Tuomas Vuorenmaa

Windows Phone 7 -sovelluksen yhdistäminen WCF-palveluun

Opinnäytetyö

Kevät 2012

Tekniikan yksikkö

Tietotekniikan koulutusohjelma

Seinäjoen ammattikorkeakoulu
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Koulutusyksikkö: Tekniikan yksikkö

Koulutusohjelma: Tietotekniikka

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Tekijä: Tuomas Vuorenmaa

Työn nimi: Windows Phone 7 -sovelluksen yhdistäminen WCF-palveluun

Ohjaaja: Petteri Mäkelä

Vuosi: 2012 Sivumäärä: 58 Liitteiden lukumäärä: 0

Tässä opinnäytetyössä esitellään Windows Phone 7 -mobiilikäyttöjärjestelmää, sekä siihen läheisesti liittyviä tekniikoita. Opinnäytetyön toisena pääaihealueena on Windows Communication Foundation, eli WCF, joka on Microsoftin sovelluskehys palvelusovelluksia varten. Palvelusovellusten tarkoitus on toimia taustajärjestelmänä muille sovelluksille, esimerkiksi mobiilisovelluksille.

Työn aihe on ajankohtainen, koska nykyiset mobiilisovellukset eivät ole enää suljettuja, puhelimen sisällä toimivia sovelluksia. Useat sovellukset, esimerkkinä sosiaalisen median sovellukset käyttävätkin tietolähteenään erilaisia taustajärjestelmiä Internet-yhteyden avulla. Ajankohtaisuutta lisää se, että Windows Phone 7 on yksi uusimmista mobiilikäyttöjärjestelmistä.

Opinnäytetyön tarkoituksena on tarkastella miten Windows Phone 7 -sovellus voidaan yhdistää tietolähteenä toimivaan WCF-palveluun. Liikkeelle lähdetään Windows Phone 7 -sovelluksen ja WCF-palvelusovelluksen luomisesta. Sen jälkeen tarkastellaan, miten WCF-palvelu otetaan käyttöön Windows Phone 7 -sovelluksessa ja mitä toimenpiteitä se vaatii.

Avainsanat: Windows Phone 7, Windows Communication Foundation, mobiilisovellus, palvelusovellus, taustajärjestelmä

SEINÄJOKIUNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Faculty: School of Technology

Degree programme: Information Technology

Specialisation: Software Engineering

Author: Tuomas Vuorenmaa

Title of thesis: Connecting Windows Phone 7 application to WCF service

Supervisor: Petteri Mäkelä

Year: 2012 Number of pages: 58 Number of appendices: 0

The aim of this thesis was to introduce the Windows Phone 7 mobile operating system and technologies related to it as Windows Phone 7 is one of the latest mobile operating systems. A secondary topic was the Windows Communication Foundation (WCF) which is Microsoft's framework for service applications. Service applications' purpose is to work as a background system for other applications.

The subject of this thesis is topical because today's mobile applications are no longer just applications inside the device. Many of the latest mobile applications use background systems as a source of their data, for example applications related to the social media.

The purpose of this thesis was to find out how to connect a Windows Phone 7 application to a WCF Service which works as a data source. The process started with the creation of Windows Phone 7 application and WCF Service application. The next phase was to take into consideration what procedures are needed to deploy the service application in the mobile application.

Keywords: Windows Phone 7, Windows Communication Foundation, mobile application, service application, data source

SISÄLTÖ

KUVIO- JA TAULUKKOLUETTELO

KÄYTETYT TERMIT JA LYHENTEET

WP7	Lyhenne Windows Phone 7 -käyttöjärjestelmästä.
WCF	Lyhenne Microsoftin Windows Communication Foundation -palvelusovellusalustasta.
Protocol Buffers	Googlen kehittämä sarjallistamismenetelmä.
.**NET**	Microsoftin kehittämä virtuaalinen ajoympäristö Windows-käyttöjärjestelmän päälle.
Silverlight	Microsoftin sovelluskehys rikkaille käyttöliittymäsovelluksille.
XNA	Microsoftin kehittämä sovelluskehys pelikehitystä varten.
C#	Microsoftin luoma ohjelmointikieli .NET-sovelluskehystä varten.
XAML	Extensible Application Markup Language. Kuvauskieli, jota käytetään etenkin käyttöliittymien kuvaukseen.
IIS	Internet Information Services. Microsoftin kehittämä palvelinohjelmistokokonaisuus.
WAS	Windows Process Activation Service. Prosessien aktivointi mekanismi, joka toimii IIS:n päällä.
SOA	Service-Oriented Architecture on palvelusovelluksille suunnattu sovellusarkkitehtuuri.
WSDL	Web Service Definition Language. Kuvauskieli, jota käytetään Web Servicen kuvaukseen.

(Järvinen 2012, Löwy 2010, Protocol Buffers: Developer Guide. 2011.)

1 JOHDANTO

1.1 Työn tausta

Kuljetusliike A. Myllymäki otti yhteyttä Seinäjoen ammattikorkeakouluun erään toisen projektin tiimoilta. Projektin edetessä, kävi ilmi, että heillä olisi tarvetta toiselle sovellukselle. Siitä tämä projekti sai alkunsa.

Kuljetusliike A. Myllymäellä on laaja asiakasrekisteri, joka sisältää nimien ja osoitteiden lisäksi muun muassa puhelinnumeroita ja ajo-ohjeita. Asiakasrekisteri sijaitsee toimiston tietokoneella, johon ei päästä ulkoapäin käsiksi. Ongelmaksi on muodostunut se, että kuljettajat, jotka tarvitsevat ajo-ohjeita ongelmatilanteissa, eivät näe asiakasrekisteriä mistään. Ongelmatilanteissa kuljettajat ovat tähän asti joutuneet soittamaan toimistolle ja pyytämään sieltä asiakkaiden ajo-ohjeita sekä puhelinnumeroita.

1.2 Työn tavoite

Työn tavoitteena on rakentaa kuljetusliike A. Myllymäki Oy:lle järjestelmä, jonka avulla heidän kuljettajansa pystyisivät lukemaan yrityksen asiakasrekisteriä myös autosta käsin. Asiakkaan kanssa sovittiin, että kuljettajien käyttöön toteutetaan mobiilisovellus ja toimiston tietokoneelle hallintasovellus. Heti aluksi päätettiin yhteisesti, että asiakasrekisteri sijoitettaisiin ulkoistetun palveluntarjoajan palvelimelle, jotta vältyttäisiin palvelimen ylläpidolta. Samassa yhteydessä päätettiin, että palvelimelle tehtäisiin rajapinta, jonka kautta hallinta- ja mobiilisovellus keskustelevat tietokannan kanssa. Päätavoitteeksi asetettiin se, että kuljettajat pääsevät lukemaan asiakasrekisteriä Internet-yhteyttä käyttäen. Myöhemmin asetettiin lisätavoitteeksi mahdollisuus soittaa asiakkaille suoraan mobiilisovelluksesta.

Projektin toteuttamiseen annettiin vapaat kädet. Kyseisen sovelluksen voisi toteuttaa lukemattomilla tavoilla, monille eri alustoille ja käyttäen useita eri tekniikoita. Kaikki käytetyt tekniikat voitiin siis valita itse. Mobiilialustaksi valittiin Windows

Phone 7 ja palvelimelle tuleva palvelu päätettiin toteuttaa Windows Communication Foundationia käyttäen.

Windows Phone 7:n valinta mobiilialustaksi johtui siitä, että kyseessä on tuore mobiilialusta, joka ei ollut tekijälle entuudestaan tuttu. Kiinnostusta lisäsi tieto siitä, että Nokia valitsi kyseisen alustan omiin älypuhelimiinsa. Windows Communication Foundation valittiin palvelusovellusalustaksi, koska se kuuluu .NET-sovelluskehykseen, kuten Windows Phone 7. Tämä mahdollisti sen, että molemmat sovellukset voitiin tehdä samalla työkalulla ja samalla ohjelmointikielellä.

1.3 Työn rakenne

Työn toinen luku käsittelee teoriaa Windows Phone 7:sta niiltä osin, kun se tätä työtä koskettaa. Luku alkaa käyttöjärjestelmän esittelyllä, jonka jälkeen käsitellään sovelluskehitystyön aloitusta. Lopuksi kerrotaan tarkemmin Windows Phone 7:n tärkeimmästä tekniikasta, eli Silverlightista.

Kolmannessa luvussa käydään läpi yleistä teoriaa Windows Communication Foundationista.

Luku neljä kertoo käytännön toteutuksesta, jossa yhdistetään Windows Phone 7 -sovellus WCF-palveluun. Luvussa käsitellään ensin palvelusovellus, sen jälkeen mobiilisovellus ja lopuksi kerrotaan miten ne yhdistetään. Luvun lopussa kerrotaan vielä luokkakirjastoista, joita työssä hyödynnettiin.

Viimeisessä luvussa esitellään johtopäätökset ja jatkokehitysideoita.

2 WINDOWS PHONE 7

2.1 Windows Phone -käyttöjärjestelmä

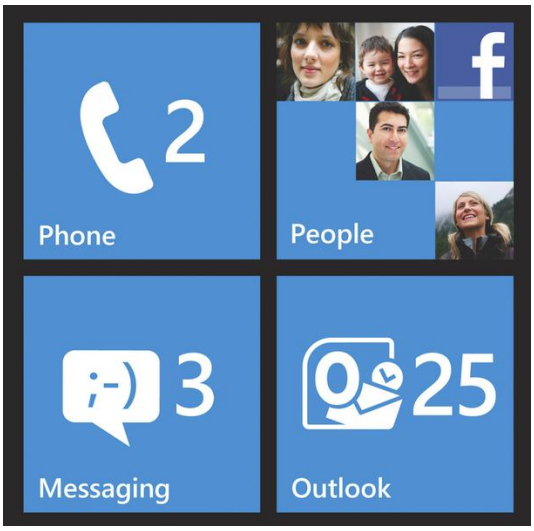
Windows Phone 7 on Microsoftin kehittämä mobiilikäyttöjärjestelmä, joka julkaistiin vuoden 2010 helmikuussa Mobile World Congress -tapahtumassa. Windows Phone 7 on seuraaja Windows Mobile -käyttöjärjestelmälle, jota ei kehitetä enää eteenpäin. Uutta käyttöjärjestelmää luotaessa aloitettiin lähes puhtaalta pöydältä, eikä Windows Phone 7:ssa hyödynnetty vanhasta Windows Mobilesta kuin käyttöjärjestelmän ytimen osia. Käyttöjärjestelmän ytimenä toimii edelleen Windows CE, joka on alun perin suunniteltu kämmentietokoita varten. Näkyvimmän muutoksen koki käyttöliittymä, ja Windows XP:tä muistuttanut käyttöliittymä sai seuraajakseen ainoastaan kosketusnäytöille suunnitellun Metro-käyttöliittymän. (Randolph & Fairbairn 2010, 1-2; Järvinen 2012, 6-13.)

Kuvio 1. Windows Phone 7:n aloitusnäkymä.

Ohjelmistojen kannalta suurin muutos on se, että vanhat Windows Mobilelle kehitetyt sovellukset eivät enää toimi Windows Phone 7:ssa. Windows Mobilelle kehitetyt sovellukset saadaan toimimaan ainoastaan uudelleenkirjoittamalla ne C#- tai Visual Basic -kielillä. Ohjelmointiin käytetään Silverlight- ja XNA-tekniikoita, jotka molemmat kuuluvat Microsoftin .NET-sovelluskehykseen. (Järvinen 2012, 12.)

2.1.1 Windows Phone 7:n ominaisuudet

Windows Phone 7:n käyttöliittymä on suunniteltu eri näkökulmasta, kuin muut älypuhelinkäyttöjärjestelmät. Muut käyttöjärjestelmät ovat sovelluskeskeisiä, mutta Windows Phonessa filosofia lähtee käyttäjästä ja hänelle tärkeistä asioista. Erilaisuus ilmenee varsinkin aloitusnäkymästä, jossa on keskitytty enemmän tuomaan käyttäjälle tietoa ja korostamaan sisältöä, kuin tuomaan silmäniloa. Aloitusnäkymä koostuu tiilistä (live tile), eli tapahtumaruuduista, jotka toimivat pikakuvakkeina sovelluksiin sekä sovellukset voivat päivittää tietojaan niihin. Esimerkiksi sähköpostisovelluksen tiilessä voi logon lisäksi olla lukemattomien sähköpostien lukumäärä. Tiilet ovat kaksipuolisia, joka mahdollistaa useamman tiedon näyttämisen samassa tiilessä. Tämä koskee niin Microsoftin tekemiä sovelluksia, kuin itse tehtyjä sovelluksia. Aloitusnäkymän voi muokata haluamansa näköiseksi lisäämällä tiiliä, siirtelemällä niitä ja poistamalla turhia tiiliä. (Järvinen 2012, 50-54; Ferracchiati & Garofalo 2011, 3.)



Kuvio 2. Live Tile.

Microsoft sanoo, että Windows Phone 7 on käyttäjän näkökulmasta erinomainen puhelin, koska siinä on kaikki ominaisuudet, jotka Applen iPhone sekä Googlen Android-laitteet ovat tehneet tutuiksi. Näihin ominaisuuksiin kuuluu muun muassa näytön monikosketusominaisuus, tyylikäs ja selkeä käyttöliittymä, tuki tunnetuimmille sähköpostitileille sekä sosiaalisen median palvelut. Windows Phone 7:n voi myös yhdistää XBox Live -palveluun, tämä ominaisuus on suunnattu nimenomaan konsolipelaajille. (Lee & Chuvyrov 2011, 4.)

Käyttäjän kannalta tärkeimmät toimintokokonaisuudet on kerätty yhteen ja näitä kokonaisuuksia kutsutaan keskuksiksi (hubs). Windows Phone 7:ssa on yhteensä kuusi keskusta: Ihmiset, kuvat, pelit, musiikki + videot, Marketplace ja Office. Esimerkiksi Ihmiset-keskuksen ansiosta viestintä ja yhteydenpito on yksi puhelimen parhaimmista ominaisuuksista. Syynä tähän on se, että keskukseen on sidottu kaikki henkilöihin liittyvät palvelut, kuten puhelinnumero, sähköpostiosoite ja Facebook-profiili yhden nimen alle. Ihmiset-keskuksen avulla yhteydenoton voi tehdä vaikkapa Facebookin chatilla, jatkaa sitä tekstiviestillä ja hyvästellä Messengerissä. Kaikkia viestintävälineitä käytettäessä ei tarvitse vaihtaa sovellusta. Yrityskäyttöä ajatellen on Office-keskus, jonka avulla saadaan Officen työpöytäversiolla luodut dokumentit suoraan puhelimeen. (Järvinen 2012, 54; Ferracchiati & Garofalo 2011, 3.)

Kuvio 3. Ihmiset-keskus.

