Opinnäytetyö (AMK)

Tietotekniikan koulutusohjelma

Sulautetut ohjelmistot

2010

Petteri Lehtonen

Rikkaiden internetsovellusten toteuttaminen Microsoft -teknologioilla

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikka | Sulautetut ohjelmistot

Huhtikuu 2010 | Sivumäärä 50

Ohjaaja: TkT Janne Roslöf

Petteri Lehtonen

Rikkaiden internet-sovellusten toteuttaminen Microsoft -teknologioilla

Tässä opinnäytetyössä tehtiin esiselvitystä myöhemmin alkavaan projektiin, jonka tarkoituksena on rakentaa internet-sovellus, johon käyttäjäkunta voi lisätä viittomakielen viittomia videomuodossa.

Toteutus tapahtuu käyttäen Microsoft www-tuotteita ratkaisuvaihtoehtoina, joista eniten huomiota sai Microsoft Silverlight. Silverlight on rikkaisiin internet-sovelluksiin tarkoitettu selain liitännäinen, jota kyseisessä videopainotteisessa sivustossa tullaan tarvitsemaan. Rikkaalla internet-sovelluksella tarkoitetaan internet-sivustoa, jolla on saman tyyppisiä ominaisuuksia kuin perinteisillä työpöytäsovelluksilla. Niiden tarkoituksena on lisätä käyttäjätyytyväisyyttä ja tuottavuutta.

Asiakkaan kanssa toteutettu prototyypitystyö tapahtui Microsoft Expression Blend +SketchFlow -tuotteella, jolla saatiin hyvinkin helposti ja nopeasti sivuston ulkoasua ja toimintoja suunniteltua. Prototyyppivaiheen jälkeen ryhdyttiin varsinaiseen toteutukseen, jonka ominaisuuksia työssä selvitetään.

Työssä valmistui 3-kerros arkkitehtuurin Silverlight sovellus käyttäen WCF RIA Services -alustaa. RIA services helpottaa kirjoittamaan liiketoiminta logiikan RIA-ympäristöön helposti ja tuottavasti. Työddä käydään läpi miten sovellus rakennetaan alusta lähtien työkaluilla, joita RIA services tarjoaa.

Tehty esiselvitystyö tulee toimimaan merkittävänä apuna projektin toteuttajalle. Toteutus tulee tapahtumaan lähivuosina, jossa esiselvityksen tuloksia voidaan hyödyntää.

ASIASANAT: video-ohjelmisto, Rich Internet Application, prototyypitys, www-sovellus

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Embedded Software

April 2010 | 50

Instructor(s): Janne Roslöf, D.Sc.

Petteri Lehtonen

Designing rich internet applications with Microsoft technologies

This thesis is a prestudy a research project that will start later on. Its objective was to build an internet application, where users could add sign language signs in the form of a video.

The project will be carried out using Microsoft Web products as the solution, where the most important technology is Silverlight. Silverlight is a browser plug-in meant for developing rich experiences, which this video heavy site will need.

Prototyping was done with the customer using Microsoft Expression Blend +Sketchflow product, which made it easy and quick to design the look and feel of the solution.

In this thesis a 3-tier Silverlight application was made using the WCF RIA Services platform. RIA Services makes writing business logic to a RIA environment easy and productive. I go through the process of building an application from start to finish using tools that RIA Services offers.

The finished solution will work as an important help for the future project implementation. The project will be implemented in a few years when the results of this research can be utilized.

KEYWORDS: Video solution, Rich internet Application, Prototyping, web application

SISÄLTÖ

SYMBOLIT- JA LYHENTEET

API	(Application Programming Interface) käyttöliittymä jolla eri ohjelmat voivat tehdä pyyntöjä ja vaihtaa tietoja eli keskustella keskenään
ATOMPub	(Atom Publishing protocol) XML-kieleen pohjaava standardi verkkosyötteille ja teidon syndikoinnille
ASP	(Active Server Pages) on web-palvelinympäristössä yleisesti käytetty dynaamisten web-sivujen luontiin käytetty ohjelmointiympäristö
BCL	(Base Class Library) .NET Frameworkin osa, joka sisältää luokkakirjastoja
CLR	(Common Language Runtime) CLR tarjoaa ns. virtuaalikoneen, joka kääntää kehitysympäristön tuottaman ohjelmakoodin
CSS	(Cascading Style Sheets) on erityisesti WWW-dokumenteille kehitetty tyyliohjeiden laji
HTML	(Hyper Text Markup Language) on yleisin web-sivujen luontiin käytetty kuvauskieli
JSON	(JavaScript Object Notation) on yksinkertainen tiedonsiirtomuoto, jota on helppo käyttää JavaScript-ohjelmissa. Nimestään ja JavaScript-perustastaan huolimatta JSON on JavaScriptistä riippumaton
LINQ	(Language Integrated Query) tiedon hakemiseen käytettävä kirjastokokoelma
OData	(Open Data Protocol) on avoin protokolla tiedon jakamiseen
RIA	(Rich internet Application) tarkoittaa internet-sovelluksia, jotka toimivat kuten perinteiset työpöytäsovellukset
WCF	(Windows Communication Foundation) WCF muodostaa viestijärjestelmän. WCF yhdenmukaistaa ja luo ohjelmille yhtenäisen käytännön välittää tietoa keskenään

XAML	(Extensible Application Markup Language) kuvauskieli, jolla Silverlight ja WPF ohjelmat toteutetaan
XML	(eXtensible Markup Language) on merkintäkieli tai standardi, jolla tiedon merkitys on kuvattavissa tiedon sekaan
.NET	on Microsoftin kehittämä ohjelmistokomponenttikirjasto, jota Microsoftin VisualStudio.NET-ympäristössä kehitetyt ohjelmistot käyttävät

1. Johdanto

Tässä työssä tehtiin esiselvitystä myöhemmin alkavaan projektiin. Esiselvitys tulee toimimaan apuna varsinaisen sovelluksen kehityksessä. Projekti on Turun ammattikorkeakoulun ja Diakonia-ammattikorkeakoulun yhteistyöhanke.

Työssä rakennetaan rikkaalla käyttölittymällä varustettu 3-kerros arkkitehtuurin Silverlight sovellus. Perinteisenä ongelmana rikkaiden internet-sovellusten toteuttamisessa on ollut kommunikointi www-palvelimen kanssa. Microsoft .NET RIA Services yksinkertaistaa tavallisten n-kerros sovellusiden mallin tuomalla yhteen ASP.NET ja Silverlight -alustat. RIA Services tarjoaa mallin kirjoittaa sovelluksen logiikan, joka toimii keskikerroksella ja kontrolloi pääsyä dataan suorittamaan kyselyitä, muutoksia ja mukautettuja operaatiota.

Ohjelmistojen rakentaminen n-kerros-maailmaan onkin monelle kehittäjälle melko häilyvä käsite. Monessa ympäristössä ollaankin totuttu rakentamaan ohjelmistoja 2-kerrosmaisesti, jolloin tietokantaan pääsy tapahtuu suoraan käyttöliittymästä. Tämä aiheuttaa monia ongelmia, jos ohjelmistoon tarvitsee tehdä muutoksia. Lisäksi Silverlight-toteutuksissa se ei ole ollenkaan mahdollista.

Ohjelman käyttäjälle 3-kerros rakenne näkyy käytettävyyden parantumisena. Kaikki kyselyt tietokantaan suoritetaan asynkronisesti, joka tarkoittaa sitä, että käyttöliittymä pysyy kokoajan käytettävänä. .NET-ympäristö tekee työstä helpompaa, sillä osattaessa kirjoittaa C#.NET-kieltä, osataan kirjoittaa koodia sekä asiakkaalle että palvelimelle.

