kuvankaappaukset	Vähintään yksi, maksimissaan 8 kuvaa kaikista tuetuista näyttötarkkuuksista

Sovelluksen kuvat	Oletus ruutu-kuvat, sovelluslistakuva ja Windows Phone Store- sovelluskuva, joka lähetetään Dev Centerin kautta
Sovelluksen paketin (XAP) kuvat	Sovelluslistan kuva: 99x99 pikselin PNG-kuva, oletusruutukuvat: PNG- tai JPEG-kuvat pienelle ja keskikokoisille ruuduille
Windows Phone Storen kuvat	300x300 pikselin kokoinen läpinäkymätön PNG-kuva

Aloitusnäytön ruutujen (eng. Tile) avulla käyttäjä saa sovelluksesta tietoja, kun se ei ole käynnissä. Ruudut voivat olla staattisia tai live-ruutuja, joiden esittämää informaatiota päivitetään ilmoituksilla. (Microsoft 2014 A). Sovelluksen Manifest-tiedostossa määritetty oletusruutu on aina staattinen ja se koostuu määritetystä kuvasta. Ruutuja on kolmea eri kokoa; pieni, keskikokoinen ja leveä. Leveä-versio ei ole pakollinen. Näistä eri kooista käyttäjä voi valita haluamansa aloitusnäytölleen. (Microsoft 2014 A). Sovelluksen dynaamisia ruutuja on eri kokojen lisäksi kolmea eri tyyppiä; flip, iconic ja cycle. Ne eroavat toisistaan ulkonäöltään; flip-tyypin toinen puoli on yksivärinen, joka sisältää tekstiä, iconic-tyyppi on perinteinen Windows 8-ruutu sekä cycle-tyyppi, jonka kuvat vaihtuvat, kuten PowerPoint-esityksessä.

2.4.3 Kuvien koot eri laitteille

Windows Phone tukee neljää eri näytöntarkkuutta; WVGA, WXGA, 720p ja 1080p (kuva 8). Kuvien suunnittelussa eri laitteille, näytöntarkkuudet on otettava huomioon luomalla jokaiselle tarkkuudelle omat tiedostonsa jokaisesta sovellukseen liitettävästä kuvasta. Tämä sen vuoksi, jotta suunniteltu näkymä pysyisi alkuperäisen muotoisena ilman skaalauksen aiheuttamaa venymistä tai tarkkuuden vääristymistä.



Kuva 8. Windows Phonen tukemat resoluutiot. (Microsoft 2015 T).

Visual Studio 2013:ssa on sisäänrakennettu toiminto sekä korkeakontrastisten kuvien että kuvakoon skaalauksen automaattiseen valintaan. (Microsoft 2015 U). Kuvat saadaan valittua laitteeseen sopivaksi nimeämällä ne kokonsa mukaisesti skaalaussuhteen avulla taulukon 8 mukaisesti.

Taulukko 8. Kuvien nimeäminen skaalaussuhteen mukaan. (Microsoft 2015 U).

Kuva	**Suhde**	**Koko**
logo.scale-100.png	100 %	Alkuperäinen koko, 100x100 pikseliä, 96dpi
logo.scale-140.png	140 %	140x140 pikseliä
logo.scale-240.png	240 %	240x240 pikseliä

Alkuperäisen kuvan nimen ollessa logo.png, siihen viitataan koodissa ilman skaalaussuhteen lisäämistä (kuva 9), jolloin ajon aikana Windows Phone valitsee parhaan mahdollisen kuvakoon esitettäväksi käytettävän laitteen ominaisuuksien mukaan. Ominaisuudet, joita vertaillaan, ovat näytön fyysinen koko, näytöntarkkuus, kuvapisteiden määrä tuumalla ja kuvasuhde. (Microsoft 2015 V).

Kuva 9. Kuvalähteeseen viittaaminen XAML-koodissa. (Microsoft 2015 U).

Kuvien kokoa ei kannata muokata skaalaamalla valmista kuvaa, vaan ne tulisi luoda oikean kokoisiksi erikseen. Skaalaamalla kuvista tulee helposti epätarkkoja ja siksi rumia. Myös suurikokoisesta ja tarkasta kuvasta pienemmäksi skaalaamalla, kuvien suhteet häiriintyvät helposti ja ne menettävät alkuperäisen ulkonäkönsä. (Microsoft 2015 U). Esimerkiksi kuvat, jotka sisältävät tekstiä, jonka reunat ovat eriväriset, kuin muu osa tekstiä, muuntuvat helposti epäsuhtaisiksi eriväristen reunojen tullessa kuvaa pienennettäessä suhteessa liian paksuiksi. Myöskään kuvakokoja, jotka eivät ole 5 pikselin kerronnaisia, ei tulisi käyttää, koska skaalatessa kuvia, niihin muodostuu pikseleiden siirtymiä. (Microsoft 2015 U).

3 Markkinointi ja ansaintamallit

Tässä kappaleessa tutustutaan Windows Phone- sovellusten erilaisiin rahoitusmalleihin, eli siihen, kuinka sovelluksia julkaisemalla voidaan ansaita rahaa Windows Phone- kaupassa. Eri rahoitusmalleja ovat maksullinen sovellus, sovelluksien sisäiset maksut ja mainokset. Sovelluksia voidaan julkaista myös testiversioina, jotka ovat ominaisuuksiltaan riisuttuja, mutta käyttäjille ilmaisia. Näin sovellusta ja sen mahdollisia kiinnostuksen kohteita käyttäjille saadaan esiteltyä paremmin, kuin pelkästään kuvailemalla toimintoja sanallisesti ja kuvankaappausten avulla. (Whitechapel, A., McKenna, S. 2013. S. 811).

Microsoftin kehittäjille suuntaaman developer dashboardin (Microsoft 2015 X) kautta kehittäjä voi muokata sovelluksensa kaikkia julkaisuun liittyviä asetuksia, seurata kuinka paljon latauksia sovellus on kerännyt sekä kuinka paljon latauksien tuotot ovat yhteensä. (Microsoft 2015 Y).

3.1 Maksullisen sovelluksen trial-versio

Maksullisen version rinnalla julkaistava ilmainen trial-versio on hyvä tapa tutustuttaa käyttäjäkunta julkaistuun sovellukseen. Sillä on kuitenkin myös haittapuolensa kehittäjän kannalta; kehittäjän on ylläpidettävä kahta erillistä versiota sovelluksesta, jolloin usein myös päivitys trial-versiosta täyteen versioon hankaloituu tai voi uupua kokonaan. (Whitechapel, A., McKenna, S. 2013. S. 823).

Windows Phone- ympäristö tarjoaa tähän ratkaisun trial mode- tilallaan. Sen avulla yksittäinen sovellus voidaan julkaista kerrallaan sekä trial- että täytenä versiona. (Whitechapel, A., McKenna, S. 2013. S. 823). Kehittäjä voi päättää täysin itsenäisesti, mitä osia sovelluksestaan voidaan käyttää trial-versiossa; onko se toiminnaltaan rajoitettu ajallisesti vai toiminnallisesti, sekä voidaanko sovelluksen tietyn hetkinen tila siirtää sellaisenaan täyteen versioon, kun sovellus päivitetään. (Microsoft 2015 Z). Kun sovellusta julkaistaessa valitaan trial mode app- valinta, käyttäjät voivat valita kokeiluvaihtoehdon sovelluksen sivulla Windows Phone- kaupassa.

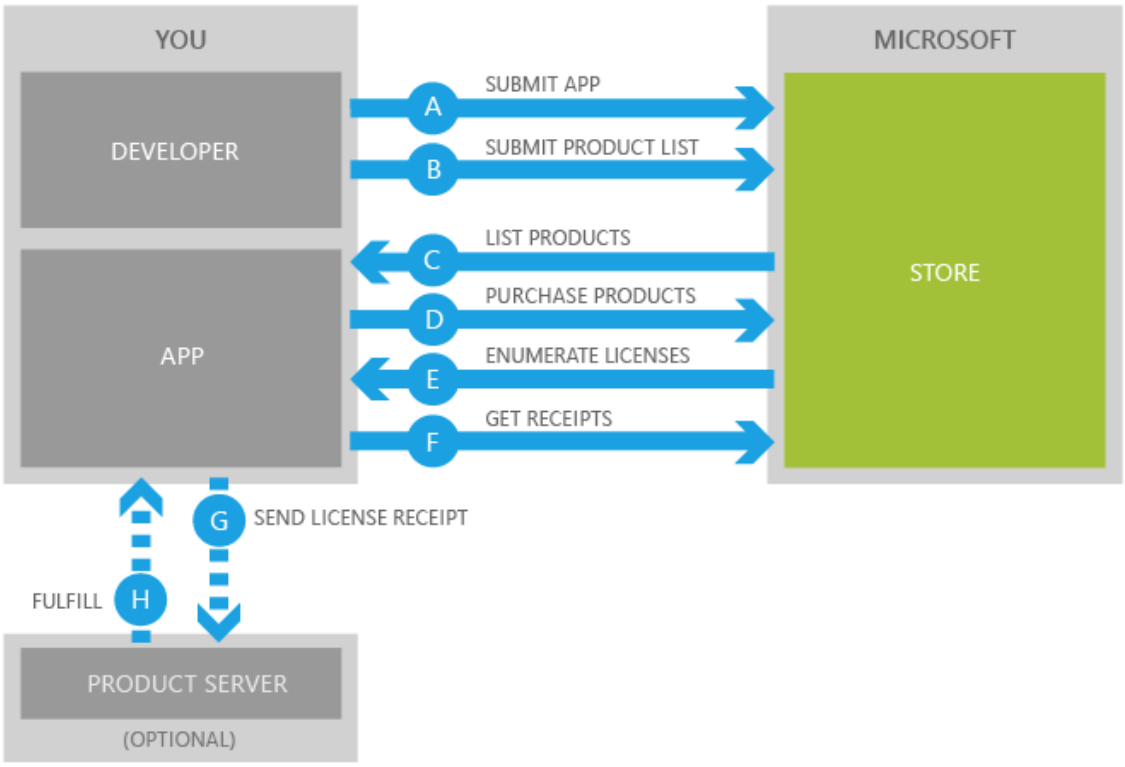
Sovelluksen on kyettävä tunnistamaan tila, jossa sitä suoritetaan. Sitä voidaan ajaa trial-modessa tai täydessä tilassa. Tämä tunnistus on koodattava julkaistavaan sovellukseen. Itse oikeudet kuitenkin määräytyvät ladattavan sovelluksen mukana tulevasta lisenssistä. Kun käyttäjä haluaa kokeilla ilmaiseksi sovellusta, sen mukana ladataan trial-lisenssi. Kun sovellus käynnistetään, lisenssin taso tarkastetaan ja käyttäjälle annetaan sen mukaiset oikeudet suorittaa sovellus. Trial-lisenssit eivät vanhene. Kun käyttäjä haluaa päivittää sovelluksen täyteen versioon, on sovelluksessa oltava suora päivityspolku, eli toiminto, jolla käyttäjä ohjataan Windows Phone- kauppaan ohjelman sivulle. Kun täysi versio ladataan, samalla latautuu uusi lisenssin, joka korvaa trial-aikaisen lisenssin, jolloin käyttäjä saa täydet oikeudet suoritettavaan sovellukseen. (Microsoft 2015 Z).

3.2 In-app maksut

Sovellus voi olla käyttäjälle myös asennettaessa maksuton, mutta kehittäjä voi tienata sen avulla pelin tai sovelluksen edetessä. Windows Phone 8 esitteli In-app purchase (IAP)- strategian, joka tunnetaan myös Premium downloadble content (PDLC)- nimikkeellä. Sen avulla rahoitusstrategiaan saadaan huomattavaa joustavuutta ja uusia mahdollisuuksia. (Whitechapel, A., McKenna, S. 2013. S. 829). Trial-versioihin verrattuna käyttökokemus on huomattavasti sujuvampi ilman alkuperäisen asennuksen jälkeisiä päivityksiä. Rahoitusstrategiana IAP on julkaisijan kannalta joustava, koska sen avulla heikosti tuloja kerännyt sovellus voi lisätä kassavirtaa, kun siihen julkaistaan ladattavaa lisäsisältöä, joka on maksullista. Näin ei jäädä pelkästään yhden kortin varaan sovelluksen julkaisuhetken kiinnostuksen mukaan ja jo olemassa olevalta asiakas kunnalta voidaan saada lisätuloja. (Whitechapel, A., McKenna, S. 2013. S. 829).

Termi In-app purchase tarkoittaa sovelluksen sisäinen osto, mutta silti siihen ottaa osaa myös Windows Phone- kauppa, kuten tavallisenkin oston tai latauksen yhteydessä. Käyttämällä hyväkseen In-app Purchase API:a, kehittäjä voi julkaista ladattavan sovelluksen Windows Phone- kauppaan, sisällyttää siihen erikseen ostettavien kohteiden listan, jota sovelluksella voi selata ja valita ladattavaksi maksua vastaan. Ostettu lisäsisältö tulee vielä pystyä toimittamaan käyttäjän sovellukseen. (Microsoft 2015 AA). Kun kehittäjä huolehtii edellä mainituista asioista, Microsoft puolestaan huolehtii sovellusten hallinnasta Windows Phone Dev Centerin kautta, maailmanlaajuisista levitysasetuksista,

käyttäjäystävällisestä ostokokemuksesta kaikilla yhteensopivilla Windows Phone- laitteilla, mahdollisuudesta ostaa sisältöä 190:ssä eri maassa ja ostoturvasta turvallisten kuittien avulla (kuva 10). (Microsoft 2015 AA).