2.1.2 Mangon tuomat uudistukset

Windows Phone 7.1, koodinimeltään Mango, on päivitys, joka sisältää sovelluskehittäjien kannalta merkittäviä uusia ominaisuuksia (Järvinen 2012, 55).

Yksi merkittävin parannus päivityksen myötä on moniajo-tuki muidenkin kuin Microsoftin kehittämille sovelluksille. Ennen tätä päivitystä sovellus oli suljettava aina kun siitä poistuttiin, jolloin sovelluksesta syntyi ”hautakivi” (tombstone). Hautakivi

sisälsi sovelluksen palauttamiseen tarvittavia tilatietoja. Tilatietojen tallennus täytyi ottaa huomioon jo sovelluksen kehitysvaiheessa. Uusi moniajo ei edelleenkään vastaa PC-järjestelmän moniajoa, jossa ohjelmia voidaan ajaa useita yhtä aikaa omissa säikeissään (threads). Mangon myötä ennen sovelluksen menoa hautakivitilaan on lisäksi tullut horrostila (dormant). Horrostilassa käyttöjärjestelmä pitää täysin huolta sovelluksen tilasta, mutta myös sovellus itse voi vaikuttaa tilaansa. Horrostilasta sovellus palaa erittäin nopeasti suoritustilaan (running), eikä käyttäjä tiedä, että ohjelma on ollut horroksessa. Tälle toimintamallille on englanninkielinen termi: Fast Application Switching. (Järvinen 2012, 55-57; Additions in the Windows Phone SDK 7.1. 2012.)

Ennen Windows Phonen 7.1 -versiota kaikki ulospäin suunnattu tietoliikenne toimi pelkästään http-protokollaa käyttäen. Mango-päivityksen myötä käyttöön saatiin matalamman tason tiedonvälitys protokollat TCP ja UDP. Tämä päivitys mahdollistaa esimerkiksi kommunikoinnin molempiin suuntiin pilvipalveluiden kanssa, pikaviestinnän verkon yli sekä moninpelaamisen. (Järvinen 2012, 57; Additions in the Windows Phone SDK 7.1. 2012.)

Kaikki Windows Phone 7 -sovellukset oli pakko ohjelmoida C#-kielellä, mutta Mangon mukana lisättiin tuettuihin kieliin Visual Basic. Vaikka Visual Basic tuli mukaan, niin C# pysyy todennäköisesti silti suosituimpana Windows Phone 7 -ohjelmointikielenä. Ohjelmointitekniikoina käytettävät Silverlight ja XNA on aiemmin pidetty erillään, mutta nyt niitä voi sekoitella keskenään. XNA on pelikehitykseen tarkoitettu tekniikka. Silverlightista kerrotaan tarkemmin myöhemmin. (Järvinen 2012, 60-61; Additions in the Windows Phone SDK 7.1. 2012.)

Päivityksen mukana tulleita muita uudistuksia:

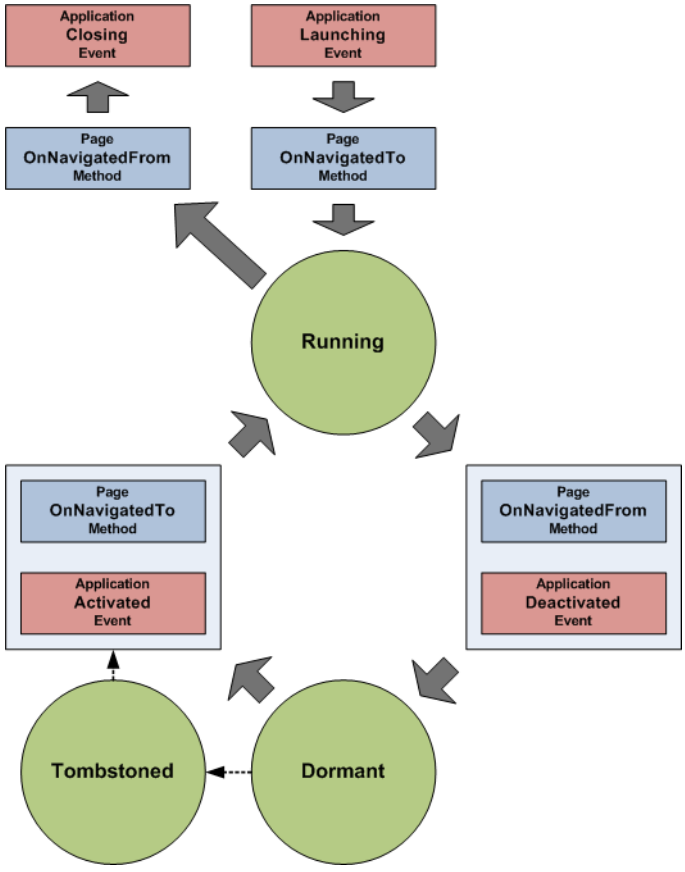
- tausta-agentit
- tiedostojen lataus taustalla
- Silverlight 4
- etusivun kaksipuoliset tiilet
- push-päivitysten parannus
- paikallinen tietokanta
- kameran käyttö ohjelmallisesti
- mainokset

– sensoreita. (Additions in the Windows Phone SDK 7.1. 2012.)

2.1.3 Sovelluksen elinkaari

Sovelluksen elinkaari tarkoittaa sovelluksen suoritusprosessia aina käynnistyksestä sulkemiseen asti. Sovelluksella on kolme tilaa: suoritustila, horrostila ja hautakivitila. Kuten edellisessä kappaleessa mainittiin, horrostila ei ollut alusta asti, vaan tuli lisäyksenä Mango -päivityksen myötä. (Execution Model Overview for Windows Phone. 2012.)

Sovelluksen ollessa etualalla, se on suoritustilassa. Kun käyttäjä vaihtaa sovellusta, siirtyy vanha sovellus horrostilaan. Jos käyttöjärjestelmä tarvitsee itselleen tai toisille sovelluksille lisää muistia, asettaa se vanhimman sovelluksen hautakivitilaan, jolloin sovellus tuhoutuu, ainoastaan tilatiedot ja yksittäiset sivut jäävät muistiin. (Execution Model Overview for Windows Phone. 2012.)



Kuvio 4. Sovelluksen elinkaari. (Execution Model Overview for Windows Phone. 2012.)

Kuvion 4 punaiset laatikot kuvaavat sovellustason tapahtumia. Sovellustason tapahtumat vaihtelevat sovelluksen tiloja. Sovellustason tapahtumia on neljä, Launching, Deactivated, Activated, ja Closing. Nämä tapahtumat kattavat sovelluksen käynnistyksen, sulkemisen, taustalle menon sekä paluun taustalta. (Execution Model Overview for Windows Phone. 2012.)

Kuvion 4 siniset laatikot kuvaavat yksittäisen sivun tapahtumia ja nämä vastaavasti vaihtelevat näkyvissä olevan sivun tiloja. Tapahtumia on kaksi: OnNavigatedTo ja OnNavigatedFrom. Nämä tapahtumat kattavat navigoinnin sivulle ja sivulta pois. (Execution Model Overview for Windows Phone. 2012.)

2.1.4 Laitevaatimukset

Parhaan mahdollisen käyttäjäkokemuksen takaamiseksi Microsoft on määritellyt tarkat laitevaatimukset, jotka jokaisen Windows Phone 7 -yhteensopivan laitteen on täytettävä. Minimivaatimukset ovat tiukat, mutta eivät mahdottomat nykytekniikalle. Laitevaatimukset pätevät niin Windows Phone 7.0:n, kuin Mango-päivityksen kohdalla. Nämä laitevaatimukset helpottavat sovellusten kehitystä ja testausta, sillä kehittäjä tietää mitkä ominaisuudet löytyvät jokaisesta laitteesta. (Järvinen 2012, 50-51; Hardware Specifications for Windows Phone. 2012.)

Yksi kehittäjän työtä helpottava ominaisuus on se, että näyttöresoluutioita on tällä hetkellä vain yksi, eikä käyttöliittymiä ole pakko suunnitella skaalautuvaksi usealle resoluutiolle. Kannattaa pitää kuitenkin mielessä, että tulevaisuudessa resoluutioita tulee olemaan muitakin. Suunnitteilla on jo laitteita resoluutiolle 320x480. (Ferracchiati & Garofalo 2011, 2.)

Windows Phone 7 laitteistovaatimukset ovat:

- kolme fyysistä painiketta (aloita, etsi, takaisin)
- WVGA (800 x 480) näyttö
- kapasitiivinen kosketusnäyttö, joka tukee neljää samanaikaista kosketusta
- datayhteys mobiiliverkon tai WLAN:n kautta
- 256 mt keskusmuistia ja 8gt flash-muistia

- A-GPS
- kiihtyvyysanturi. (Hardware Specifications for Windows Phone. 2012.)

Vaihtoehtoisia laitteistoja ovat:

- kompassi
- gyroskooppi
- ensisijainen kamera
- toissijainen kamera puhelimen etupuolella. (Hardware Specifications for Windows Phone. 2012.)

2.2 Sovelluskehityksen aloitus

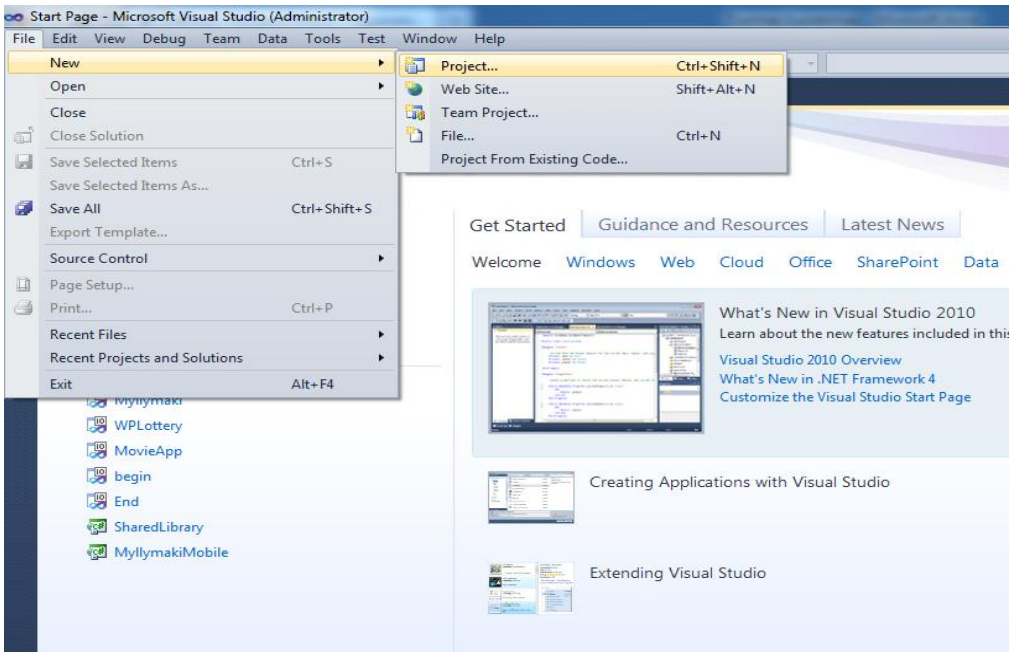
Windows Phone 7 -kehitysvälineinä toimivat Microsoftin Visual Studio 2010 sekä Expression Blend. Visual Studiolla kirjoitetaan ohjelmakoodi ja sitä voidaan käyttää yksinkertaisten käyttöliittymien suunnitteluun. Expression Blendin tarkoitus on toimia graafisen suunnittelun työkaluna ja siksi se tarjoaakin paremmat ominaisuudet käyttöliittymien suunnitteluun kuin Visual Studio. Molemmista ohjelmistoista on saatavilla ilmaiset versiot, joilla pääsee alkuun. (Järvinen 2012, 68-69; Lee & Chuvyrov 2011, 7.)

Visual Studio 2010 ja Expression Blendin lisäksi täytyy ladata puhelintyökalut (Windows Phone SDK), jotka ovat saatavissa ilmaiseksi osoitteesta create.msdn.com. Puhelintyökalut tuovat puhelinsovellusten kehitykseen välttämättömät ominaisuudet Visual Studioon sekä Blendiin. Lisäksi mukana asentuu puhelinemulaattori, joka on Windows Phone 7 -puhelinta matkiva tietokonesovellus. Puhelinemulaattori mahdollistaa sovelluskehityksen ilman oikean laitteen hankintaa. Työkaluista kannattaa aina ottaa käyttöön uusimmat, mutta on hyvä huomioida että myös vanhat versiot ovat yhä ladattavissa. Asennusjärjestyksenä suositellaan seuraavaa: ensin Visual Studio 2010, sen jälkeen Expression Blend ja vasta lopuksi puhelintyökalut. (Järvinen 2012, 74-77.)

2.2.1 Projektin luominen

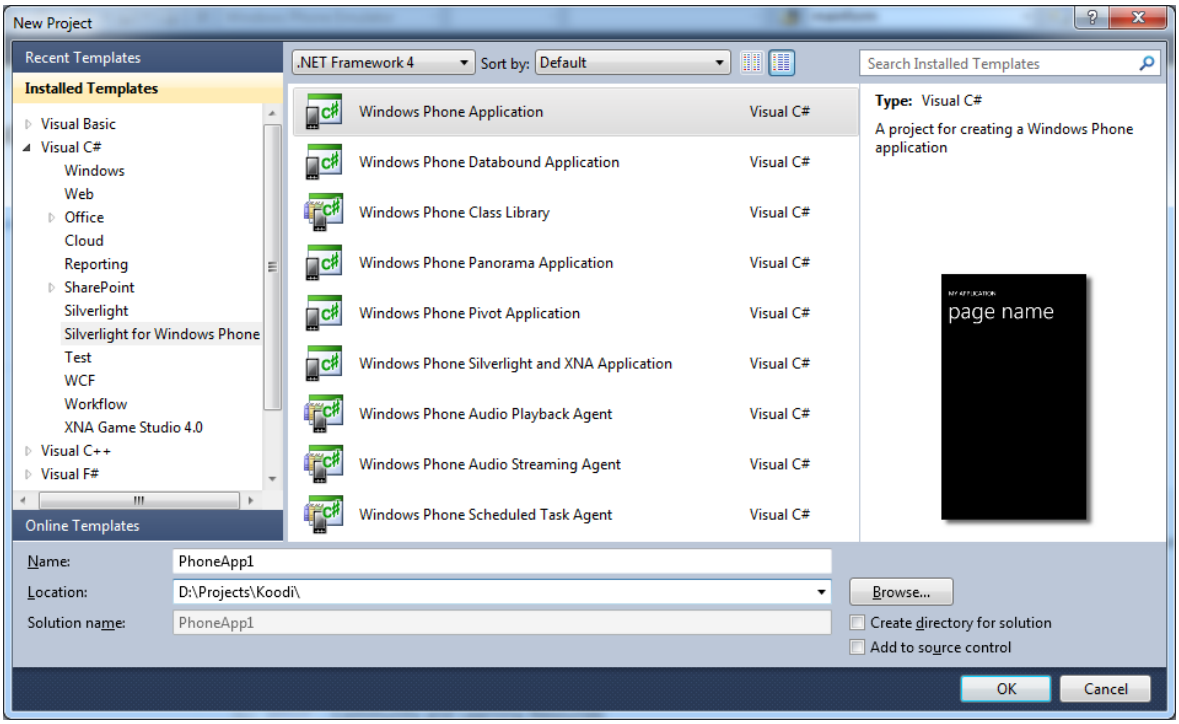
Projektin voi luoda joko Visual Studio 2010:llä tai Expression Blend 4:lla. Yleensä sovelluskehittäjät luovat projektin Visual Studiolla ja niin tehdään tässäkin tapauksessa.