Raportissa tutustutaan ensin teoriassa käytettyihin teknologioihin ja loppuosassa tarkastellaan niiden toimintaa käyännössä

2. Ohjelmiston tarkoitus

Tässä työssä tehtiin esiselvitystä myöhemmin alkavaan projektiin. Esiselvitys tulee toimimaan apuna varsinaisen sovelluksen kehityksessä. Projekti on Turun ammattikorkeakoulun ja Diakonia-ammattikorkeakoulun yhteistyöhanke. Hanke ja siinä rakennettava sovellus tulevat olemaan erittäin laajoja ja teknisesti haastavia, joten tämän kaltainen esiselvitystyö on kriittistä. Vaikka työssä onkin tarkoituksena ollut tehdä esiselvitystä, monet sen toiminnoista ovat valmiita ja ideat on tuotu hyvin esille. Esiselvityksen aikana on myös tullut paljon uusia ideoita, joita alkuperäisessä suunnitelmassa ei ollut. Toteutukseen liittyvät ideat, graafinen ulkoasu ja sivuston käytännön toimivuus on myös kokonaan esiselvityksessä ideoitua ja toteutettua. Ohjelmistoa on toteutettu Turun ammattikorkeakoulun insinööriopiskelijaryhmässä: minä ja Kimmo Koski olemme olleet ohjelmiston kehittäjiä, Tomi Härkönen on ollut järjestelmäasiantuntijana ja Johannes Maliranta on vastannut graafisesta ilmeestä. Laajassa sisältöpainoitteisessa sivustossa käyttöliittymän suunnittelu ja toiminta onkin tärkeässä osassa.

2.1 Ongelma

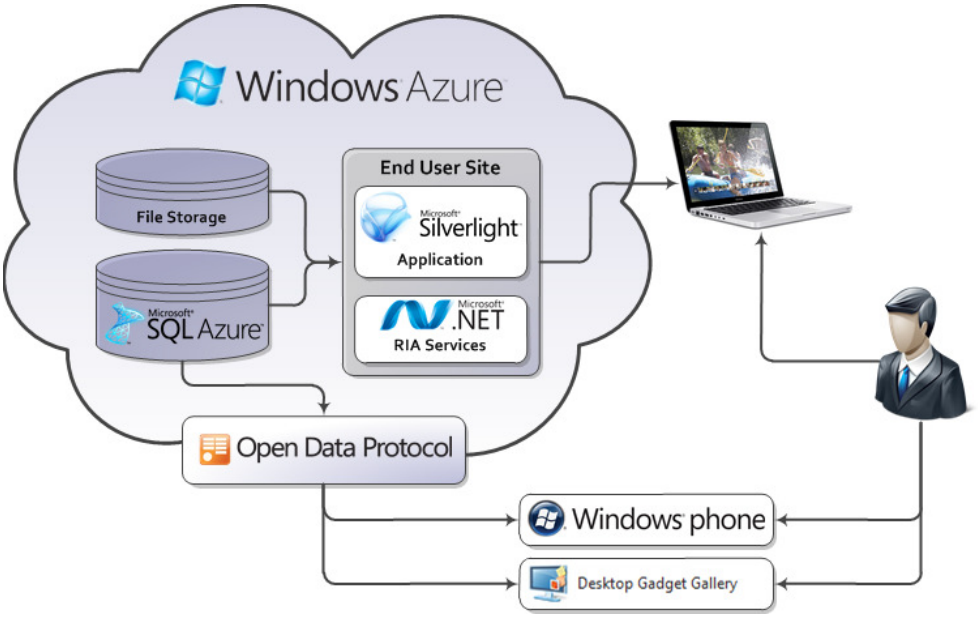
Useimmilla ohjelmistoilla on taustallaan olla jokin ongelma, jota se yrittää ratkaista. Sitä vartenhan ohjelmistoja tehdään. Ongelma, jota tässä työssä pyritään ratkaisemaan keskittyy pääosin viittomakielen opiskeluun. Suurin osa viittomankielen opiskelijoista aloittaa opintonsa nuorina aikuisina. Verrattuna lapsena tehtyyn kielen opiskeluun aikuisilla on vaikeuksia oppia viittomakielistä sanastoa ja muistaa se.

Viittomakielessä ei ole olemassa vakiintunutta kirjallista muotoa. Vaikka ei viittomankieltä osaisikaan, pystyy varmasti kuvittelemaan, miten vaikeaa on kirjoittaa paperille viittomia, joissa on kolmiulotteista liikettä, vartalon, kasvojen, pään ja huulion liikkeitä sekä muita ei-manuaalisia elementtejä.

internetissä on olemasa sanakirjoja, mutta ne ovat usein suljettuja, eivätkä tarjoa sisällön kommentointi-, arviointi- tai lisäysmahdollisuuksia. Suomessakin toimii esimerkkinä Suvi - Suomalaisen Viittomakielen verkkosanakirja. Tästä syystä eri ammateissa tai harrastuksissa käytettyä erikoissanastoa ei yleensä ole saatavilla ollenkaan. Siispä viittomankielen opiskelijoilta on tullut toive sivustoon, jossa tätä ongelmaa ei olisi

2.1 Ratkaisu

Työssä rakennetaan alusta viittomille, joissa käyttäjät pystyvät arvioimaan ja tuottamaan sen sisältöä. Tarkoituksena ei ole tehdä sivustosta sanakirjamaista, missä on yksi viittoma sanaa kohden ja kaikki lisätyt termit käyvät läpi tarkan tarkistusprosessin, vaan pitää se avoimena, joka kasvaa ja muuttuu koko ajan. Tämä kasvattaa viittomakielen tietoisuutta ihmisissä, jotka opiskelevat viittomankieltä ja ihmisissä, jotka jo työskentelevät sen kanssa. Varianttien esittäminen kasvattaa kielenopiskelijoiden ja –käyttäjien aktiivista sekä passiivista sanavarastoa. Ratkaisuun on tuotu mukaan myös sosiaalisen median elementtejä ihmisten vuorovaikutuksen ja sisällön laadunvarmistamiseksi. Esiselvityksessä tästä viittomakielen termipankista käytetään ohjelmistonimeä Signbook, mutta virallisesti nimeä ei vielä ole päätetty.



Kuva 2.1: Signbookin tekninen arkkitehtuuri.

Kuten kuvassa 2.1 näkyy Signbook tullaan toteuttamaan Windows Azure-pilvialustalle. Azure tarjoaa mahdollisuuden suorittaa sovelluksia, tietokantoja ja tiedon tallennusta virtuaalisilla pilvipalvelimilla. Azuren avulla on mahdollista rakentaa skaalautuvia sovelluksia, jolloin pystytään helposti reagoimaan esimerkiksi käyttäjämäärän nopeaan kasvuun.

Signbook-sovellus on kehitetty kokonaan Microsoftin Silverlightilla, koska se tarjoaa hyviä tapoja rikkaiden käyttöliittymien rakentamiseen. Silverlight toimii samalla tapaa

kaikissa selaimissa, mutta se tarvitsee erillisen selain-liitännäisen. Sovelluksen kaikki päätoiminnot liittyvät videoiden toistoon, joten Silverlight on luonnollinen valinta asiakasohjelmistoksi.

Silverlight-sovelluksen takana on SQL-tietokanta, jossa sijaitsee kaikki Signbookin sisältö. Signbook käyttää WCF RIA Servicesia tietokantayhteyksiin. RIA Services tarjoaa monia yksinkertaisettuja tapoja n-kerrossovelluksen rakentamiseen.

Silverlight 4 tarjoaa mahdollisuuden käyttää käyttäjän web-kameraa nauhoittamaan videot suoraan sovellukseen. Tämä on yksi suurimmista eduista, joita Silverlight tarjoaa, mikä tekee viittomien lisäyksen sovellukseen helpoksi ja interaktiiviseksi.

Signbookiin on sisällytetty myös Open Data Protocol, joka muuntaa www-sivuston web api:ksi. OData tarjoaa HTTP, Atom Publishing Protocol (AtomPub) and JSON -syötteen sivuston sisältöön, jota voidaan käyttää useassa eri laitteessa ja alustassa. Tämän avulla Signbookkiin on rakennettu Windows Desktop Gadget ja Windows Phone -sovellukset. Tässä opinnäytetyössä keskityn Signbookin Silverlight käyttöliittymään ja sitä isännöivään palvelinsovellukseen.

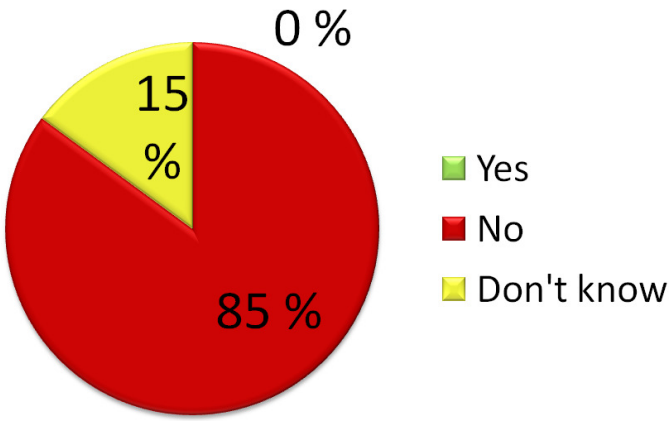
2.2 Yhteistyö

Kenelläkään kehitystiimiin kuuluvalla ei ole kuuroja ihmisiä kaveripiirissä tai perheissään. Jotta ymmärtäisimme ongelman ja pystyisimme rakentamaan sille ratkaisun on työn aikana tehty yhteistyötä Diakonia-ammattikorkeakoulun kanssa. Diakonia-ammattikorkeakoulu kouluttaa viittomakielentulkkeja, joten heille ongelma on todellinen ja arkipäiväinen. Yhteistyön tuloksena on saatu esiselvitykseen sisältöä, kuten viittomia ja tietoa ongelmasta ja viittomakielestä.

