

RSS-syötteiden lukijan toteutus Windows Phone 8 -alustalle

Tuomas Becks

Opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

2013



Tietojenkäsittelyn koulutusohjelma

Tekijä tai tekijät Tuomas Becks	Ryhmätunnus tai aloitusvuosi 2009
Raportin nimi RSS-syötteiden lukijan toteutus Windows Phone 8 -alustalle	Sivu- ja liitesivumäärä 29 + 1
Opettajat tai ohjaajat Juhani Välimäki	
<p>Tämän opinnäytetyön tarkoituksena on toteuttaa RSS-syötteiden lukijan prototyyppi Windows Phone 8 -käyttöjärjestelmälle. Syntyvän ohjelman avulla käyttäjä pystyy hallinnoimaan RSS-syötteitä ja lukemaan niitä omalla älypuhelimellaan. Sovellus ei tähtää täysin valmiiseen sovellukseen, vaan vasta prototyyppiin, jota on tarkoitus kehittää eteenpäin. Projektin tavoitteena on myös oman tietämyksen lisääminen ja uuden oppiminen.</p> <p>Työssä toteutettiin vain RSS-lukijan peruskäytön kannalta tarvittavat ominaisuudet. Kosmeettisiin virheisiin ei kiinnitetty tässä vaiheessa huomiota. Tuotteen vaatimuksissa olevien ominaisuuksien tuli silti toimia ilman ongelmia.</p> <p>Ennen varsinaista sovelluksen toteuttamista otettiin selvää, miten sovellus voitaisiin toteuttaa Windows Phone 8 -käyttöjärjestelmälle. Sovellus päätettiin toteuttaa oletusten mukaisesti C#- ja XAML-kielillä. Tietokantaratkaisuksi valittiin SQLite. Työssä kehitetty sovellus täytti määritetyt tuotteen vaatimukset.</p> <p>Projektissa syntynyt sovellus tarjoaa hyvän pohjan jatkokehitystä varten. Lisäksi projektin aikana opittu tieto on ollut erittäin hyödyllistä oman tietämyksen lisäämisen kannalta.</p>	
Asiasanat Windows Phone 8, RSS, C#, SQLite	

Degree Programme in Business Information Technology

Authors Tuomas Becks	Group or year of entry 2009
The title of thesis Implementing an RSS reader for Windows Phone 8	Number of report pages and attachment pages 29 + 1
Advisor(s) Juhani Välimäki	
<p>The purpose of this thesis was to create a RSS reader prototype for the Windows Phone 8 operating system. By using this program, users can manage and read RSS feeds on their smartphones. The aim was not to produce a totally complete program but a prototype that is intended to be developed further in the future.</p> <p>Before the actual implementation, it was necessary to find information about how to create such a program for the Windows Phone 8 operating system. The actual implementation of the program was carried out in C# and XAML programming languages. SQLite was used as the database solution.</p> <p>Within this thesis project, only the basic RSS reader features were implemented. Cosmetic errors did not receive attention at this stage. However, the features stated in the product requirement list were to work without issues.</p> <p>The program developed in this thesis project met the requirements established in the program requirements and it provides a good basis for further development.</p>	
Key words Windows Phone 8, RSS, C#, SQLite	

Sisällys

1	Termit	1
2	Johdanto	2
2.1	Työn tavoite	2
3	Teoriatausta.....	4
3.1	XML.....	4
3.1.1	RSS	4
3.2	SQLite.....	6
3.2.1	Vaihtoehtoisia tietokantoja.....	7
3.3	Windows Phone 8 kehitysalustana.....	8
3.4	Käytettävyys & käyttöliittymä.....	9
4	Tutkimussuunnitelma	12
5	Projektin toteutus ja kehitys.....	13
5.1	Määrittäminen.....	13
5.2	Teknologian selvitys.....	13
5.3	Työkalujen valinta	14
5.4	Suunnittelu ja toteutus	14
5.4.1	Käyttöliittymän toteutus	15
5.4.2	Tietokannan suunnittelu ja toteutus.....	21
5.5	Laadunvarmistus ja testausta	23
6	Toteutunut työ, lopputulokset ja johtopäätökset.....	24
6.1	Ongelmat ja niiden ratkaisut	24
6.2	Valittujen työkalujen ja menetelmien arviointi	25
6.3	Projektin eteneminen ja työn lopputulos	26
6.4	Oppiminen	27
7	Yhteenveto	28
7.1	Jatkokehitysehdotukset.....	28
	Lähteet.....	30
	Liitteet.....	33

Liite 1. Tuotteen vaatimukset	33
-------------------------------------	----

1 Termit

Termi	Selitys
Windows Phone 8	Microsoftin älypuhelinkäyttöjärjestelmä
SQLite	relaatiotietokanta
XAML	XAML on deklarativinen merkinmäkekieli, joka on XML:n mukaista
artikkeli	RSS-syötteen item-elementti
syöte	RSS-dokumentti
Visual Studio	Microsoftin ohjelmointiympäristö
C#	ohjelmointikieli
MVVM	Model-View-ViewModel, sovellusarkkitehtuurimalli
Context menu	Valikko, josta käyttäjä voi valita aiheeseen liittyviä valintoja
Application bar (app bar)	Palkki näytön alaosassa nappuloille. Laajennettuna toimii myös valikkona.
SLAT	Lyhenne sanoista Second Level Address Translation. Prosessorin rautatason ominaisuus, jota voidaan käyttää virtualisoinnin tehostamiseen.
Hyper-V	Hypervisor Windowsille. Hypervisorin avulla hallitaan virtuaalikoneita.

2 Johdanto

Tämä työ toimii opinnäytetyönä. Työn tarkoituksena on toteuttaa RSS-syötteiden lukija Windows Phone 8 -käyttöjärjestelmälle. Koska Windows Phone 8 -kehitys on minulle uusi asia, tämä projekti on ennen kaikkea myös oppimiskokemus ja tärkeä osa oman osaamisen kehittämistä. Projekti tähtää toimivan prototyypin luomiseen, ja tarkoitus on julkaista valmis ohjelma myöhemmin Windows Phone Store -sovelluskaupassa. Toisin sanoen tässä opinnäytetyössä syntyvä prototyyppi on tarkoitettu jatkokehitettäväksi.

Tämä työ on kokonaisuus, joka on enemmän kuin pelkkä sovelluksen toteutus, sillä se vaatii itseopiskelua ja soveltamista. Työn tavoitteena syntyvä sovellus ei itsessään ole kovin monimutkainen. Haastavuutta työssä lisää uusi alusta, kieli ja kehitysympäristö. C# ja Visual Studio eivät ole aivan uusia tuttavuuksia, mutta uuden opettelemista on silti paljon. Java sen sijaan on varsin tuttu ohjelmointikieli, ja sen osaaminen auttaa huomattavasti C#:n opettelemisessa, sillä syntaksiltaan C# on kielenä hyvin samantyyppinen kuin Java. Projektin tavoite on kohtuullinen ja erittäin hyvin toteutettavissa sillä tietämyksellä, mitä minulla on. Työ on sopivan haastava, mutta ei missään nimessä mahdoton toteuttaa.

Idea tämän työn toteuttamiselle on lähtöisin käytännön tarpeesta. Tahdon sellaisen sovelluksen, joka ei vaadi kolmannen osapuolen palveluita tai käyttäjätilejä. Lisäksi haluan sovelluksen olevan sellainen, mikä soveltuu omaan käyttötarkoitukseeni parhaiten ja mitä voi käyttää älypuhelimella. Windows Phone 8 ei myöskään itsessään sisällä RSS-lukijaa. Halu oppia Windows Phone 8 -sovelluskehitystä on myös yksi tärkeä syy tälle työlle.

2.1 Työn tavoite

Työn tavoitteena on toteuttaa toimiva prototyyppi Windows Phone 8 -käyttöjärjestelmälle, jonka avulla käyttäjä pystyy lisäämään, poistamaan, muokkaamaan ja lukemaan RSS (Really Simple Syndication) -syötteitä omalla älypuhelimellaan. Käyttäjän tulisi pystyä myös merkitsemään syötteitä sekä luetuiksi että lukemattomiksi. Atom-standardia ei tueta tässä prototyypissä. Lisäksi käyttäjän tulisi nähdä selvästi, mitkä koh-

teet on jo luettu ja mitkä ovat uusia eli lukemattomia. Toteutettavassa prototyypissä syötteet lisätään kirjoittamalla tai kopioimalla URL-osoite sovellukseen.

Sovellus ei tule vaatimaan mitään käyttäjätilejä, eikä se tule olemaan pilvipalvelu. Kaikki tapahtuu käyttäjän puhelimessa. Syötteiden hakeminen palvelimilta tarvitsee luonnollisesti internetyhteyden, mutta kaikki syötteisiin liittyvä käsittely tehdään puhelimessa. Syötteet haetaan palvelimelta manuaalisesti käyttäjän käskystä, eikä hakemista ole automatisoitu taustaprosessiksi. Tässä vaiheessa sovellukselle ei toteuteta esimerkiksi live tile -tukea.

Työssä ei toteuteta sovellusta, joka olisi valmis kaupalliseen käyttöön. Tarkoituksena on luoda syötteidenlukijan runko ja tutkia, miten kyseinen sovellus pystytään toteuttamaan Windows Phone 8 -käyttöjärjestelmälle ja mitä työkaluja ja teknologioita sen toteuttamiseksi vaaditaan

3 Teoriatausta

3.1 XML

XML (Extensible Markup Language) -kieli on kehitetty tekstimuotoisen datan esittämiseen. Se on mukautuva, ja sen käyttötarkoitukset ovat monenlaiset. XML on yksi eniten käytetyistä formaateista strukturoidun tiedon esittämisessä. Ensimmäinen luonnos XML -määrittäyksestä julkaistiin lokakuun 14. päivänä 1996. Kielenä XML on siis jo kohtuullisen vanha. XML on alusta asti suunniteltu unicode-merkistöä silmällä pitäen, mikä takaa hyvän yhteensopivuuden eri kielten kanssa. XML:ää on hyödynnetty monessa eri käyttötarkoituksessa. Muun muassa XHTML perustuu XML:ään ja on täysin XML-standardin mukainen. (W3C 1996; W3C 2013a.)

XML-dokumentti voi olla rakenteeltaan hyvin yksinkertainen (esimerkki 1).

```
<?xml version="1.0" encoding="UTF-8"?>
<autot>
  <auto>
    <merkki>Audi</merkki>
    <malli>A4</malli>
    <omistaja>Matti Meikäläinen</omistaja>
  </auto>
  <auto>
    <merkki>BMW</merkki>
    <malli>M6</malli>
    <omistaja>Maija Meikäläinen</omistaja>
  </auto>
</autot>
```

Esimerkki 1. Kuvitteellinen XML-dokumentti, jossa esitetään kaksi autoa sekä autojen merkki, malli ja omistaja

3.1.1 RSS

RSS eli Really Simple Syndication on XML:n murre, joka noudattaa World Wide Web Consortiumin XML 1.0 -standardia. RSS-syöte on XML-dokumentti, joka sisältää yleensä esimerkiksi uutisia tai tietoja uusista julkaisuista internetsivuilla. (RSS Advisory Board 2009.)