Visual Studion käynnistyttyä avataan File -> New -> Project… (Kuvio 5).



Kuvio 5. Uuden projektin luominen.

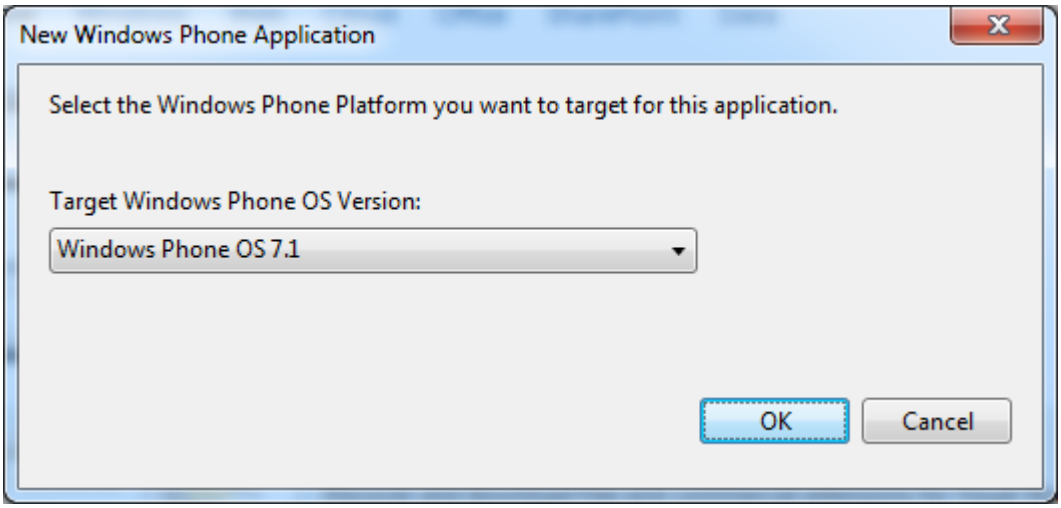
New Project -valikosta löytyy C#- ja Visual Basic -kielien alta Windows Phonelle tarkoitetut sovellusmallit. Windows Phone -sovelluksille on kahden tyyppisiä sovellusmalleja, Silverlight for Windows Phone, joka on tarkoitettu yleissovelluksille sekä XNA Game Studio 4.0, joka on pelejä varten. Lisäksi Visual Studiolla pystyy tallentamaan omia sovellusmalleja, jolloin itse suunniteltua sovelluksen runkoa ei tarvitse kirjoittaa aina alusta asti. (Järvinen 2012, 106; kuvio 6.)



Kuvio 6. New Project -valikko.

Silverlight for Windows Phonen sovellusmalleja on kolmen tyyppisiä. Ensimmäinen tyyppi on tavallisille sovelluksille tarkoitetut mallit. Niiden erona on se, että ne luovat oletuksena pohjalle erilaisilla käyttöliittymäkomponenteilla varustetun sovelluksen. Ne toimivat täysin samalla tavalla ja jokaiseen voi lisätä samat käyttöliittymäkomponentit. Toinen tyyppi on luokkakirjasto. Luokkakirjasto sisältää uudelleenkäytettäviä toimenpiteitä, joita tarvitaan usein ja monessa eri projektissa. Viimeinen tyyppi on erilaisia taustalla toimivia sovelluksia varten. Teknisesti ne eivät ole sovelluksia vaan luokkakirjastoja, jotka toteuttavat tiettyjä käyttöjärjestelmän rajapintoja. (Järvinen 2012, 105-106.)

Kun sopiva projektimalli on valittu, annetaan sovellukselle nimi sekä polku ja klikataan OK-painiketta. Kehittäjältä kysytään vielä mille Windows Phonen versiolle sovellus suunnataan. Käytännössä vaihtoehdot ovat 7.0 tai 7.1. Kun valinta on tehty, voi sovelluksen kehitys alkaa.



Kuvio 7. Kohdealustan valitseminen.

2.2.2 Metro-suuntaviivat

Metro-käyttöliittymä on suunniteltu huolellisesti. Suunnitteluprosessin edetessä Microsoft on julkaissut tarkkoja dokumentaatioita siitä millaisia Metro-sovellusten tulisi olla. Kuitenkin sovelluksia, jotka eivät noudata Metro-ulkoasua saa laittaa kauppapaikkaan. Sovelluksien käyttäjien kannalta Metro-ulkoasua on hyvä noudattaa, sillä sovellukset ovat silloin yhtenäisiä ja täten helppoja omaksua. Usealle alustalle tehtäviä sovelluksia suunniteltaessa onkin hyvä pitää aina mielessä alustan omat ohjeet käyttöliittymille. Uusin käyttöliittymäohjeistus on saatavilla Microsoftin MSDN-kehittäjäpalvelusta. (Järvinen 2012, 54-55; Randolph & Fairbairn 2010, 5-7.)

Metro-sovelluksen kuuluisi olla pääpiirteittäin:

- moderni
- ”On The Go”
- siisti
- liikettä näytöllä
- yksinkertainen ja luettava
- yhtenäinen
- innovatiivinen. (General Design Principles. 2012.)

2.2.3 CodePlex

CodePlex on Microsoftin avoimen lähdekoodin yhteisö, jonne kehittäjät voivat perustaa projekteja. Muut kehittäjät voivat seurata itseään kiinnostavien projektien etenemistä sekä ladata lähdekoodit ja luokkakirjastot omaan käyttöönsä. CodePlex mainitaan tässä yhteydessä siksi, että sitä kautta saadaan ladattua laajennuksia myös Windows Phone 7 -kehitykseen. (CodePlex: Open Source Project Community. 2012.)

Yksi tärkeimmistä Windows Phone 7 -laajennuksista on Silverlight for Windows Phone Toolkit, jonka uusin versio on julkaistu vuoden 2011 marraskuussa. Kyseinen laajennus sisältää uusia käyttöliittymäkomponentteja sekä parannuksia vanhoihin komponentteihin. Parannuksen liittyivät muun muassa suorituskykyyn. (CodePlex: Open Source Project Community. 2012.)

Toinen mainittavan arvoinen laajennus on MVVM Light Toolkit. MVVM tulee sanoista Model-View-View-Model. MVVM on sovelluksen suunnittelumalli, joka perustuu Martin Fowlerin esitysmalliin. Sen tarkoitus on erotella tieto, logiikka ja näkymä toisistaan. Näiden kolmen asian toisistaan erottaminen auttaa luomaan siistimpiä, helpommin ylläpidettäviä sekä paremmin laajennettavissa olevia sovelluksia. MVVM Light Toolkit tarjoaa valmiit apuvälineet suunnittelumallin toteuttamiseksi. (MVVM Light Toolkit [viitattu 5.3.2012]; Ferracchiati & Garofalo 2011, 418.)

2.3 Silverlight

Silverlight on visuaalisesti rikkaiden käyttöliittymien luomiseen tarkoitettu tekniikka, jota käytetään webissä, työpöydällä ja puhelimella. Silverlight yhdistää vektorigrafiikan, animaatiot, sekä kuvan, videon ja äänen. Silverlight perustuu Windows Presentation Foundation -tekniikkaan, joka tuli .NET 3.0:n mukana Windows-työpöytäsovellusten käyttöön. WPF:stä tehtiin kevennetty, alustariippumaton versio, joka toimi aluksi nimellä WPF Everywhere ja nykyisin siis nimellä Silverlight. Nykyisin Silverlightia käytetään varsinkin web-sovelluksissa. Silverlight sopii puhelinkäyttöjärjestelmän käyttöliittymätekniikaksi sen alustariippumattomuuden takia.

Valintaa puoltaa myös se, että Silverlightin ajonaikainen ympäristö vie tilaa ainoastaan muutaman megatavun. (Järvinen 2012, 96.)

2.3.1 Silverlight For Windows Phone

Mobiiliympäristössä käytetään Silverlight For Windows Phone -tekniikkaa, joka pohjautuu Silverlightin versioon 4. Mobiiliversio ei tue kaikkia täyden Silverlightin ominaisuuksia, mutta siihen on myös lisätty ominaisuuksia, joita muut versiot eivät tue. Osa täyden Silverlightin ominaisuuksista on sellaisia, joiden tukemisessa ei olisi muutenkaan järkeä, kuten hiiren klikkaukset. Mobiiliympäristössä Silverlightia käytetään lähes kaikessa käyttöliittymäohjelmoinnissa, jos jätetään pelit pois laskuista. Kehittäjien kannalta juuri Silverlightin käyttämä XAML-kuvauskieli tuottaa eniten haasteita. XAML suosii imperatiivisen ohjelmoinnin sijaan deklaratiivista, johon yleensä ohjelmoijat eivät ole tottuneet. (Järvinen 2012, 96; Features Differences Between Silverlight and Silverlight for Windows Phone. 2012.)

2.3.2 XAML

XAML tulee sanoista Extensible Application Markup Language. XAML on käyttöliittymäsuunnitteluun tarkoitettu kuvauskieli. Kieli muistuttaa HTML-kieltä, mutta on tehokkaampi. Käsin kirjoitettuna XAML-kieli vaatii tarkkuutta, sillä se on XML-pohjainen kieli. Kieli on XML-pohjainen, jotta sitä olisi mahdollisimman tehokasta käsitellä koneellisesti. Visual Studio ja Expression Blend hyödyntävät tätä ominaisuutta. (Järvinen 2012, 97; XAML 2010.)

XAML-kielen syntaksi vastaa XML-kielen syntaksia. Jokainen käyttöliittymäkomponentti kuvataan elementeillä ja niiden attribuuteilla. Nämä vastaavat ohjelmakoodissa luotuja olioita ja niiden ominaisuuksia. Sisäkkäisistä käyttöliittymäelementeistä syntyy puurakenne, jonka mukaan Silverlight piirtää ne järjestyksessä näytölle. Piirtojärjestys kannattaa pitää mielessä grafiikan suorituskyvyn vuoksi, koska jos jokin elementti jää toisen alle, ei sitä kannata piirtää. (Järvinen 2012, 97.)

XAML-koodin kirjoittamiseen Visual Studiolla ja Expression Blendillä on kaksi tapaa. Koodin voi kirjoittaa käsin, joka on hyvä tapa varsinkin silloin, kun tietyt elementit täytyy saada tietyn elementin sisään. Toinen tapa on raahata komponentit työkalupaletista suoraan käyttöliittymään, jolloin Visual Studio tai Expression Blend luo koodin käyttäjän puolesta. (Järvinen 2012, 98-99.)

3 WINDOWS COMMUNICATIONS FOUNDATION

3.1 Mikä on WCF?

Nykyään sovelluksilla on tapana toimia erilaisia palveluita hyödyntäen. Tätä ajatusmallia tukemaan Microsoft on kehittänyt Windows Communications Foundationin eli WCF:n. WCF on laajennus .NET-sovelluskehikkoon, joka tarjoaa yhtenäisen alustan palvelusovelluksille. WCF yksinkertaistaa palvelusovelluksen rakennetta, koska se yhdistää useita eri tekniikoita yhden tekniikan alle (kuvio 8). (Lövy 2010, 1-2; Introducing Windows Communication Foundation in .NET Framework 4 2010.)

	ASMX	.NET Remoting	Enterprise Services	WSE	System. Messaging	System. Net	WCF
Interoperable Web Services	X						X
Binary .NET –.NET Communication		X					X
Distributed Transactions, etc.			X				X
Support for WS-* Specifications				X			X
Queued Messaging					X		X
RESTful Communication						X	X

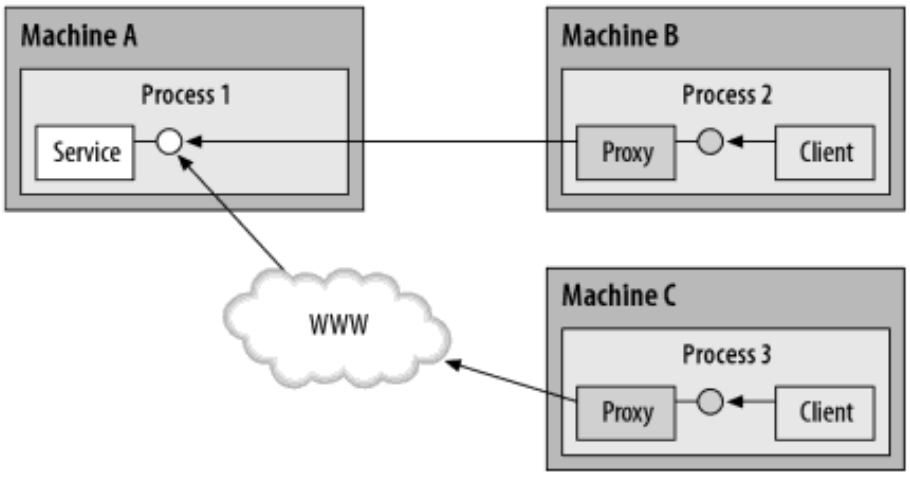
Kuvio 8. WCF:n yhdistämät tekniikat. (Introducing Windows Communication Foundation in .NET Framework 4. 2010.)

Itse WCF-palvelusovellukset pyörivät aina Windows-ympäristössä, mutta asiakassovellus voi toimia myös Linux- tai Mac-ympäristössä. WCF, joka tunnettiin aiemmin nimellä Indigo, julkaistiin .NET-sovelluskehikon 3. version yhteydessä. Uusimman .NET-sovelluskehikon mukana WCF-sovelluksiin tuli tuki Windowsin Azure-pilvipalvelualustaan. (Löwy 2010, 2.)

3.2 WCF-palvelu

Palvelu tässä yhteydessä tarkoittaa sovellusta, jolla on joukko toimintoja, jotka ovat muiden sovellusten käytettävissä. Palvelu on itsenäinen kokonaisuus. Palveluihin lukeutuvat myös WCF-palvelut. WCF-palvelut voivat olla paikallisia, samalla tietokoneella toimivia tai sitten niihin otetaan yhteyttä verkon yli, jolloin ne ovat etäpalveluita. WCF-palvelut pystyvät kommunikoimaan muillakin tekniikoilla toteutettujen palveluiden kanssa sekä toisin päin. WCF -palvelun asiakas siis voi olla käytännössä mikä tahansa. (Löwy 2010, 2.)