2.3 Taustatutkimus

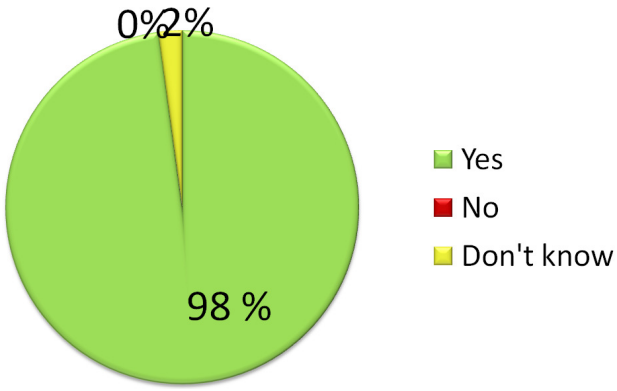
Työn aikana tehtiin myös tutkimusta, joissa kysyttiin peruskysymyksiä ongelmasta ja sen ratkaisusta. Kysymyksiin vastasi 47 viittomakielen opiskelijaa ja tulokset antavat melko selkeän kuvan ongelman olemassaolosta ja ratkaisun toimivuudesta. Seuraavissa osissa käydään tärkeimmät kysymykset ja niiden tulokset läpi. Diakin tekemän tutkimuksen mukaan opiskeluolot ovat samat muissa maissa, joten tuloksia voi soveltaa myös muualla.

Kysymyksessä nro 1 kysyttiin tyytyväisyyttä viittomakielen opiskelumetodeihin. Kuten kuvasta 2.2 voidaan todeta, ongelma on todellinen jo pelkästään sen perusteella, että kukaan ei vastannut olevansa tyytyväinen.



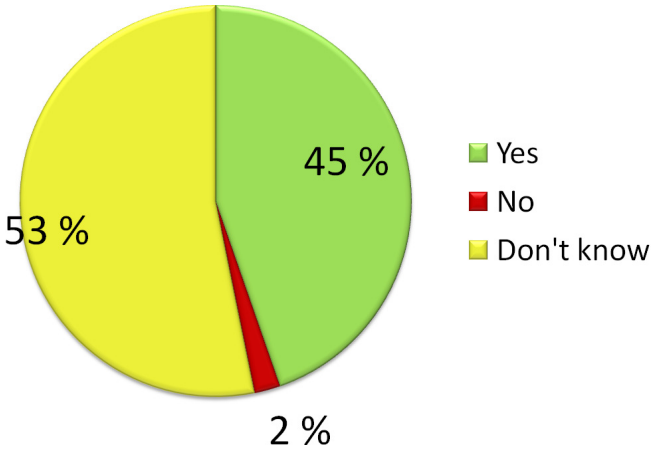
Kuva 2.2: Oletko tyytyväinen nykyisiin viittomakielen opiskelumetodeihin?

Kysymyksessä nro 2 kysyttiin viittomakielen opiskelua ja Signbookin apua siihen. Kuvasta 2.3 nähtävistä tuloksista voidaan todeta, että ratkaisu on toimiva ja käyttäjät olisivat siihen tyytyväisiä. Kysymykset olivat alunperin erillään, mutta tulokset olivat täysin identtiset, joten ne näkee myös samasta kaaviosta.



Kuva 2.3 Auttaisiko Signbook viittomakielen opiskelua ja käyttäisitkö sitä, jos se olisi saatavilla?

Viimeisessä kysymyksessä selvitettiin käyttäjien sisällöntuoton toimivuutta. Kuvasta 2.4 löytyvistä tuloksissa mielenkiintoista oli se, että lähes puolet lisäisi sisältöä ja vain 2% ei lisäisi. Tämän pohjalta voidaan olettaa, että termipankkiin saadaan runsaasti sisältöä.



Kuva 2.4 Lisäisitkö sisältöä Signbook:iin?

2.4 Imagine Cup

Esiselvityksessä tehdyllä työllä kehitystiimi osallistuu Microsoftin järjestämään vuosittaiseen Imagine Cup -kisaan. Imagine Cup on suurin opiskelijoille järjestettävä vuosittainen teknologiakilpailu maailmassa. Vuonna 2009 yli 350 000 opiskelijaa 100 eri maasta otti osaa kilpailuun. Suomessa vuoden 2009 kilpailuun rekisteröityi yli 250 opiskelijaa.

Tämän vuoden maailmanlaajuiset finaalit järjestetään heinäkuussa Puolan Varsovassa. Vuoden 2010 Imagine Cup teemana on “Imagine a world where technology helps solve the toughest problems facing us today."

Imagine Cupissa työ voitti Suomen finaalin ja pääsi jatkamaan maailmanlaajuisiin finaaleihin. Maailmanlaajuisia finaaleja ei vielä keväällä 2010 ole pidetty, mutta aiemmissa kisoissa menestyminen kertoo myös osaltaan työn laadusta.

3. Rikkaat käyttöliittymät

Rikkaalla internet -sovelluksella (Rich internet Application, RIA) tarkoitetaan www-sovellusta, jolla on ominaisuuksia, jotka ovat tyypillisiä työpöytäohjelmissa. RIA:t toteutetaan useimmiten selainliitännäisillä, jotka käyttäjän tulee asentaa omaan koneeseensa. Useimmiten näitä toteutuksia nähdään internetissä selainpeleissä, sekä videopalveluissa, kuten Youtube. Useimmat näistä on toteutettu Adoben Flashilla. Useimmilla RIA-tekniikoilla pystytään rakentamaan sovellusita sekä internetiin että työpöydälle

RIA:lla, pystytään toteuttamaan interaktiivisia ja käyttäjäystävällisiä ohjelmia, joihin tavalliset www-standardit, kuten HTML ja CSS eivät pysty. Hyötynä on käyttäjätyytyväisyyden ja tuottavuuden paraneminen.

3.1 Haasteet

Yhtenä suurimpana haasteena on selain-liitännäisten tarve. Ilman, että käyttäjä on asentanut koneelleen liitännäisen sovellukset eivät toimi ollenkaan. Silverlightissa tämä on vielä ongelma, mutta esimerkiksi Flash on asennettuna jo lähes kaikkiin koneisiin, joten voidaan olettaa sen löytyminen koneesta. Tämä tarve voidaan kiertää tekemällä omat versio sovelluksesta, joka on tarkoitettu koneisiin ilman liitännäistä. Tosin se vaatii paljon enemmän työtä yhden version tekemiseen verrattuna. Tämä haaste tulee erityisesti vastaan laitteissa, joihin liitännäisiä ei saa, kuten useimmissa mobiililaitteissa. Tässä tapauksessa pitää rakentaa erillisiä asiakassovelluksia mobiililaitteille, jos sellaisilla haluaa sovellusta käyttää.

3.2 Silverlight ja vaihtoehdot

Silverlight on alusta, joka tarjoaa puitteet rakentaa rikkaita, internet selaimesssa toimivia sovelluksia, jotka toimivat monessa eri käyttöjärjestelmässä. Silverlight toimii selaimen liitännäisen kautta. Kun tullaan Web-sivulle, joka sisältää Silverlight sisältöä, tämä selainliitännäinen suorittaa koodia, ja tekee tästä sisältöä sille erikseen tarkoitetulle alueelle sivulla. Silverlight tarjoaa paljon rikkaamman ympäristön kuin perinteinen sekoitus HTML:llää ja JavaScriptia tavallisilla web-sivuilla. Silverlight-sivulla voidaan käyttää interaktiivista grafiikkaa, käyttää vektorianimaatioita ja toistaa video ja äänitiedostoja.