Kuva 10. In-app purchase ostotapahtuma. (Kuva Microsoft 2015 AA).

3.3 Mainosrahoitteinen sovellus

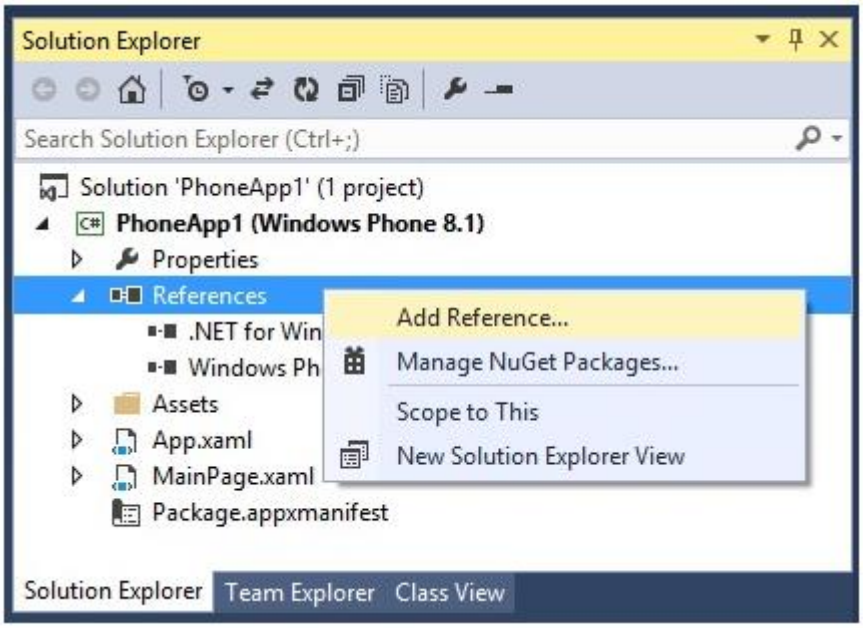
Sovelluksia voidaan rahoittaa mainoksilla, jolloin ne ovat käyttäjille ilmaisia. Useasti myös mainoksia sisältäviä sovelluksia voi päivittää maksullisiksi, jolloin mainoksista päästään eroon. Monet hittipelit ovat ilmaisia, mutta ovat tuottaneet kehittäjilleen miljoonia juuri mainonnan avulla.

3.3.1 Microsoft Advertising

Microsoft Advertising SDK for Windows Phone on esiasennettuna kaikissa uusissa Windows Phone- laitteissa lähtien versiosta 7.1 eteenpäin. Sen avulla kaikissa laitteissa voidaan näyttää standardeja mainoksia ennalta määritettyinä kokoina ja lähteinä. SDK:n avulla sovelluksiin voidaan lisätä helposti tekstiä ja bannereita sekä kohdentaa mainoksia haluttuihin käyttäjäryhmiin. (Microsoft 2015 AB).

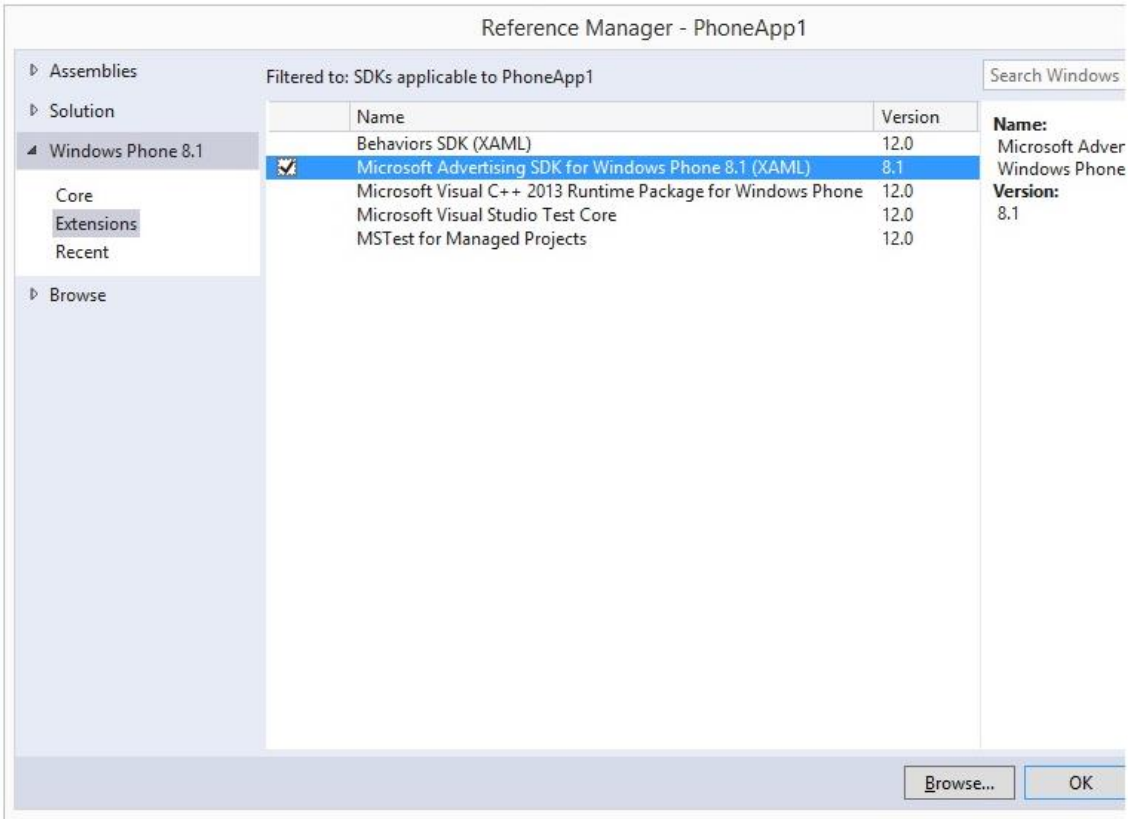
PubCenter (Microsoft pubCenter)- tunnus linkitetään sovellukseen lisättävään AdControl-kirjastoon, jonka avulla Microsoft Advertising Networkin mainokset tulevat näkyviin sovelluksen ruudulle. PubCenterissä luodaan uusi AdUnit, jonka yksilöllinen ApplicationId-tunnus konfiguroidaan sovelluksen AdControl-parametreihin. (Microsoft 2014 B).

Windows Phone 8.1- sovellukseen voidaan integroida AdControl-kirjasto käyttämällä XAML-merkintätapaa. Vaatimuksina ovat Windows 8.1, johon on asennettu Visual Studio 2013 Update 2 for Windows 8.1 sekä Windows Phone 8.1 Advertising SDK (XAML). (Microsoft 2014 C). Visual Studioon luodussa Windows Phone 8.1- projektissa lisätään Solution Explorer- ikkunassa hiiren oikealla klikkaamalla References-kohtaan uusi viite (kuva 11).



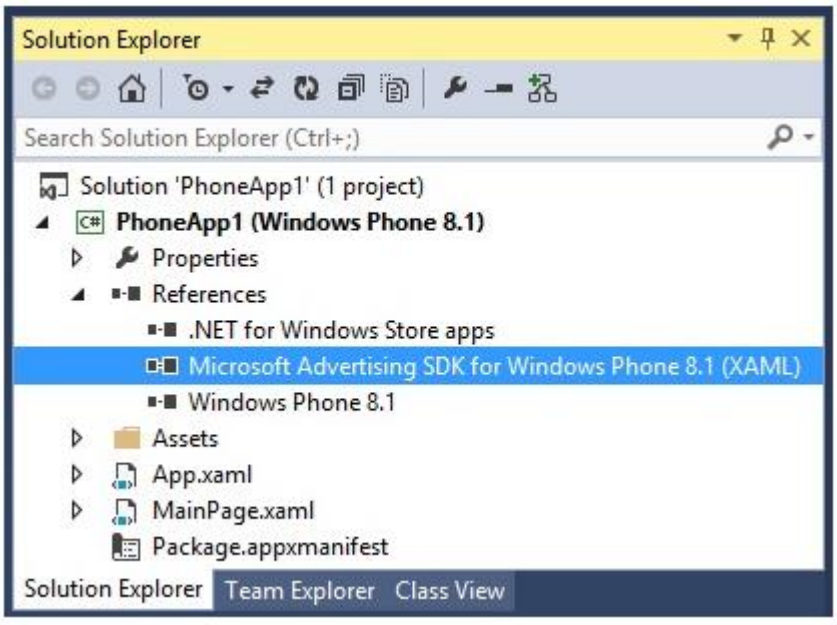
Kuva 11. Lisää viite. (Kuva Microsoft 2014 C).

Avautuvasta Reference Manager- ikkunasta valitaan Windows Phone 8.1 ja Extensions. Kun Extensions on valittuna, ikkunan keskikohdasta valitaan Microsoft Advertising SDK for Windows Phone 8.1 (XAML) (kuva 12).



Kuva 12. Reference Manager- ikkuna. (Kuva Microsoft 2014 C).

Viitteen lisäämisen jälkeen sen tulisi näkyä References-kohdassa Solution Explorer-ikkunassa (kuva 13).



Kuva 13. Solution Explorer- ikkuna. (Kuva Microsoft 2014 C).

Viitteen ollessa paikoillaan, sitä voidaan hyödyntää XAML-koodissa. Jotta käyttö onnistuu, on Pagen attribuutteihin lisättävä viitteen nimiavaruus Microsoft.Advertising.Mobile.UI (kuva 14), jonka jälkeen AdControl voidaan lisätä esimerkiksi Grid-objektiin parametreineen. (Microsoft 2014 C).

XAML

Copy

```
<Page
    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"
    xmlns:UI="using:Microsoft.Advertising.Mobile.UI"
    x:Class="App1.MainPage"
    mc:Ignorable="d"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}">

    <Grid>
        <UI:AdControl
            AutoRefreshIntervalInSeconds="60"
            ApplicationId="test_client"
            AdUnitId="Image320_50"
            HorizontalAlignment="Left"
            Height="50"
            IsAutoRefreshEnabled="True"
            Margin="35,35,0,0"
            VerticalAlignment="Top"
            Width="320"/>
    </Grid>
</Page>
```

Kuva 14. XAML-koodi AdControllille. (Kuva Microsoft 2014 C).

Tämän jälkeen sovellusta suoritettaessa, emulaattorin ruudulla tulee näkyä testibanneri, jossa on valkoisella pohjalla Microsoftin logo.

3.3.2 Ad mediation

Windows Phone 8.0 ja uudemmat käyttöjärjestelmät tukevat myös ad mediationia, jonka avulla mainonnan tuottoja voidaan optimoida. Sen avulla sovelluksessa voidaan näyttää yhden tai useamman mainosverkoston mainoksia ilman mainosverkostokohtaisia SDK:ta. Eri mainosverkostojen tarjoamilla SDK:lla on erilaisia ominaisuuksia; jotkin tuottavat kehittäjälle enemmän per tuhat katsomiskertaa, jotkin tarjoavat enemmän mainoksia näytettäväksi, kun sovellus pyytää API:n kautta niitä näyttämään. Microsoftin ad mediation pyrkii tasoittamaan näitä epätasaisuuksia. (Microsoft 2015 AC).

Ad mediationin avulla kehittäjä voi määrittää, kuinka usein yksittäisen mainosverkoston mainoksia esitetään sovellukseen lisätyssä kontrollissa. Tämä asetus tehdään sovelluksen ulkopuolella Dev Centerissä. (Microsoft 2015 AC). Myös markkina-aluekohtaiset määritykset ovat mahdollisia, joten mainosverkostojen sisältöä voidaan hyödyntää kohdennetusti.

Ad mediationin käyttö vaatii tilien luontia kaikkiin haluttuihin mainosverkostoihin. Jotta kontrollin lisääminen omaan projektiin olisi mahdollisimman helppoa, tilit kannattaa luoda ja muokata kuntoon etukäteen. (Microsoft 2015 AD). Windows Phone 8.1 XAML- koodi ei tue tällä hetkellä kuin kolmea eri mainosverkostoa, mutta Silverlight-koodin avulla voidaan käyttää yhteensä seitsemää eri verkostoa (taulukko 9).

Taulukko 9. Tuetut mainosverkostot. (Microsoft 2015 AD).

Verkosto	**WP 8.1 XAML**	**WP 8.1 Silverlight**	**WP 8 Silverlight**
Microsoft Advertising	Tuettu	Tuettu	Tuettu
AdDuplex	Tuettu	Tuettu	Tuettu
Smaato	Tuettu	Tuettu	Tuettu
AdMob (Google)	*Ei tuettu*	Tuettu	Tuettu
MobFox	*Ei tuettu*	Tuettu	Tuettu
InMobi	*Ei tuettu*	Tuettu	Tuettu
Inneractive	*Ei tuettu*	Tuettu	Tuettu

Jokaisella mainosverkostolla on omat asetuksensa, jotka tulee muokata ad mediation-kontrollia lisätessä. Vähintään tulee määritellä mainosverkostokohtainen sovellustunnus, mutta lähes kaikki vaativat myös sovelluksen sisäisen mainostunnuksen. (Microsoft 2015 AD). Kun kontrolli lisätään omaan projektiin ja sille määritetään käytettävät mainosverkostot, Visual Studio noutaa automaattisesti vaadittavat mainosverkostokohtaiset käännökset, joiden avulla verkostojen kanssa keskustellaan.