Asiakkaaksi sanotaan sovellusta, joka käyttää palvelua. Mikäli asiakassovellus halutaan toteuttaa Microsoftin tekniikoilla, niin se voidaan luoda esimerkiksi Windows Forms-, WPF-, tai Silverlight-tekniikoiden avulla. Asiakas voi myös olla toinen WCF-palvelu. Jos palvelu halutaan ottaa käyttöön muilla tekniikoilla, on yksi mahdollinen tapa julkaista sen tiedot WSDL:n avulla ja antaa WSDL eteenpäin asiakassovellukselle. WSDL on tekniikan suhteen neutraali. Myös WCF-palvelu pystyy lataamaan toisella tekniikalla tehdyn palvelun WSDL:n. (Löwy 2010, 2-4; Windows Communication Foundation Architecture Overview 2010.)

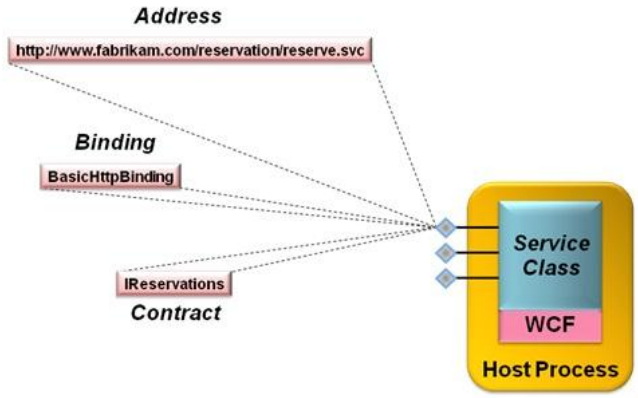


Kuvio 9. Kommunikointi välityspalvelimen kautta. (Löwy 2010, 4.)

WCF-tekniikkaa käytettäessä asiakassovellus ei koskaan ole suoraan tekemisissä palvelun kanssa. Asiakkaan ja palvelun väliin asettuu välityspalvelin, joka välittää asiakkaan kutsut itse palvelulle (kuvio 9). Välityspalvelin avaa kanavan, jonka kautta se kommunikoi palvelun kanssa. (Löwy 2010, 2-4; Windows Communication Foundation Architecture Overview 2010.)

3.3 Päätepisteet

Päätepisteet toimivat portaaleina palveluun (Windows Communication Foundation Architecture Overview. 2010). Näihin päätepisteisiin asiakkaat ottavat yhteyden, eli niitä täytyy olla yksi tai useampi. Päätepisteet sisältävät osoitteen, sidostiedon (binding) sekä sopimuksen. Päätepisteen osoite tarkoittaa sen verkko-osoitetta, jossa se sijaitsee. Sidostieto sisältää tiedon miten kyseiseen päätepisteeseen otetaan yhteys. Sopimus määrittää minkä WCF-palveluluokan päätepiste tuo asiakkaalleen näkyville. (Introducing Windows Communication Foundation in .NET Framework 4. 2010.)



Kuvio 10. WCF-palvelun päätepiste. (Introducing Windows Communication Foundation in .NET Framework 4. 2010.)

3.4 Sopimukset

Sopimukset ovat alustariippumattomia sekä standardeja tapoja kuvailla, mitä palvelu tekee. Kaikki WCF-palvelut julkistavat sopimuksensa. WCF määrittelee neljä sopimusta: palvelusopimus, datasopimus, viestisopimus ja virhesopimus. Niistä yleisimmät ovat palvelusopimukset ja datasopimukset, joten niistä tarkemmin seuraavaksi. (Löwy 2010, 7.)

3.4.1 Palvelusopimus

Jokainen WCF-palveluluokka toteuttaa jonkin palvelusopimuksen. Palvelusopimukseen määritellään, mitkä toiminnot halutaan julkistaa asiakkaille, mitä viestin-

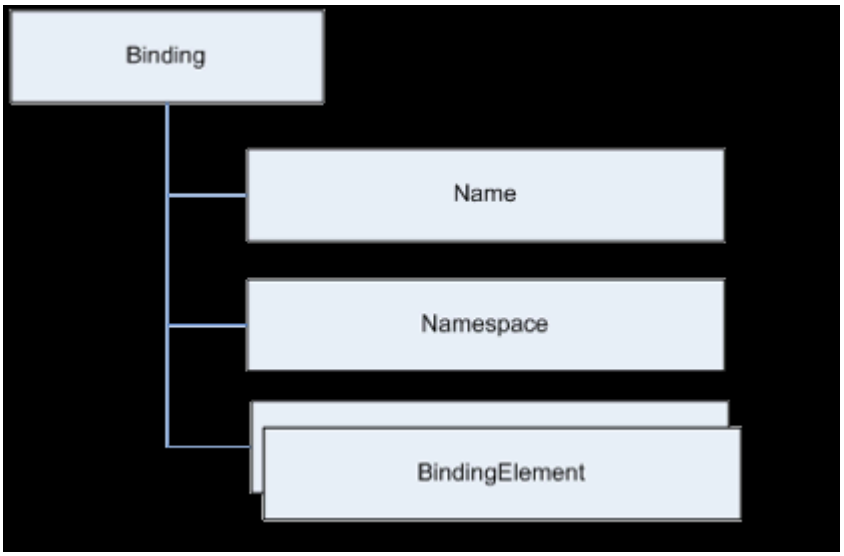
välitys menetelmää käytetään ja mitä formaattia viestit käyttävät. Jotta asiakas pystyy kutsumaan suoraan jotain toimintoa, täytyy se vielä merkitä operaatiosopimusmerkinnällä. Palveluluokkaan voidaan siis määritellä toimintoja, joita operaatiosopimuksen toteuttavat toiminnot kutsuvat, mutta asiakkaalla ei ole suoraa pääsyä niihin. (Introducing Windows Communication Foundation in .NET Framework 4. 2010; Bustamante 2007, 89.)

3.4.2 Tietotyyppisopimus

WCF-palveluluokkaan voidaan määritellä tarpeen mukaan myös tietotyyppisopimuksia. Tietotyyppisopimus määrittelee tietotyypit, joita tuodaan palveluun ja viedään palvelusta. Yksinkertaiset palveluluokat eivät tätä välttämättä tarvitse. Mikäli operaatioiden parametreinä on monimutkaisia olioita, tulee tietotyyppisopimus tarpeeseen. Tietotyyppisopimuksella määritellään, miten monimutkaiset oliot muutetaan sellaiseen muotoon, että niitä voidaan välittää verkon yli puolin ja toisin. Tätä prosessia kutsutaan sarjallistamiseksi. (Introducing Windows Communication Foundation in .NET Framework 4. 2010.)

3.5 Sidokset

Kuten jo aiemmin todettiin, sidokset kuuluvat päätepisteille. Hieman tarkemmin sanottuna, palvelulla voi olla useita päätepisteitä ja päätepisteillä yksi sidosmenetelmä. Sidokset sisältävät muun muassa käytettävän tietoliikenneprotokollan, viestin koodausmenetelmän ja turvallisuusvaatimukset kuten SSL-salaus ja Soap-viestin salaus. (Bustamante 2007, 159.)



Kuvio 11. Sidoksen rakenne. (Windows Communication Foundation Architecture Overview. 2010.)

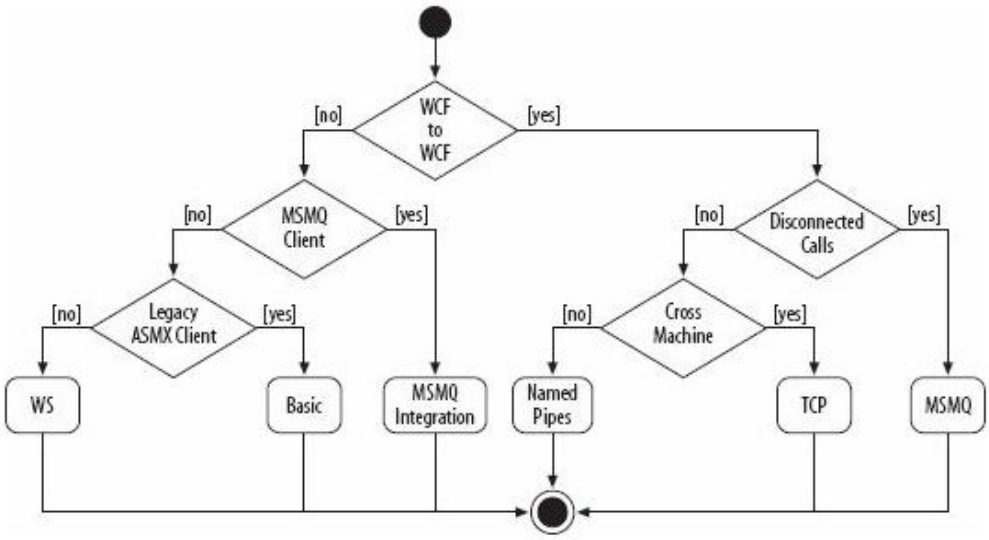
Sidos muodostuu sidoselementeistä. Jokainen elementti kuvaa, miten palvelun kanssa kommunikoidaan. Sidoksessa täytyy määritellä ainakin yksi kuljetuselementti, ainakin yksi viestin koodauselementti sekä tarvittava määrä muita elementtejä. WCF tarjoaa useita yleiskäyttösiä sidoksia, jotka sisältävät valmiiksi konfiguroidut sidoselementit. Näitä voidaan käyttää sellaisenaan tai muokata omaan tarpeeseen sopivaksi. WCF mahdollistaa myös omien sidoksien luomisen. Omissa sidoksissa voidaan käyttää samoja WCF:n tarjoamia sidoselementtejä tai itse luotuja elementtejä. Omilla elementeillä voidaan mahdollistaa muun muassa uusien tietoliikenneprotokollien tai salausmenetelmien käyttö palvelussa. (Windows Cummunications Foundation Bindings. 2011; Custom Bindings. 2012.)

WCF:n tarjoamat sidokset ovat:

- BasicHttpBinding
- WSHttpBinding
- WS2007HttpBinding
- WSDualHttpBinding
- WSFederationHttpBinding
- WS2007FederationBinding
- NetTcpBinding
- NetNamedPipesBinding
- NetMsmqBinding
- NetPeerTcpBinding

- WebHttpBinding
- MsmqIntegrationBinding. (Configuring System-Provided Bindings. 2012.)

Jokaiselle sidokselle on oma suositeltu käyttötarkoituksensa, joten ennen sen valintaa kannattaa tutustua vaihtoehtoihin (kuvio 12). Ensimmäisenä kannattaa miettiä, tukevatko kaikki tulevat asiakkaat WCF-tekniikkaa vai ei. Mikäli palvelua on tarkoitus käyttää usealla eri alustalla web servicenä, valitaan BasicHttpBinding. BasicHttpBinding julkistaa palvelun ulkomaailmalle samanlaisena, kuin tavallisen ASMX web servicen. Haittapuolena tässä sidoksessa on, että se ei tue WS-protokollia. Mikäli ei-WCF-asiakas tukee WS-standardeja, kannattaa harkita jotain seuraavista: WSHttpBinding, WSFederationBinding tai WSDualHttpBinding. WCF-asiakkaille, jotka vaativat yhteydetöntä tiedonvälitystä, on NetMsmqBinding ja yhteydellistä kommunikointia varten on NetTcpBinding. NetMsmqBinding käyttää viestinvälitykseen Microsoftin MSMQ-tekniikkaa ja NetTcpBinding kommunikoi TCP-protokollan avulla. Palvelu ja asiakas voivat sijaita samalla tietokoneella, jolloin hyvä sidosvalinta olisi NetNamedPipesBinding. NetNamedPipesBinding sallii yhteyksiä ainoastaan siltä tietokoneelta, jolla se sijaitsee. (WCF Essentials-A Developer's Primer. 2006.)



Kuvio 12. Sidoksen valinta. (WCF Essentials-A Developer's Primer. 2006.)

3.6 Turvallisuus

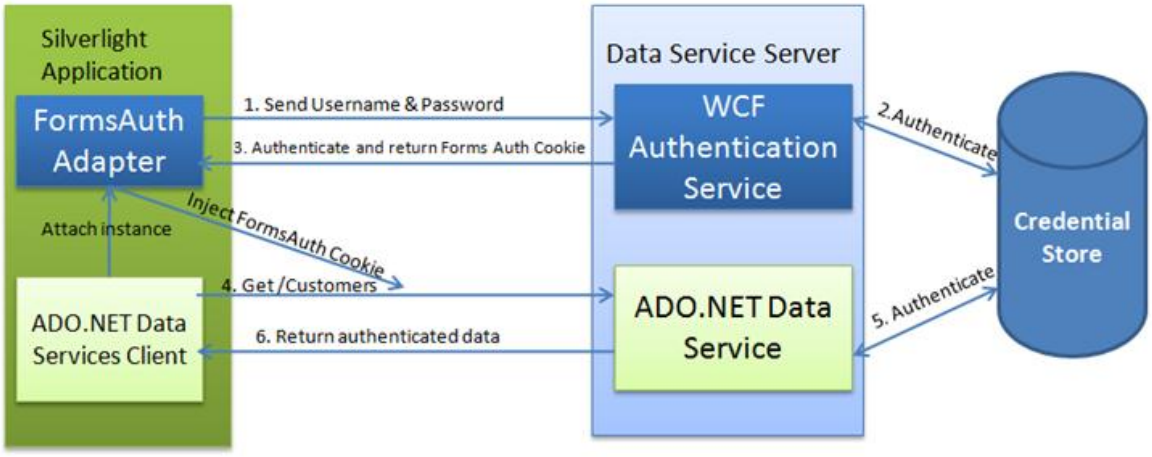
WCF-palvelusovelluksen kehityksen aikana täytyy miettiä, onko sovellus sellainen, joka vaatii suojausta. Mikäli sovellus sisältää arkaluontoista tai arvokasta tietoa, täytyy verkon yli kommunikoinnin hoitua turvallisesti. WCF tarjoaa useita tapoja hoitaa liikennöinti turvallisesti ja varmistua siitä, että asiakkaalla on oikeus käyttää palvelua. Kehittäjän tehtäväksi jää päättää, mitkä ovat millekin sovellukselle tärkeitä tapoja. Turvallisuusasiat korostuvat varsinkin silloin, kun palvelu on julkinen, eikä esimerkiksi yrityksen sisäverkossa toimiva palvelu. WCF:n tarjoamiin tapoihin varmistaa turvallinen liikennöinti kuuluvat muun muassa: autentikointi (authentication), auktorisointi (authorization) ja viestin suojaaminen (transfer security). (Bustamante 2007, 409; Löwy 2010, 525.)