Samat asiat on tehty ennenkin. Useat muut tekniikat käyttävät liitännäisiä venyttämään selaimen rajoja, kuten Java, ActiveX, Shockwave, ja (eniten levinnyt) Adobe Flash. Vaikka nämä kaikki vaihtoehdot ovat vielä käytössä, yksikään niistä ei ole tullut hallitsevaksi alustaksi rikkaiden www-sovellusten kehitykseen. Monet niistä kärsivät useista ongelmista, kuten huonosta asennuskokemuksista, huonoista kehitystyökaluista, ja riittämätömästä yhteensopivuudesta kaikkiin selaimiin ja käyttöjärjestelmiin. Ainoa teknologia, joka on pystynyt välttämään nämä karikot on Flash, joka tarjoaa erinomaisen usean alustan tuen ja laajan levikin. Kuitenkin Flash vasta hiljattain kehittynyt multimediasoittimesta joukoksi dynaamisia ohjelmointityökaluja, joten verrattuna moderniin .NET-ohjelmointiympäristöön sen tarjonta on vähäisempää.

Silverlightin tarkoitus on yhdistää suoritusteho ja monen alustan tuki Flashista, todella hyvällä ohjelmointi ympäristöllä, joka sisällyttää .NETin peruskäsitteet. Tällä hetkellä Flashilla on etulyöntiasema Silverlightiin verrattuna, johtuen sen laajasta käyttöönotosta ja kypsyydestä. Kuitenkin Silverlightissa sijaitsee muutamia arkkitehtuurisia piirteitä, joissa Flash ei pysty kilpailemaan. Tärkeimpänä se, että se perustuu .NET Common Language Runtime:n (CLR) supistettuun versioon ja kehittäjät voivat kirjoittaa asiakaspuolen koodin C #:lla. [4]

Flash

Silverlightista puhuttaessa tulee mieleen maailman suosituin selain liitännäinen Adobe Flash. Se on asennettuna yli 90 %:iin maailman internet-selaimista. Flashilla on todella pitkä historia, ja se on lähtöisin yksinkertaisesta animaation lisäämisestä Web maailmaan. On myös .NET www-kehittäjille todella yleistä käyttää Flashia sivustoillaan, mutta sen käyttäminen vaatii omat työkalunsa ja oman ohjelmointikielensä. Flashista on myös todella hankalaa kutsua palvelinpään koodia. Tässä tilanteessa Silverlight tarjoaa paremman vaihtoehdon, rikkaan nettisivuston kehittämiseen.

Taulukko 3.1. Selain liitännäisten levinneisyys käyttäjien selaimiin kirjoitushetkellä.[1]

Plug in	Flash	Silverlight	Sun Java
Asennettuna	97.26%	52.51 %	72.7% %
Ei löytynyt	2.74%	47.49 %	27.30 %

HTML 5

HTML5 on tuleva standardi, jolla voidaan tulevaisuudesa tehdä mediarikkaita sovelluksia. HTML5 ja CSS3 yhdistettynä AJAXiin voidaan tehdä myös hyvin rikkaita käyttökokemuksia. AJAX on teknologia, jolla voidaan suorittaa asynkronisesti palvelinpuolen koodia, joka tarkoittaa, että käyttöliittymä pysyy kokoajan käytettävissä. HTML5:lla ei vielä pysty tekemään kaikkea, joka erillisillä liitännäisillä voidaan tehdä, mutta vähentää niiden tarvetta esimerkiksi videon toistossa. Niin kuin kaikissa HTML ratkaisuissa tässäkin tullaan hyvin paljon toimimaan selainten ja standardien ehdoilla. Eri selaimet toistavat HTML:ää eri tavoin, joka saa koodin tekemään eri asioita eri selaimissa. Silverlight toimii samaan tapaan kaikissa selaimissa ja ympäristöissä, mikä helpottaa suuresti kehitystä ja testausta.

4. Silverlight

4.1 Yleiskatsaus

Silverlight tarjoaa säilytetyssä tilassa toimivan grafiikkajärjestelmän, kuten Windows Presentation Foundation. Silverlight integroi multimediaa, grafiikkaa, animaatioita ja vuorovaikutteisuusuutta yhdeksi ajonaikaiseksi ympäristöksi. Silverlightissa, käyttöliittymät ovat Extensible Application Markup Language (XAML) kuvauskieltä ja ne ohjelmoidaan käyttäen .NET alustaa. XAML:a voidaan käyttää kuvaamaan vektorigrafiikkaa ja animaatioita. Silverlightia voidaan käyttää myös luomaan Windowsin gadgetteja.

Silverlight tukee Windows Media Video (WMV), Windows Media Audio (WMA) ja MPEG Layer III (MP3) mediasisältöä kaikissa tuetuissa selaimissa ilman Windows Media Playeria, Windows Media Playerin ActiveX-komponentteja tai Windows Media-selainliitännäisiä. Koska Windows Media Video 9 on Society of Motion Picture and Television Engineers (SMPTE) VC-1-standardin toteutus Silverlight tukee VC-1 videota, mutta vain Advanced Systems Format (ASF) muodossa. Lisäksi ohjelmiston lisenssisopimus sanoo, että VC-1:n käyttöön on lupa vain kuluttajien henkilökohtaiseen ja ei-kaupalliseen käyttöön. Versio 3:sta lähtien Silverlight tukee H.264-videon toistoa. Silverlightiin voidaan dynaamisesti ladata Extensible Markup Language (XML) -sisältöä, jota voidaan manipuloida Document Object Model (DOM)-liitännän kautta,

tämä tekniikka vastaa perinteisen Ajaxin tekniikkaa. Silverlight paljastaa Downloader-olion, jota voidaan käyttää lataamaan sisältöä, kuten komentokieltä, media-assetteja tai muuta dataa, joita saatetaan tarvita sovellusssa. Versio 2:sta lähtien ohjelmalogiikka voidaan kirjoittaa millä tahansa. NET kielellä, mutta myös jollain johdannaisilla dynaamisista ohjelmointi kielistä, kuten IronRuby ja IronPython. [2]

Silverlight-sivustot tai -sivustojen alueet eivät ole hakukoneiden haettavissa ja indeksoitavissa, joten kehittäjien täytyy luoda ja ylläpitää päällekkäistä HTML-sisältöä, joka on lähetettävä hakukoneiden roboteille, jos sivuston sisältö halutaan indeksoida. Sama HTML lähetetään myös käyttäjille, jos Silverlight ei ole asennettuna. Tämä johtuu siitä, Flash Headless Player for Search Enginestä, jonka avulla hakukoneet, kuten Google ja Yahoo indeksoivat Flash-sisältöä, ei ole Silverlight-versiota. Silverlightin hakukoneoptimoinnin (SEO) -tuen puute selittää, miksi sitä yleisesti pidetään työkaluna kehittämään extranetsovelluksia, julkisten Web-sivustojen sijaan. Poikkeuksia tähän on kuitenkin olemassa video-sivustoissa, kuten Netflix, viime olympialaiset, ja Sunday Night Football, jossa Silverlightin vahvuudet ovat voittaneet SEO:n puutteet.

4.2 Versiot

Silverlight 1

Silverlight 1.0, jota kehitettiin koodinimellä Windows Presentation Foundation / Everywhere (WPF/E), koostuu ydin alustasta, joka vastaa käyttöliittymästä, interaktiivisuudesta, perus-UI kontrolleista (grafiikat ja animaatiot , median toisto, DRM ja DOM integraatio). Se koostuu seuraavista osista:

- Input käsittelee laitteiden kuten näppäimistön, hiiren, kynän jne. toimintaa
- UI core hallitsee bitmap, vektorigrafiikka, teksti ja animaatio renderöintiä
- Media hallitsee MP3, WMA Standard, WMV7, WMV8 ja WMV9/VC-1 toistoa
- XAML-käyttöliittymä voidaan luoda käyttämällä XAML-kuvauskieltä.

Silverlight-sovellus käynnistyy viittaamalla Silverlight komponenttiin HTML sivusta, joka lataa sitten XAML tiedoston. XAML tiedosto sisältää Canvas -objektin, joka toimii paikkamerkkinä muille elementeille. Silverlight tarjoaa erilaisia geometrisia elementtejä kuten viivoja, ellipsejä ja muita kuvioita, sekä elementtejä, kuten tekstiä, kuvia, ja mediaa, jne. Nämä elementit voidaan animoida käyttäen Event triggereitä. Tapahtumat

kuten näppäimistön painallus tai hiiren liike voi myös tehdä eventtejä, joihin voidaan reagoida mukautettuja komentosarjojen avulla.