Muun muassa Yleisradio julkaisee RSS-syötteitä uusista uutisista, jotta Yleisradion uutisia pystyttäisiin seuraamaan helpommin.

RSS-syötteen rakenne on varsin yksinkertainen. Ensiksi RSS-dokumentissa on rss-elementti, jonka attribuutissa määritetään RSS:n versio. Seuraavaksi tulee channel-elementti, joka sisällä täytyy olla pakolliset elementit, eli title, link ja description. Title-elementti sisältää kanavan nimen, ja link sisältää url:in, joka linkittää pääsivulle, jossa syötteen sisältäviä artikkeleita julkaistaan. Description-elementti sisältää RSS-syötteen kuvauksen. Channel-elementtiin voidaan myös laittaa muutamia vaihtoehtoisia elementtejä, jotka on määritelty RSS-määrittelyissä. (RSS Advisory Board 2009.)

Syötteen tärkein tieto sisältyy item-elementtiin, joka yleensä sisältää esimerkiksi uutisen tiedot. Item-elementti ei varsinaisesti sisällä pakollisia elementtejä, mutta siinä on oltava ainakin title- tai description-elementti. Title sisältää esimerkiksi uutisen otsikon ja description taas sisältää artikkelin tiivistelmän. Jossain tapauksissa description voi sisältää myös koko artikkelin. Description-elementti voi myös sisältää HTML-koodia puhtaan tekstin sijasta. Link-elementti sisältää url-osoitteen kyseiseen artikkeliin, jotta syötettä seuraava henkilö pystyisi lukemaan koko artikkelin siltä sivustolta, missä se on julkaistu. Item-elementtien lukumäärää ei ole rajoitettu. (RSS Advisory Board 2009.)

Muita paljon käytössä olevia elementtejä ovat author, category, enclosure sekä pubDate. Author-elementti sisältää sen henkilön sähköpostiosoitteen, joka kirjoitti kyseisen artikkelin. Category sisältää artikkelin kategorian, esimerkiksi ”Ulkomaan uutiset”. Enclosure on tarkoitettu tiedostoihin linkittämiseen, ja siihen kuuluvat seuraavat pakolliset attribuutit: url, length ja type. Url on linkki tiedostoon, length on kohteen koko bitteinä ja type sisältää kohteen MIME-tyypin. Enclosure-elementtiä käytetään yleensä media-tiedostoihin linkittämiseen. PubDate sisältää artikkelin julkaisuaikaleiman, jonka tulee noudattaa RFC 822 -standardia. (RSS Advisory Board 2009.)

Koska RSS-määrittelyn author-elementti on tehty sähköpostiosoitetta varten, eikä pelkkää nimeä varten, on varsin yleistä, että esimerkiksi author-elementti korvataan Dublin Core Metdata Initiativen creator-elementillä (esimerkki 2). (Dublin Core Metadata Initiative 2012; RSS Advisory Board 2009.)

```

<item>
<title><![CDATA[Mint's personal finance app comes to Windows devices with
Live Tile support]]></title>
<link>http://www.engadget.com/2013/12/04/mint-windows-8-windows-phone-
8/?ncid=rss_truncated</link>
<description><![CDATA[ It's been a long time
coming, but Mint has finally released a version of its personal finance
app for both Windows 8.1 and Windows Phone 8. Just like its Android and
iOS iterations, the app lets users track their account balances, spending
habits and ...]]></description>
<dc:creator><![CDATA[Nicole Lee]]></dc:creator>
<pubDate>Wed, 04 Dec 2013 08:30:00 -0500</pubDate>
<guid isPermaLink="false">http://www.engadget.com/2013/12/04/mint-
windows-8-windows-phone-8/</guid>
</item>

```

Esimerkki 2. Item-elementti on otettu Engadget.com-sivuston RSS-syötteestä. Author-elementtiä ei ole, mutta sen sijaan on käytetty Dublin Core Metadata Initiativeen creator-elementtiä

Atom-standardi on vaihtoehtoinen standardi RSS:n rinnalle, ja se on kattavampi ja joustavampi kuin RSS. Atom-standardin author-elementtiin voidaan esimerkiksi määrittää nimi ja sähköposti erikseen omille elementeilleen. (Nottingham, M. & Sayre, R. 2005.)

3.2 SQLite

SQLite on itsenäinen avoimen koodin SQL-tietokanta, joka ei vaadi konfigurointia, asennusta tai palvelinta, ja sen vuoksi SQLiten käytettävyys on varsin monipuolinen ja se soveltuu monenlaiseen käyttötarkoitukseen. SQLite on kirjoitettu ANSI-C:lla, minkä ansiosta sen kääntäminen pitäisi onnistua melkein millä tahansa standardilla C-kääntäjällä. Tämän takia SQLite toimii monessa eri käyttöjärjestelmässä, muun muassa myös mobiilikäyttöjärjestelmissä, kuten Androidissa ja Windows Phone 8:ssa. SQLite tukee monia muista SQL-tietokannoista tuttuja ominaisuuksia, kuten esimerkiksi trigge-reitä ja näkymiä. Myös transaktiot ovat tuettuja, mikä lisää kannan vikasietoisuutta. Tie-tokanta, jonka SQLite luo, on myös täysin alustariippumaton. Esimerkiksi Androidilla luotu tietokanta toimii suoraan 64-bittisessä työpöytä-Windowsissa. SQLiten vanhemman version on myös osoitettu olevan varsin kilpailukykyinen nopeudeltaan. (Android Developers; SQLite a; SQLite b; SQLite c; SQLite d.)

3.2.1 Vaihtoehtoisia tietokantoja

Yksi vaihtoehto SQLiten tilalle on käyttää tekstitiedostopohjaista tietokantaratkaisua, kuten esimerkiksi XML:ää tiedontallennusta varten. XML:lle on kehitetty XQuery-kyselykieli, jonka kehittämistä vastaa W3C:n työryhmä. XQuery mahdollistaa kyselyiden tekemisen jäsennetylle tai osittain jäsennetylle XML:lle samantyyppisesti kuin yleisiin SQL-relaatiotietokantoihin. (Microsoft TechNet ; W3C 2013b.)

Tekstitiedostopohjaisten tietokantojen käsitteleminen suurella tietomäärällä on hyvin hidasta verrattuna relaatiotietokantoihin. Relaatiotietokannat ovat hyvin optimoituja ja soveltuvat raskaisiin palvelinsovelluksiin. Lisäksi relaatiotietokannat tukevat ominaisuuksia, jotka tekevät niistä paremmin soveltuvia raskaaseen käyttöön. Esimerkiksi XML-pohjainen tietokanta soveltuu aivan hyvin sellaisiin sovelluksiin, joissa tietoa ei ole kuin muutama sata riviä. Pienikokoiselle tietokannalle tiedostopohjainen ratkaisu voi olla jopa nopeampi kuin relaatiotietokanta, mutta käytännön nopeudessa ei silti ole juuri havaittavaa eroa, koska molemmat olisivat nopeita. (Bourret, R. 2005.)

Tekstitiedostopohjainen tietokanta on hyvin siirräntärajoitteinen. Myös tiedon järjestyksellä on merkitystä hakemisnopeuteen. Relaatiotietokannat ovat hyvin optimoituja suuren datamäärän käsittelyyn, oli kyse sitten poistamisesta, lisäämisestä tai muokkauksesta. Kun kyselyt ovat yksinkertaisia ehdoiltaan, XML-pohjainen tietokantaratkaisu voi olla relaatiotietokantaa nopeampi. (Bourret, R. 2005; Williams, A. 2005. 45.)

Ei silti ole aivan yksiselitteistä, mikä tietokantaratkaisu on nopeampi, sillä nopeus vaihtelee tilanteesta riippuen. Pääasia on, että valitaan käyttötarkoitukseen parhaiten sopiva tietokantaratkaisu.

Esimerkiksi Windows Phone 8 -käyttöjärjestelmässä sovelluskehittäjä voi valita SQLiten sijasta Windows Phone local databasen. Local database on siinä mielessä hyvin lähellä SQLiteä, että myös se on SQL-relaatiotietokanta. Local databasea voi pitää Microsoft SQL Serverin kevytversiona, joka on räätälöity lokaalisti käytettävää sovellusta varten. Se ei myöskään SQLiten tapaan vaadi palvelinta, eikä se liioin pyöri taustalla

koko ajan. Local database ei luonnollisesti tue kaikkia ominaisuuksia, mitä esimerkiksi Microsoftin SQL Server tukisi. (Windows Phone Dev Center 2013e.)

Windows Phone local databasen huonona puolena on se, että sitä käyttävä sovellus voi olla varsin hankala siirtää toiselle käyttöjärjestelmälle, sillä Windows Phone local database on Microsoftin oma Windows Phone tietokanta. SQLite toimii myös muilla käyttöjärjestelmillä. (SQLite a; Windows Phone Dev Center 2013e.)

3.3 Windows Phone 8 kehitysalustana

Microsoftilla on pitkä historia mobiilikäyttöjärjestelmien kehittämisessä jo 2000-luvun alkupuolelta alkaen. Microsoft päätti koota mobiilikäyttöjärjestelmänsä alusta alkaen, ja samalla käytettävyys muuttui radikaalisti Windows Phone 7:ssä. Lisäksi Microsoft hylkäsi Windows Mobile -nimen ja siirtyi Windows Phone -nimen käyttöön. Windows Phone 8 perustuu käyttöliittymältään ja visuaaliselta ilmeeltään Windows Phone 7 -käyttöjärjestelmään. Suurin ero on pinnan alla. Windows Phone 8:ssä on siirrytty Windows NT -ytimen käyttöön. Windows Phone -käyttöjärjestelmä on koodiltaan suljettu. (Wikipedia a; Wikipedia b; Wikipedia c; Wikipedia d.)

Microsoft kontrolloi tiukasti sovellusten jakelua käyttöjärjestelmälleen, eikä sovelluksia voida ladata, ostaa tai asentaa muualta kuin Microsoftin omasta sovelluskaupasta. Windows Phone 8 on suljettu käyttöjärjestelmä toisin kuin avoimeen lähdekoodiin perustuva Android. (Android; Wikipedia d; Windows Phone Dev Center 2013d.)

Windows Phone 8 tuo monia uudistuksia sovelluskehittäjiä silmällä pitäen, sillä uusia ohjelmointirajapintoja on lisätty ja aikaisemman C#:n lisäksi ohjelmoija voi halutessaan myös koodata natiivisti C++:lla, mutta tällöin XAML UI:ta ei tueta. Samalla myös natiivi SQLite-tuki tuli mahdolliseksi, mikä myös helpottaa kehitettäessä samaa sovellusta eri alustoille samanaikaisesti. C++-tuen ansiosta pelien siirtäminen alustalta toiselle helpottui ja suorituskyky parani, sillä esimerkiksi graafisesti vaativia pelejä on hankala toteuttaa C#:lla tehokkaasti. (SQLite c; Windows Phone Dev Center 2013f.)