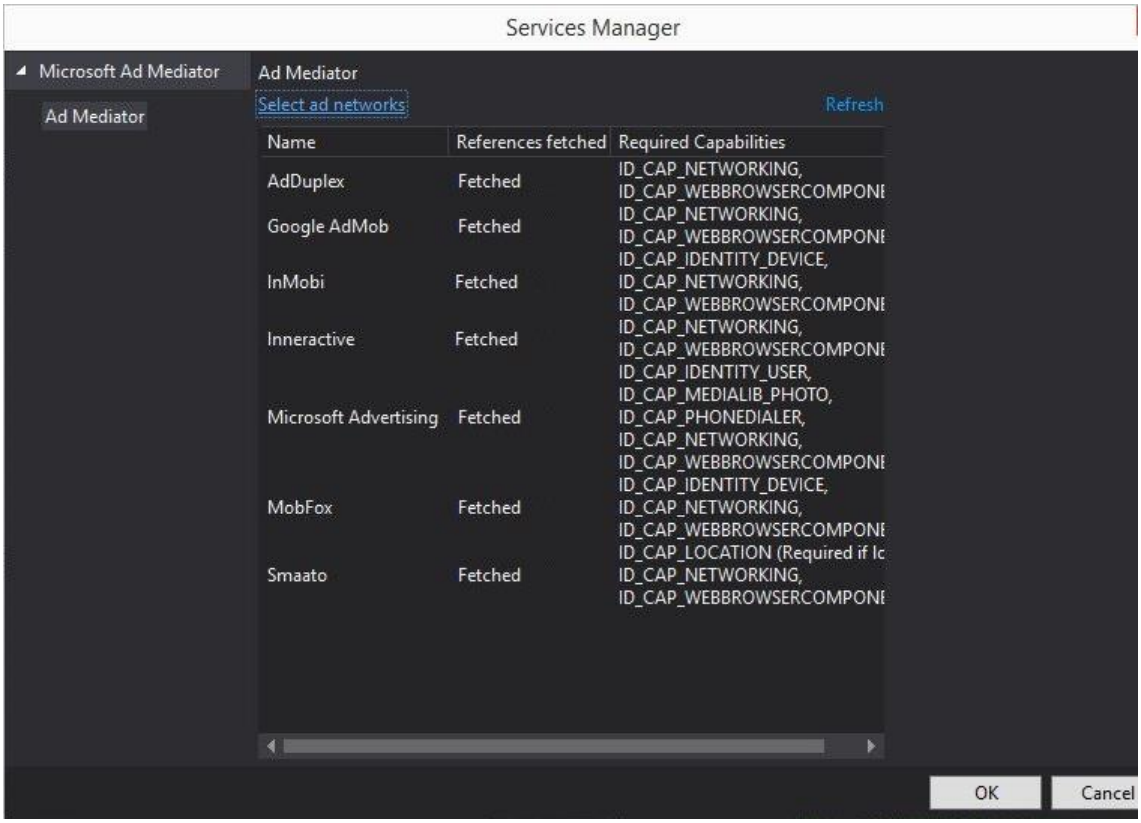
AdMediatorControl-kontrollin lisääminen sovellukseen tapahtuu raahaamalla se työkalulaatikosta haluttuun kohtaan sovelluksessa. Sovelluksen näkymän tulee olla designer,

kontrollia ei saa raahata suoraan XAML-koodiin. XAML-koodiin luodaan tällöin automaattisesti yksilöllinen tunnus (Id) ja kontrollin nimi (kuva 15) Kontrollin nimen voi vaihtaa haluamakseen, mutta tunnusta ei saa vaihtaa. (Microsoft 2015 AE).

```
XAML                                                                    Copy

// Code that gets added to the phoneapplication page header

xmlns:WindowsPhone8="clr-namespace:Microsoft.AdMediator.WindowsPhone8;
assembly=Microsoft.AdMediator.WindowsPhone8"

// Code that gets added for the ad mediator control

<WindowsPhone8:AdMediatorControl x:Name="AdMediator_038F91"

    HorizontalAlignment="Left" Height="80"
    Id="AdMediator-Id-2B28CAA4-57E6-45D8-8B0B-044BD2F65D8C"
    Margin="215,612,0,0" VerticalAlignment="Top" Width="480"/>
```

Kuva 15. AdMediatorControl XAML-koodi. (Kuva Microsoft 2015 AE).

Kontrollin lisäämisen jälkeen sen asetukset voidaan määrittää. Määritys tapahtuu Connected Servicen lisäämisen kautta. Projektiin lisätään palvelu painamalla hiiren oikealla painikkeella projektin nimen kohdalla Solution Explorerissa, valitsemalla lisää (Add) ja valitsemalla Connected Service. (Microsoft 2015 AE). Microsoft Ad Mediator- palvelu löytyy lista ja sen alla näkyy oletuksena Microsoft Advertising- mainosverkosto. Visual Studio on noutanut tarvittavat käännökset jo valmiiksi verkoston toimintaa varten. Muiden verkostojen lisääminen tapahtuu Select ad network- linkin kautta. Vain projektin kanssa yhteensopivat verkostot löytyvät esitettävältä listalta. Kaikki käytettävissä olevat mainosverkostot on hyvä lisätä projektiin, vaikkei niitä vielä aikoisi käyttääkään (kuva 16). Jos ne myöhemmin valitaan käytettäväksi, sovellukseen ei tarvitse tehdä muutoksia. (Microsoft 2015 AE).



Kuva 16. Connected Services manager. (Kuva Microsoft 2015 AE).

Kontrollin oletuskoko on 400x66 pikseliä XAML-projektissa, 400x80 pikseliä Silverlightilla. (Microsoft 2015 AE). Oletuskoon voi vaihtaa helposti sopivaksi AdMediator-Controlin parametreissa.

Kontrollin pyytäessä mainosverkostolta mainoksia esitettäväksi, se odottaa oletuksena vastausta 15 sekuntia, kunnes pyyntö hylätään ja uusi pyyntö suoritetaan toiselle mainosverkostolle. Tämä timeout-asetus voidaan vaihtaa koodissa 2- 60 sekuntiin. (Microsoft 2015 AE).

Jos kontrollin kanssa ilmenee ongelmia, viimeistään siinä vaiheessa kannattaa lisätä projektiin AdMediatorin tapahtumien (kuva 17) ja käsittelemättömien poikkeuksien käsittely (kuva 18). (Microsoft 2015 AE). Microsoft on löytänyt testeissään useita käsittelemättömiä poikkeuksia, joita tapahtuu kontrollin ajon aikana ja kehottaa sivuillaan lisäämään esimerkkikoodin projektin App.xaml.cs- tiedostoon.

Kuva 17. Ote AdMediatorin tapahtumien käsittelyn-koodista. (Kuva Microsoft 2015 AE).

```
C#                                                                    Copy

// In App.xaml.cs file, register with the UnhandledException event handler.
UnhandledException += App_UnhandledException;

void App_UnhandledException(object sender, UnhandledExceptionEventArgs e)
   {
      if (e != null)
      {
         Exception exception = e.Exception;
         if (exception is NullReferenceException && exception.ToString().ToUpper().Contains("
         {
            Debug.WriteLine("Handled Smaato null reference exception {0}", exception);
            e.Handled = true;
            return;
         }
      }
// APP SPECIFIC HANDLING HERE

   if (Debugger.IsAttached)
      {
         // An unhandled exception has occurred; break into the debugger
         Debugger.Break();
      }
   }
```

Kuva 18. Ote käsittelemättömien poikkeuksien hallinnan koodista. (Kuva Microsoft 2015 AE).

Ad mediationin käytön testaaminen tapahtuu parhaiten käyttämällä Windows Phone-emulaattoria, mieluiten kuitenkin oikean rekisteröidyn laitteen kanssa. Visual Studion sisäisen emulaattorin kanssa testimainokset, eikä itse kontrolli, aina näy joten ulkoinen laite on paikallaan. Testin aikana mainokset pyörivät läpi vuorotellen, jolloin kaikkien mainosverkostojen testibannerit tulisi toimia. Yllä olevien tapahtumien ja poikkeusten hallinnan avulla voidaan lukea konsolista mahdolliset virhetilanteet ja korjata ne ennen sovelluksen julkaisua. (Microsoft 2015 AF).

Kun sovellusta julkaistaan, Dev Center havaitsee automaattisesti ad mediationin käytön sekä käytettävät mainosverkostot. (Microsoft 2015 AG). Paketin lataussivulla kehittäjälle näytetään Ad mediation configuration- osio, jossa mainonnan asetuksia voi muokata. Asetuksia voidaan muokata vapaasti myös julkaisun jälkeen. Asetukset muodostuvat kolmesta eri osa-alueesta: päivityksen tiheys, mainosverkostojen jakauman asetukset ja

jokaisen verkoston pakolliset asetukset. (Microsoft 2015 AG). Mainoksia voidaan esittää sovelluksessa 30- 120 sekunnin välein. Tämän asetuksen tulee olla sama, kuin itse mainosverkoston asetus. Mainosverkostojen välinen jakauma voidaan asettaa prosentuaalisesti vaihteluvälillä 10- 100 %. Jos halutaan käyttää kahta eri verkostoa yhtäläisesti, molemmille asetetaan arvo 50 %. Muita arvoja ovat ”backup” ja ”do not use”. ”Backup” tarkoittaa, että mainosverkostoa käytetään vain siinä tapauksessa, kun primäärin verkoston kanssa ei onnistuta kommunikoimaan. ”Do not use” tarkoittaa sitä, ettei mainosverkostoa ei käytetä. Tätä voidaan hyödyntää markkina-aluekohtaisesti, eli yhdellä markkina-alueella voidaan käyttää eri mainosverkostoja kuin toisilla. (Microsoft 2015 AG). Näitä asetuksia muuttamalla mainontaa voidaan optimoida. Kun sovellusta ladattaessa sallitaan tasainen jakauma kaikkien käytettävien mainosverkostojen mainosten välillä, raporteista voidaan nähdä kaikkein aktiivisimmat ja parhaiten tuottavat verkostot alueittain. Keskittämällä mainonta niihin, tuotot voivat olla huomattavasti suuremmat.

4 Shakesbeat- sovellus

Opinnäytetyön tuloksena tuotetaan Windows Phone 8.1- sovellus nimeltään ShakesBeat, joka julkaistaan Windows Phone- kaupassa vapaasti ladattavaksi. Sovelluksella ei ole ulkopuolista tilaajaa, joten se julkaistaan henkilökohtaisen kehittäjätilin alaisuudessa. Tässä kappaleessa kuvataan sovelluksen kehitys pääpiirteittäin vaatimuksista ja suunnittelusta lähtien sovelluksen julkaisuun saakka peilaten edellä teoriaosuudessa käsiteltyjä asioita.

4.1 Tavoite

Tavoitteena on kehittää toimiva sovellus, joka täyttää sille asetetut vaatimukset, ja julkaista se Windows Phone- kaupassa. Sovellukselle asetettavat vaatimukset ovat seuraavat:

- Sovelluksen värimaailman tulee olla runsas ja värikäs
- Sovelluksen grafiikoiden tulee olla tarkkoja
- Sovelluksen grafiikoiden tulee toimia muuttumattomina kaikilla laitekooilla
- Sovelluksen käytön tulee olla nopeaa ja helppoa
- Sovelluksen tulee hyödyntää laitteen liikeantureita ja reagoida sen heilutukseen
- Sovelluksen tulee toistaa musiikkia ja muita äänilähteitä laitteen paikallisesta muistista
- Sovelluksen tulee pystyä asettamaan toistettavan äänilähteen toistonopeus perustuen laitteen fyysisen heilutuksen nopeuteen
- Sovelluksen tulee säilyttää tilatietonsa, eli jos sovelluksesta siirrytään toiseen ja sen jälkeen palataan takaisin, sovelluksen suorituksen tulee jatkua samasta kohdasta, johon sen suoritus keskeytyi
- Sovelluksen tulee kyetä hyödyntämään Microsoftin ad mediation- kontrolleja, eli näyttämään mainoksia halutuista mainosverkostoista

Sovelluksen kohdeyleisö on lapset ja nuoret. Sovelluksen tarkoitus on olla yksinkertainen ja helppokäyttöinen sekä käyttäjiä hauskuuttava. Kantavana ideana pidetään hidas-

tetun tai nopeutetun musiikin ja puheen toiston aiheuttamaa iloa ja naurua. Yksinkertaisuutensa vuoksi sovelluksen käyttö on nopeaa, jolloin sen satunnainen käyttö on todennäköisempää. Kohdeyleisön vuoksi myös värimaailman tulee olla yksinkertainen mutta samalla houkutteleva.