Ohjelmallinen käyttöliittymän manipulointi saavutetaan käyttämällä komentosarjakieliä, joilla voidaan muuttaa Silverlight Canvas olion Document Object Modellia. Tämän helpottamiseksi Silverlight paljastaa DOM Application Programming Interfacen (API), jota voi käyttää millä tahansa Silverlighton tukemalla skriptikielellä. Silverlight versio 1.0 on rajoitettu JavaScriptiin, jota käytetään internet-selaimessa.

Silverlight 2

Silverlight 2 sisältää version. NET Frameworkista, käyttäen samaa koko Common Language Runtime (CLR) versiota, kuin .NET Framework 3.0, jotta se voi suorittaa ohjelmia millä tahansa .NET kielellä. Kuten .NET Framework 4.0 CLR:ssa, useita CoreCLR:n Silverlight-esiintymiä voi sijaita samassa prosessissa. Tämän avulla XAML layout -tiedosto voidaan täydentää code-behind koodilla. Tämä taustakoodi on kirjoitetty .NET kielellä, jossa on sisällä ohjelman logiikka. Sitä voidaan käyttää ohjelmallisesti manipuloimaan sekä Silverlight -sisältöä, että HTML-sivua, joka isännöi Silverlightia. XAML kuvauskieli sekä koodi käännetään. NET-assemblyihin, jotka sitten pakataan ZIP-menetelmällä ja tallennetaan .xap-tiedostoon.

Silverlightin mukana tulee kevyen luokan kirjasto, joka sisältää ominaisuuksia, kuten laajennettavissa olevia kontrolleja, XML Web Servicesiä, verkkokomponentteja ja Language Integrated Query (LINQ) API:n. Tämä luokkakirjasto on .NET Framework Base Class Libraryn (BCL).osa ja on huomattavasti pienempi kuin sen kokoversio.

Silverlightin .NET Framework versioon on lisätty osia Windows Presentation Foundation (WPF) käyttölittymäohjelmointi -mallista, mukaan lukien tuki geometrisille muodoille, asiakirjoille, medialle ja animaatio objekteille. Beta 2 versiosta lähtien Silverlightin mukana tulee yli 30 käyttöliittymäkontrollia, kuten TextBox, CheckBox, Slider, ScrollViewer ja Calendar. Silverlightin kontrolleja voidaan käyttää kaksisuuntaiseen tiedon sitomiseen, automaattiseen ulkoasun hallintaan (StackPanel, Gridin, jne. avulla), sekä datankäsittelyyn (DataGrid ja ListBox). Käyttöliittymäkontrollit ovat räätälöitävissä käyttämällä template -mallia. Kolmannen osapuolen kirjastoja laajentamaan käyttöliittymäkontrollien valikoimaa on myös saatavilla.

Silverlightin BCL:n mukana tulee luokkia collectionien, reflectionien, regular expressionien, stringien käsittely ja data accessiin. Se tukee myös LINQ:a, ja siinä on

täysi tuki LINQ to Objectsiin ja Expression trees rakenteisiin. Silverlight voi käsitellä tietoja Really Simple Syndication (RSS) tai JSON-muodossa, XML:n lisäksi. BCL tarjoaa tehostetun tuen XML-tietojen kanssa työskentelyys, myös XMLReader ja XMLWriter -luokissa.

Silverlightissa on myös luokkia tietojen pääsyyn XML-pohjaisten Web-palvelujen yli, kuten Representational State Transfer (REST), Windows Communication Foundation (WCF) Services ja ADO.NET Data Services. Silverlight käyttää XML-pohjaista konfiguraatio tiedostoa kontrolloimaan verkkoyhteyksiä sekä HTTP että socket yhteyksillä. Sivuston ylläpitäjät voivat käyttää sitä kontrolloimaan, mitä resursseja Silverlight-sovellus voi käyttää, kun tämä sovellus ole ole peräisin samasta verkkotoimialueesta

Media Stream Source API vastaa mukautuvan median virtaustoistosta. Adaptive streamingin avulla toistosovellus voi valita bittinopeuden, joka perustuu asiakkaan käytettävissä olevaan kaistanleveyteen ja prosessorin nopeuteen. Media Stream Sourcen avulla kehittäjä voi määrittää mukautetun menetelmän mediadatan hakuun. Ainoa vaatimus on, että lopullinen video ja ääniraita esitetään Silverlight sovellusIle sellaisessa muodossa, jonka Silverlight voi purkaa (VC-1, H.264, WMA, MP3, jne.).

Silerlight 2 antaa myös rajoitetun mahdollisuuden käyttää tiedostojärjestelmää Silverlight-sovelluksissa. Se voi käyttää käyttöjärjestelmän natiivia tiedosto-valintaikkunaa selaamaan, mihin tahansa tiedostoon johon on käyttöoikeus. Tiedostosta esitetään siistitty polkuinformaatio, jotta sovellus ei saa tietoja, kuten käyttäjän nimeä, ja ne voidaan avata ainoastaan vain luku-tilassa. Paikalliseen tallentamiseen Silverlight tarjoaa Isolated Storagen eli selaimen välimuistinulkopuolella yksityisen käyttäjän profiilissa sijaitseva piilotettu kansion. Se on 1 Mt / URL oletuksena, mutta käyttäjä voi muuttaa tämän. Silverlight sovelluksen tallentama data identifioidaan URL:n mukaan, josta se ladataan, ja sinne pääsee vain sama sovellus. Kaikki instanssit Silverlightista käyttävät samaa Isolated Storagea, joten kaikki esiintymät samasta Silverlight sovelluksesta voivat jakaa tallennetut tiedot, vaikka ne olisivat käynnissä eri selaimissa.

Silverlight 3

Silverlight 3 sisältää aiempaa enemmän kontrolleja. Osa näistä kontrolleista on lähtien Silverlight Toolkitista. Lisäksi Silverlight 3 sisältää navigoinnin, joka antaa

mahdollisuuden linkkityyliseen navigointiin, sekä mahdollistaa syvälinkityksen (yhdistämisen suoraan tietylle sivulle) Silverlight sovelluksen sisällä.

Mediapuolella, Silverlight 3 tukee Advanced Audio Coding (AAC) äänen purkua sekä rautakiihdytettyä H.264 videon purkua. Silverlight 3 tarjoaa myös 1080p smooth streamingin. Silverlight 3 tukee perspektiivistä 3D:tä, joka mahdollistaa 2D osien 3D transformaation. Nämä transformaatiot, samoin kuin monet 2D operaatiot kuten venytys, alpha-sekoitus jne. ovat rautakiihdytettyjä. Kustomoidut animaatiot, kuten transformaatiot ja blendaukset, voidaan luoda Silverlightin elementteihin käyttämällä High Level Shader Language (HLSL):a käyttäen pikselivarjostimia. Silverlight 3 tukee myös ClearType tekstin renderointiä.

Käyttöliittymäelementit Silverlight 3:ssa tukevat elementistä--elementtiin sitoutusta, joka mahdollistaa yhden elementin tilan sitomisen toiseen elementtiin, sekä validointi -mekanismina tietojen sitomista. Silverlight 3 sovellukset voivat tallentaa tiedostoja minne tahansa Save File dialogin kautta. Kuitenkin polku, johon tiedosto tallennetaan on edelleen piilossa Silverlight sovelluksille.

Silverlight 3 sisältää myös LocalConnection API:n, jonka avulla pystyy keskustelemaan muille samalla koneella käynnissä oleville sovelluksille riippumatta selaimesta. Se voi myös valvoa verkkoyhteyden tapahtumia. Silverlight 3 voi vaihtoehtoisesti käyttää Binary XML:llää kommunikoimaan WCF palvelujen kanssa.

Silverlight 3 tukee Out-of-browser -kokemuksia, eli Silverlight sovelluksia voidaan asentaa järjestelmän offline-käyttöön (jos sovellus ilmeisesti on suunniteltu siten, että se tukee paikallista asennusta), jossa ne toimivat selaimen ulkopuolella. Ne käynnistetään käynnistä-valikon avulla tai työpöydän pikakuvakkeesta, ja toimivat ilman selainikkunaan. Ohjelmat voivat tarkistaa, ovatko ne käynnissä selaimen sisällä tai ei. Paikallisesti asennettu Silverlight-sovellus ajetaan edelleen hiekkalaatikossa.