Tavanomaiset sovellukset toteutetaan .NET:illä tai/ja XAML:illa. C#:n, ja Visual Basicin lisäksi ohjelmoija voi halutessaan toteuttaa sovelluksensa myös C++:aa avuksi käyttäen. C++ on paras valinta silloin, kun halutaan toteuttaa Direct3D-sovellus. XAML on Microsoftin kehittämä deklaratiiivinen merkintäkieli, joka on XML:n mukaista. XAML on kehitetty helpottamaan ja nopeuttamaan käyttöliittymän kehittämistä .NET Framework -sovelluksille. (Microsoft Developer Network; Windows Phone Dev Center 2013a.)

Windows Phone 8 -kehitystyökalut ovat saatavissa ilmaiseksi Microsoftilta. Nämä työkalut sisältävät muun muassa kehitysympäristön (Visual Studio Express 2012 for Windows Phone) ja Windows Phone 8 -emulaattorin. Windows Phone 8.0 SDK vaatii toimiakseen Windows 8 Professional x64 - tai Enterprise x64 -käyttöjärjestelmän. Tietokoneen suorittimen on tuettava muun muassa SLAT-teknologiaa. Käytännössä sovelluskehittäjä ei voi asentaa Windows Phone -sovelluskehitystyökaluja muille käyttöjärjestelmille kuin Windowsille. Jotta sovelluskehittäjä voisi käyttää esimerkiksi omaa Windows Phone -puhelintaan sovelluksen testaamiseen, pitää kyseinen puhelin ensiksi rekisteröidä sovelluskehitykseen. (Microsoft Download Center 2012; Windows Phone Dev Center 2013c.)

Model-View-ViewModel (MVVM) -suunnittelumalli on hyvin yleisesti käytössä Windows Phone 8 -sovelluksissa. Microsoft käyttää esimerkeissään varsin usein MVVM-mallia. Yleisesti ottaen MVVM:n ideana on se, että tietojen käsittely ja käyttöliittymä pidetään erillään. MVVM-mallin mukainen toteutus voitaisiin tehdä esimerkiksi toteuttamalla Model-luokka C#:lla, jossa käsitellään dataa, ja View-käyttöliittymänäkymät, jotka on toteutettu XAML-kielellä. (Windows Phone Dev Center 2013b.)

3.4 Käytettävyys & käyttöliittymä

Jakob Nielsen määrittelee käytettävyyden koostuvan viidestä osasta. Näitä ovat opittavuus, tehokkuus, muistettavuus, virheet ja tyytyväisyys. (Nielsen 1993, 26)

Opittavuus on tärkeää, jotta käyttäjä pystyisi nopeasti aloittamaan työskentelyn. Tehokas käyttöliittymä voi olla hankala oppia, mutta kokenut käyttäjä pystyy tekemään työn-

sä tehokkaammin. Käyttöliittymän muistettavuus helpottaa käytettävyyttä sovelluksen ollessa luonteeltaan sellainen, ettei sitä käytetä jatkuvasti. Virheitä pitäisi välttää, ja käyttäjälle pitäisi tarjota yksinkertainen ratkaisu niistä selviämiseen. Systeemin täytyisi olla sellainen, että käyttäjä on tyytyväinen sen käytettävyyteen. (Nielsen 1993, 26)

Nielsen on tiivistänyt käytettävyyden kymmeneen periaatteeseen, joita olisi hyvä noudattaa ja jotka olisi hyvä pitää mielessä käytettävyyttä ajatellen. Nielsen itse kutsuu näitä periaatteita käytettävyyden heuristiikoiksi. (Nielsen 1993, 115.)

Ensimmäinen periaate on se, että dialogien on hyvä olla yksinkertaisia ja luonnollisia. Käyttöliittymän tulisi olla mahdollisimman pelkistetty ja yksinkertainen. Muun muassa käyttöliittymän komponenttien ryhmittelyllä saadaan käyttöliittymästä selkeämpi. (Nielsen 1993, 115–123.)

Toinen periaate on se, että käyttöliittymän termien on oltava helposti ymmärrettäviä. Ne eivät saa olla liian teknisiä, koska niiden on oltava käyttäjälle itsestään selviä. (Nielsen 1993, 123.)

Kolmas periaate on pienentää käyttäjän muistitaakkaa. Käyttäjän muistitaakka täytyisi näet pitää mahdollisimman pienenä. Ei voida olettaa, että käyttäjä muistaa kaiken tiedon. Käyttäjälle pitäisi näyttää tarvittavat tiedot koko ajan. (Nielsen 1993, 129.)

Neljäs periaate on yhteneväisyys. Käytettävyyden pitäisi olla yhdenmukainen koko ajan. Saman toiminnon pitäisi toteuttaa sama asia, eikä käyttäjän pitäisi joutua arvelemaan, mitä toiminto tekee. (Nielsen 1993, 132.)

Viides periaate on palautteen antaminen. Käyttöliittymän tulisi antaa palautetta esimerkiksi jonkin tilanteen etenemisestä. Jos jokin toiminto on käytännössä välitön, toiminnon edistymisen näyttäminen ei ole tarpeen. Jos toiminto kumminkin kestää kauan, on esimerkiksi edistymispalkki hyvä tapa näyttää käyttäjälle sovelluksen tila. (Nielsen 1993, 134–136.)

Kuudes periaate on ulosmenojen selkeät merkinnät. Käyttäjän täytyisi pystyä esimerkiksi pääsemään pois nopeasti jostakin erikoisnäköymästä perustilaan ilman monimutkaista navigointia. Käyttäjälle pitäisi tarjota selkeä mahdollisuus peruuttaa jokin toiminto. (Nielsen 1993, 138–139.)

Seitsemäs periaate on pikavalintojen tarjoaminen. Kokeneelle käyttäjälle tulisi tarjota toimintoja, jotka nopeuttavat käytettävyyttä ja lisäävät tehokkuutta. Erittäin usein käytettäville ominaisuuksille olisi hyvä löytyä pikavalinta. (Nielsen 1993, 139–140.)

Kahdeksas periaate on antaa käyttäjälle hyviä virheilmoituksia. Virheilmoitusten on oltava selkeitä ja helposti ymmärrettäviä, ja niiden on ilmoitettava selvästi, missä tai mikä virhe tapahtui. Lisäksi virheilmoituksesta pitäisi käydä selväksi, mitä käyttäjä voi tehdä tilanteen ratkaisemiseksi. (Nielsen 1993, 142–143)

Yhdeksäs periaate on virheiden estäminen. Virheitä voi estää antamalla selkeitä ja yksiselitteisiä virheilmoituksia ja vahvistamalla, käyttäjältä ennen kun jokin toiminto suoritetaan. (Nielsen 1993, 145–146)

Kymmenes periaate on ohjeiden ja dokumentaation antaminen. Käyttäjälle pitäisi tarjota ohjeet ja dokumentaatio, josta systeemin käyttö käy helposti ja selkeästi selville. Parasta tietysti olisi, jos systeemi olisi niin yksiselitteinen, ettei sen käyttöön tarvittaisi dokumentaatiota. (Nielsen 1993, 148–149)

4 Tutkimussuunnitelma

Projektissa toteutetaan sovellus, jolla luetaan RSS-syötteitä Windows Phone 8 -käyttöjärjestelmässä. Sovelluksella pitäisi pystyä lisäämään, poistamaan, muokkaamaan ja lukemaan RSS-syötteitä. Lisäksi käyttäjän täytyisi pystyä merkitsemään syötteitä luetuiksi tai lukemattomiksi. Tuotteen vaatimukset löytyvät Tuotteen vaatimukset -liitteestä (liite 1). Työhön kuuluu sopivien menetelmien tutkiminen ennen työn aloittamista. Oletuksena oli, että sovellus voitaisiin toteuttaa C#:lla ja tietokanta SQLitellä. Muita mahdollisia kielii ja teknologioita olivat XAML, HTML5 ja CSS. Projektissa oli käytössä vapaamuotoinen iteratiivinen menetelmä, jossa tuotteen vaatimukset toteutetaan ja testataan ennen siirtymistä seuraavaan tuotevaatimukseen.

Sovellus täyttää laadulliset vaatimukset silloin, kun ohjelmalla pystytään tekemään Tuotteen vaatimukset -liitteessä mainitut toiminnalliset vaatimukset (liite 1). Lisäksi sovelluksen täytyy toimia Windows Phone SDK 8.0:n mukana tulevassa emulaattorissa sekä Nokia Lumia 920 -älypuhelimessa, jotta sovelluksen käytännön toimivuus voitaisiin todentaa. Projektissa syntyvä sovellus on prototyyppi, jota on tarkoitus jatkokehittää projektin jälkeen kohti lopullista valmista sovellusta.

Suurin riski on aikataulusta lipeäminen, mikä tarkoittaisi sitä, että opinnäytetyön valmistuminen venyisi. Työmäärän arviointi on hankalaa, koska minulla ei ole aiempaa kokemusta Windows Phone 8 -ohjelmistokehityksestä. Tuotteen vaatimuksiin on valittu sellaiset ominaisuudet, jotka olisi realistisesti mahdollista toteuttaa niillä tiedoilla, mitä minulla on, ilman että niiden toteuttaminen venyisi.

5 Projektin toteutus ja kehitys

5.1 Määrittäminen

Projektin määrittäksenä toimivat tuotteen vaatimukset (liite 1). Tuotteen vaatimuksissa kuvailtiin, mitä ohjelmalla tulisi pystyä tekemään. Vaatimuksissa ei otettu ennalta kantaa, millä teknologioilla tai työkaluilla sovellus tulisi toteuttaa, joten sopivien työkalujen ja teknologioiden selvittäminen oli työn ensimmäinen osa ennen itse toteuttamista.

Tuotteen vaatimukset koostuivat yhdeksästä eri vaatimuksesta, jotka oli jaoteltu kahden eri prioriteettikategoriaan. Tärkeimmät vaatimukset kuuluivat prioriteettiin 11X, ja vähemmän tärkeät vaatimukset kuuluivat prioriteettiin 21X.

5.2 Teknologian selvitys

Ohjelmointikieleksi valittiin C#, muun muassa siksi, että siihen on helpointa löytää esimerkkejä. Ylivoimainen enemmistö Windows Phone Dev Centerin esimerkeistä on C#:lla toteutettuja. C# tarjoaa kaikki toiminnollisuudet, joita ohjelmassa tarvitaan. Lisäksi C# ei ole minulle kielenä täysin tuntematon ja syntaksi on hyvin lähellä Javaa, mikä helpottaa huomattavasti ohjelmointia. Kokonaan uuden ja tuntemattoman kielen, kuten esimerkiksi VB.NET, opettelu olisi hidastanut projektin toteutusta huomattavasti.