3.6.1 Autentikointi

Autentikointi tarkoittaa, että palvelu varmistaa sen kutsujan olevan oikeasti se, joka kutsuja väittää olevansa. Tämä asia on tärkeää myös toiseen suuntaan. Asiakas varmistaa, että kyseessä on juuri se palvelu, jota halusikin kutsua. Autentikointia suositellaan käytettäväksi aina, kun palvelu toimii Internetissä. Mikäli autentikointia ei käytetä, palveluun voi ottaa yhteyttä kuka tahansa. (Löwy 2010, 525.)

WCF:n tarjoamat tavat toteuttaa autentikointi ovat:

- Windows-autentikointi
- käyttäjätunnus ja salasana
- X509-sertifikaatti
- omatekemä autentikointi
- Issued token. (Löwy 2010, 526.)

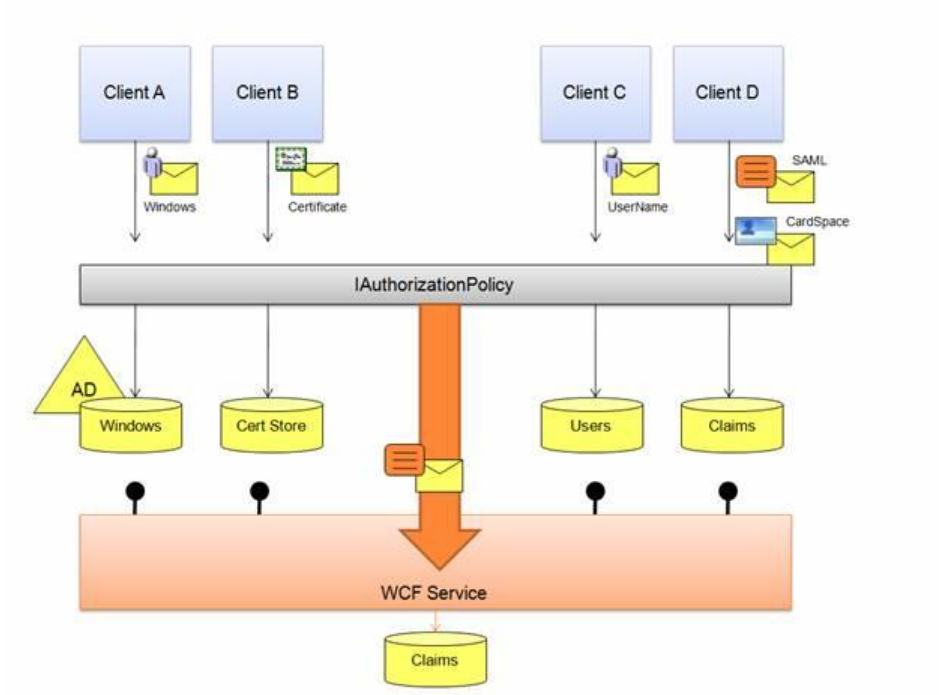


Kuvio 13. Esimerkki WCF-autentikoinnista.

3.6.2 Auktorisointi

Auktorisoinnin avulla määritellään, mitä operaatioita kullakin käyttäjällä on lupa käyttää. Auktorisointi suoritetaan sillä olettamuksella, että käyttäjä on se, jonka väittää olevansa. Tämä tarkoittaa sitä, että auktorisointia käytettäessä täytyy olla käytössä myös autentikointi. Auktorisointi ilman autentikointia on mahdollista toteuttaa, mutta menettää merkityksensä, sillä käyttäjän identiteettiä ei tarkistettaisi. (Löwy 2010, 526.)

Auktorisointiin liittyy käyttäjätietovarasto. Varastossa on kaikki käyttäjät sekä tieto siitä, mihin rooleihin käyttäjät kuuluvat. Palvelun toiminnot kuuluvat aina jollekin roolille. Kun palvelu auktorisoi pyyntöä, se katsoo varastostaan onko käyttäjällä roolia, jolla on oikeus suorittaa operaatio. WCF tukee kahdenlaisia käyttäjätietovarastoja. Ensimmäinen niistä on Windowsin käyttäjätiedot. Toinen vaihtoehto on käyttää jotain ASP.NET-tarjoajista tallentamaan käyttäjätunnukset ja roolit. ASP.NET-tarjoaja on yleensä Microsoftin SQL Server -tarjoaja. (Löwy 2010, 526.)



Kuvio 14. WCF-auktorisoinnin paikka asiakkaan ja palvelun välissä.

3.6.3 Siirron suojaaminen

Autentikointi ja auktorisointi takaavat sen, että vain käyttäjät, joilla on oikeudet palveluun, voivat käyttää sitä. Autentikointi ja auktorisointi eivät takaa sitä, että viesti saapuu perille koskemattomana. Tästä vastaa siirronsuojausmenetelmät (transfer security). Siirronsuojausmenetelmiä on yhteensä viisi: ei ollenkaan, kuljetuksen suojaus, viestin suojaus, sekoitettu sekä molemmat. Koska sekoitettu- ja molemmat-menetelmät sisältävät ominaisuuksia kuljetuksen suojaus- ja viestin suojaus -menetelmistä, niitä ei käsitellä tarkemmin. (Löwy 2010, 527; Bustamante 2007, 410-411.)

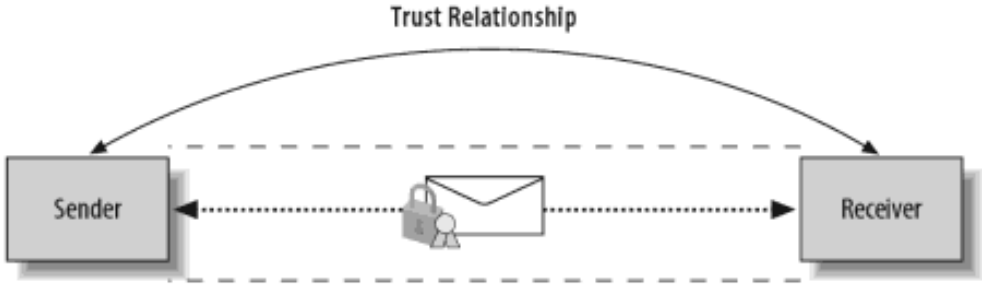
Kun siirron suojausmenetelmäksi valitaan kuljetuksen suojaus (transport transfer security), WCF ottaa käyttöön suojatun kommunikointiprotokollan. (Löwy 2010, 528.)

Käytössä olevat protokollat ovat:

- HTTPS
- TCP
- IPC

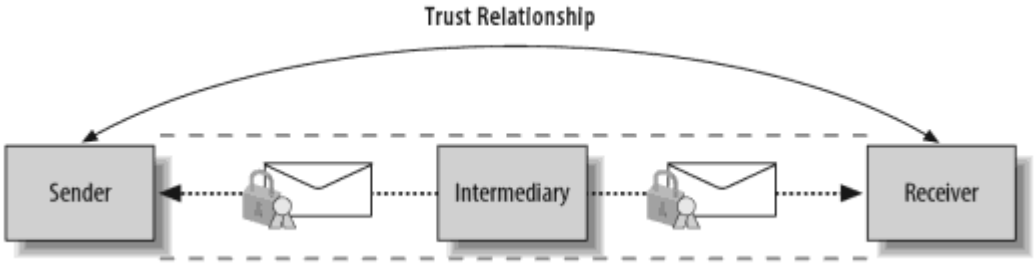
- MSMQ. (Löwy 2010. 528.)

Kuljetuksensuojausmenetelmä salaa kommunikoinnin lähettäjän ja vastaanottajan välillä. Menetelmä salaa käytettävän kuljetuskanavan. Tämä menetelmä takaa viestin eheyden, sen että kukaan muu ei pysty lukemaan sitä ja molemminpuolisen autentikoinnin. Molempien, lähettäjän ja vastaanottajan täytyy tietää, millä menetelmällä viesti on salattu. Tämä tieto saadaan molempiin päihin palvelun päätepisteen sidokselta (binding). Kuljetuksen suojausmenetelmällä on mahdollista suorittaa osa salaamisesta ja salauksen purkamisesta verkkokortilla, joka säästää tietokoneen prosessorin resursseja. Tämä menetelmä on yksinkertainen ja tehokas tapa salata liikenne asiakkaan ja palvelun välillä. Menetelmä on tehokas, mutta miinuspuolena tässä menetelmässä on, että se salaa liikenteen ainoastaan kahden pisteen välillä (kuvio 15). Jos lähettäjän ja vastaanottajan välissä on välityspalvelimia, jotka eivät ole suojattuja, on kyseenalaista onko menetelmä turvallinen. Tätä menetelmää suositellaankin vain Intranet-käyttöön. (Löwy 2010, 528; Bustamante 2007, 410-412.)



Kuvio 15. Kuljetuksen suojaus. (Bustamante 2007, 412.)

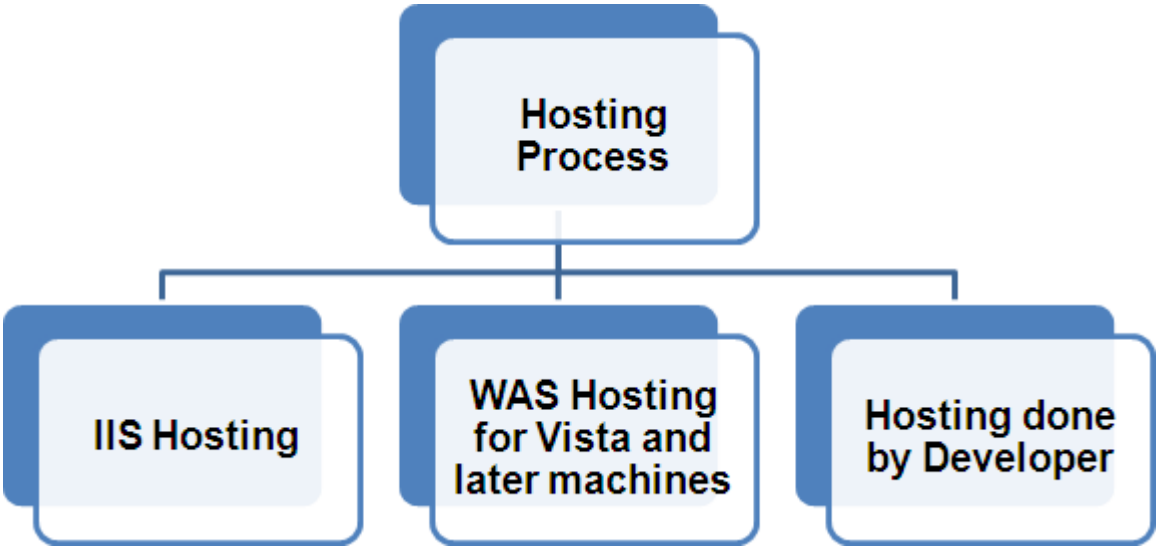
Viestin suojausmenetelmä sen sijaan on parempi tapa suojata julkinen liikennöinti. Kuten nimikin kertoo, kyseinen tapa salaa itse viestin, kuljetuskanavan sijaan. Kun viesti salataan, on sitä turvallista liikuttaa verkon yli esimerkiksi http-protokollaa käyttäen, joka ei ole itsessään turvallinen tapa. Tätä menetelmää käytettäessä ei haittaa, vaikka viestin kulkisi suojaamattomien välityspalvelimien kautta (kuvio 16). Myös tämä menetelmä takaa viestin eheyden, sen että kukaan muu ei pysty lukemaan sitä sekä molemminpuolisen autentikoinnin. (Löwy 2010, 528-529; Bustamante 2007, 411.)



Kuvio 16. Viestin suojaus. (Bustamante 2007, 412.)

3.7 Palvelun isännöinti (Hosting Service)

Jotta palvelu saadaan aktiiviseksi, täytyy se isännöidä ajonaikaisessa ympäristössä. Isännän tehtävä on huolehtia palvelun luomisesta, kontekstista ja sen elinkaaresta. WCF-palvelut on suunniteltu toimimaan missä tahansa Windows-prosessissa, joka vaan pystyy tulkitsemaan tulkattua ohjelmakoodia. Isännöintiin on kolme vaihtoehtoa: itseisännöinti, WAS ja IIS (kuvio 17). WCF:n SOA-arkkitehtuurista johtuen näyttää ohjelmakoodi lähes samalta, valinnasta riippumatta. (Hosting Services. 2012.)



Kuvio 17. Isännöintivaihtoehdot. (Kumar 2009.)

3.7.1 Itseisännöinti (Self-hosting)

Ensimmäinen tapa on itseisännöinti. Itseisännöity palvelu tarkoittaa palvelua, jota isännöi esimerkiksi sovelluskehittäjän itse tekemä sovellus. Tämä on joustavin

isännöintitapa, sillä se tarvitsee vähiten rakennetta ympärilleen toimiakseen. Palvelu saadaan käytettäväksi sulauttamalla palvelun ohjelmakoodi sovelluksen ohjelmakoodiin, luomalla ServiceHost-instanssi ja avaamalla se. Itseisännöinnillä saadaan palvelu pyörimään esimerkiksi konsolisovelluksen tai käyttöliittymäsovelluksen sisällä. Palvelun kehitysvaiheessa itseisännöidyn palvelun toimintaa on helppo seurata ja virheiden havaitseminen helpottuu. Tätä käytetään useissa WPF- ja WinForms-sovelluksissa, jotka kommunikoivat verkon yli. (Hosting Services. 2012.)

3.7.2 IIS-isännöinti (Internet Information Services)

IIS-isännöinti tarkoittaa sitä, että palvelu toimii IIS:n alla. IIS-isännöinti on integroitu ASP.NET-tekniikkaan ja siitä hyödynnetäänkin muun muassa prosessin kierrätystä, prosessin tilan seuraamista ja viestipohjaista aktivointia. Viestipohjaisen aktivoinnin ansiosta palvelu aktivoidaan vasta, kun ensimmäinen viesti lähetetään palveluun (Various Options in Hosting of WCF Services). Tätä tapaa suositellaan käytettäväksi web service -sovellusten isännöintitavaksi. Palvelun isännöintimäärityksiä ei kirjoiteta ohjelmakoodiin, vaan asetukset määritellään IIS:ään. IIS-isännöinti rajaa viestinvälitysprotokollan ainoastaan http-protokollaan. (Hosting Services. 2012.)

3.7.3 WAS-isännöinti (Windows Process Activation Service)

WAS on uusi prosessien aktivointimenetelmä, joka on Windows Server 2008:ssa ja Vistassa, sekä uudemmissä Microsoftin käyttöjärjestelmissä. Siinä on sama prosessimalli, kuin IIS:ssä, mutta se ei rajaa viestinvälitysprotokollaa http-protokollaan. WAS:n myötä viestipohjainen aktivointi onnistuu muillakin protokollilla, kuten TCP, ICP ja MSMQ, pelkän http:n sijaan. Nyt myös sovellukset, jotka käyttävät edellä mainittuja protokollia, pystyvät hyödyntämään IIS:n ominaisuuksia, kuten prosessien kierrätystä. Isännöintimäärityksiä ei tässäkään tavassa kirjoiteta ohjelmakoodiin, vaan WAS konfiguroidaan käyttäytymään halutulla tavalla.