Silverlight 4

Termipankin sivuston tärkein ominaisuus lienee sen ominaisuus videoiden tallennukseen suoraan selaimesta. Silverlight 3 versiossa ei ole minkäänlaista tukea webcamiin, joten tässä tapauksessa on käytettävä Silverlightin 4 versiota. Silverlight 4 on vielä beta -vaiheessa, eikä sen virallista julkaisupäivää ole kerrottu. Visual Studion seuraava versio 2010 tosin julkaistaan 13.4.2010, joten Silverlightin uutta versiota voitaneen odottaa hieman sen jälkeen. Silverlight 4 tuo paljon toivottuja ominaisuuksia

ja sillä voidaan jo rakentaa paljon kattavampia liiketoimintasovelluksia kuin aikaisemmilla versioilla. Muutamia ominaisuuksia, joita 4 versio tuo mukanaan ovat

- tuki Googlen Chrome-selaimelle
- verkkokamera ja mikrofoni tuki
- Tulostus-tuki
- Parempi hiiri-tuki, kuten oikea nappi ja hiiren rulla
- Leikepöytä ja drag & drop -tuki
- Multi-touch
- WCF Rich internet Application (RIA) Services. [3, s. 9–14].

4.3 Yhteensopivuus

Kuten millä tahansa www-keskeisellä teknologialla, on tärkeää saada yhteensopivuus laajaan valikoimaan tietokoneita ja laitteita. Vaikka Silverlight kehittyy edelleen, sen kehitys pyrkii selvästi on tukemaan kaikkia tärkeimpiä selaimia Mac OS X ja Windows tietokoneissa. Tällä hetkellä Silverlight 3:n yhteensopivuus on seuraavat:

- Windows-tietokoneissa: Silverlight toimii Windows 7, Windows Vista ja Windows XP:ssä. Vähimmäis selainversiot, joilla Silverlight 3 toimii on internet Explorer 6, Firefox 1.5, Safari ja Google Chrome. Silverlight toimii myös Windows 2000:lla, mutta vain internet Explorer 6:lla. Opera toimii myös, mutta sitä ei ole virallisesti tuettu. Muilla selaimilla Silverlight 3 ei ole tällä hetkellä tuettu.
- Mac-tietokoneet: Silverlight toimii Mac-tietokoneiden kanssa OS X:n versiolla 10.4.8 tai uudempi, jos ne ovat Intel laitteita (toisin kuin vanhemmat PowerPC koneet). Vähimmäisselainversiot, jota Silverlight 2 tukee on Firefox 2 ja Safari 3.
- Linux-tietokoneet: Vaikka Silverlight ei tällä hetkellä toimi Linuxissa, Mono-tiimi luo avoimen lähdekoodin Linux toteutusta Silverlight 1:stä ja Silverlight 2:sta. Tämä hanke on nimeltään Moonlight, ja sitä kehitetään Microsoftin tukemana. Lisätietoja monosta löytyy verkosta osoitteesta http://www.mono-project.com/Moonlight. [4, s. 14-15].

4.4 Kehitystyökalut

4.4.1 .NET

Microsoft .NET on ohjelmistokomponenttikirjasto, joka voidaan asentaa Windows käyttöjärjestelmäisiin tietokoneisiin. Sillä voidaan kehittää ohjelmia verkkoon, laitteisiin, palveluihin ja moneen muuhun. Tämän hetkinen vakaa versio on 3.5 SP1. Uusimpien Windows käyttöjärjestelmien mukana tulee versio 3.0. NET:in tehokkuus ja helppokäyttöisyys perustuvat sen suuriin valmiisiin kirjastoihin, joilla pystyy helposti toteuttamaan yleisiä ohjelmoinnin ongelmia. Lähes kaikki Windows ohjelmistokehitys tehdään sitä käyttämällä, minkä ansiosta kaikki työkalut sopivat hyvin yhteen ja niillä saa vähemmällä työllä aikaan oikean lopputuloksen.

.NET koostuu seuraavista osista:

- Common Language Runtime – käyttöjärjestelmässä toimiva ajoympäristö, koka kääntää koodin tietokoneen ymmärtämään muotoon
- Base Class Libraries – koodikirjastoja, joilla saa monia yleisiä ohjelmointiongelmia ratkaistua
- Development frameworks and technologies – laajempiin ongelmiin tarkoitettuja kokonaisuuksia. [9]

Työssä käytettävää Silverlightia voidaan kutsua .NETin pienoisversioksi, joka on tarkoitettu web-ohjelmointiin. Silverlightista onkin jouduttu jättämään suuri osa ominaisuuksista pois, koska on pyritty mahdollisimman pieneen asennuspakettiin. [5]

4.4.2 C#

C# on Microsoftin kehittämä ohjelmointikieli, joka muistuttaa sekä syntaksiltaan että kirjastoiltaan hyvin paljon Javaa. Se on nykyaikainen kieli, jolla on paljon vahvuuksia. Se on moderni ja monikäyttöinen olio-ohjelmointikieli. Nykypäivänä kun tutustutaan Windows-ohjelmointiin ei voi olla törmäämättä C# -kieleen. Yksi sen hyvistä puolista onkin suuri ja aktiivinen käyttäjäkunta, joilta löytyy apua ongelmaan jos toiseenkin. Kehittäjät, jotka ovat tottuneet Javaan, pystyvät helposti siirtymään siihen ja ymmärtämään sillä kirjoitettua koodia.

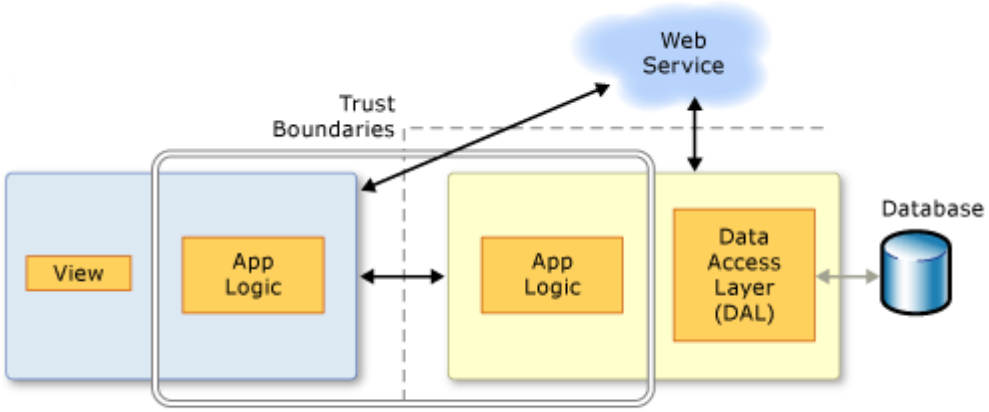
4.4.3 Visual Studio

Visual Studio on Microsoftin ohjelmankehitysympäristö. Sitä voidaan käyttää monien alustojen sovellusten kehittämiseen, konsolisovellusista nettisovelluksiin ja natiivista managed ympäristöihin. Visual Studion koodieditori on ehdottomasti markkinoiden huippua ja sillä ohjelmien työstäminen on todella tehokasta. Visual Studion Intellisense tarjoaa näppärän tavan nopeuttaa koodin kirjoittamista ennustamalla haluttua tekstiä. Intellisense myös karsii pois kirjoitusvirheitä. Sen WYSIWYG (What You See is What You Get), editorin avulla on helppo luoda graafisia käyttöliittymiä, mikä osin varmasti selittääkin suurelta osin sen suosiota. [6]

5. WCF RIA Services

RIA Services helpottaa n-kerrosarkkitehtuuristen RIA (Rich internet Application) ohjelmien, kuten Silverlight-toteutusta. Yleinen ongelma n-kerros arkkitehtuurien ohjelmissa on sovelluslogiikan koordinointi keskikerroksen ja käyttöliittymäkerroksen kanssa. Halutessamme parhaan mahdollisen käyttökokemuksen, käyttöliittymän tulee tietää sovelluksenlogiikka, joka sijaitsee palvelimella. Sovellusta tehtäessä ei kuitenkaan haluta kirjoittaa sovellusenlogiikkaa sekä käyttöliittymäkerrokselle että keskikerrokselle. RIA Services ratkaisee tämän ongelman tarjoamalla valmiita komponentteja, työkaluja ja palveluja, jotka välittävät sovelluksen logiikan käyttöliittymälle. Tämän avulla voidaan luoda käyttöliittymän, joka päivittyy viimeisimmällä keskikerroksen logiikalla, aina kun se käännetään.