Windows Phone 8:ssa käyttöliittymä toteutetaan oletusarvoisesti XAMLilla ja Visual Studio tarjoaa hyvän graafisen työkalun käyttöliittymän toteuttamiseen XAMLilla, joten XAML oli hyvä ja helppo ratkaisu sen toteuttamiseen. Koko käyttöliittymän pystyy toteuttamaan hyvin pitkälti kokonaan XAMLilla. Code behind -näkyvässä voi tarvittaessa ohjelmoida käyttöliittymää myös C#:lla.

Tietokannaksi tarvittiin ratkaisu, joka soveltuisi hyvin mobiilikäyttöön ja olisi vikasetoinen. Kanta saattaa kasvaa suureksi kohtuullisen nopeasti ja kerralla lisättävää rivimäärä voi myös olla suuri, joten tekstipohjaiset tietokannat olisivat olleet selvästi hitaampia kuin SQLite:n tapainen relaatiokanta. SQLite soveltui käyttötarkoitukseen hyvin, sillä se on varsin helppokäyttöinen ja se on tuettu hyvin Windows Phone 8:ssa.

SQLite tukee transaktioita, minkä ansiosta se tarjoaa hyvää vikasietoisuutta muun muassa silloin, jos laitteesta katkeaa virta kesken transaktion. Toinen vaihtoehto olisi ollut käyttää Microsoftin Windows Phone local database -tietokantaa, mutta koska se olisi ollut minulle täysin tuntematon tietokanta, toisin kuin SQLite, se ei mielestäni ollut yhtä järkevä ratkaisu tähän sovellukseen. SQLiten käyttö mahdollistaa myös sen, että sovellus on helpompi siirtää muille alustoille. Windows Phone local database toimisi vain Windows Phone 8:ssa.

5.3 Työkalujen valinta

Windows Phone 8 -työkaluissa ei ole juuri valinnanvaraa. Windows Phone SDK 8.0 sisältää joukon työkaluja, joita sovelluskehittäjä voi käyttää oman ohjelmansa luomiseen. Tärkeimmät kaksi työkalua ovat Visual Studio Express 2012 for Windows Phone, jolla itse ohjelmointi tapahtuu, sekä Windows Phone 8 -emulaattori. Windows Phone SDK 8.0:n mukana tulee myös Blend for Visual Studio 2012, jonka avulla XAML-sovellusten toteuttaminen onnistuu helposti visuaalisesti. Syvällisempi toiminnollisuus ja C#-ohjelmointi pitää silti tehdä Visual Studiossa.

Windows Phone SDK 8.0 vaatii toimiakseen 64-bittisen Windows 8 Professional - tai Enterprise -käyttöjärjestelmän, koska emulaattori hyödyntää käyttöjärjestelmän Hyper-V-ominaisuutta. Jotta emulaattori toimisi, pitää Hyper-V-palikan olla asennettuna. Hyper-V vaatii toimiakseen myös prosessorilta SLAT (Second Level Address Translation) -tuen. Prosessorin SLAT-tuen lisäksi pitää tietokoneen virtualisointituen olla päällä. SLAT-tuki puuttuu vanhahkoista suorittimista. Projektissa käytetyssä kokoonpanossa oli moderni suoritin, eikä kehitysympäristön asennuksessa tai toiminnassa ilmennyt ongelmia.

5.4 Suunnittelu ja toteutus

Suunnittelu ja toteutus tapahtuivat käsi kädessä, sitä mukaa kuin toiminnollisuuksia toteutettiin. Käyttöliittymään tehtiin muutoksia, jos uusi toteutettu ominaisuus sitä vaati. Tuotteen vaatimuksissa ei otettu kantaa muuhun kuin toiminnallisiin seikkoihin, joten käytettävyyden ja käyttöliittymän suhteen ei ollut mitään tiettyä ennalta määrättyä tavoitetta.

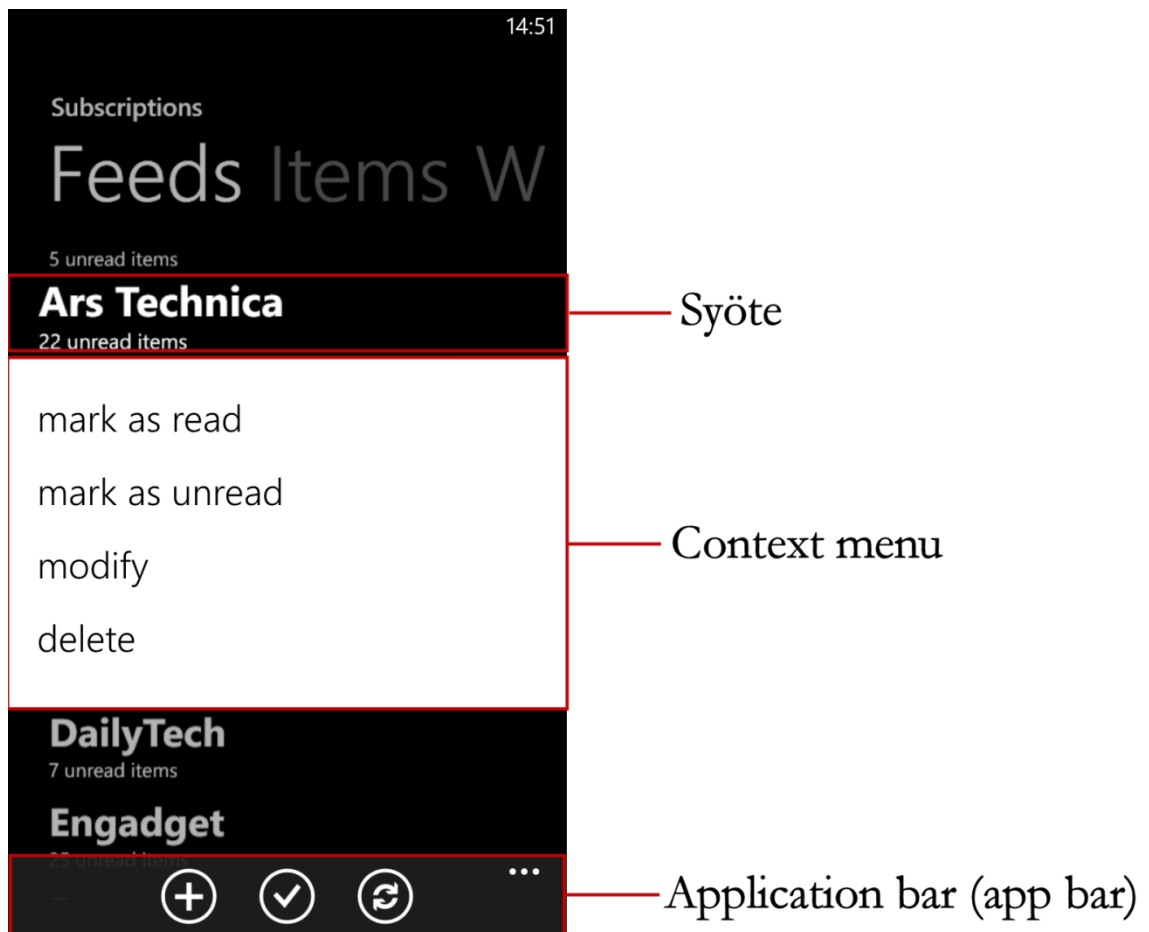
Aivan ensiksi keskityttiin toiminnollisuuksien lisäämiseen ja vasta sitten käyttöliittymän toiminnollisuuteen. Käytännössä kävi kuitenkin niin, että massiivisia muutoksia ei tarvinnut tehdä uusien ominaisuuksien lisäämisen yhteydessä.

Toteutettavan ohjelman on tarkoitus olla prototyyppi, jota olisi helppo jatkokehittää. Tästä johtuen ohjelman käyttökokemus ei ole paras mahdollinen. Käyttäjälle ei esimerkiksi anneta selkokielisiä virheilmoituksia, ja muun muassa päivityskomento ei kerro, milloin se on valmis. Käyttäjän antamia tietoja ei myöskään tarkisteta, vaan niiden oletetaan olevan oikeita. Käyttöliittymän pikkuvikoihin ei myöskään juuri kiinnitetä huomiota. Monet asiat tulevat parantumaan ja muuttumaan jatkokehityksessä.

5.4.1 Käyttöliittymän toteutus

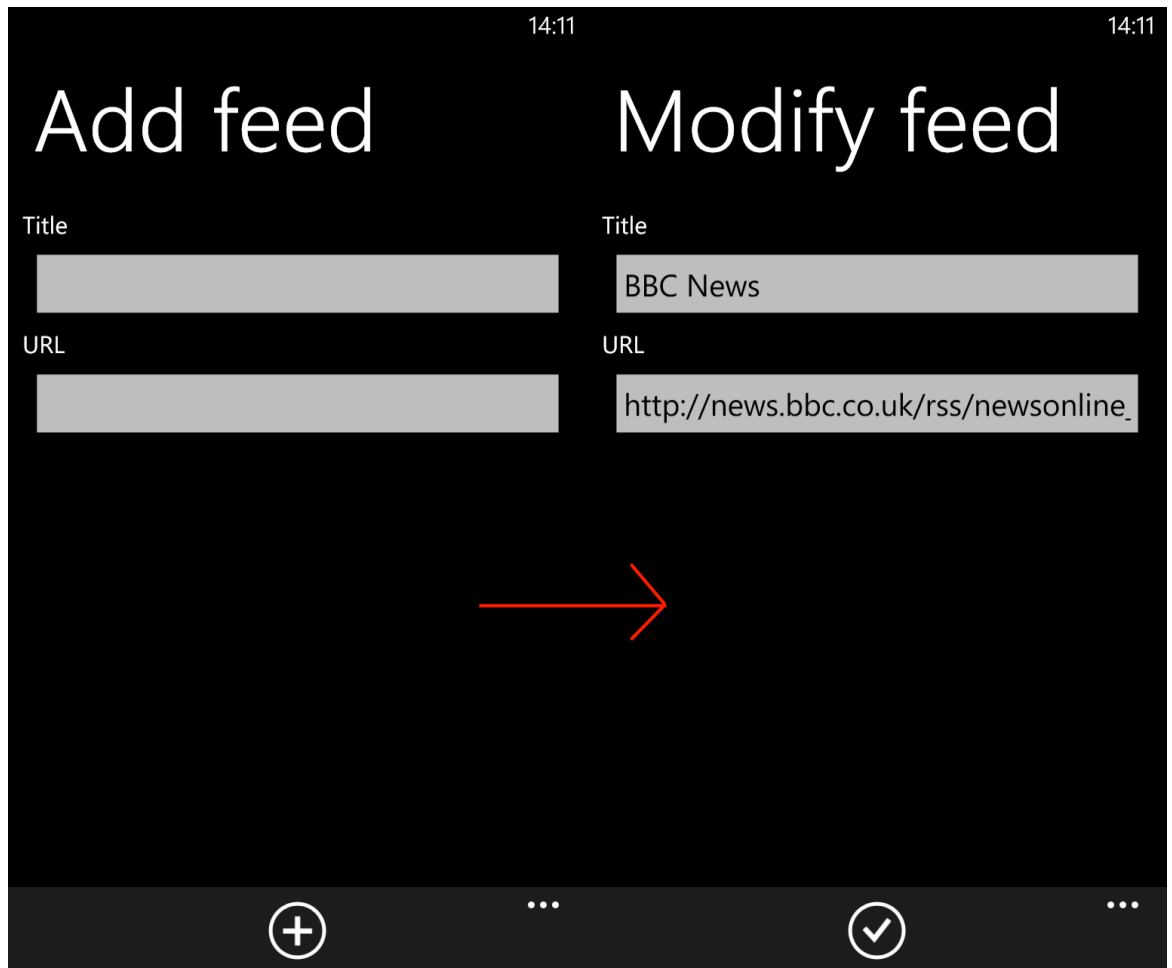
Käyttöliittymässä päädyttiin ratkaisuun, joka hyödyntää Windows Phone 8:n pivot-kontrollia. Tässä tapauksessa pivot sisältää kolme pivot-iteä, jotka vastaavat kolmea eri näkymää: Feeds, Items ja What's New. Pivot-iteet ovat käytännössä eri näkymiä vaakatasossa, ja niiden välillä pystyy siirtymään pyyhkäisemällä näyttöä oikealle tai vasemmalle. Sovellus on toiminnaltaan ja ulkonäöltään varsin yhtenevä käyttöjärjestelmän sisäisten sovellusten kanssa. Se, ettei esimerkiksi Items-näkymässä tarvitse painaa puhelimen takaisin-näppäintä haluttaessa päästä takaisin Feeds-näkymään, tekee käytettävyydestä sulavampaa. Kolmen päänäkymän välillä liikkuminen on tämän ansiosta nopeaa ja selkeää.