Opinnäytetyötä ohjaavina tutkimuskysymyksinä, jotka tuotoksen on ratkaistava, toimivat liikeantureiden ja laitteeseen tallennetun media hyödyntäminen, samanlaisen käyttökokemuksen mahdollistaminen laitteesta ja laitekoosta riippumatta ja Windows Phone-kaupan mainospohjaisen rahoituksen hyödyntäminen.

4.2 Suunnittelu

4.2.1 Yleistä

Sovellus toteutetaan Visual Studio 2013 Ultimate Update 4- kehitysympäristössä käyttäen XAML- ja C#- ohjelmointikieliä. XAML-sivujen tukena käytetään code behind-tyylisesti C#-luokkia. Kehittäjätilinä käytetään henkilökohtaista tiliä.

Projektiksi valitaan 'Blank App (Windows Phone)', eli toteutus aloitetaan tyhjältä pöydältä. Tarvittavat apu- ja lisäluokat saadaan kopioitua valmiista projektista sekä päivitettyä Visual Studion omalla NuGet- paketinhallinnalla.

Sovellus kehitetään iteratiiviseen tyyliin toteuttamalla toiminnallisuus kerrallaan, kuitenkin sallien rinnakkaisten toiminnallisuuksien muokkauksen. Kaikki edellä luetellut vaatimukset toteutetaan käyttäen hyväksi teoriaosuuden käsittelemiä keinoja.

Sovelluksen testauksessa käytetään sekä Visual Studion sisäänrakennettua emulaattoria että fyysistä laitetta, joka on kytketty tietokoneeseen USB-liittymän avulla. Testausta varten laitteessa tulee olla vähintään yksi äänitiedosto.

Sovellus julkaistaan Windows Phone- kaupassa heti sen valmistuttua. Sovelluksen julkaisun jälkeen sen tilaa voidaan seurata Windows Phone Dev Centerin kautta. Myös ad mediation- asetuksia voidaan muuttaa tarpeen mukaan samasta paikasta.

4.2.2 Toiminnallisuudet

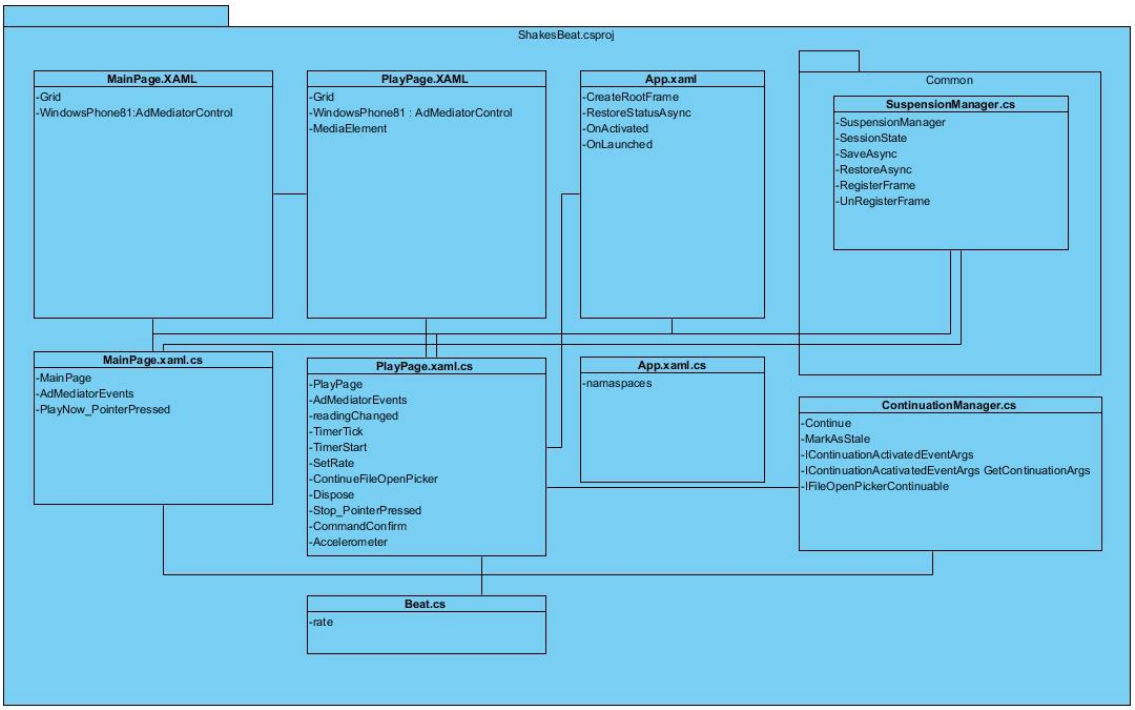
Tiedostojen valinta ja avaaminen toteutetaan tietoperustan kappaleen 2.2.3 mukaisesti käyttäen FileOpenPicker-luokkaa ContinuationManager- ja SuspensionManager- apuluokkien kanssa. ”Play now”- linkkiä painettaessa käyttäjälle avautuu Windows Phone-tiedostovalitsin, jossa äänitiedosto valitaan ja viittaus tiedostoon toimitetaan seuraavalle XAML-sivulle parametrina.

Sivujen välillä siirtymiseen käytetään kappaleen 2.2.4 mukaisesti Frame.Navigate-metodia, jolle annetaan parametriksi kohdesivun nimi. Sovelluksessa siirrytään PlayPage.XAML-sivulle, jossa sivua alustettaessa tulee käynnistää edellä mainittu FileOpenPicker- toiminne.

Median toisto, keskeyttäminen ja lopetus toteutetaan MediaElement-luokan avulla, kuten kappaleessa 2.2.1 kuvataan. Sovelluksessa toistettava äänitiedosto tulee valita edellä mainitun mukaisesti, jolloin tiedoston toisto aloitetaan automaattisesti ContinuationManager-luokan ContinueFileOpenPicker-metodin sisällä. Median PlaybackRate-asetus voidaan muuttaa heiluttamalla puhelinta, joka aiheuttaa tilatun Accelerometer-luokan ReadingChanged-tapahtuman. Sen avulla lasketaan ensimmäisen ja neljännen heilautuksen välinen aika millisekunteina, josta johdetaan määritetyn asteikon mukaisesti asetettava double-arvo MediaElement.DefaultPlaybackRate-ominaisuudelle.

Kun toisto halutaan lopettaa, painetaan ”Stop”-linkkiä, joka tuottaa kappaleessa 2.2.5 esitellyn MessageDialog-luokan mukaisen ilmoituksen näytön yläosaan ja pysäyttää samalla äänitiedoston toiston. Ilmoitus on varmistus, jonka avulla varmistetaan, että käyttäjä haluaa varmasti lopettaa äänitiedoston toiston ja siirtyä aloitussivulle. Jos käyttäjä vastaa ei, toisto jatkuu normaalisti, jos kyllä, käyttäjä ohjataan aloitussivulle Frame.Navigate-metodin avulla ja toiston varaamat resurssit vapautetaan IDisposable-rajapinnan Dispose-metodin avulla.

Sovelluksen tulee muodostua MainPage.XAML ja PlayPage.XAML sivuista, sekä niiden taustalla käytettävistä *.cs- luokista. Sovellukseen lisätään myös tarvittavat apuluokat kuvan 19 mukaisesti.

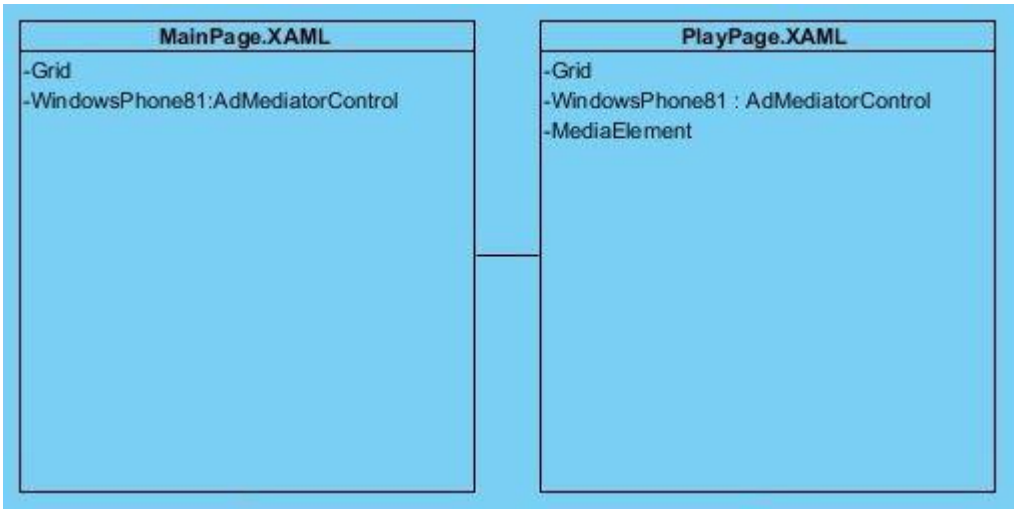


Kuva 19. ShakesBeat.csproj.

Sovelluksen referensseihin tulee lisätä Microsoftin AdMediator-kirjasto NuGet-paketinhallinnan avulla kappaleen 3.3.2 mukaisesti.

4.2.3 Käyttöliittymä

Sovelluksen käyttöliittymän tulee muodostua kahdesta sivusta, MainPage.xaml ja PlayPage.xaml, joiden välillä käyttäjä voi navigoida (kuva 20). Molemmilla sivuilla näytetään AdMediator- kontrollin kautta mainoksia. Kontrollit tulee nimetä yksilöllisesti, jotta ne toimivat oikein. Molemmilla sivuilla on myös kuvamuotoinen tekstilinkki, joiden kautta toiminteet toteutetaan.



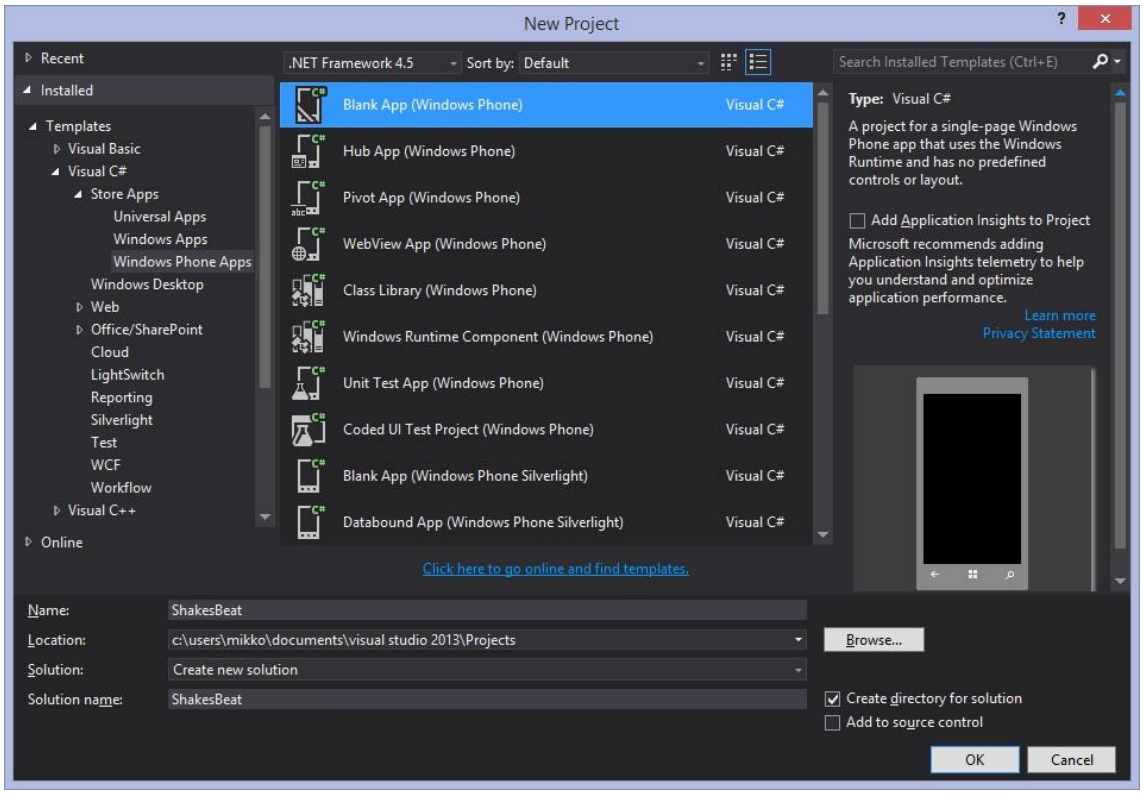
Kuva 20. XAML-sivut.

Sovelluksen grafiikat toteutetaan Photoshopilla ja niiden tulee skaalautua kolmelle eri tarkkuudelle (kappale 2.4.3): WVGA, 720p ja 1080p. Tämän vuoksi jokainen erillinen kuva (taustakuva, logot ja tekstit) suunnitellaan ensin 1080p-muotoon ja skaalataan sen jälkeen alemmille tarkkuuksille. Skaalatessa tulee ottaa huomioon kuvien ja tekstien kokojen ja suhteiden muutokset; ne eivät saa muuttua, vaan kokonaisuuden tulee pysyä ehjänä. Kuvasyvyyden on oltava vähintään 100 dpi. Kuvat tallennetaan PNG-formaatissa. Grafiikoiden sisällön tulee toteuttaa vaatimuksissa määritellyt asiat.