Kuva 5.1 näyttää yksinkertaisen version n-kerrosohjelmasta. RIA services keskittyy laatikkoon, joka sijaitsee käyttöliittymäkerroksen ja DAL (Data Access Layer) välissä, josta se helpottaa rikkaan käyttöliittymän kehitystä



Kuva 5.1 RIA Services -kaavio.

RIA Services lisää Visual Studioon työkalut, joilla pystytään linkittämään asiakas- ja palvelinprojektit yhteen kokonaisuuteen ja generoimaan koodia asiakkaalle, joka heijastuu keskikerroksen koodista. Alustan komponentit tukevat preskriptiivistisiä tapoja ohjelman logiikan kirjoittamiseen, jotta niitä voidaan uudelleenkäyttää käyttöliittymäkerroksella. Preskriptiivistiset tavat tarkoittavat sitä, että asiakkaan koodia kirjoittaessa alusta pystyy ehdottamaan oikeita ilmaisuja. Usein tarvittuja palveluita, kuten autentikointi ja käyttäjätietojen hallinta, ovat sisällytettynä, mikä lyhentää kehitykseen tarvittavaa aikaa. [7]

WCF RIA Services on saatavilla RIA Services lataussivulta osoitteesta http://silverlight.net/getstarted/riaservices/ ja siitä on 2 versiota saatavilla:

- WCF RIA Services Beta for Visual Studio 2008 SP1
- WCF RIA Services Preview for Visual Studio 2010

Molemmat versiot ovat ilmaisia ja niiden kehitystä voidaan tehdä ilmaisilla työkaluilla, jotka tulevat pysymään ilmaisina myös tulevaisuudessa. Versio tulee tosin vaihtumaan tulevaisuudessa Preview-versiosta valmiiseen tuotantoversioon.[8]

5.1 RIA Services -toiminta

Yrityssovelluksia luotaessa WCF RIA-palveluiden avulla voidaan toteuttaa ratkaisuja, jotka käsittelevät erilaisia skenaarioita. Esimerkiksi yksinkertainen skenaario voi olla yksittäinen Silverlight-sovellus, joka käyttää vain muutamaa Domain Serviceä keskitasolla. Monimutkaisessa skenaariossa voi olla useita Silverlight-sovelluksia, jotka linkittyvät yhteiseen keskikerrokseen, joka näyttää monta Domain Serviceä.

Kaikissa RIA Services ratkaisuissa on linkki keskimmäisen kerroksen ja käyttöliittymäkerroksen välissä. RIA Services -linkki on erityinen projektista projektiin referenssi, joka helpottaa generoimaan käyttöliittymätason koodin keskimmäisen tason koodissa.

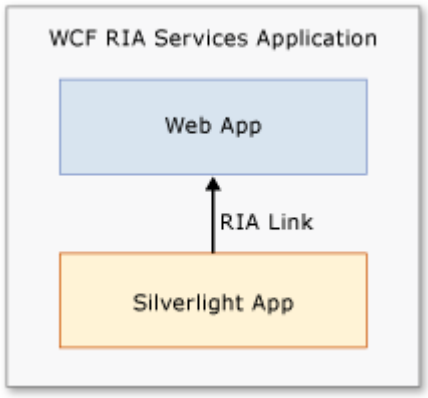
Kun projektien välillä on linkki, käyttöliittymäkerrosprojekti saa kaikki keskimmäisen kerroksen koodit. Ei siis pystytä määrittämään, että vain tietyt osat keskitason koodista koskevat käyttöliittymäkerroksen projektia. Seuraavia sääntöjä sovelletaan RIA Services linkkien käytössä:

- Linkki on määritetty Silverlight-projektissa.
- Linkki osoittaa aina Silverlight-projektista .NET palvelimen projektiin tai luokkakirjastoon.
- Voi olla vain yksi linkki Silverlight-projektia kohti.
- Linkkiä ei voi osoittaa muihin Silverlight-projekteihin.
- Useita Silverlight-projekteja voi viitata samaan palvelimen projektiin tai luokkakirjastoon.
- Silverlight-projektia ei voi linkittää suoraan luokkakirjasto-projektiin.

5.2. Ohjelmistorakenne

Oletusohjelmistorakenteessa RIA Services luo yhden asiakasprojektin ja yhden palvelinprojektin. Kun luodaan uusi projekti Silverlight Application-templatella ja valitaan "Enable WCF RIA Services" valintaruutu, luodaan ohjelmisto oletusrakenteen kanssa. RIA Services-linkki on Silverlight ja palvelin -projektien välillä. Kun ohjelmisto

käännetään, asiakkaan koodi generoidaan Domain Serviceille ja Shared Codelle. Kuvassa 5.3 esitetään oletusrakenne:



Kuva 5.2: WCF RIA Services oletusrakenne

Oletusrakenne on kätevä, koska kaikki Domain Servicet ja Shared Codet, jotka on lisätty palvelinprojektiin ovat automaattisesti käytettävissä Silverlight projektissa sen jälkeen, kun ohjelmisto käännetään. Tämä rakenne toimii hyvin, kun ei ole monta Domain Serviceä palvelinprojektissa eikä tarvitse käyttää samaa liikentoimintalogiikkaa useissa eri Silverlight-sovelluksissa.

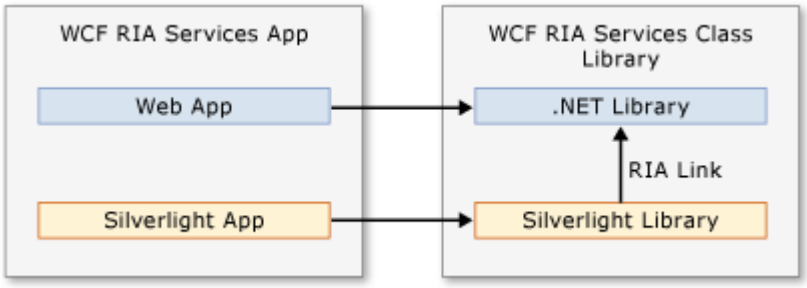
Oletusrakenteen kanssa voi lisätä enemmän Silverlight-sovelluksia, joissa on RIA Services linkki ohjelmiston palvelinprojektiin. Oletusrakenteessa on kuitenkin rajoituksia. Jokaiselle Silverlight-projektille generoitu koodi sisältää kaikki palvelinprojektin keskimmäisen kerroksen koodit. Esimerkiksi jos kolme Silverlight-sovellusta liittyvät yhteen palvelinprojektiin ja haluataan lisätä Domain Service, jota käyttää vain yksi Silverlight-sovellus, kaikki kolme Silverlight sovellusta saavat palvelimelta generoidun Domain Servicen ja voivat käyttää sitä.

RIA Services tarjoaa myös Silverlight Business Application templaten. Tämä malli on kätevä lähtökohta yrityssovelluksessa, joka hyödyntää Silverlightia käyttöliittymän rakentamiseen. Malli perustuu Silverlight Navigation templateen ja käyttää RIA Servicesiä tukemaan autentikointia ja käyttäjän rekisteröintiä. Kun luodaan uusi ohjelmisto Silverlight Business Application-templaten avulla, RIA Services luo projektin oletusrakenteen kanssa. Silverlight Business Application lisää automaattisesti seuraavat ominaisuudet:

- Login-ikkuna

- Rekisteröinti-ikkuna
- Silverlight-navigointi.

RIA-services tarjoaa WCF RIA Services Class Library -projektityypin tukemaan koodin jakamista kirjastojen avulla. Class Libraryjen avulla pakataan liiketoimintalogiikka uudelleenkäytettäviin n -kerros Class Library komponentteihin. Kuvassa 5.3 on rakenne, joka käyttää RIA Services luokkakirjastoja.