Feeds-näkymä sisältää listauksen syötteistä, joita käyttäjä seuraa. Jos uusia artikkeleita löytyy, näytetään tilaus korostettuna ja tilauksen nimen alla näkyy uusien lukemattomien artikkeleiden lukumäärä. Jos käyttäjä painaa syötettä pitkään, aukeaa context menu, josta käyttäjä voi merkitä syötteen artikkelit luetuiksi, lukemattomiksi, poistaa tilauksen tai muokata tilausta. Näytön alaosassa olevasta application barista käyttäjä voi lisätä tilauksia, merkitä kaikki kerralla luetuksi ja antaa päivityskomennon (kuva 1).



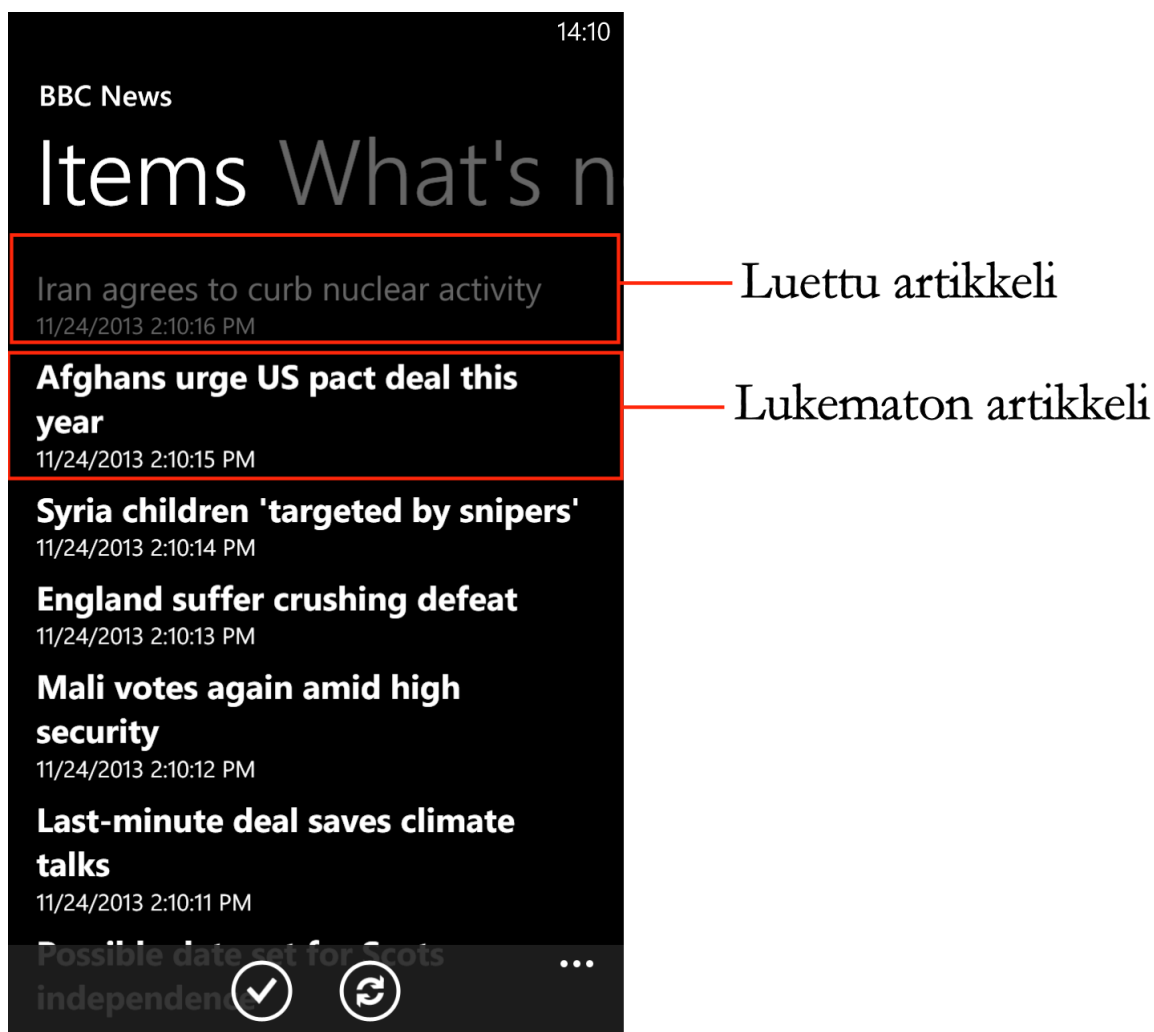
Kuva 1. Feeds-näkymä

Feeds-näkymän lisäys- ja muokkauspainike aukaisevat uuden näkymän, jossa käyttäjä voi antaa syötteen nimen sekä syötteen URL-osoitteen. Näytön alaosassa olevassa application barissa on lisää-nappula. Lisäys ja muokkaus käyttävät samaa näkymää, mutta muokkauksessa Title- ja URL-kentät ovat esitäytettyinä. Myös otsikon teksti sekä application barin kuvake ja kuvakkeen teksti vaihdetaan (kuva 2).



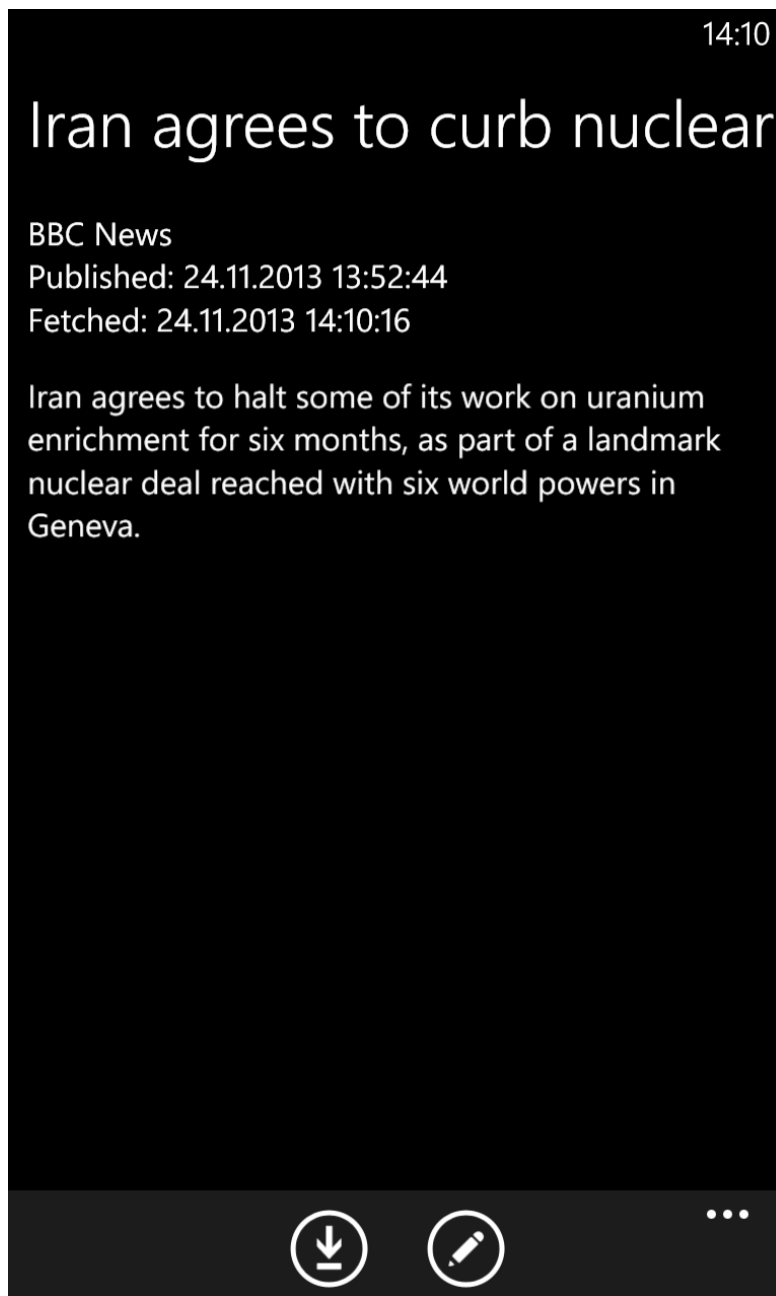
Kuva 2. Lisäys- ja muokkaus-näkymä

Items-näkymä näyttää oletuksena tuhat uusinta artikkelia. Jos käyttäjä painaa syötettä Feeds-näkymässä, siirrytään Items-näkymään ja näytetään käyttäjän painaman tilauksen syötteet. Lukemattomat artikkelit näytetään korostettuina, ja jo luetut artikkelit näytetään harmaina. Jos käyttäjä painaa artikkelia pitkään, aukeaa context menu, josta käyttäjä voi merkitä artikkelin luetuksi tai lukemattomaksi (kuva 3).