4.3 Toteutus

4.3.1 Projekti

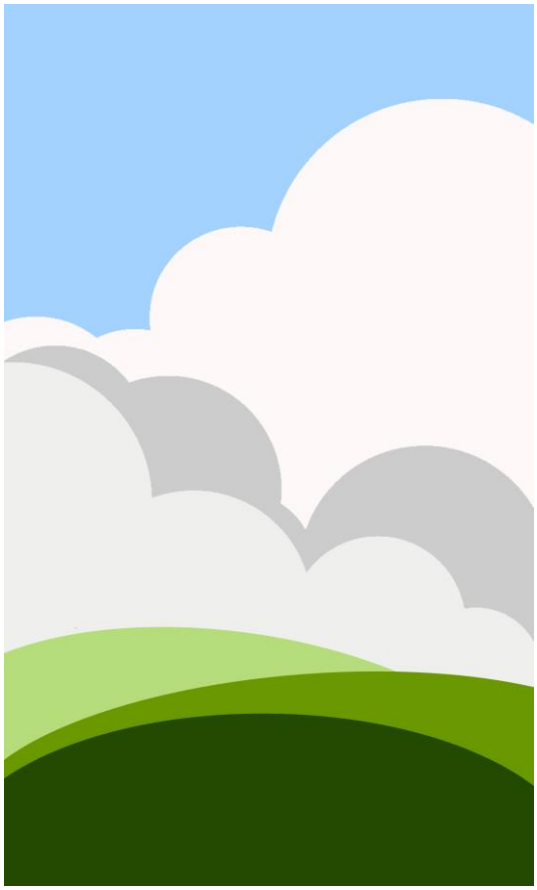
Luodaan Visual Studio 2013 projekti nimeltään ShakesBeat.csproj. Projektiksi valitaan uusi ”Blank App (Windows Phone)” (kuva 21).



Kuva 21. Uusi projekti Visual Studiossa.

4.3.2 XAML-sivut ja grafiikat

Luotuihin MainPage.xaml ja PlayPage.xaml (liite 1 ja 2) lisätään ulommaisen Grid-objektiin taustakuvaksi (Image-brush) Photoshopissa luotu kuva Shakesbeat.png (kuva 22), joka skaalataan eri laitteisiin sopivaksi käyttämällä Visual Studion tukemaa nimeämiskäytäntöä (taulukko 8).



Kuva 22. Shakesbeat.scale-240.png.

MainPage.xaml-sivulle lisätään taustakuvan lisäksi tulevan AdMediator-kontrollin alle sisemmän Grid-objektin taustakuvaksi Logo.png (kuva 23) sekä PlayPage.xml-sivulle vastaavasti Shake.png (kuva 24).

Kuva 23. Logo.scale-240.png

Kuva 24. Shake.scale-240.png

MainPage.xaml-sivulle lisätään alimpaan Grid-objektiin taustakuvaksi PlayNow.png (kuva 25), johon liitetään PlayNow_PointerPressed()-tapahtuma code behind- tiedostossa (liite 3), jolla siirrytään PlayPage.xaml-sivulle valitsemaan toistettavaa äänilähdettä standardin tiedostonvalitsimen avulla (kuva 26).

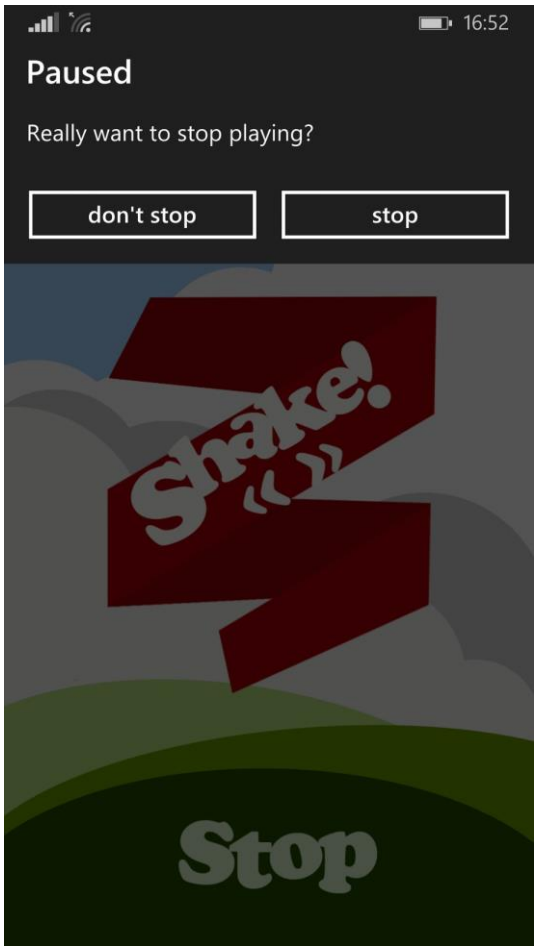
Kuva 25. PlayNow.scale-240.png.

Kuva 26. Windows Phonen standardi tiedostonvalitsin.

PlayPage.xaml-sivulle lisätään samaan objektiin taustakuvaksi Stop.png (kuva 27), johon liitetään myös Stop_PointerPressed()-tapahtuma (liite 4), jonka avulla äänilähteen toisto keskeytetään (Taulukko 2, Pause-metodi) ja käyttäjälle esitetään MessageDialog-luokan avulla ilmoitus kappaleen 2.2.5 mukaisesti (kuva 28). Jos valinta on ”don’t

stop”, äänitiedoston toisto jatkuu samasta paikasta samoilla asetuksilla, jos valinta puolestaan on ”stop”, siirrytään automaattisesti aloitussivulle.

Kuva 27. Stop.scale-240.png.



Kuva 28. Stop-linkin valinnan aiheuttama ilmoitus.

4.3.3 Apuluokat

Jotta sovelluksen tilatieto voidaan tallentaa ja käyttää sitä hyväksi eri sovellusten ja ikkunoiden välillä siirryttäessä, projektiin tulee lisätä Microsoftin tarjoamat apuluokat ContinuationManager ja SuspensionManager, kuten kappaleet 2.2.3 ja 2.2.4 kertovat. SuspensionManager-apuluokan löytää projektin Common-kansiosta, kunhan projekti

on jokin muu kuin ”Blank App (Windows Phone)”. Kopiointia varten joudutaan siis luomaan toinen projekti, josta SuspensionManager.cs-tiedosto voidaan kopioida.

ContinuationManager.cs-tiedosto saadaan kopioitua MSDN:stä kappaleen 2.2.3 mukaisesti. Luokka luodaan projektin juureen ja App.xaml.cs-luokkaan lisätään tarvittavat viitteet sen käyttöön. ContinuationManager-luokkaa käyttävän luokan, eli PlayPage.xaml.cs:n, on toteutettava interface IFileOpenPickerContinuable, joten se tulee myös määrittää luokan luonnissa (liite 2).

4.3.4 Median toisto

Median toisto aloitetaan automaattisesti FileOpenPickerin ContinueFileOpenPicker-metodin avulla, jolle syötetään parametrina valittu äänitiedosto. Jos parametri on tyhjä, palataan automaattisesti aloitussivulle, muuten jatketaan PlayPage.xaml:ssa (liite 2) luodun MediaElementin (x:Name=”ShakesPlayer”) alustuksen kanssa. MediaElement luodaan siis XAML-tiedostoon parametreineen, mutta sen toiminnallisuudet määritetään code behind- tiedostossa. MediaElementille määritetään taulukon 1 ja taulukon 2 mukaiset asetukset, eli määritetään SetSource- ja DefaultPlaybackRate-ominaisuudet, sekä aloitetaan äänitiedoston toisto alusta Play-metodin avulla.

Median toisto keskeytetään painamalla Stop-objektia, joka kutsuu ensin MediaElement-luokan Pause-metodia. MessageDialog-luokan ilmoituksen kautta kutsutaan joko continueSong-metodia, joka suorittaa MediaElement-luokan Play-metodin, jonka avulla äänitiedoston toistoa jatketaan oletuksena sen hetkisestä toistokohdasta, tai stopSong-metodia, joka puolestaan suorittaa MediaElement-luokan Stop-metodin ennen navigointia aloitussivulle (liite 4).

MediaElementin rajoituksena on vain yhden sound streamin käytön mahdollistaminen. Tämän vuoksi toisen MediaElement-objektin käyttö ei ole mahdollista samassa projektissa. Sovellusta testattaessa huomattiin pieni hiljainen hetki toiston PlaybackRatea muutettaessa heiluttamalla laitetta. Tämä haluttiin peittää lisäämällä tapahtumaan levysoittimen scratch-ääni. MediaElementin avulla se ei onnistunut, koska default stream oli jo käytössä ja peitti alleen kaikki muut streamit. Ratkaisuna tähän projektiin lisättiin

SharpDX.XAudio2-kirjasto, jonka avulla toisen päällekkäisen äänen toisto onnistuu. Kirjaston avulla scratch-ääni (”Assets/skratch2.wav”) toistetaan aina, kun toiston DefaultPlaybackRate saa uuden arvon.

4.3.5 Kiihtyvyysanturin käyttö

Accelerometer-luokkaa varten projektin PlayPage code behind- luokkaan lisätään Windows.Devices.Sensors- viittaus. Luokalle luodaan privaatti Accelerometer-objekti nimeltään _accelerometer ja sen arvoja tallennetaan myös privaattiin AccelerometerReading-objektiin _lastReading (liite 4). PlayPage()-päämetodin alustuksessa myös _accelerometer alustetaan kutsumalla GetDefault-metodia, joka kyselee käytettävän laitteen sisältämän liikesensorin ominaisuuksia ja tallentaa ne luotuun objektiin. Jos laite löytyy, _accelerometerin näytteenottotaajuus määritetään halutun mukaiseksi. ReportInterval-ominaisuus määritetään lukemalla laitteen oletusminimiraportointitaajuus, joka on yleensä 16 ms. ShakesBeat-sovelluksen tarpeisiin se on liian tiuha, joten _accelerometer.ReportInterval- ominaisuus asetetaan 60 ms:iin. Alustus päätetään lukemalla laitteesta sen hetkinen arvo, johon tulevia muutoksia voidaan verrata.

Alustuksen jälkeen kiihtyvyysanturin tapahtumille on luotava readingChanged-tapahtuman toteuttava metodi, eli mitä sovelluksen tulee tehdä, kun laite rekisteröi lukemien muutoksia. Tässä metodissa kutsutaan asynkronista Dispatcheria, jonka avulla lasketaan laitteen rekisteröimiä heilautuksia ja niiden määrää, joka tallennetaan privaattiin _shakeCount Integer-muuttujaan. Muuttujan arvoa kasvatetaan yhdellä joka heilautuksella, joka rekisteröidään. Jos _shakeCountin arvo on 1, heilautuksen aikaleima tallennetaan _firstReadingTime-muuttujaan (DateTimeOffset-tyyppi). Jos muuttujan arvo on 4, sen hetkisen heilautuksen aikaleima tallennetaan _secondReadingTime-muuttujaan, jonka jälkeen tallennettujen aikaleimojen erotus lasketaan System.TimeSpan-tyypin muuttujaa diff ja kutsutaan setRate-metodia parametreina saadun arvon millisekunnit (diff.TotalMilliSeconds).

Jotta heilautuksen aiheuttama arvon muuttuminen voitaisiin rekisteröidä, tapahtuma täytyy ”tilata”, eli sovellukselle tulee kertoa, mitä _accelerometer.ReadingChanged-

tapahtuman jälkeen tehdään. Tämä tapahtuu jo sivulle siirryttäessä, eli ContinueFileOpenPicker-metodissa lisätään subscription muodossa:

_accelerometer.ReadingChanged += new TypedEventHandler<Accelerometer, AccelerometerReadingChangedEventArgs>(readingChanged);

SetRate-metodissa asetetaan parametrina saadun millisekuntien mukaan toiston nopeus. Millisekunteja verrataan asetettuihin raja-arvoihin ja double-tyyppinen newRate-muuttuja saa sen mukaan arvokseen määritetyn arvon. Tässä huomioitiin double-tyypin oletustarkkuus ja sen haluttiin ohittaa tarkemmalla arvolla, jolloin myöhempi vertailu on helpompaa. Sen vuoksi newRate-muuttuja saa arvokseen Math-luokan Round-metodin mukaisen pyöristetyn arvon, joka sisältää vain yhden desimaalipilkun jälkeisen luvun. Luotuun beat-objektiin asetetaan arvoksi newRate-muuttujan arvo.

Taustalla pyörii DispatchTimer nimeltään timer, jonka intervalliksi on asetettu 3 sekuntia. Tämä aiheuttaa beat-objektin sisältämän double-arvon asettamiseksi MediaElementin DefaultPlaybackRate-ominaisuuden arvoksi joka kolmas sekunti, mutta vain jos arvo on eri kuin sen hetkinen arvo.