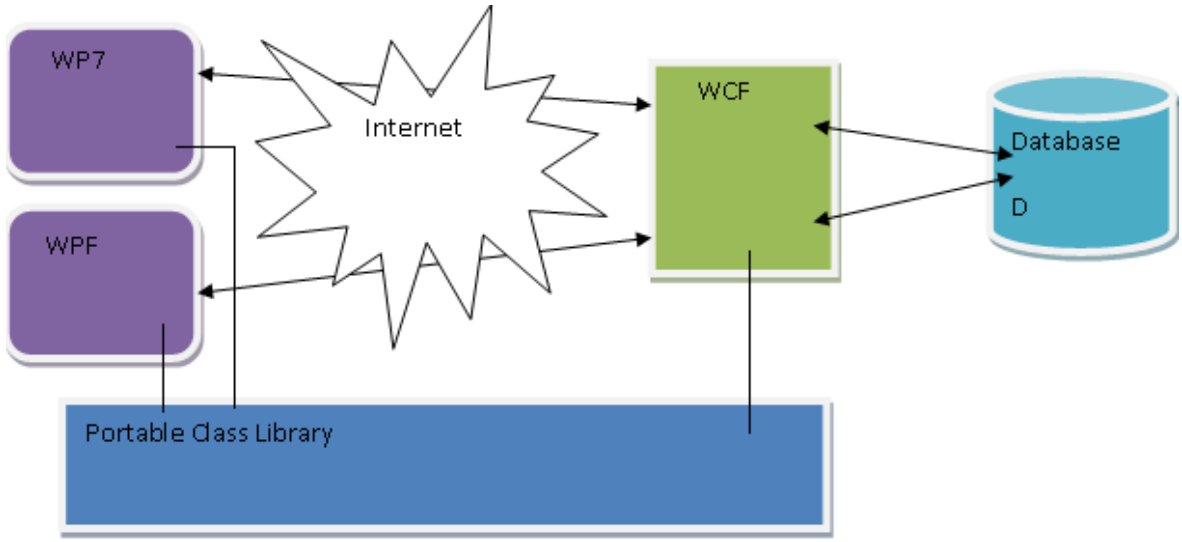
Myös IIS 7.0 käyttää WAS:ia viestipohjaisen aktivoinnin toteuttamiseen. (Hosting Services. 2012.)

4 WINDOWS PHONEN YHDISTÄMINEN WCF-PALVELUUN

4.1 Projektin yleiskuvaus

Sovellusten taustalla tietolähteenä toimii MySql-tietokanta, johon on siirretty koko A. Myllymäki Oy:n asiakasrekisteri yrityksen vanhasta Microsoft Access -tietokannasta. Lisäksi tietokanta sisältää sovellusten käyttäjien tiedot. Ensimmäisessä versiossa käyttäjistä tallennetaan ainoastaan käyttäjätunnukset ja salasanat.

Sovelluskokonaisuus koostuu kolmesta sovelluksesta: työpöytäsovellus, mobiilisovellus sekä palvelusovellus. Palvelusovelluksen tarkoitus on tarjota työpöytäsovellukselle ja mobiilisovellukselle rajapinta tietokantaan. Työpöytäsovellus on tietokannan hallintaa varten ja mobiilisovelluksen tehtävänä on saada asiakasrekisteri luettavaksi kuljettajille, sijainnista riippumatta. Sovellusten lisäksi kokonaisuuteen kuuluu Portable Class Library, joka on kaikkien sovellusten yhteinen luokkakirjasto.



Kuvio 18. Sovelluksen rakenne.

4.2 Palvelusovelluksen kuvaus

Kuten aiemmin on jo todettu, palvelusovellus on toteutettu WCF-tekniikalla ja isännöintitavaksi valittiin IIS-isännöinti. Palvelu toimii siis itsenäisesti web-palvelinympäristössä samalla tavalla kuten ASMX-tekniikalla toteutetut web service -sovellukset. IIS toimii ainoastaan Microsoftin Windows-käyttöjärjestelmissä. IIS-isännöintiin päädyttiin sen vuoksi, että tällä tavalla palvelu saatiin toimimaan itsenäisenä sovelluksena web-palvelimella. Tässä tapauksessa ei ollut tarvetta saada palvelimelle muuta kuin palvelusovellus.

4.2.1 Palvelusopimus ja palveluluokka

Palvelusovellukseen kuuluu palvelusopimusrajapinta sekä palveluluokka, joka toteuttaa palvelusopimusrajapinnan. Palvelusopimus on osa, joka näkyy palvelun asiakkaille. Kaikki toiminnot, joita ei ole määritelty palvelusopimukseen, eivät näy asiakkaille. Kuviossa 19 on esitelty tämän työn palvelusopimus, josta nähdään toiminnot, joita asiakkaat voivat suorittaa. Tähän palvelusopimukseen kuuluu kirjautuminen palveluun, asiakasrekisterin hallinta, sekä käyttäjien hallinta.

```
[ServiceContract]
public interface IMyllymakiService
{
    [OperationContract]
    byte[] Login(byte[] username, byte[] password);

    [OperationContract]
    byte[] GetCustomers(byte[] user);

    [OperationContract]
    bool DeleteCustomer(int customerId, byte[] user);

    [OperationContract]
    string InsertCustomer(byte[] customerBytes, byte[] user);

    [OperationContract]
    bool InsertUser(byte[] currentUser, byte[] userToInsert);

    [OperationContract]
    byte[] FindUser(string username, byte[] user);

    [OperationContract]
    byte[] GetUsers(byte[] user);

    [OperationContract]
    bool DeleteUser(int userId, byte[] user);
}
```

Kuvio 19. Palvelusopimus.

Kuviosta 20 nähdään, miten palveluluokka Service1 toteuttaa IMyllymakiService-palvelusopimusrajapinnan. Kuviossa 20 on näkyvillä myös esimerkkinä toiminto asiakastiedon lisäämisestä. Koska kyseinen toiminto on määritelty palvelusopimukseen, voidaan uusia asiakkaita lisätä palvelun asiakkaiden toimesta.

```
public class Service1 : IMyllymakiService
{
    public string InsertCustomer(byte[] customerBytes, byte[] user)
    {
        if (UserManager.AuthenticateUser(user))
        {
            return CustomerManager.InsertCustomer(customerBytes);
        }
        else
        {
            return "Authentication failed";
        }
```

Kuvio 20. Palveluluokka.

4.2.2 Apuluokat

Palveluluokan lisäksi sovellukseen kuuluvat erilliset luokat hallitsemaan käyttäjiä, asiakasrekisteriä, sekä salausta ja salauksen purkamista. Nämä apuluokat helpottavat virheiden paikantamista ja yhtenäistävät toimintoja. Apuluokkia käyttämällä palveluluokka pystyttiin pitämään mahdollisimman yksinkertaisena, eikä sen sisältämä ohjelmakoodi päässyt paisumaan hallitsemattomaksi. Mikäli luokka pääsee kasvamaan isoksi, käy helposti niin, että samoja asioita toistetaan useaan kertaan ja kun halutaan muuttaa yhtä, niin täytyy muuttaa kaikkia.

UserManager-luokka on staattinen luokka, joka on nimensä mukaan tarkoitettu hallitsemaan palvelun käyttäjiä. UserManager-luokka toimii palvelulle rajapintana tietokannan käyttäjätietoihin. Kyseisen luokan vastuulla on luoda, muokata, poistaa sekä listata käyttäjiä. Turvallisuuden kannalta sen tärkein tehtävä on autentikoida ja auktorisoida käyttäjät. Tämä on ainoa luokka palvelussa, jonka kautta pystytään hallitsemaan tietokannassa sijaitsevia käyttäjätietoja. Käyttäjien selaaminen ja muokkaaminen onnistuu ainoastaan admin-tunnuksilla. Tavallinen käyttäjä ei siis pysty tämän luokan avulla tekemään muuta kuin yrittää kirjautumista järjestelmään.

```
public static byte[] FindUser(string username)
{
    using (MySqlConnection connection = new MySqlConnection(connStr))
    {
        try
        {
            PortableUser portableUser = null;
            MyLLY db = new MyLLY(connection);
            User dbUser = db.Users.SingleOrDefault(x => x.Username == username);

            if (dbUser != null)
            {
                portableUser = new PortableUser();
                portableUser.Username = dbUser.Username;
                portableUser.Password = dbUser.Password;
                portableUser.Id = dbUser.Id;
                return SerializePortableUser(portableUser);
            }
            return null;
        }
        catch (Exception ex)
        {
            return null;
        }
    }
}
```

Kuvio 21. Esimerkki UserManager-luokan toiminnosta.

CustomerManager on luokka, joka puolestaan hallitsee asiakasrekisteriä. Kaikki asiakasrekisteriä koskevat haut ja muutokset menevät keskitetysti tämän kautta tietokantaan. CustomerManager-luokan toimintoihin pääsee käsiksi ainoastaan ne käyttäjät, jotka UserManager-luokka on autentikoinut onnistuneesti, ja heillä on oikeus käyttää toimintoa. Kaikilla käyttäjillä, jotka on lisätty järjestelmään, on oikeus lukea asiakasrekisteriä, mutta vain admin-tunnuksilla voidaan muokata sitä.

```
public static byte[] GetCustomers()
{
    using (MySqlConnection connection = new MySqlConnection(connStr)) {
        try {
            MyLLY db = new MyLLY(connection);
            List<Customer> customersList = new List<Customer>();
            foreach (OsOItTeeT source in db.OsOItTeeT) {
                Customer c;
                try {
                    c = ToDesktopCustomer(source);
                    customersList.Add(c);
                }
                catch (Exception ex) {
                    string s = ex.Message;
                }
            }
            CustomerList customersListObject = new CustomerList();
            customersListObject.customers = customersList;
            MemoryStream stream = new MemoryStream();
            Serializer.Serialize<CustomerList>(stream, customersListObject);
            return CryptoManager.Encrypt(stream.ToArray());
        }
        catch (Exception e) {
            return null;
        }
    }
}
```

Kuvio 22. Asiakkaiden hakeminen tietokannasta.

Salausta ja salauksen purkua varten tehtiin oma luokka: CryptoManager. Palveluun tuleva ja lähtevä tieto kuljetetaan CryptoManager-luokan kautta, joka purkaa ja salaa tiedon tietyillä salasanoilla. CryptoManager-luokka pystyy vaihtamaan tarvittaessa tekstin koodausta, UTF-8:n ja Base64:n välillä. Tekstin koodauksen muuttaminen on tarpeellista silloin, kun käyttäjien salasanat tallennetaan tietokantaan. Luokka käyttää salaustapana System.Security.Cryptography-nimiavaruudesta löytyvää symmetristä 128-bittistä AES-algoritmia. AES on suhteellisen turvallinen salausmenetelmä. Kuviossa 23 nähdään toiminto, joka purkaa salatun, Base64-muotoisen tekstin selkokieliseksi UTF-8:lla koodatuksi tekstiksi.

```
public static string DecryptFromBase64(string dataToDecrypt)
{
    AesManaged aes = null;
    MemoryStream memoryStream = null;
    CryptoStream cryptoStream = null;
    try {
        Rfc2898DeriveBytes rfc2898 = new Rfc2898DeriveBytes(pw, Encoding.UTF8.GetBytes(salt));
        aes = new AesManaged();
        aes.Key = rfc2898.GetBytes(aes.KeySize / 8);
        aes.IV = rfc2898.GetBytes(aes.BlockSize / 8);
        memoryStream = new MemoryStream();
        cryptoStream = new CryptoStream(memoryStream, aes.CreateDecryptor(), CryptoStreamMode.Write);
        //Decrypt Data
        UTF8Encoding enc = new UTF8Encoding(false);
        byte[] data = Convert.FromBase64String(dataToDecrypt);
        cryptoStream.Write(data, 0, data.Length);
        cryptoStream.FlushFinalBlock();
        //Return Decrypted String
        byte[] decryptBytes = memoryStream.ToArray();
        return enc.GetString(decryptBytes);
    }
    finally {
        if (cryptoStream != null)
            cryptoStream.Close();
        if (memoryStream != null)
            memoryStream.Close();
        if (aes != null)
            aes.Clear();
    }
}
```

Kuvio 23. Esimerkki salauksen purkutoiminnosta.

4.2.3 Palvelun asetukset

IIS-isännöidyn palvelun asetukset sijaitsevat Web.config-tiedostossa, jonka Visual Studio luo automaattisesti kun aloitetaan uusi WCF Application -projekti. Tässä kyseisessä palvelusovelluksessa on käytössä vain yksi päätepiste, eli palveluun voidaan ottaa yhteys ainoastaan yhdellä tavalla. Kun päätepisteitä on vain yksi, tarkoittaa se myös sitä, että ainoastaan yksi palvelusopimus julkistetaan asiakkaille. Päätepisteelle kuuluu myös käytetty sidos. Sidos määritellään erillään, mutta päätepisteelle kerrotaan käytettävän sidoksen nimi. Kuviosta 24 nähdään käytössä olevan päätepisteen tämän hetkiset asetukset. Käytetty sidos on nimeltään basicHttpBinding ja päätepisteen julkaisema palvelusopimus on MyllymakiService-nimiavaruudessa sijaitseva IMyllymakiService. Osoitekenttä on tyhjä, jotta osoitteeksi saadaan suoraan IIS:n virtuaalisen kansion nimi.

```
<service name="MyllymakiService.Service1">
  <endpoint address="" binding="basicHttpBinding"
            contract="MyllymakiService.IMyllymakiService"  />
</service>
```

Kuvio 24. Palvelun päätepisteen asetukset.

Palvelun sidostavaksi valittiin basicHttpBinding, koska palvelu haluttiin saada käytettäväksi samalla tavalla, kuin tavallinen web service. Alustava sidos, joka tässä palvelussa on käytössä, on esitelty kuviossa 25. Sidokseen on vasta määritelty viestin kuljetustapa, suurin sallittu koko sekä aikaraja, jonka puitteissa yksittäinen viesti täytyy saada perille. Tietokannassa on paljon tietueita, jonka takia aikarajaa ja viestin kokoa on kasvatettu.

Kuten jo aiemmin mainittiin, ei basicHttpBinding ole sellaisenaan turvallinen. Samasta syystä aiemmin kerrottiin salauksesta huolehtivasta apuluokasta, jonka ansiosta kukaan muu ei pysty lukemaan viestejä. Tässä vaiheessa palvelun kehitystä ei toteuteta WCF:n omia salausmenetelmiä, sillä Windows Phone 7 -sovellukset eivät tällä hetkellä tue niitä.

```
<basicHttpBinding>
  <binding maxReceivedMessageSize="697108864" receiveTimeout="01:00:00" transferMode="Streamed" >
  </binding>
</basicHttpBinding>
```

Kuvio 25. Sidoksen asetukset.