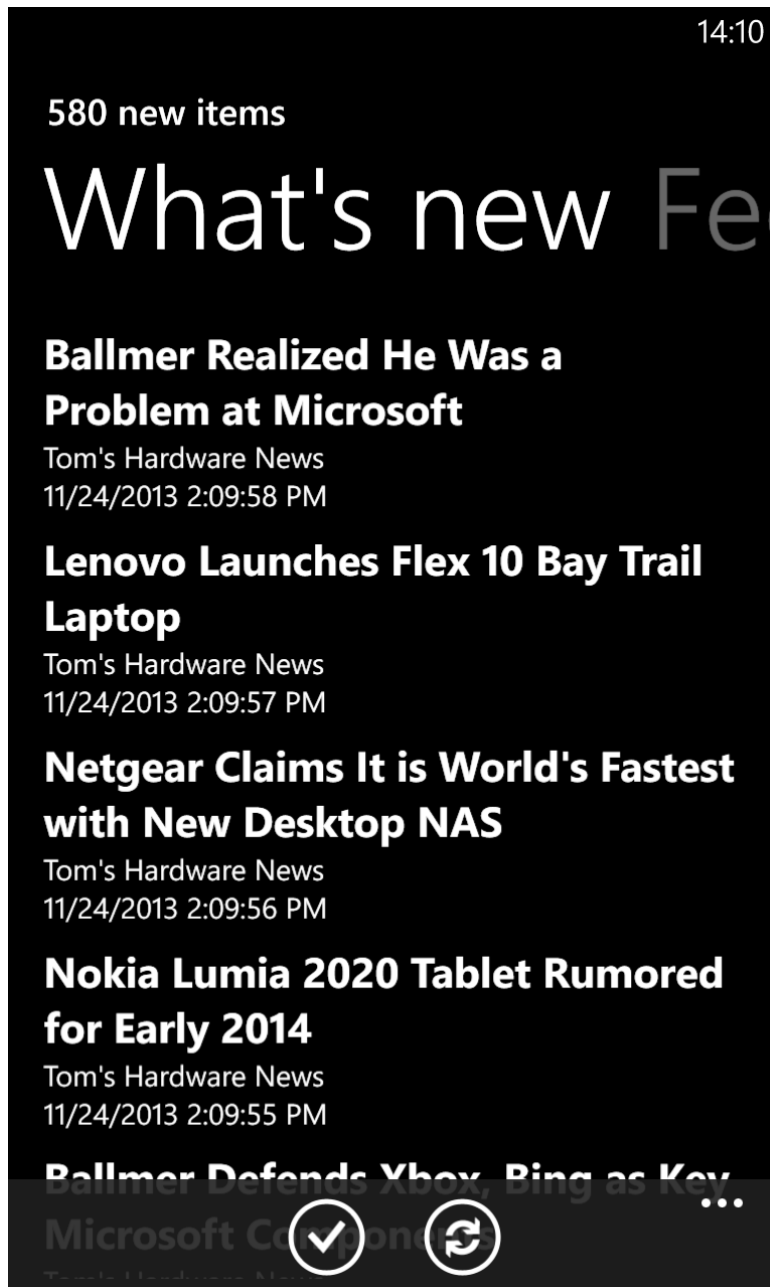
Kuva 3. Items-näkymä

Jos Items-näkymän artikkelia painetaan kerran, aukeaa uusi sivu, jossa näkyy otsikon lisäksi artikkelin tarkemmat tiedot, kuten esimerkiksi description-elementin sisältö. Artikkeleli merkkautuu automaattisesti luetuksi, kun sitä painetaan. Yksittäisen artikkelinäkymän application barista käyttäjä voi avata selaimen ja lukea koko artikkelin. Käyttäjä voi myös merkitä artikkelin lukemattomaksi ja samalla siirtyä takaisin aikaisempaan näkymään (kuva 4).



Kuva 4. Yksittäisen artikkelin näkymä

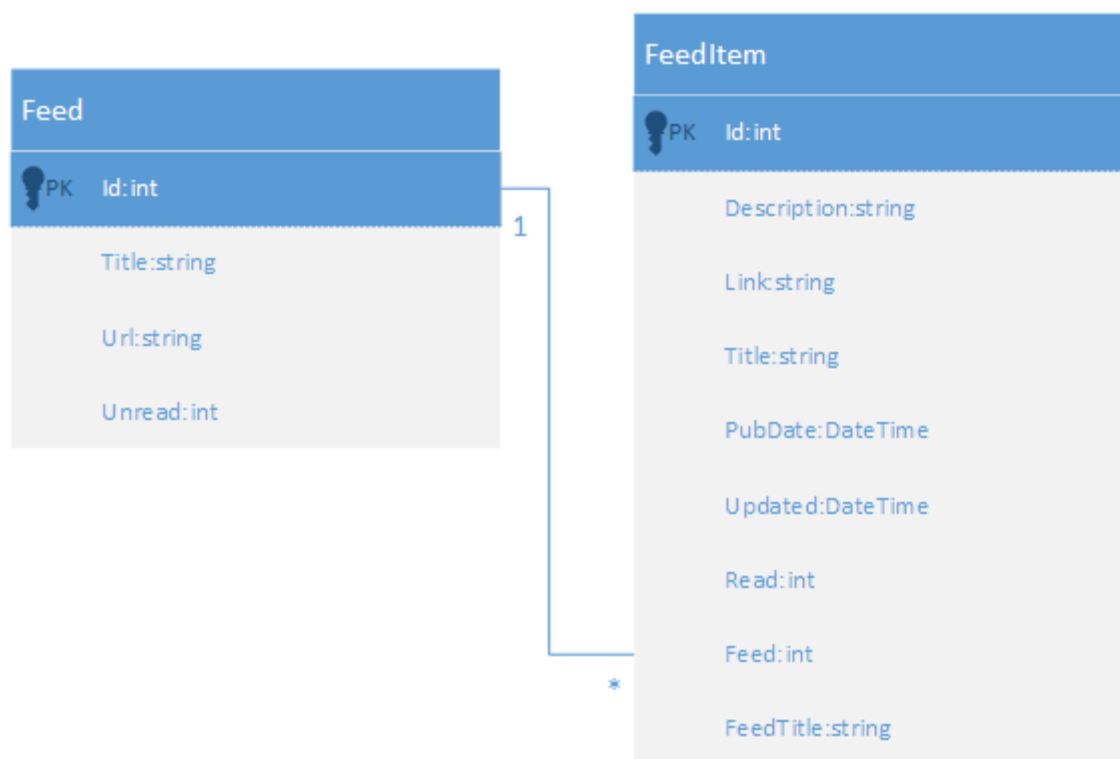
What's new -näköymä sisältää katsauksen kaikista uusista lukemattomista artikkeleista sekä uusien artikkeleiden lukumäärän. Artikkeleiden painaminen toimii vastaavalla tavalla kuin Items-näköymässä. Luetut artikkelit poistuvat tästä näköymästä. Näköymän alaosassa olevasta application barista käyttäjä voi merkitä kaikki luetuiksi ja antaa päivityskomennon uusien artikkelien tarkistamiseksi (kuva 5).



Kuva 5. What's new -näköymä

5.4.2 Tietokannan suunnittelu ja toteutus

Kävi varsin nopeasti selväksi, ettei sovellus tule vaatimaan monimutkaista tietokantaa. Tämän takia kahden taulun tietokanta oli järkevä ratkaisu. Projektissa käytössä oleva sqlite-net-kirjasto mahdollistaa LINQ:n käytön, mikä helpottaa huomattavasti muun muassa kannan luontia ja kantaan tallentamista. Tietokannan on helpompaa ajatella koostuvan kahdesta olioluokasta kuin tauluista Feed ja FeedItem (kuva 6). Kanta luodaan automaattisesti näiden kahden luokan perusteella. Kannan normalisoimattomuus on tietoinen ratkaisu tehokkuutta ajatellen, jotta kyselyiden määrää voitaisiin vähentää. Myös listausten käsittely helpottuu.



Kuva 6. Kaavio tietokannan havainnollistamiseksi

Alla on Feed-olioluokka, jonka perusteella yksi kannan taulu luodaan. Id:n yläpuolella olevat attribuutit kertovat, mikä muuttuja, eli tässä tapauksessa Id, on pääavain ja mikä on auto increment-muuttuja (esimerkki 2).

```
using System;

namespace FeedReader
{
    public class Feed
    {
        [SQLite.PrimaryKey, SQLite.AutoIncrement]
        public int Id { get; set; }

        public string Title { get; set; }
        public string Url { get; set; }
        public int Unread { get; set; }
    }
}
```

Esimerkki 2. Feed-luokka, josta kannan taulu luodaan

Feed sisältää syötteen tiedot ja FeedItem yksittäisen artikkelin tiedot. Molemmissa on pääavaimena int Id auto increment.

Feed -”taulu” sisältää seuraavat sarakkeet Id:n lisäksi: string Title, string Url ja int Unread. Title sisältää syötteen nimen, Url sisältää syötteen URL-osoitteen ja Unread sisältää syötteen lukemattomien artikkelien lukumäärän. Unread-kenttä päivitetään uusien artikkeleiden tarkistuksen yhteydessä.

FeedItem-”taulu” sisältää Id:n lisäksi seuraavat: string Description, string Link, string Title, DateTime PubDate, DateTime Updated, int Read, int Feed ja string FeedTitle. Description, Link, Title ja PubDate on nimetty RSS-määrittelyn samannimisten elementtien mukaan ja ne vastaavat niitä sisällöltään. PubDate sisältää julkaisuaikaleiman, jos se on ollut määritetty syötteessä. Updated on sen ajan aikaleima, jolloin syöte haettiin. Jos PubDate puuttuu syötteestä, sekä Updated että PubDate, tulevat olemaan samat. Read-arvo kuvaa artikkelin tilaa. Tässä vaiheessa 1 tarkoittaa luettua ja 0 lukemattomaa. Feed sisältää Feed-taulun Id:n, jotta tiedetään, mihin syötteeseen artikkeli kuuluu. FeedTitle sisältää syötteen otsikon, ja sitä päivitetään, jos käyttäjä nimeää syötteen uudestaan.

Koska toteutettava prototyyppi hyödyntää vain muutamaa yleisintä RSS-syötteen elementtiä, tietokanta on varsin yksinkertainen.

5.5 Laadunvarmistus ja testausta

Kaikki uudet ominaisuudet testattiin heti toteutuksen jälkeen. Kun syötteitä alettiin tallentaa tietokantaan, jotta artikkeleiden merkitseminen luetuksi tai lukemattomaksi toimisi, piti testausta tehdä pitkään, jotta voitiin varmistua ominaisuuden toimivan käytännössä. Ohjelma täyttää laadunvarmistuksen kriteerit silloin, kun käyttäjä pystyy tekemään ilman ongelmia kaiken, mitä tuotteen vaatimuksissa listataan (liite 1).