4.3.6 Windows Phone- kaupan grafiikat

Windows Phone- kaupan vaatimat kuvat (kappale 2.4.2, kuva 7) luodaan Photoshopilla. Niissä hyödynnetään jo olemassa olevia kuvia, joista muokataan pienempiä kuvia. Kuvat (kuvat 29 ja 30) tallentuvat suoraan projektin bundle-pakettiin, mutta osa niistä tulee ladata Dev Centeriin sovellusta julkaistaessa.

Kuva 29. Sovelluksen logot kauppaa varten.

Kuva 30. Sovelluksen ruutukaappaukset kauppaa varten.

4.3.7 Ad mediation

Sovellukselle luotujen kuvien lisäksi ruudulla näytetään mainoksia AdMediator- kontrollin avulla (kappale 3.3.2). Ruudun ylälaidassa on kontrollille varattu 480x80-kokoinen alue. Kontrollin lisääminen on helppoa ja se tulee tehdä raahaamalla hiirellä työkalulaatikosta kyseenomainen kontrolli graafiseen suunnittelunäkymään, ei siis code behind- luokkiin. Kontrollin lisäämisen jälkeen sen toimintaa tulee muokata lisäämällä projektiin Connected Service (kuva 16).

Ennen kontrollin lisäämistä kannattaa luoda haluttuihin mainosverkostoihin tunnukset ja mainosobjektit. Niiden tiedot tulee syöttää Connected Servicea lisättäessä. Windows Phone Runtime- sovelluksen kanssa käytettävissä on ainoastaan Microsoftin, Adduplexin ja Smaaton verkostot. Ad mediation asetukset konfiguroidaan ladattaessa sovellusta Dev Centeriin. Nämä asetukset tallentuvat projektin Admediator.config-tiedostoon,

joka sijaitsee projektin juuressa. Testattaessa sovellusta kontrolli ei välttämättä näytä oikein testimainoksia.

4.3.8 Testaus

Sovellusta testataan kehityksen aikana jatkuvasti kehitystyylin ollessa iteratiivinen. Testaus tapahtuu Visual Studion avulla, joko sen sisäänrakennetun laite-emulaattorin (kuva 32) avulla tai kehityskoneeseen liitetyn rekisteröidyn Windows Phone- laitteen avulla. Jälkimmäinen vaihtoehto on suositeltava sisäänrakennetun emulaattorin jäykkyyden ja rajoitusten vuoksi. Emulaattorin avulla voidaan kylläkin simuloida erinäisiä paikkatietoja ja orientaatioita, joita oikealla laitteella ei aina voida saavuttaa testitilanteessa.

Kuva 31. Visual Studion laite-emulaatori.

AdMediator-kontrollia testattaessa tulee huomata, että emulaattori ei osaa näyttää kaikkia mainosverkostoja ja niiden testimainoksia oikein. Sama voi toistua myös oikean laitteen kanssa testattaessa. Siihen on olemassa helppo korjaus, jonka avulla mainokset saadaan toimimaan. Muokkaamalla admediator.config-tiedostoa <testconfiguration>-

elementin osin vastaavaksi kuin tiedoston alkuosan asetukset, mainokset saadaan näkyviin. Testiosan mainoskohtaiset id:t nollaantuvat, josta johtuen testiympäristössä ne eivät toimi täysin oikein ilman muokkausta.

4.3.9 Julkaisu Windows Phone- kauppaan

Sovelluksen ollessa valmis julkaistavaksi, siitä luodaan Visual Studion avulla appx-paketti. Paketointi tehdään Visual Studiossa Project-menun kautta Store- ja Create app packages-valinnoilla. Avautuvan paketointivelhon avulla kirjaudutaan kehittäjätilin avulla Windows Phone- kauppaan ja valitaan paketoitavaksi haluttava sovellus, jos se on jo alustavasti Dev Centeriin luotu. Jos ei, niin uuden nimen sovellukselle voi varata myös velhon kautta. Kun paketti on luotu onnistuneesti, sen yhteensopivuus tulee tarkastaa Windows App Certification Kitin avulla.

Testattavasta sovelluksesta luodaan velhon avulla kaksi versiota, testattava *.appxbundle ja kauppaan ladattava *.appxupload. Paketointivelho tarjoaa automaattisesti linkin paikallisesti asennettuun Windows App Cert Kitiin, joten sen voi käynnistää heti paketoinnin päätyttyä ilman muita konfigurointeja. Sovelluksen voi käynnistää myös suoraan Start Menusta joko graafisena tai komentokehotteessa. Kun kaikki valitut testit menevät läpi ja yleisarvosana sovellukselle on ”passed”, on se valmis ladattavaksi kauppaan.

Julkaisu tehdään kirjautumalla Dev Centeriin Windows Phone Storen puolelle omilla kehittäjätunnuksilla. Varatulle sovellusnimelle muokataan puuttuvat tiedot ja ladataan appxupload-paketti sille varatusta linkistä. Samalla Dev Centerin sivulla ladataan julkaisun vaatimat kuvat erikokoisine versioineen. Windows Phone- kauppa osaa luoda automaattisesti 720p-muotoisesta kuvasta muut vaaditut tarkkuudet kuvaruutukaappauksia varten, joten kaikkia kuvia ei tarvitse ladata itse.

Julkaisun yhteydessä Windows Phone- kauppa havaitsee käytetyn AdMediator-kontrollin admediator.config-tiedostosta ja pyytää muokkaamaan sen asetukset (kappale 3.3.2). Paketin sisältämän admediator.config-tiedoston asetukset näkyvät valinnoissa automaattisesti, vain jakeluasetukset tulee määrittää ja tallentaa.

Kun sovellus julkaistaan, sen tulee käydä läpi vielä Microsoftin tarkastukset. Niiden suorittamista voi joutua odottamaan muutaman tunnin, olettaen, että Windows App Cert Kit- testi meni hyväksytysti läpi. Kun Microsoftilta saapuu sähköpostiviesti hyväksytystä sovelluksesta, odottamista on vielä päivästä kahteen, ennen kuin sovellus tulee julkisesti saataville Windows Phone- kauppaan.

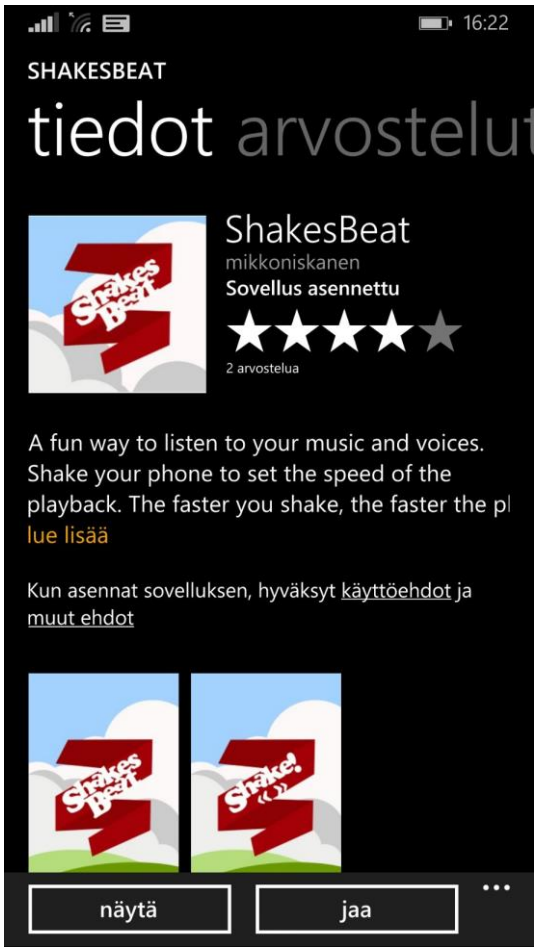
4.4 Yhteenveto

ShakesBeat-sovellus julkaistiin onnistuneesti Windows Phone- kaupassa. Sen kehityksessä käytettiin hyväksi tietoperustassa käsiteltyjä asioita, joiden avulla sovelluksen sisältämä koodi pysyi Microsoftin julkaisuille määrittämien vaatimusten mukaisena. Luotua ShakesBeat.appxbundle-pakettia testattaessa, siitä ei löytynyt yhtäkään virhettä, joten kaikki testit läpäistiin ensimmäisellä kerralla.

ShakesBeat-sovellus toteuttaa kaikki sille asetetut toiminnalliset sekä graafiset vaatimukset. Julkaisun jälkeen nämä voitiin vielä todentaa eksoottisemmilla laitteilla, joiden näyttöjen koot poikkesivat normaalista noin viidestä tuumasta.

ShakesBeat-sovelluksen mainontaa, latauksia ja mahdollisia virhetilanteita voidaan seurata Dev Centerissä. Sieltä onnistuu myös sovelluksen päivittäminen, eli uuden appxupload-paketin lataaminen sekä käytössä olevan admediator.config-tiedoston tallennus päivityksiin liitettäväksi, jotta asetukset pysyisivät ennallaan myös päivityksen jälkeen. Muiden mainosverkostojen seuranta onnistuu verkostojen omilta kotisivuilta kirjautumalla niihin rekisteröidyillä tunnuksilla.

ShakesBeat on vapaasti ladattavissa kaikille yhteensopiville Windows Phone 8.1- laitteille Windows Phone- kaupasta osoitteesta http://www.windowsphone.com/en-us/store/app/shakesbeat/6634b729-61d8-4e85-a8e6-8f01825cd377 (kuva 32).



Kuva 32. ShakesBeat Windows Phone- kaupassa.

5 Pohdinta

Windows Phone 8.1 alkaa olla jo kypsä versio markkinoille. Sen helppokäyttöisyys, intuitiivisuus ja selkeä värimaailma tekevät siitä vahvan markkinahaastajan kahden johtavan mobiilikäyttöjärjestelmän, Androidin ja iOS:in, kannoille. Jatkuvasti kasvava sovellustarjonta ja kasvava ekosysteemi luovat hyvät edellytykset menestykselle. Vanhoista lastentaudeista eroon pääseminen on ollut helppoa verrattuna niistä kärsittyyn imagotappioon, jonka ylipääsemiseen saattaa mennä hiukan kauemmin aikaa.

Itse IT-alalla ja Microsoftin tuotteiden parissa yli 15 vuotta työskennelleenä koin Windows Phonen heti omakseni, eikä opinnäytetyön aiheen valintaakaan tarvinnut miettiä kovin kauaa. Olin suorittanut 'Programming in C#'- sertifikaatin jo ennen työskentelyn aloitusta, mutta minulla ei ollut kokemusta Windows Phonen ohjelmoinnista. Sen vuoksi odotin innolla uusien tekniikoiden oppimista ja niiden kanssa työskentelyä. Opinnäytetyön prosessissa tutkiva oppiminen, ohjelmointi ja opitun dokumentointi kulkivat käsi kädessä heti ensi metreiltä lähtien. Tutkiva oppiminen oli luonteeltaan iteroivaa, eli koodia muutettiin ja sitä testattiin jatkuvasti, ja näin eteen tulleiden ahaa-elämysten kautta suunnitelmaa muutettiin tarpeen mukaan. Ohjelmoitavan sovelluksen tarvitsemat komponentit vaativat syventymistä ja yksityiskohtaisen tiedon omaksumista, mutta itse ohjelmointityö eteni sangen rivakasti alkuun päästyään.

ShakesBeat-sovellus julkaistiin Windows Phone- kaupassa ja se on vapaasti ladattavissa. Sovelluksen kehitys, testaus ja julkaisu ovat kaikki suhteellisen suoraviivaisia tehtäviä. Mainonnan käyttö rahoitusmallina sovellukselle on puolestaan haastavampaa juuri mainonnan vuoksi. Mainonta sovelluksessa ei ole sama asia kuin sovelluksen mainonta. Niinpä julkaistun sovelluksen mainontaan on keskityttävä huolellisesti. Moni kehittäjä pettyy ladattuaan hienon sovelluksensa kauppaan saataville, kun latauksia ei tulekaan omaa ja perheen muiden jäsenten latauksia enempää. Jos sovellus on todellinen "Killer app", niin toki se voi saavuttaa kuuluisuutta käyttäjältä toiselle kulkeutuvan tiedon myötä, mutta kaikki muut sovellukset vaativat mainontaa ja näkyvyyttä. Tässä tulee tietenkin näkyville mainosverkostojen ansaintamalli, eli jos haluat näkyvyyttä omalle sovelluksellesi, maksa mainoksista. Toisaalta kun omaa sovellustasi aletaan ladata suu-

remmissa määrissä, myös omat mainostulosi nousevat samassa suhteessa. Mutta jokainen sovellus tarvitsee näkyvyyttä erottuakseen Windows Phone- kaupan sadasta tuhannesta muusta sovelluksesta. Mainontaan voi käyttää myös perinteisempiä sosiaalisen media kanavia, kuten esimerkiksi Facebookia, LinkedIniä ja Twitteriä. Luomalla omat sivunsa sovellukselleen tai kehittäjätililleen, niitä on helppo mainostaa ilmaiseksi yhteisöjen muiden osapuolten kesken.

ShakesBeat toimi osaltaan testisovelluksena muille tuleville julkaisuille. Niiden seassa on suunnitelmien mukaan vauvoille suunnattu sovellus, musiikin tekoon ja toistoon suunniteltu sovellus ja pelejä. Pelien luontiin tullaan yhdistämään muitakin kehitysalustoja kuin Visual Studio, kuten Construct 2 ja Unity 5. Myös ShakesBeat saa lähiaikoina päivityksiä niin taustakuvilleen, kuin muihinkin toimintoihinsa.