4.3 Mobiilisovelluksen kuvaus

Windows Phone 7 -mobiilisovelluksen tehtävänä on toimia kuljetusliikkeen kuljettajien apuna. Mobiilisovelluksen avulla he voivat selata asiakasrekisteriä tien päällä. Sovelluksen avulla kuljettaja saa tietoonsa muun muassa asiakkaan nimen, osoitteen, ajo-ohjeen paikan päälle sekä mahdolliset puhelinnumerot. Puhelinnumeroihin pystytään myös soittamaan suoraan sovelluksesta.

4.3.1 Käyttöliittymä

Sovelluksen käyttöliittymä on pyritty pitämään mahdollisimman yksinkertaisena ja selkeänä, Metro-suuntaviivat mielessä pitäen. Kuitenkin tärkein syy siihen, että käyttöliittymän täytyy olla yksinkertainen ja helppokäyttöinen, johtuu siitä, että sovelluksen tulevat käyttäjät ovat keskimäärin melko iäkkäitä. Koska käyttöliittymä on niin yksinkertainen, on se tehty Visual Studiolla, eikä Expression Blendillä. Käyttö-

liittymä koostuu kolmesta eri näkymästä: aloitusnäkymästä, asiakasrekisterinäkymästä sekä tarkemmat tiedot -näkymästä.

Aloitusnäkymässä on käytetty PanoramaControl-komponenttia, joka on hyvin tyypillinen komponentti Windows Phone 7 -sovellusten aloitusnäkymissä (kuvio 26). Aloitusnäkymä ei sisällä tällä hetkellä muita toimintoja, kuin linkin asiakasrekisterinäkymään. Aloitusnäkymään voidaan tulevaisuudessa tuoda esimerkiksi ilmoituksia, jotka halutaan kuljettajille näyttää, mikäli asiakas niin haluaa.

Kuvio 26. Aloitusnäkymä.

Asiakasrekisterinäkymä on sovelluksen tärkein näkymä. Se listaa palvelusovelluksen kautta saadut asiakastiedot. Asiakasrekisterinäkymässä on mahdollista myös hakea ladatuista asiakkaista näkyviin vain tietyt asiakkaat, nimen tai osoitteen perusteella.

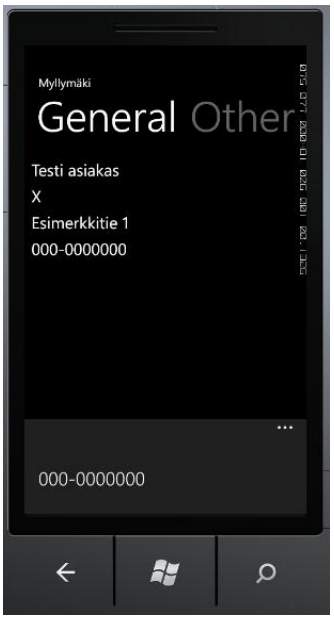
Kuvio 27. Asiakasrekisterinäkymä.

Kuviossa 28 on asiakasrekisterinäkymässä käytetyn ListBox-komponentin asetukset. Asetukset käsittävät listaan liittyvät toiminnot, miltä lista näyttää sekä sen, miten listan rivit esitetään. Jokainen asiakas on omalla rivillään, eli listan rivi kuvaa aina yhtä asiakasta.

```
<ListBox x:Name="lbCustomers" SelectionChanged="lbCustomers_SelectionChanged" DoubleTap="lbCustomers_DoubleTap">
    <ListBox.ItemTemplate>
        <DataTemplate>
            <StackPanel Orientation="Vertical" Margin="0,0,0,10" Background="Green">
                <TextBlock FontSize="24" VerticalAlignment="Center" Text="{Binding Name}" Width="400"></TextBlock>
            <TextBlock FontSize="22" VerticalAlignment="Center" Text="{Binding Address}" Width="400"></TextBlock>
            </StackPanel>
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>
```

Kuvio 28. Asiakasrekisterin ListBox-komponentti.

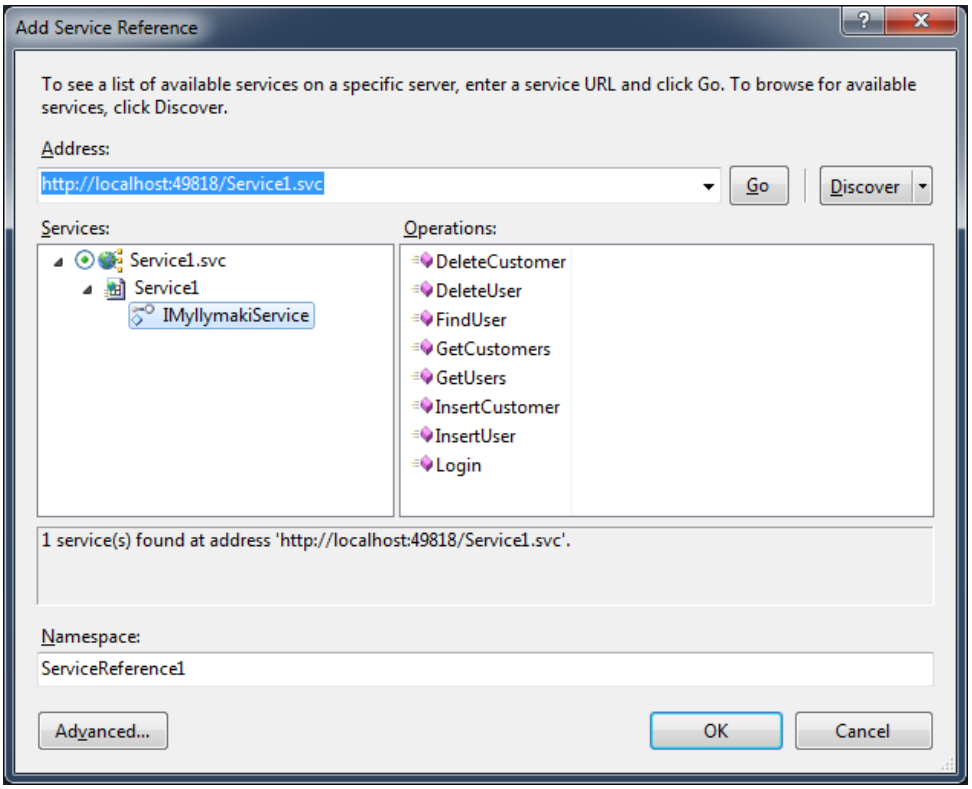
Tarkemmat tiedot -näkymän tarkoitus on näyttää selkeässä muodossa yksittäisen asiakkaan kaikki tiedot. Näkymä mahdollistaa myös soittamisen asiakkaalle, mikäli asiakkaan puhelinnumero löytyy tietokannasta.



Kuvio 29. Tarkemmat tiedot -näkymä.

4.3.2 Palvelun käyttöönotto asiakassovelluksessa

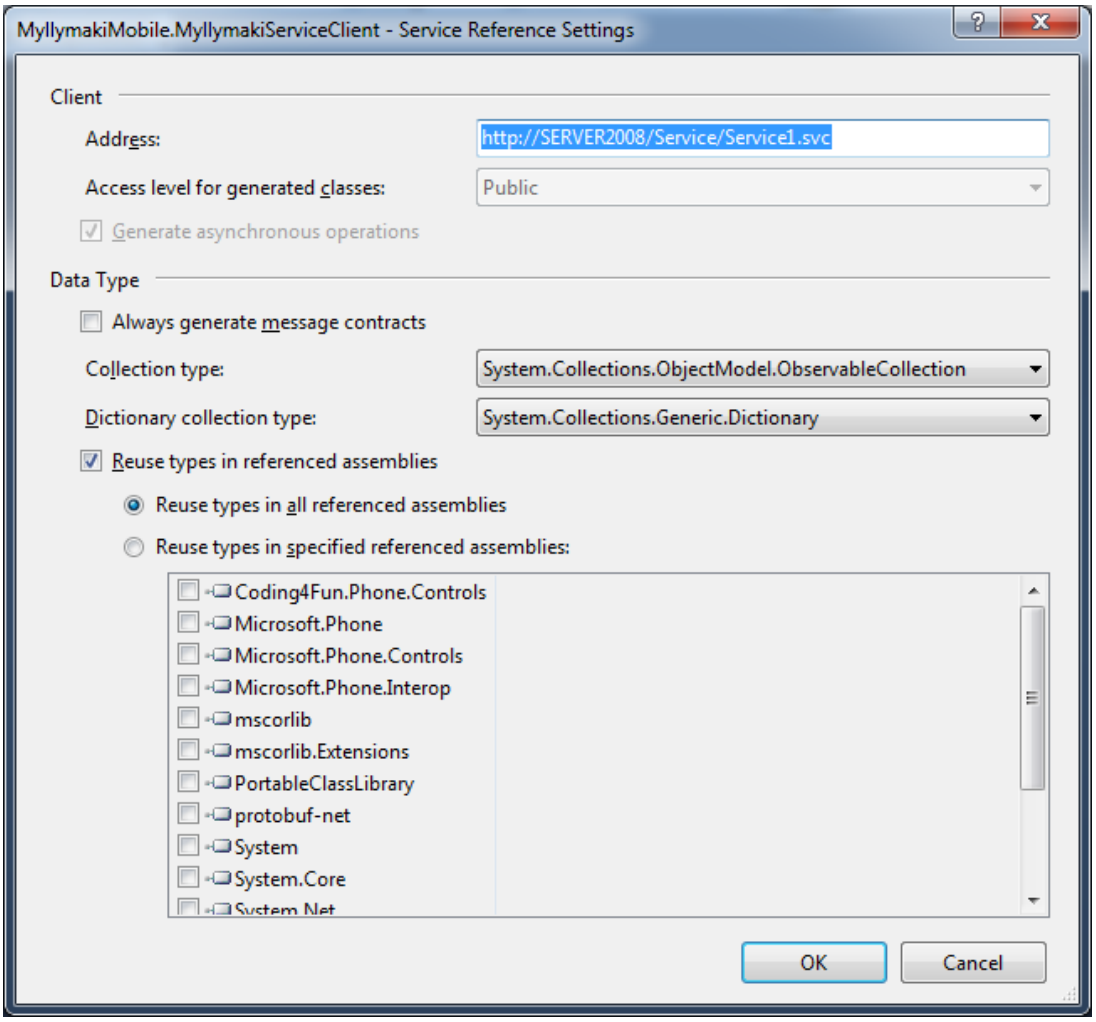
Mikäli mobiilisovellusta halutaan käyttää WCF -palvelusovelluksen asiakkaana, täytyy mobiilisovelluksen tietää minkä palvelun asiakas se on. Palvelu on itsenäinen, eikä ota kantaa siihen, mikä tai kuka asiakas on. Windows Phone 7 -sovelluksen tapauksessa sille täytyy lisätä viittaus haluttuun palveluun. Visual Studio 2010:llä se onnistuu pienellä vaivalla. Windows Phone 7 -projektin alta löytyvää Service References kansiota klikataan hiiren kakkospainikkeella ja valitaan valikosta Add Service Reference. Add Service Reference -ikkunan avauduttua, voidaan kirjoittaa palvelun osoite Address-kenttään ja painaa Go. Jos palvelusovellusprojekti kuuluu samaan Solutioniin, niin voidaan unohtaa osoite ja painaa pelkästään Discover. Discover etsii kaikki palvelut, jotka kuuluvat saman Solutionin alle. Kuviossa 30 on klikattu Discover-painiketta ja Visual Studio on löytänyt Service1-palvelun, joka onkin ainut palvelu tässä Solutionissa. Kuviosta 30 nähdään myös kaikki palvelun toiminnot Operations-listassa. Nämä ovat ne toiminnot, jotka on määritelty palvelun julkistamaan palvelusopimukseen.



Kuvio 30. Palveluviittauksen lisääminen.

On hyvä muistaa, että aina kun palvelusovellusta muutetaan, täytyy myös viittaus palveluun päivittää asiakkaille. Visual Studiossa palveluviittauksen päivitys onnistuu klikkaamalla hiiren kakkospainiketta palveluviittauksen päällä ja avautuvasta valikosta klikataan Update Reference.

Kun viittaus palveluun on olemassa, sen asetuksiin voidaan vielä vaikuttaa. Jälleen klikataan hiiren kakkospainikkeella palveluviittausta ja tällä kertaa valitaan Configure Service Reference. Asetuksia, joihin voidaan vaikuttaa, ovat muun muassa palvelun osoite ja minkä tyyppisiä kokoelmia palautetaan palvelusta. Merkittävin asetus sovelluksen toiminnallisuuden kannalta on se, luodaanko myös asynkroniset versiot toiminnoista synkronisten rinnalle. Windows Phone 7 -asiakkaat eivät pysty vaikuttamaan tähän asetukseen vaan heille luodaan aina pelkästään asynkroniset toiminnot. Windows Phone 7:n asynkronisista palvelutoiminnoista, kerrotaan seuraavassa luvussa.



Kuvio 31. Palveluviittauksen asetukset.

4.3.3 Windows Phone 7 ja asynkroniset palvelutoiminnot

Edellisessä kappaleessa mainittiin, että kaikki Windows Phone 7 -asiakkaiden käyttämät palvelutoiminnot suoritetaan aina asynkronisesti. Synkroninen toiminto tarkoittaa sellaista toimintoa, joka suoritetaan välittömästi ja sovellus jää odottamaan kunnes toiminto on suoritettu. Asynkroninen toiminto on synkronisen vastakohta. Kun asynkroninen toiminto käynnistetään, sen suoritus tapahtuu toisessa säikeessä, jolloin sovellus pystyy jatkamaan eteenpäin ilman, että jäisi odottelemaan toiminnon loppuun suoritusta. Kun asynkroninen toiminto on suoritettu loppuun, se lähettää tapahtumankäsittelijälleen tapahtuman siitä, että on valmis. Asynkronisten toimintojen etu on siinä, että niillä pystytään kätevästi suorittamaan

raskaita, aikaa vieviä toimintoja taustalla, jolloin käyttöliittymä pysyy kokoajan reagoivana. Asynkronisia toimintoja käytetään usein sovelluksissa, jotka kommunikoivat verkon yli tai käsittelevät laajoja tietokantoja.

4.4 Sarjallistaminen Protocol Buffers -kirjastolla

Tässä työssä mainittiin sarjallistaminen aiemminkin. Sarjallistaminen siis tarkoittaa sitä prosessia, jossa jokin monimutkainen olio muutetaan sellaiseen muotoon, että se voidaan siirtää verkon yli ja muuttaa toisessa päässä takaisin samaksi olioksi (MSDN Serialization). Tässä työssä sarjallistamiseen käytetään Googlen kehittämän Protocol Buffers -kirjaston C#-versiota: Protobuf-net. Protocol Buffers on alustariippumaton binäärisarjallistamismuoto ja se on toteutettu jo usealle ohjelmointikielelle. Näihin ohjelmointikieliin kuuluvat muun muassa C#, Java, C++ ja Python. Protocol Buffers on tehokas sarjallistamismuoto. Se toimii kuten XML, mutta on pienempi, tehokkaampi ja yksinkertaisempi. (Use "Protocol Buffers" serialization from your .NET code. 2011; Protocol Buffers: Developer Guide).