Käytännön pitkäaikainen testaus tapahtui Nokia Lumia 920 -älypuhelimella, ja perustoiminnallisuudet testattiin aina ensin Windows Phone 8 -emulaattorilla. Testausta helpotti se, ettei ohjelmassa ole tietoturvan kannalta kriittisiä osa-alueita. Käyttäjällä on vain kaksi kenttää, joihin hän voi kirjoittaa. Toteutettava ohjelma on kaiken lisäksi prototyyppi, eivätkä pienet visuaaliset ongelmat tai muut pikku puutteet tule olemaan ongelma. Esimerkiksi listan viimeisen kohteen jääminen hieman piiloon application barin alle ei ole sovelluksen toiminnan kannalta kriittistä. Tarkoituksenmukaista on nyt, että ominaisuudet toimivat käytännössä. Monet ominaisuudet tulevat muuttumaan jatkokehityksessä, joten siitäkin syystä täysin virheettömään sovellukseen pyrkiminen ei ole päätavoite tässä vaiheessa.

6 Toteutunut työ, lopputulokset ja johtopäätökset

6.1 Ongelmat ja niiden ratkaisut

Käyttöliittymän toteutuksen kannalta suurin ongelma oli, miten tietoa pystyy helpoiten siirtämään näkymästä toiseen ja miten yhdeltä sivulta pystytään päivittämään jotakin toista sivua. Kaikista yksinkertaisemmaksi ratkaisuksi osoittautui staattisten muuttujien käyttö. Esimerkiksi päänäköymän tietojen, kuten otsikon, muuttaminen onnistui helposti toisesta luokasta tekemällä pieni lisäys konstruktoriin (esimerkki 3).

```
public static MainPage mainPage;  
public MainPage()  
{  
    mainPage = this;  
}
```

Esimerkki 3. MainPage on luokka, joka vastaa päänäköymästä

Ohjelman loogisen toiminnan kannalta tärkein luokka on syötteiden päivityksestä vastaava luokka. Jotta uusia tietoja pystytään hakemaan, pitää lähettää http-kysely ja odottaa palvelimen vastausta. Windows Phone 8 ei tue C#:n HttpResponseMessage-luokan käyttöä synkronisesti, minkä takia toiminto piti toteuttaa asynkronisesti. Kävi myös ilmi, että Windows Phone 8 käyttää http-kyselyille välimuistia. Tämä teki päivityksen ongelmalliseksi, sillä vastaus tuli suoraan välimuistista eikä saman syötteen päivitys enää toiminut toivotulla tavalla ensimmäisen päivityksen jälkeen, jos käyttäjä ei tämän jälkeen ollut yrittänyt päivittää jotain toista syötettä. Satunnaisten attribuutin lisääminen URL-osoitteeseen olisi periaatteessa ratkaissut ongelman, mutta se ei toiminut käytännössä, sillä jotkin palvelimet antavat 404-virheen, jos osoitteeseen lisätään attribuutteja. Ongelma ratkaistiin muokkaamalla http-kyselyn otsikko satunnaisilla arvoilla, minkä ansiosta välimuisti lakkasi toimimasta (esimerkki 4).

```
HttpRequest request = (HttpRequest)HttpRequest.Create(url);  
request.Headers["If-Modified-Since"] = DateTime.Now.Ticks.ToString();
```

Esimerkki 4. If-Modified-Since-kenttään lisättiin satunnaisia arvoja

Vastaukseksi saatu XML-dokumentti oli helppo käsitellä C#:n XElement-luokan avulla, minkä ansiosta RSS-syötteen tiedot saatiin helposti käsiteltävään muotoon oliolis-

taan. Tätä listaa verrattiin sitten tietokannassa olevien tietojen kanssa, ja jos artikkeleita ei tietokannassa jo ollut, ne lisättiin kantaan. Artikkeleiden vertailu tapahtui käyttämällä hyväksi title (otsikko)-, description (kuvaus/lyhennelmä)- ja link (linkki)-elementtien arvoja. Oletuksena vertailtiin otsikkoa ja linkkiä, mutta jos esimerkiksi syötteessä on vain kuvaus elementti-artikkelille, tällöin vertaamiseen käytettiin vain sitä. Julkaisupäivämäärä ei ole pakollinen elementti RSS-määritysten mukaan, joten sitä ei käytetä vertailussa. RSS-määritysten mukaan syötteen item-elementtissä pitää olla joko otsikko, kuvaus tai linkki-elementti. Kannasta ei tarvitse hakea kaikkia aikaisempia syötteitä.

Context menu -valikon tekeminen onnistui helposti suoraan XAML-puolella Windows Phone Toolit -palikan avulla. Jostain syystä context menu -valikko vaati arvon tyhjentämisen aina sulkeutumisen jälkeen, tai muuten context menu -valikko saattoi viitata väärään listan kohteeseen. Context menu -valikon arvon pystyi tyhjentämään lisäämällä valikolle Unloaded-tapahtumakäsittelijän, jossa arvo sitten tyhjennettiin, kun valikko poistui näkyvistä.

6.2 Valittujen työkalujen ja menetelmien arviointi

SQLite osoittautui toimivaksi ratkaisuksi, eikä sen käyttöönotossa ilmennyt ongelmia. Tietokannan sai toimimaan sqlite-net-palikkaa ja sqlite-net-wp8 C++/CX -käärettä hyväksi käyttäen. SQLite toimi varsin nopeasti, mutta sovelluksen käynnistysnopeus alkoi hidastua kannan kasvaessa suureksi. Kannan käyttöä ei ole optimoitu, joten hidastuminen ei ole suuri yllätys. Voi olla mahdollista, että tehokkaampiakin kantaratkaisuja olisi olemassa. Jatkokehityksessä voisi olla järkevää tutkia myös, miten muut tietokantaratkaisut toimisivat. Muun muassa Lex.DB voisi olla järkevämpi tietokantaratkaisu kannan yksinkertaisuuden takia.

XAML toimi hyvin käyttöliittymän toteuttamisessa ja nopeutti sen toteuttamista. Siitä on varsin helppo saada selvää jopa ilman visuaalista työkalua, ja suurin osa käyttöliittymästä pystyttiin toteuttamaan suoraan visuaalisella editorilla. Tiettyjen muutosten tekeminen oli yksinkertaisinta tehdä suoraan XAML-koodia muokkaamalla. Projektissa käytössä oleva Windows Phone Toolkit -palikka helpotti käyttöliittymän luontia. Muun muassa context menun luominen onnistui sen avulla vaivattomasti XAML-koodilla.

Lisäksi erilaisten visuaalisten tehosteiden, kuten esimerkiksi siirtymien, tekeminen helppotui eikä niitä tarvinnut ohjelmoida C# puolella.

C#:lla logiikan ohjelmointi onnistui ilman suuria pulmia, ja tarpeen mukaan käyttöliittymän osien muokkaaminen C#:lla onnistui kohtuullisen pienellä vaivalla. Suurin yllätys oli se, miten hyvät työkalut C#:ssa on XML-dokumentin läpikäymiseksi. Http-kyselyn ja -vastauksen tekeminen olisi voinut olla yksinkertaisempaa ja toiminta selkeämpää.

Kehitysympäristönä Visual Studio oli hyvin käyttäjäystävällinen ja vakaa. Windows Phone 8 -emulaattori oli myös oivallinen, ja sen avulla sovelluksen debuggaaminen onnistui mainiosti. Debuggaaminen sujui yhtä hyvin myös suoraan puhelimessa.

6.3 Projektin eteneminen ja työn lopputulos

Projektin luonne oli sellainen, että suunnittelu ja toteutus oli järkevintä tehdä samaan aikaan. Tämä toimi sinänsä varsin hyvin, mutta lopputuloksena syntyi hieman sekavaa koodia. Varsinaisen suunnittelun puute johti siihen, että jotkin sovelluksen osat olivat varsin optimoimattomia ja tehottomia uuden toiminnon liittämisen jälkeen. Tämän takia muun muassa uusien syötteiden hakemiseen ja päivittämiseen tehty luokka on varsin tehoton. Varsinkin kannan käyttöön liittyvää toistoa esiintyy paljon.

Koska tietämykseni Windows Phone 8 -sovelluskehityksestä oli olematon, ei tulos myöskään vastaa kunnolla mitään arkkitehtuurimallia. Sovellus muistuttaa hieman Model View ViewModel -arkkitehtuurimallin sovellusta, eikä sen muuttaminen MVVM-mallin mukaiseksi sovellukseksi todennäköisesti tulisi olemaan kovin haastavaa.

Ajallisesti suurin osuus kului tutkimiseen, miten mikäkin ominaisuus voitaisiin toteuttaa. Käyttöliittymä oli työmäärällisesti suurin, koska siihen liittyvien asioiden opetteluun ja tiedon etsimiseen kului eniten aikaa. Itse ohjelman logiikka oli varsin suoraviivaisesti toteutettavissa.

Työn toteutuksessa ei ilmennyt suurempia ongelmia, minkä ansiosta toimintoja saatiin toteutettua aikataulusta edellä. Syntynyt sovellus vastasi täysin sitä, minkälaiseksi tuote

oli suunniteltu. Projektin aikana tuli ilmi useita eri asioita, jotka kannattaa ottaa huomioon kehitettäessä sovellusta edelleen. Työstä saatu tieto ja jatkokehitysehdotukset ovat tärkeitä, eikä niitä välttämättä tullut ajatelleeksi suunnitteluvaiheessa ennen varsinaista toteutusta.

6.4 Oppiminen

Tämä projekti oli oppimisen ja itsensä kehittämisen kannalta todella tärkeä. Ennen tässä projektissa toteutettua sovellusta en ole tehnyt mitään sovelluksia älypuhelimille. Myös työssä käytetyt työkalut ja teknologiat olivat minulle varsin tuntemattomia. Kokemukseni Visual Studiosta, C#:sta ja SQLitestä olivat perin pintapuolisia ennen tätä projektia. Työn tekemisen kannalta helpottava tekijä oli se, etteivät edellä mainitut teknologiat, työkalut ja kielet olleet kuitenkaan täysin tuntemattomia.

Projektissa pääsin soveltamaan asioita, joita olen oppinut muualla. Oli mielenkiintoista muun muassa huomata, mitä keskinäisiä eroja on C#:lla ja Javalla. Kehitysympäristöistä minulla on aikaisempaa kokemusta enimmäkseen Eclipsestä, joten nyt oli hyvä päästä tutustumaan perusteellisesti Visual Studion toimintaan ja käyttöön.

Sovelluskehitys älypuhelimille on varmasti jatkossa yhä merkityksellisempää, joten tässä opitut taidot ovat epäilemättä minulle hyödyllisiä tulevaisuudessa.

7 Yhteenveto

Projekti oli mielenkiintoinen ja hyödyllinen oman osaamisen kehittämisen kannalta. Työn tekeminen vaati sitä, että joutui itse ottamaan asioista selvää. Oli mukava huomata, etteivät tavoitteet olleet liian korkealla ja että ne täyttyivät. Työssä valmistunut lopputulos on sellainen, mitä projektin alussa asetettiin tavoitteeksi. Kun ottaa huomioon lähtökohdan, voidaan lopputulokseen mielestäni olla varsin tyytyväisiä.