Kuvat

Taulukot

Lähteet

IDC 2014. Press Release: Worldwide Smartphone Growth Forecast to Slow from a Boil to a Simmer as Prices Drop and Markets Mature, According to IDC. Luettavissa: http://www.idc.com/getdoc.jsp?containerId=prUS25282214. Luettu: 22.4.2015.

Whitechapel, A., McKenna, S. 2013. Windows Phone 8 Development Internals. O'Reilly Media Inc. Sebastopol, California.

Falafel Software. 2013. Professional Windows Phone App Development, third edition. Apress. New York, NY.

Kuusinen, N. 2013. Windows 8 Store- ja Windows Phone 8 -sovelluskehitys: C#-ohjelmointikielellä ja XAML-merkintäkielellä. Amk-opinnäytetyö. Hämeen ammatti-korkeakoulu. Visamäki. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/63970/Kuusinen_Niko.pdf?sequence=1. Luettu: 7.3.2015.

Nevala, J. 2013. Windows Phone 8 -sovelluskehityksen perusteet. Amk-opinnäytetyö. Vaasan ammattikorkeakoulu. Vaasa. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/68389/Windows_Phone_8_sovelluskehityksen_perusteet_Jarno_Nevala.pdf?sequence=1. Luettu 20.2.2015.

Brushan, C. 2013.The Brief History of Windows Phone. Luettavissa: http://www.blogginghits.com/2013/07/04/the-brief-history-of-windows-phone/. Luettu: 8.3.2015.

Microsoft 2015 A. Microsoft Support Lifecycle. Luettavissa: http://support.microsoft.com/en-gb/lifecycle?p1=17945. Luettu: 16.3.2015.

Microsoft 2015 B. Windows Phone 8.1:n uudet ominaisuudet. Luettavissa: http://www.windowsphone.com/fi-fi/how-to/wp8/basics/whats-new-in-windows-phone Luettu: 16.3.2015.

Clayton, C. 2014. Modern design at Microsoft: Going beyond flat design. Luettavissa http://www.microsoft.com/en-us/stories/design/. Luettu: 18.3.2015.

Microsoft 2015 C. Microsoft design principles. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/hh781237.aspx. Luettu: 18.3.2015.

Microsoft 2015 D. Quickstart: video and audio (XAML). Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/xaml/hh465160.aspx. Luettu: 23.3.15.

Microsoft 2015 E. Supported audio and video formats (Windows Runtime apps). Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/hh986969.aspx. Luettu: 23.3.2015.

Microsoft 2015 F. MediaElement class. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.ui.xaml.controls.mediaelement.aspx. Luettu: 23.3.2015.

Microsoft 2015 G. Optimize media resources (XAML). Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/xaml/hh994642.aspx. Luettu: 23.3.2015.

Hoekstra, M. 2013. How to play a WAV sound file with DirectX in C# for Windows 8. Luettavissa: http://matthijs.hoekstraonline.net/2013/01/13/how-to-play-a-wav-sound-file-with-directx-in-c-for-windows-8/. Luettu: 23.3.2015.

SharpDX.org 2012. SourceVoice Class (ShrpDX.XAudio2). Luettavissa: http://sharpdx.org/documentation/api/t-sharpdx-xaudio2-sourcevoice. Luettu: 23.3.2015.

Microsoft 2015 H. Accelerometer class. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.devices.sensors.accelerometer.aspx. Luettu: 24.3.2015.

Microsoft 2015 I. AccelerometerReading class. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.devices.sensors.accelerometerreading.aspx?cs-save-lang=1&cs-lang=csharp#code-snippet-1. Luettu: 25.3.2015.

Windows Phone Dev Center. 2015. Converged sensors driver model. Luettavissa: https://dev.windowsphone.com/en-US/OEM/docs/Driver_Components/Converged_sensors_driver_model. Luettu: 25.3.2015.

Microsoft 2015 J. How to continue your Windows Phone app after calling a file picker (XAML). Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/xaml/dn614994.aspx. Luettu: 25.3.2015.

Microsoft 2015 K. Quickstart: Accessing files with file pickers (XAML). Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/xaml/hh771180.aspx. Luettu: 27.3.2015.

Microsoft 2015 L. FileOpenPicker class. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/xaml/windows.storage.pickers.fileopenpicker.aspx. Luettu: 27.3.2015.

Microsoft 2015 M. How to continue your Windows Phone Store app after calling an AndContinue method. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/xaml/dn631755.aspx. Luettu: 27.3.2015.

Microsoft 2015 N. Quickstart: Navigating between pages (XAML). Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/xaml/hh771188.aspx. Luettu: 27.3.2015.

Microsoft 2015 O. MessageDialog class. Luettavissa: https://msdn.microsoft.com/en-US/library/windows/apps/windows.ui.popups.messagedialog. Luettu: 30.3.2015.

Microsoft 2015 P. Application lifecycle (Windows Runtime apps). Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/hh464925.aspx. Luettu: 30.3.2015.

Microsoft 2015 Q. Color. Luettavissa: https://msdn.microsoft.com/library/windows/apps/dn439319.aspx. Luettu: 04.04.2015.

Microsoft 2015 R. Choosing your app images. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/xaml/hh846296.aspx. Luettu: 04.04.2015.

Microsoft 2014 A. Tile design guidelines for Windows Phone. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/jj662929.aspx. Luettu: 04.04.2015.

Microsoft 2015 S. App submission requirements for Windows Phone. Luettavissa: https://msdn.microsoft.com/en-US/library/windows/apps/hh184844(v=vs.105).aspx. Luettu: 04.04.2015.

Microsoft 2015 T. Multi-resolution apps for Windows Phone 8. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/jj206974(v=vs.105).aspx. Luettu: 05.04.2015.

Microsoft 2015 U. Quickstart: Using file or image resources (XAML). Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/xaml/hh965325.aspx. Luettu: 05.04.2015.

Microsoft 2015 V. Guidelines for scaling to pixel density. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/xaml/hh465362.aspx. Luettu: 05.04.2015.

Microsoft 2015 X. Windows/ Phone Dev Center. Luettavissa: http://dev.windows.com/en-us. Luettu: 05.04.2015

Microsoft 2015 Y. Monetizing apps for Windows Phone 8. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/hh202939(v=vs.105).aspx. Luettu: 05.04.2015.

Microsoft 2015 Z. Creating trial apps for Windows Phone 8. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/ff967558(v=vs.105).aspx. Luettu: 05.04.2015.

Microsoft 2015 AA. In-app purchase for Windows Phone 8. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/jj206949(v=vs.105).aspx. Luettu: 5.4.2015.

Microsoft 2015 AB. Advertising in apps for Windows Phone 8. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/hh286399(v=vs.105).aspx. Luettu: 05.04.2015.

Microsoft pubCenter. 2015. Luettavissa: https://pubcenter.microsoft.com. Luettu: 05.04.2015.

Microsoft 2014 B. Introducing the SDK. Luettavissa: https://msdn.microsoft.com/en-us/library/advertising-mobile-windows-phone-sdk-intro(v=msads.20).aspx. Luettu: 05.04.2015.

Microsoft 2014 C. Windows Phone 8.1. XAML Apps – Integrating the AdControl Using XAML Markup. Luettavissa: https://msdn.microsoft.com/en-us/library/advertising-mobile-windows-phone-ads-walkthroughs-81-xaml-markup(v=msads.20).aspx. Luettu: 05.04.2015.

Microsoft 2015 AC. Using ad mediation to maximize ad revenue. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/xaml/dn864359.aspx. Luettu: 06.04.2015.

Microsoft 2015 AD. Selecting and managing your ad networks. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/xaml/dn864356.aspx. Luettu: 06.04.2015.

Microsoft 2015 AE. Adding and using the ad mediation control. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/xaml/dn864355.aspx. Luettu: 06.04.2015.

Microsoft 2015 AF. Testing your ad mediation implementation. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/xaml/dn864358.aspx. Luettu: 06.04.2015.

Microsoft 2015 AG. Submitting your app and configuring ad mediation. Luettavissa: https://msdn.microsoft.com/en-us/library/windows/apps/xaml/dn864357.aspx. Luettu: 06.04.2015.

Liitteet

Liite 1. MainPage.xaml

```
<Page
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   xmlns:local="using:ShakesBeat"
   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
   xmlns:toolkit="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
   xmlns:WindowsPhone81="using:Microsoft.AdMediator.WindowsPhone81"
   x:Class="ShakesBeat.MainPage"
   mc:Ignorable="d">

   <Grid>
      <Grid.Background>
         <ImageBrush Stretch="Fill" ImageSource="Assets/ShakeBeat.png"/>
      </Grid.Background>
      <WindowsPhone81:AdMediatorControl x:Name="AdMediator_8FE733" HorizontalAlignment="Center"
Height="80" Id="AdMediator-Id-239F9727-90DA-47A1-AF58-CBB048FF4BE4" Margin="10" VerticalAlign-
ment="Top" Width="auto"/>
      <Grid HorizontalAlignment="Center" Height="374" Width="374" Margin="0,100,0,120" VerticalAlign-
ment="Top">
         <Grid.Background>
            <ImageBrush Stretch="Uniform" ImageSource="Assets/ShakesBeat_Logo.png"/>
         </Grid.Background>
      </Grid>
      <Grid HorizontalAlignment="Center" Height="100" Margin="0,500,0,20" VerticalAlignment="Bottom"
Width="374" IsDoubleTapEnabled="False" IsHoldingEnabled="False" IsRightTapEnabled="False"
          PointerPressed="PlayNow_PointerPressed">
         <Grid.Background>
            <ImageBrush Stretch="Uniform" ImageSource="Assets/PlayNow.png"/>
         </Grid.Background>
      </Grid>
   </Grid>
</Page>
```

Liite 2. PlayPage.xaml

```
<Page
   x:Class="ShakesBeat.PlayPage"
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
   xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
   xmlns:local="using:ShakesBeat"
   xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
   xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
   xmlns:toolkit="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone.Controls.Toolkit"
   xmlns:WindowsPhone81="using:Microsoft.AdMediator.WindowsPhone81"
   mc:Ignorable="d">
   <Page.Background>
      <ImageBrush Stretch="Fill"/>
   </Page.Background>

   <Grid>
      <Grid.Background>
         <ImageBrush Stretch="Fill" ImageSource="Assets/ShakeBeat.png"/>
      </Grid.Background>
      <WindowsPhone81:AdMediatorControl x:Name="AdMediator_CF46FF" HorizontalAlignment="Center"
Height="80" Id="AdMediator-Id-05933C37-66D0-4F7F-A00F-FF29FD16A86D" Margin="10" VerticalAlign-
ment="Top" Width="auto"/>
      <Grid HorizontalAlignment="Center" Height="374" Width="374" Margin="0,100,0,120" VerticalAlign-
ment="Top">
         <Grid.Background>
            <ImageBrush Stretch="Uniform" ImageSource="Assets/Shake.png"/>
         </Grid.Background>
      </Grid>
      <Grid HorizontalAlignment="Center" Height="100" Margin="0,500,0,20" VerticalAlignment="Bottom"
Width="374" IsDoubleTapEnabled="False" IsHoldingEnabled="False" IsRightTapEnabled="False"
          PointerPressed="Stop_PointerPressed">
         <Grid.Background>
            <ImageBrush Stretch="Uniform" ImageSource="Assets/Stop.png" />
         </Grid.Background>
      </Grid>
      <MediaElement x:Name="ShakesPlayer" Visibility="Collapsed" AudioCatego-
ry="ForegroundOnlyMedia">
      </MediaElement>
   </Grid>

</Page>
```

Liite 3. MainPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;
using Windows.Storage.Pickers;
using ShakesBeat.Common;
using Windows.ApplicationModel.Activation;
using Windows.Storage;
using Microsoft.AdMediator;
using System.Diagnostics;

// The Blank Page item template is documented at http://go.microsoft.com/fwlink/?LinkId=391641

namespace ShakesBeat
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>
    public sealed partial class MainPage : Page
    {

        public static MainPage Current;

        public MainPage()
        {
            this.InitializeComponent();
            Current = this;
            this.NavigationCacheMode = NavigationCacheMode.Required;
            AdMediator_8FE733.AdSdkError += AdMediator_8FE733_AdError;
            AdMediator_8FE733.AdMediatorFilled += AdMediator_8FE733_AdFilled;
```

```
        AdMediator_8FE733.AdMediatorError += AdMediator_8FE733_AdMediatorError;
        AdMediator_8FE733.AdSdkEvent += AdMediator_8FE733_AdSdkEvent;
    }

    void AdMediator_8FE733_AdSdkEvent(object sender, Microsoft.AdMediator.Core.Events.AdSdkEventArgs e)
    {
        Debug.WriteLine("AdSdk event {0} by {1}", e.EventName, e.Name);
    }

    void AdMediator_8FE733_AdMediatorError(object sender, Microsoft.AdMediator.Core.Events.AdMediatorFailedEventArgs e)
    {
        Debug.WriteLine("AdMediatorError:" + e.Error + " " + e.ErrorCode);
        // if (e.ErrorCode == AdMediatorErrorCode.NoAdAvailable)
        // AdMediator will not show an ad for this mediation cycle
    }

    void AdMediator_8FE733_AdFilled(object sender, Microsoft.AdMediator.Core.Events.AdSdkEventArgs e)
    {
        Debug.WriteLine("AdFilled:" + e.Name);
    }

    void AdMediator_8FE733_AdError(object sender, Microsoft.AdMediator.Core.Events.AdFailedEventArgs e)
    {
        Debug.WriteLine("AdSdkError by {0} ErrorCode: {1} ErrorDescription: {2} Error: {3}", e.Name, e.ErrorCode, e.ErrorDescription, e.Error);
    }