Miksi tässä työssä käytetään Protocol Buffers -kirjastoa? Jokainen tekniikka, WCF, WP7 ja WPF suorittavat sarjallistamisen eri tavalla. Käyttöön haluttiin ottaa yhtenäinen tapa ja sen Protocol Buffers pystyi tarjoamaan, sillä se toimii samalla tavalla riippumatta alustasta. Yhtenäistä sarjallistamistapaa haettaessa valinta oli melko selkeä, sillä tekijälle Protocol Buffers oli jo entuudestaan tuttu. Mikäli asiakas haluaisikin vaihtaa WP7-mobiilialusta esimerkiksi Androidiin, käy Protocol Buffers siinäkin tapauksesta. Tekijän aiempi kokemus Protocol Buffers -kirjaston käytöstä on nimenomaan .NET-, Java- ja Android-sovellusten yhteistoiminnasta verkon välityksellä.

4.4.1 Protocol Buffers -kirjaston käyttö

.NET-ympäristössä Protocol Buffersia voidaan käyttää kahdella tavalla: prototiedoston avulla tai lisäämällä sarjallistettavaan luokkaan tietyt merkinnät.

Ensimmäinen tapa, eli proto-tiedoston avulla, on alkuperäinen tapa luoda sarjallistettavia luokkia. Kuvio 32 kuvaa yksinkertaisen proto-tiedoston sisältöä. Proto-tiedosto sisältää viestejä, jotka sisältävät pakollisia, vapaaehtoisia ja toistuvia kenttiä. Jokainen kenttä merkitään yksilöivällä numerolla ja jokainen kenttä sisältää nimi-arvo parin. Viestin sisällä voidaan myös määritellä lisää viestejä, kuten kuviossa 32 on määritelty Person-viestin sisällä PhoneNumber-viesti.

```
message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;

  enum PhoneType {
    MOBILE = 0;
    HOME = 1;
    WORK = 2;
  }

  message PhoneNumber {
    required string number = 1;
    optional PhoneType type = 2 [default = HOME];
  }

  repeated PhoneNumber phone = 4;
}
```

Kuvio 32. Yksinkertainen Proto-tiedosto.

Kun proto-tiedosto on valmis, siitä käännetään viestejä vastaavat luokat Protocol Buffers kääntäjällä, jonka jälkeen luokat ovat sovellusten käytettävissä. Visual Studio 2010:een on saatavilla työkalu, joka generoi ja päivittää proto-tiedostoista luokat automaattisesti jokaisella tallennuskerralla.

Toinen tapa, jolla luokat voidaan sarjallistaa käyttäen Protocol Buffersia, on luoda luokka ja lisätä siihen tarvittavat merkinnät. Kuviossa 33 nähdään kaksi tavallista C#-kielellä toteutettua luokkaa. Kuviossa 34 nähdään Protocol Buffers sarjallistettu luokka. Luokan määrittelyn yhteydessä luokan yläpuolelle merkitään [ProtoContract] sekä jokaisen sarjallistettavan ominaisuuden yläpuolelle [ProtoMember(Numero)]. Kuten proto-tiedostoa käytettäessä, niin samoin tässäkin tavassa jokainen kenttä merkitään uniikilla numerolla.

```
class Person {
    public int Id {get;set;}
    public string Name {get;set:}
    public Address Address {get;set;}
}
class Address {
    public string Line1 {get;set;}
    public string Line2 {get;set;}
}
```

Kuvio 33. Luokat ilman Proto-merkintöjä. (Protocol Buffers: Getting Started. 2011.)

```
[ProtoContract]
class Person {
    [ProtoMember(1)]
    public int Id {get;set;}
    [ProtoMember(2)]
    public string Name {get;set:}
    [ProtoMember(3)]
    public Address Address {get;set;}
}
[ProtoContract]
class Address {
    [ProtoMember(1)]
    public string Line1 {get;set;}
    [ProtoMember(2)]
    public string Line2 {get;set;}
}
```

Kuvio 34. Luokat Proto-merkinnöillä. (Protocol Buffers: Getting Started. 2011.)

Käytettiin kumpaa tahansa aiemmin kerrotuista tavoista, niin valinta ei vaikuta luokan käyttöön. Kun halutaan sarjallistaa Person-luokan olio, se luodaan ja siitä täytetään ainakin pakolliset kentät. Sitten se annetaan parametrina Protocol Buffersin Serializer-luokan Serialize-toiminnolle. Serialize-toiminnolle annetaan myös toisena parametrina kohdevirta, johon tulos kirjoitetaan. Kuviossa 35 person-olio kirjoitetaan person.bin-tiedostoon.

```
var person = new Person {
    Id = 12345, Name = "Fred",
    Address = new Address {
        Line1 = "Flat 1",
        Line2 = "The Meadows"
    }
};
using (var file = File.Create("person.bin")) {
    Serializer.Serialize(file, person);
}
```

Kuvio 35. Person-olion sarjallistaminen. (Protocol Buffers: Getting Started. 2011.)

Kuviossa 36 on esimerkki miten kuviossa 35 sarjallistettu person-olio palautetaan takaisin samaksi olioksi. Palautukseen käytetään samaa Serializer-luokkaa ja sen toimintoa Deserialize. Deserialize-toiminnolle kerrotaan mihin luokkaan palautettava olio kuuluu, sekä virta, josta olio palautetaan.

```
Person newPerson;
using (var file = File.OpenRead("person.bin")) {
    newPerson = Serializer.Deserialize<Person>(file);
}
```

Kuvio 36. Palautus takaisin Person-olioksi. (Protocol Buffers: Getting Started. 2011.)

4.5 Lisäkirjastot

Tässä kappaleessa esitellään vielä kaksi kirjastoa, joita on hyödynnetty tässä työssä.

4.5.1 DbLinq

Tekijällä on kokemusta tietokantojen käsittelystä usealla eri tavalla. Miellyttävin tapa tietokantojen käsittelyyn on tekijän mielestä LINQ-kyselykieli. LINQ tulee sanoista Language-Integrated Query. LINQ esiteltiin Visual Studio 2008:n myötä. LINQ:lla voidaan suorittaa kyselyitä:

- .NET-kokoelmista
- SQL Server -tietokannoista
- ADO.NET-dataseteistä
- XML-dokumenteista. (DbLinq2007. 2010.)

Kuten listasta nähdään, LINQ ei tue MySql-tietokantoja. Tässä kohdassa apuun tulee DbLinq-kirjasto. DbLinq pystyy generoimaan MySql-tietokannasta samanlaisen dbml-tiedoston kuin LINQ generoi SQL Server -tietokannasta. Tämän jälkeen LINQ-kyselyitä pystytään tekemään MySql-tietokantaan samaan tapaan kuin SQL Server -tietokantaan. (DbLinq2007. 2010.)

4.5.2 Portable Class Library

Portable Class Library on luokkakirjasto, joka toimii sellaisenaan useassa eri .NET-sovelluskehyksen alustassa, kuten: Silverlight, Windows Phone 7, XBox 360. Ilman Portable Class Librarya täytyisi kohdistaa tavallinen luokkakirjasto yh-

delle alustalle ja sen jälkeen muokata se käsin sopivaksi muille alustoille. Täytyy muistaa, että kaikki alustat eivät tue kaikkia ominaisuuksia, joten onkin tärkeää valita kohteiksi ainoastaan tarvittavat alustat. Portable Class Library -projekti ei tule Visual Studion mukana, vaan se täytyy asentaa erikseen. (Portable Class Libraries. 2012.)

5 JOHTOPÄÄTÖKSET JA JATKOKEHITYS

Työlle asetetut tavoitteet ovat saavutettu. Mobiilisovelluksen avulla asiakasrekisteriä voidaan selata missä vain, kunhan Internet-yhteys on käytettävissä. Palvelusovellus on suojattu käyttäjätunnuksilla, mikä takaa sen, että kuka tahansa ei pääse asiakasrekisteriin käsiksi. Toimeksiantaja on ollut tyytyväinen tulokseen ja työstä on saatu hyvää palautetta.

Palvelusovelluksen ja mobiilisovelluksen tekeminen oli haastavaa mutta opettavaista, koska molemmat Windows Phone 7 sekä Windows Communication Foundation olivat tekijälle vieraita. Suurin haaste oli WCF-palvelun ohjelmointi, koska Windows Communication Foundation on laaja käsite ja se tarjoaa runsaan määrän vaihtoehtoja asioiden toteuttamiseen. Työn tekeminen oli mielenkiintoista, kun oppi uusia asioita, onnistui ratkaisemaan ongelmia ja lopulta onnistui toteuttamaan haluamansa asiat.

Projekti on valmis, mutta työn edetessä on tekijälle tullut mieleen ideoita sovelluksen jatkokehitykseen. Windows Phone 7 -sovellukset pystyvät vastaanottamaan ilmoituksia WCF-palvelulta ja näyttämään niitä. Näitä ilmoituksia voitaisiin hyödyntää esimerkiksi tapauksessa, jossa konttorilta halutaan lähettää jokin viesti kuljettajille. Lisäksi käyttöön voisi ottaa jonkun WCF:n tarjoamista salausratkaisuista, kun tällä hetkellä käytössä on tekijän itse tekemä salausmenetelmä.

LÄHTEET

Additions in the Windows Phone SDK 7.1. 2012. [Verkkosivu]. MSDN. [Viitattu 26.3.2012]. Saatavana: http://msdn.microsoft.com/en-us/library/ff637516%28v=vs.92%29.aspx

Bustamante, M L. 2007 Learning WCF. O'Reilly Media, Inc.

CodePlex: Open Source Project Community. Päivitetty 2012. [Verkkosivusto]. [Viitattu 29.3.2012]. Saatavana: http://www.codeplex.com/

Configuring System-Provided Bindings. 2012. [Verkkosivu]. MSDN. [Viitattu 4.3.2012]. Saatavana: http://msdn.microsoft.com/en-us/library/ms731092.aspx

Custom Bindings. 2012. [Verkkosivu]. MSDN. [Viitattu 29.3.2012]. Saatavana: http://msdn.microsoft.com/en-us/library/aa347793.aspx

DbLinq2007. Päivitetty 16.4.2010. [Verkkosivu] [Viitattu 25.3.2012]. Saatavana: http://code.google.com/p/dblinq2007/

Execution Model Overview for Windows Phone. 2012. [Verkkosivu]. MSDN. [Viitattu 3.3.2012]. Saatavana: http://msdn.microsoft.com/en-us/library/ff817008%28v=vs.92%29.aspx

Features Differences Between Silverlight and Silverlight for Windows Phone. 2012. [Verkkosivu]. MSDN. [Viitattu 2.3.2012]. Saatavana: http://msdn.microsoft.com/en-us/library/ff426931%28v=VS.95%29.aspx

Ferracchiati, F. C. & Garofalo, E. 2011. Windows Phone Recipes: A Problem-Solution Approach. 2. painos. Apress.

General Design Principles. 2012. [Verkkosivu]. MSDN. [Viitattu 3.3.2012]. Saatavana: http://msdn.microsoft.com/en-us/library/hh202906%28v=vs.92%29.aspx

Hardware Specifications for Windows Phone. 2012. [Verkkosivu]. MSDN. [Viitattu 1.3.2012]. Saatavana: http://msdn.microsoft.com/en-us/library/ff637514%28v=vs.92%29.aspx

Hosting Services. 2012. [Verkkosivu]. MSDN. [Viitattu 6.3.2012]. Saatavana: http://msdn.microsoft.com/en-us/library/ms730158.aspx

Introducing Windows Communication Foundation in .NET Framework 4. Päivitetty 2010. [Verkkosivu]. MSDN. [Viitattu 20.3.2012]. Saatavana: http://msdn.microsoft.com/library/ee958158.aspx

Järvinen, J. 2012. Windows Phone: Sovelluskehitys. Porvoo: Bookwell Oy.

Kumar, D. 2009. Various Options in Hosting of WCF Services. [Verkkojulkaisu]. Saatavana: http://www.c-sharpcorner.com/uploadfile/dhananjaycoder/various-options-in-hosting-of-wcf-services/

Lee, H. & Chuvyrov, E. 2011. Beginning Windows Phone 7 Development. 2. painos. Apress.

Lövy, J. 2010. Programming WCF Services. 3. painos. O'Reilly Media, Inc.

Löwy, J. 2006. WCF Essentials-A Developer's Primer. [Verkkojulkaisu]. [Viitattu 1.4.2012]. Saatavana: http://www.code-magazine.com/article.aspx?quickid=0605051&page=3

MVVM Light Toolkit. [verkkosivu]. GalaSoft. [Viitattu 5.3.2012]. Saatavana: http://www.galasoft.ch/mvvm/#blogs

Portable Class Libraries. Päivitetty 2011. [Verkkosivu]. MSDN. [Viitattu 30.3.2012]. Saatavana: http://msdn.microsoft.com/en-us/library/gg597391.aspx

Protocol Buffers: Developer Guide. 2011. [Verkkosivu]. Google Inc. [Viitattu 2.4.2012]. Saatavana: http://code.google.com/apis/protocolbuffers/docs/overview.html

Protocol Buffers: Getting Started. 2011. [Verkkosivu]. Google Inc. [Viitattu 2.4.2012]. Saatavana: http://code.google.com/p/protobuf-net/wiki/GettingStarted

Randolph, N & Fairbairn, C. 2010. Professional Windows Phone 7 Application Development: Building Applications and Games Using Visual Studio, Silverlight, and XNA. Wrox.

Serialization. 2012. [Verkkosivu]. MSDN. [Viitattu 1.4.2012]. Saatavana: http://msdn.microsoft.com/en-us/library/7ay27kt9%28v=vs.80%29.aspx

Use "Protocol Buffers" serialization from your .NET code. 2011. [Verkkosivu]. Google Inc. [Viitattu 30.3.2012]. Saatavana: http://code.google.com/p/protobuf-net/

What Is Windows Communication Foundation. 2012. [Verkkosivu]. MSDN. [Viitattu 10.3.2012]. Saatavana: http://msdn.microsoft.com/en-us/library/ms731082.aspx

Windows Communication Foundation Architecture Overview. 2010. [Verkkosivu]. MSDN. [Viitattu 26.3.2012]. Saatavana: http://msdn.microsoft.com/en-us/library/aa480210.aspx

Windows Cummunications Foundation Bindings. 25.6.2011. [Verkkosivu]. MSDN. [Viitattu 27.3.2012]. Saatavana: http://msdn.microsoft.com/en-us/library/ms733027.aspx

XAML. 21.6.2010. [Verkkosivu] Silverlight.net. [Viitattu 6.3.2012]. Saatavana: http://www.silverlight.net/learn/overview/what-is-silverlight/xaml-%28silverlight-quickstart%29