Työssä syntyi toimiva sovellus. Toteutunutta prototyyppiä on helppo rakentaa kohti valmista sovellusta. Työn aikana Windows Phone 8 -kehitystyökalut tulivat paremmin tutuiksi ja tietämys Windows Phone 8 kehityksestä kasvoi merkittävästi. Sovelluskehitys Windows Phone 8 -alustalle osoittautui yksinkertaisemmaksi ja suoraviivaisemmaksi kuin, mitä oli alun perin ajatellut. Microsoftin esimerkit ja ohjeet olivat hyviä ja selkeitä, minkä ansioista kehitysympäristön asentaminen ja työssä alkuun pääseminen sujui varsin helposti.

Työ oli minulle hyödyllinen, ja siitä opittu tietotaito tulee olemaan tärkeää tulevaisuudessa. Samalla tuli myös opittua, miten joitain asioita ei tulisi toteuttaa. Tehdyistä virheistä oppii, eikä niitä toivon mukaan toista tulevaisuudessa. Tästä työstä opituilla tiedoilla on paljon helppo alkaa toteuttaa jotain toista sovellusta. Kaiken kaikkiaan projekti on ollut oppimisen kannalta hyvin antoisa kokemus.

7.1 Jatkokehitysehdotukset

Projektin oli tarkoitus olla prototyyppi eikä valmis tuote, joten lista jatkokehitykseen tulevista ominaisuuksista on pitkä.

Sovellusta ei ole juuri optimoitu, ja monet luokat pystyttäisiin toteuttamaan paremmin ja tehokkaammin. Sovelluksessa on myös jonkin verran toistoa koodissa, ja esimerkiksi kannan käsittely olisi hyvä keskittää omaan luokkaansa. Jossain vaiheessa kannasta pitää poistaa vanhoja artikkeleita, jotta siitä ei tulisi liian suuri ja hidas. Kannan käyttö pitäisi myös muuttaa asynkroniseksi, jotta käytettävyyks olisi sulavampaa ilman nytkähtelyitä.

Uusia ominaisuuksia lisättiin sitä mukaa, kuin edelliset ominaisuudet valmistuivat. Tästä johtuen esimerkiksi RSS-artikkeleiden päivitysten tarkistamiseen tehty luokka ei toimi kovin hyvin, kun uusia artikkeleita etsitään useammalle kuin yhdelle syötteelle samaan aikaan. Luokka oli suunniteltu toteuttamaan vain yksi päivitys kerrallaan eikä monta samanaikaisesti. Päivitys pitäisi saada toimimaan taustalla ja tarkistamaan päivitykset käyttäjän määrittämän väliajan mukaan. Jos syötteitä tulee paljon lisää pienellä aikavälillä ja jos palvelin lähettää vain esimerkiksi 15 artikkelia syötteessä, kasvaa riski, että artikkeleita jää saamatta, jos syötteitä ei haeta palvelimelta tasaisin väliajoin.

Kannan kasvun takia myös selattavat artikkelilistat paisuvat. Tämän takia jonkinlainen artikkelien sivuttaminen tai inkrementaalinen lataus olisi järkevää. Listan selaamiseen ja listan kohdan säilyttämiseen voidaan tehdä myös muita parannuksia, jotka nopeuttaisivat käyttöä.

Tällä hetkellä syötteiden lisääminen tapahtuu niin, että käyttäjä joutuu itse kirjoittamaan RSS-syötteen URL-osoitteen ja otsikon. Käytettävyyden kannalta tämä ei ole optimaalinen ratkaisu, ja käyttäjä voi tehdä helposti virheen URL-osoitteen kopioimisessa tai kirjoittamisessa. Järkevämpi ratkaisu olisi esimerkiksi se, että sovellus yrittää hakea oikean sivun hakutermin perusteella ja etsiä RSS-syötteet siltä sivulta.

Tällä hetkellä käyttäjä ei pysty muuttamaan mitään asetuksia. Se ei vielä tässä vaiheessa ole ongelma, mutta kun ominaisuuksia tulee lisää, pitäisi myös käyttäjälle antaa mahdollisuus vaikuttaa asetuksiin. Käyttäjä voisi esimerkiksi valita haluamansa päivitysaikavälin.

Sovelluksessa on toteutettuna hyvin perustasoinen RSS-tuki, eikä se ota huomioon kuin yleisimmät elementit. Käyttäjän kannalta olisi erittäin hyvä, jos myös Atom-syötteitä tuettaisiin.

Lähteet

Android. Android Open Source Project. Luettavissa: <http://source.android.com/>. Luettu: 4.12.2013

Android Developers. android.database.sqlite. Luettavissa: <http://developer.android.com/reference/android/database/sqlite/package-summary.html>. Luettu: 3.12.2013

Dublin Core Metadata Initiative 2012. Dublin Core Metadata Element Set, Version 1.1. Luettavissa: <http://dublincore.org/documents/dces/>. Luettu: 3.12.2013

Bourret, R. 2005. XML and Databases. Luettavissa: <http://www.rpbourret.com/xml/XMLAndDatabases.htm>. Luettu: 4.12.2013

Microsoft Developer Network. XAML Overview (WPF). Luettavissa: <http://msdn.microsoft.com/en-us/library/ms752059%28v=vs.110%29.aspx>. Luettu: 3.12.2013

Microsoft Download Center 2012. Windows Phone SDK 8.0 Luettavissa: <http://www.microsoft.com/en-us/download/details.aspx?id=35471>. Luettu: 4.12.2013

Microsoft TechNet. XQuery Language Reference (SQL Server) Luettavissa: <http://technet.microsoft.com/en-us/library/ms189075.aspx>. Luettu: 3.12.2013

Nielsen, J. 1993. Usability Engineering. Academic Press Limited. London.

Nottingham, M. & Sayre, R. 2005. The Atom Syndication Format. Luettavissa: <http://tools.ietf.org/html/rfc4287>. Luettu: 3.12.2013

RSS Advisory Board 2009. RSS 2.0 Specification. Luettavissa: <http://www.rssboard.org/rss-specification>. Luettu: 3.12.2013

SQLite a. About SQLite. Luettavissa: <http://www.sqlite.org/about.html>. Luettu: 3.12.2013

SQLite b. Database Speed Comparison. Luettavissa: <http://www.sqlite.org/speed.html>. Luettu: 3.12.2013

SQLite c. SQLite Download Page. Luettavissa: <http://www.sqlite.org/download.html>. Luettu: 3.12.2013

SQLite d. SQLite Is Self-Contained. Luettavissa: <http://www.sqlite.org/selfcontained.html>. Luettu: 3.12.2013

Wikipedia a. Windows Mobile. Luettavissa: http://en.wikipedia.org/wiki/Windows_Mobile. Luettu: 4.12.2013

Wikipedia b. Windows Phone. Luettavissa: http://en.wikipedia.org/wiki/Windows_Phone. Luettu: 4.12.2013

Wikipedia c. Windows Phone 7. Luettavissa: http://en.wikipedia.org/wiki/Windows_Phone_7. Luettu: 4.12.2013

Wikipedia d. Windows Phone 8. Luettavissa: http://en.wikipedia.org/wiki/Windows_Phone_8. Luettu: 4.12.2013

Williams, A. 2005. Performance of relational databases versus native XML databases. Yliopiston opinnäytetyö. University of Otago. Dunedin. Luettavissa: <http://hdl.handle.net/10523/1200>. Luettu: 3.12.2013

Windows Phone Dev Center 2013a. Getting started with developing for Windows Phone. Luettavissa: <http://msdn.microsoft.com/library/windowsphone/develop/ff402529%28v=vs.105%29.aspx>. Luettu: 4.12.2013

Windows Phone Dev Center 2013b. Implementing the Model-View-ViewModel pattern in a Windows Phone app. Luettavissa: <http://msdn.microsoft.com/en-us/library/windowsphone/develop/gg521153%28v=vs.105%29.aspx>. Luettu: 3.12.2013

Windows Phone Dev Center 2013c. How to register your phone for development. Luettavissa: <http://msdn.microsoft.com/library/windowsphone/develop/ff769508%28v=vs.105%29.aspx>. Luettu: 3.12.2013

Windows Phone Dev Center 2013d. Join the program. Luettavissa: <http://msdn.microsoft.com/library/windowsphone/help/jj206717%28v=vs.105%29.aspx>. Luettu: 4.12.2013

Windows Phone Dev Center 2013e. Local database for Windows Phone. Luettavissa: <http://msdn.microsoft.com/en-us/library/windowsphone/develop/hh202860%28v=vs.105%29.aspx>. Luettu: 3.12.2013

Windows Phone Dev Center 2013f. Native code on Windows Phone 8. Luettavissa: <http://msdn.microsoft.com/en-us/library/windowsphone/develop/jj681687%28v=vs.105%29.aspx>. Luettu: 4.12.2013

W3C 1996. Extensible Markup Language (XML). Luettavissa: <http://www.w3.org/TR/WD-xml-961114.html>. Luettu: 3.12.2013

W3C 2013a. Extensible Markup Language (XML) 1.0 (Fifth Edition). Luettavissa: <http://www.w3.org/TR/2008/REC-xml-20081126/>. Luettu: 3.12.2013

W3C 2013b. XQuery 3.0: An XML Query Language. Luettavissa: <http://www.w3.org/TR/2013/PR-xquery-30-20131022/>. Luettu: 3.12.2013

Liitteet

Liite 1. Tuotteen vaatimukset

	Toimijana	minä ...	jotta ...	Prioriteetti	koko	Tilanne	kommentteja / avoimia asioita
1	Ohjelman käyttäjä	haluan lisätä RSS-kanavan ja nähdä artikkelit	voin jatkossa seurata RSS-kanavaa	110			
2	Ohjelman käyttäjä	haluan nähdä listauksen RSS-kanavista	voin nähdä kaikki lisäämäni RSS-kanavat	111			
3	Ohjelman käyttäjä	haluan valita yksittäisen RSS-kanavan	voin lukea pelkästään tiettyä RSS-kanavaa	112			
4	Ohjelman käyttäjä	haluan merkitä haluamani kanavan artikkelin luetuiksi/lukemattomiksi	tiedän, mitkä olen jo lukenut	113			
5	Ohjelman käyttäjä	haluan merkitä haluamani kanavan artikkelit luetuiksi/lukemattomaksi	tiedän, mitkä olen jo lukenut	114			
6	Ohjelman käyttäjä	haluan merkitä kaikki kohteet luetuiksi/lukemattomiksi	tiedän, mitkä olen jo lukenut	211			
7	Ohjelman käyttäjä	haluan muokata RSS-kanavan tietoja tai poistaa kanavan	voin nimetä sen uudelleen tai korjata URLin	212			
8	Ohjelman käyttäjä	haluan katsauksen ja listauksen kaikista uusista lukemattomista RSS-artikkeleista	näen, mitä ei ole luettu	213			
9	Ohjelman käyttäjä	haluaa lukea syötteitä offline-tilassa	voin lukea syötteitä silloin, kun ei ole internetyhteyttä	214			