    //void HardwareButtons_BackPressed(object sender, Windows.Phone.UI.Input.BackPressedEventArgs e)
    //{
    //    if (ScenarioFrame.CanGoBack)
    //    {
    //        Clear the status block when navigating
    //        NotifyUser(String.Empty, NotifyType.StatusMessage);

    //        ScenarioFrame.GoBack();

    //        Indicate the back button press is handled so the app does not exit
    //        e.Handled = true;
    //    }
```

```
        //}
        /// <summary>
        /// Invoked when this page is about to be displayed in a Frame.
        /// </summary>
        /// <param name="e">Event data that describes how this page was reached.
        /// This parameter is typically used to configure the page.</param>
        protected override void OnNavigatedTo(NavigationEventArgs e)
        {
            // TODO: Prepare page for display here.
            // TODO: If your application contains multiple pages, ensure that you are
            // handling the hardware Back button by registering for the
            // Windows.Phone.UI.Input.HardwareButtons.BackPressed event.
            // If you are using the NavigationHelper provided by some templates,
            // this event is handled for you.
        }
        private void PlayNow_PointerPressed(object sender, PointerRoutedEventArgs e)
        {
            this.Frame.Navigate(typeof(PlayPage));
        }
    }
}
```

Liite 4. PlayPage.xaml.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Runtime.InteropServices.WindowsRuntime;
using Windows.ApplicationModel.Activation;
using Windows.Foundation;
using Windows.Foundation.Collections;
using Windows.Storage.Pickers;
using Windows.UI.Xaml;
using Windows.UI.Xaml.Controls;
using Windows.UI.Xaml.Controls.Primitives;
using Windows.UI.Xaml.Data;
using Windows.UI.Xaml.Input;
using Windows.UI.Xaml.Media;
using Windows.UI.Xaml.Navigation;
using Windows.UI.Popups;
using Microsoft.AdMediator;
using Windows.Devices.Sensors;
using Windows.UI.Core;
using System.Threading.Tasks;
using System.Diagnostics;
using Windows.Media.Core;
using SharpDX;
using SharpDX.XAudio2;
using SharpDX.IO;
using SharpDX.Multimedia;

// The Blank Page item template is documented at http://go.microsoft.com/fwlink/?LinkID=390556

namespace ShakesBeat
{
    /// <summary>
    /// An empty page that can be used on its own or navigated to within a Frame.
    /// </summary>
    public sealed partial class PlayPage : Page, IFileOpenPickerContinuable, IDisposable
    {
        private Accelerometer _accelerometer;
        private AccelerometerReading _lastReading;
        private int _shakeCount = 0;
```

```
        private double _previousMilliSecs = 0.0;

        private bool disposed = false;

        DateTimeOffset _firstReadingTime;
        DateTimeOffset _secondReadingTime;
        DispatcherTimer timer = new DispatcherTimer { Interval = TimeSpan.FromSeconds(3) };

        System.TimeSpan diff;

        Beat beat = new Beat();
        System.TimeSpan spanMil = System.DateTimeOffset.UtcNow.AddMilliseconds(200) - Sys-
tem.DateTimeOffset.UtcNow;
        XAudio2 xAudio;
        SoundStream soundStream;
        WaveFormat waveFormat;
        AudioBuffer buffer;


       public PlayPage()
       {
           this.InitializeComponent();
           AdMediator_CF46FF.AdSdkError += AdMediator_CF46FF_AdError;
           AdMediator_CF46FF.AdMediatorFilled += AdMediator_CF46FF_AdFilled;
           AdMediator_CF46FF.AdMediatorError += AdMediator_CF46FF_AdMediatorError;
           AdMediator_CF46FF.AdSdkEvent += AdMediator_CF46FF_AdSdkEvent;
           beat.rate = 1.0;
          //SharpDX part
           xAudio = new XAudio2();
           var masteringVoice = new MasteringVoice(xAudio);
           var nativeFileStream = new NativeFileStream("Assets/skratch2.wav", NativeFileMode.Open, NativeFil-
eAccess.Read);
           soundStream = new SoundStream(nativeFileStream);
           waveFormat = new WaveFormat();
           waveFormat = soundStream.Format;
           buffer = new AudioBuffer
           {
              Stream = soundStream.ToDataStream(),
              AudioBytes = (int)soundStream.Length,
              Flags = BufferFlags.EndOfStream
           };
```

```
            _accelerometer = Accelerometer.GetDefault();

            if (_accelerometer != null)
            {
                uint minReportInterval = _accelerometer.MinimumReportInterval;
                uint reportInterval = minReportInterval > 60 ? minReportInterval : 60;
                _accelerometer.ReportInterval = reportInterval;
                _lastReading = _accelerometer.GetCurrentReading();
            }
        }

        void AdMediator_CF46FF_AdSdkEvent(object sender, Microsoft.AdMediator.Core.Events.AdSdkEventArgs e)
        {
            Debug.WriteLine("AdSdk event {0} by {1}", e.EventName, e.Name);
        }

        void AdMediator_CF46FF_AdMediatorError(object sender, Microsoft.AdMediator.Core.Events.AdMediatorFailedEventArgs e)
        {
            Debug.WriteLine("AdMediatorError:" + e.Error + " " + e.ErrorCode);
            // if (e.ErrorCode == AdMediatorErrorCode.NoAdAvailable)
            // AdMediator will not show an ad for this mediation cycle
        }

        void AdMediator_CF46FF_AdFilled(object sender, Microsoft.AdMediator.Core.Events.AdSdkEventArgs e)
        {
            Debug.WriteLine("AdFilled:" + e.Name);
        }

        void AdMediator_CF46FF_AdError(object sender, Microsoft.AdMediator.Core.Events.AdFailedEventArgs e)
        {
            Debug.WriteLine("AdSdkError by {0} ErrorCode: {1} ErrorDescription: {2} Error: {3}", e.Name, e.ErrorCode, e.ErrorDescription, e.Error);
        }

        //Accelerometer reading event handler
        async private void readingChanged(object sender, AccelerometerReadingChangedEventArgs e)
        {
            await Dispatcher.RunAsync(CoreDispatcherPriority.Normal, () =>
                {
```

```
            AccelerometerReading reading = e.Reading;

            if (reading != null)
            {
            if (Math.Abs(_lastReading.AccelerationX - reading.AccelerationX) > 1.0 && reading.AccelerationX
< -1.5)
            {
               _shakeCount++;
               Debug.WriteLine("XXXXXXXXXXXXXXXXX: Acceleration: " + reading.AccelerationX);
               Debug.WriteLine("Shakecount: " + _shakeCount);
               if (_shakeCount == 1)
               {
                  _firstReadingTime = reading.Timestamp;
               }
               if (_shakeCount == 4)
               {
                  _secondReadingTime = reading.Timestamp;
                  diff = _secondReadingTime - _firstReadingTime;


                  if(diff.TotalMilliseconds + spanMil.Milliseconds > _previousMilliSecs || diff.TotalMilliseconds
- spanMil.Milliseconds < _previousMilliSecs) {
                  setRate(diff.TotalMilliseconds);
                  Debug.WriteLine("diff total milliseconds to set rate: " + diff.TotalMilliseconds);
                  }
                  _shakeCount = 0;
               }

               Debug.WriteLine("diff: " + diff);
            }
            }
         });
      }

      private void timerStart()
      {
         timer.Tick += timerTick;
         timer.Start();
      }

      private void timerStop()
      {
```

```
    timer.Stop();
}

private void timerTick(object sender, object e)
{
    if (Math.Round(beat.rate,1) != Math.Round(ShakesPlayer.PlaybackRate,1))
    {
        var sourceVoice = new SourceVoice(xAudio, waveFormat, true);
        sourceVoice.SubmitSourceBuffer(buffer, soundStream.DecodedPacketsInfo);
        sourceVoice.Start();
        ShakesPlayer.PlaybackRate = beat.rate;
    }
    Debug.WriteLine("Timer: " + tickerInt++);
    Debug.WriteLine("           PlayBackRate: " + Math.Round(ShakesPlayer.PlaybackRate,1));
}

private int tickerInt;

private void setRate(double s)
{
    Debug.WriteLine("SetRate shakescount parameter: " + s);
    double newRate = 1.0;

    if (s > 1900)
    {
        newRate = Math.Round(0.4, 1);
    }
    else if (s > 1700 && s < 1901)
    {
        newRate = Math.Round(0.6, 1);
    }
    else if (s > 1500 && s < 1701)
    {
        newRate = Math.Round(0.7, 1);
    }
    else if (s > 1400 && s > 1501)
    {
        newRate = Math.Round(0.8, 1);
    }
    else if (s > 1300 && s < 1401)
    {
        newRate = Math.Round(0.9, 1);
```

```
}
else if (s > 1200 && s < 1301)
{
   newRate = Math.Round(1.0, 1);
}
else if (s > 1100 && s < 1201)
{
   newRate = Math.Round(1.0, 1);
}
else if (s > 900 && s < 1101)
{
   newRate = Math.Round(1.1, 1);
}
else if (s > 800 && s < 901)
{
   newRate = Math.Round(1.3, 1);
}
else if (s > 600 && s < 801)
{
   newRate = Math.Round(1.5, 1);
}
else if (s > 500 && s < 601)
{
   newRate = Math.Round(1.9, 1);
}
else if (s > 400 && s < 501)
{
   newRate = Math.Round(2.1, 1);
}
else if (s > 300 && s > 401)
{
   newRate = Math.Round(2.3, 1);
}
else if (s > 200 && s > 301)
{
   newRate = Math.Round(2.5, 1);
}
else if (s < 201)
{
   newRate = Math.Round(2.7, 1);
}
beat.rate = newRate;
```

```
}


/// <summary>
/// Invoked when this page is about to be displayed in a Frame.
/// </summary>
/// <param name="e">Event data that describes how this page was reached.
/// This parameter is typically used to configure the page.</param>
protected override void OnNavigatedTo(NavigationEventArgs e)
{

    var openPicker = new Windows.Storage.Pickers.FileOpenPicker
    {
        ViewMode = PickerViewMode.List,
        SuggestedStartLocation = PickerLocationId.MusicLibrary
    };
    openPicker.FileTypeFilter.Add(".wma");
    openPicker.FileTypeFilter.Add(".mp3");
    openPicker.FileTypeFilter.Add(".wav");
    openPicker.PickSingleFileAndContinue();
}

protected override void OnNavigatingFrom(NavigatingCancelEventArgs e)
{
    _accelerometer = null;
    ShakesPlayer = null;
    base.OnNavigatingFrom(e);
}

public async void ContinueFileOpenPicker(FileOpenPickerContinuationEventArgs args)
{
    var file = args.Files.FirstOrDefault();
    if (file == null)
    {
        this.Frame.Navigate(typeof(MainPage));
    }
    else
    {
        if (file != null)
        {
            var stream = await file.OpenAsync(Windows.Storage.FileAccessMode.Read);
            ShakesPlayer.SetSource(stream, file.ContentType);
```

```
                ShakesPlayer.DefaultPlaybackRate = 1.0;
                ShakesPlayer.Play();
                //assign event handler
                _accelerometer.ReadingChanged += new TypedEventHandler<Accelerometer, Accelerometer-
terReadingChangedEventArgs>(readingChanged);
                timerStart();
            }
        }
    }

    public void Dispose()
    {
        Dispose(true);
        GC.SuppressFinalize(this);
    }

    private void Dispose(bool disposing)
    {
        if (disposed)
            return;

        if (disposing)
        {
            soundStream.Dispose();
            xAudio.Dispose();
            // Free any other managed objects here.
            //
        }

        // Free any unmanaged objects here.
        //
        disposed = true;
    }

    private void stopSong()
    {
        ShakesPlayer.Stop();
        timerStop();
        this.Frame.Navigate(typeof(MainPage));
    }

    private void continueSong()
```

```
        {
            ShakesPlayer.Play();
        }

        private async void Stop_PointerPressed(object sender, PointerRoutedEventArgs e)
        {
            ShakesPlayer.Pause();
            var dialog = new MessageDialog("Really want to stop playing?", "Paused");
            dialog.Commands.Add(new UICommand("Don't stop", new UICommandIn-
vokedHandler(commandConfirm)));
            dialog.Commands.Add(new UICommand("Stop", new UICommandIn-
vokedHandler(commandConfirm)));
            await dialog.ShowAsync();
        }

        private void commandConfirm(IUICommand command)
        {
            var commandLabel = command.Label;
            switch (commandLabel)
            {
                case "Stop": stopSong(); break;
                case "Don't stop": continueSong(); break;
            }
        }
    }
}
```