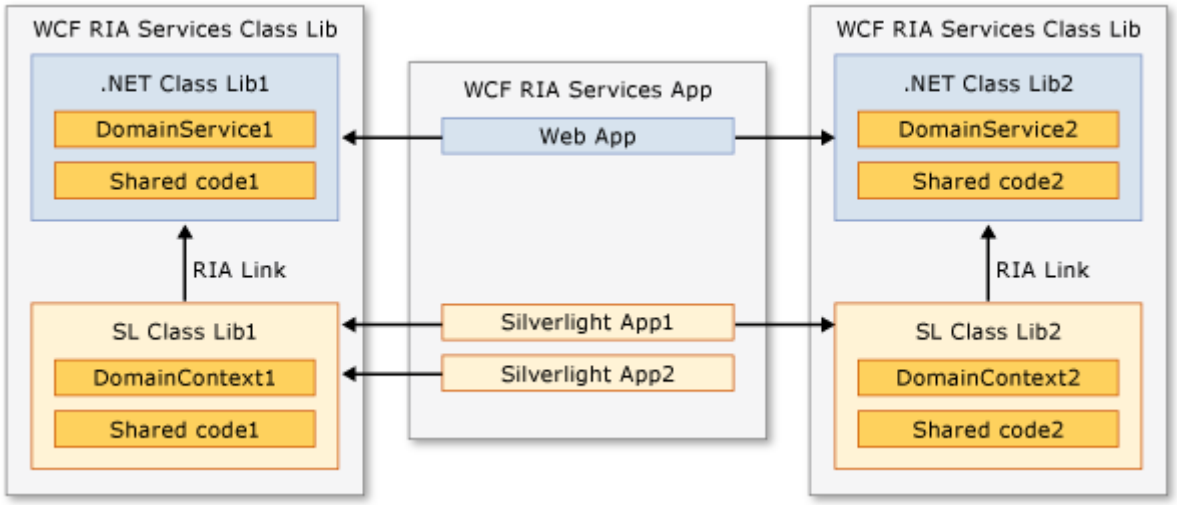
Kuva 5.3: WCF RIA Services Class Library

Huomaa, että RIA Services-linkki ei ole sovelluksen projektien välillä. Sen sijaan luokan luokkakirjastojen välillä on linkki. Luokkakirjastojen määrää ei ole rajoitettu, ja niitä voidaan käyttää uudelleen myös rajoittamattomassa määrässä sovelluksia.

RIA Services -luokkakirjastojen käyttäminen tarjoaa seuraavat edut:

- Palvelimen ja asiakkaan osat samasta data domainista voidaan kehittää ja pakata yhteen komponenttiin. Tämä komponentti voidaan käyttää uudelleen useassa sovelluksessa.
- Lähdekoodin jakaminen tapahtuu vain yhdessä paikassa. Sijainti on komponentin kerrosten välissä, eikä joka Silverlight sovellusssa.
- Useat Silverlight-sovellukset yhdessä Web-sovellusprojektissa voi viitata vain tarvitsemaansa luokka kirjastoihin. Jokainen Silverlight asiakas ei näe koko keskikerroksen liiketoimintalogiikkaa.

RIA Services class libraryilla voidaan luoda joustava rakenne, joka sisältää vain osat, jotka tarvitaan sovelluksessa. Seuraavassa kuvassa on rakenne, joka käyttää useita RIA Services class libraryja. [9 s.18-20]



Kuva 5.4: Rakenne, joka käyttää useita luokkakirjastoja

5.3. Keskimmäinen kerros

Kolmitasoisessa sovelluksessa keskimmäinen taso sisältää logiikan, joka hallitsee vuorovaikutusta käyttöliittymätason ja datatason välillä. Sinne asetetaan liiketoimintasääntöjä ja vahvistuksia, jotka varmistavat, että tiedot ovat oikeassa muodossa. Esimerkiksi henkilöstönhallintasovelluksessa tehdään käyttöliittymä, jonka avulla työntekijät voivat jättää loma-anomuksia. Ennen kuin lomat myönnetään pitää kuitenkin varmistaa, että työntekijän lomasaldo ei ole koskaan pienempi kuin nolla. Keskikerrokselle voidaan lisätä logiikka tarkistamaan työntekijän lomasaldo ennen kuin loma myönnetään.

Kun halutaan luoda parhaan mahdollisen käyttökokemuksen RIA-asiakasohjelma, kuten Silverlight-sovellus, usein halutaan samat liiketoimintasäännöt niin käyttöliittymään kuin palvelimellekin. Tämän vuoksi on välttämätöntä, että keskitason koodi synkronoidaan asiakkaan ja palvelimen välillä. RIA Servicesin avulla voidaan käyttää .NET Frameworkkia kirjoitettaessa sovelluksen keskitason logiikkaa. RIA Services generoi käyttöliittymän koodin keskitason koodista siten, että nämä tasot pysyvät aina synkronoituna.

Data Access Layer

RIA servicesin kanssa voi käyttää mitä tahansa Data Access Layeria. Sen voi yhdistää esimerkiksi seuraaviin:

- Entity Data Model, jota käytetään Signbookissa

- LINQ to SQL object model (vain jos RIA Services Toolkit on asennettuna).
- Common Language Runtime (CLR) object.
- Web service, joka tuottaa dataa lähteestään.

Data layeriin voi asettaa validointisääntöjä, jotka vahvistavat rajoituksia arvoille, joita käyttöliittymästä lähetetään.

Joissakin tapauksissa on käytössä enemmän kuin yhden taulun tietoja. RIA Services sisältää ohjelmointi alustan, joka tukee hierarkkisten tietomallien muokkaamista (kuten tilaus- ja tilaustiedot yhteys), periytymistietomalleja (kuten Parent and Child suhde) ja tietojen projektiomalleja (kuten tietojen denormalisointi yhteen tietomalliin ottamalla arvoja Asiakas- ja Osoite tauluista).

Sovelluksessa voidaan joutua näyttämään tietoja useista tietolähteistä tai paljastamaan yksi entiteetti useisiin Domain Serviceihin. WCF RIA Servicesin avulla tämä skenaario onnistuu tekemällä referenssit eri entiteettien välillä.

Oletusarvon mukaan RIA Services ei lähetä koko alkuperäistä entiteettiä muutettujen arvojen kanssa Data Access Layeriin tarkistamaan tietoja yhteneväisyyttä. Sen sijaan RIA Services tallentaa ja välittää takaisin vain ne jäsenet, jotka on merkitty RoundtripOriginalAttribute, ConcurrencyCheck tai TimeStamp-määritteillä.

Domain Service

Domain Service on julkinen abstraktio toimialueen liiketoimintalogiikasta. Se sisältää entiteetit ja operaatiot, jotka muodostavat toimialueen liiketoimintalogiikan. RIA Services tarjoaa DomainService -luokan pohjaluokaksi kaikkiin luokkiin, jotka toimivat liittymänä keskikerroksen liiketoimintalogiikkaan. Kun Domain Service toteutetaan, määritetään entiteetit, jotka halutaan näyttää käyttöliittymälle. On myös mahdollista määrittää dataoperaatiot, jotka ovat sallittuja toimialueen kautta, ja lisätä sovelluslogiikkaa Domain Serviceen. Jokaiseen Domain Serviceen, joka sallitaan asiakassovelluksen käyttöön, RIA Services generoi DomainContext luokan asiakassovellukseen. [10 s.21-22]

5.4 Silverlight-käyttöliittymä

WCF RIA Servicesin avulla voidaan luoda Silverlight-asiakaita, jotka ovat tietoisia keskitasolla olevasta sovelluksen logiikasta, kun tietoja käsitellään. Käyttöliittymän voi antaa tarkastella ja muokata tietoja ja käyttää validointiehtoja ennen muutosten lähettämistä. Silverlight-kontrollit käyttävät luokkia, jotka generoidaan automaattisesti keskitason koodista.

DomainContext-luokka luodaan jokaiselle keskikerroksen projektissa olevalle Domain Servicelle, joka paljastuu entiteettiluokkana. Käytännössä tämä tarkoittaa kaikkia Domain Servicen metodeita, jotka palautusarvona palauttavat entiteettejä, joita Silverlight-kontrollit osaavat käsitellä. DomainContext-luokka sisältää kysely -ja muuttamismetodit, jotka keskustelevat vastaavien toimialueen operaatioden kanssa DomainServicessä. Kun kyselymetodia kutsutaan, Silverlight sovelluksen DomainContextissa kyselymetodi kutsuu vastaavaa metodia DomainServicessä, joka palauttaa kysytyn datan. DomainContextin metodit suoritetaan asynkronisesti, jotta käyttöliittymä pysyy koko ajan käytettävissä, kun dataa ladataan.

Silverlight-kontrolleille, kuten DataGridilla voidaan esittää DomainContextin kautta noudetut tiedot. Kontrolliin voidaan sitoa kyselyn tulokset, minkä jälkeen se näyttää ne automaattisesti, kun tuloksia on näytettävänä.

RIA Servicesin avulla voidaan päivittää, lisätä ja poistaa dataa Silverlight-kontrolleista, kun nämä toiminnot on näytetty DomainServicessä. Kun datanmuokkausoperaatioita kutsutaan DomainServicestä, Silverlight-asiakkaalta saatu data käsitellään toisen tason logiikalla, joka varmistaa, että liiketoiminnansääntöjä sovelletaan muutoksissa.

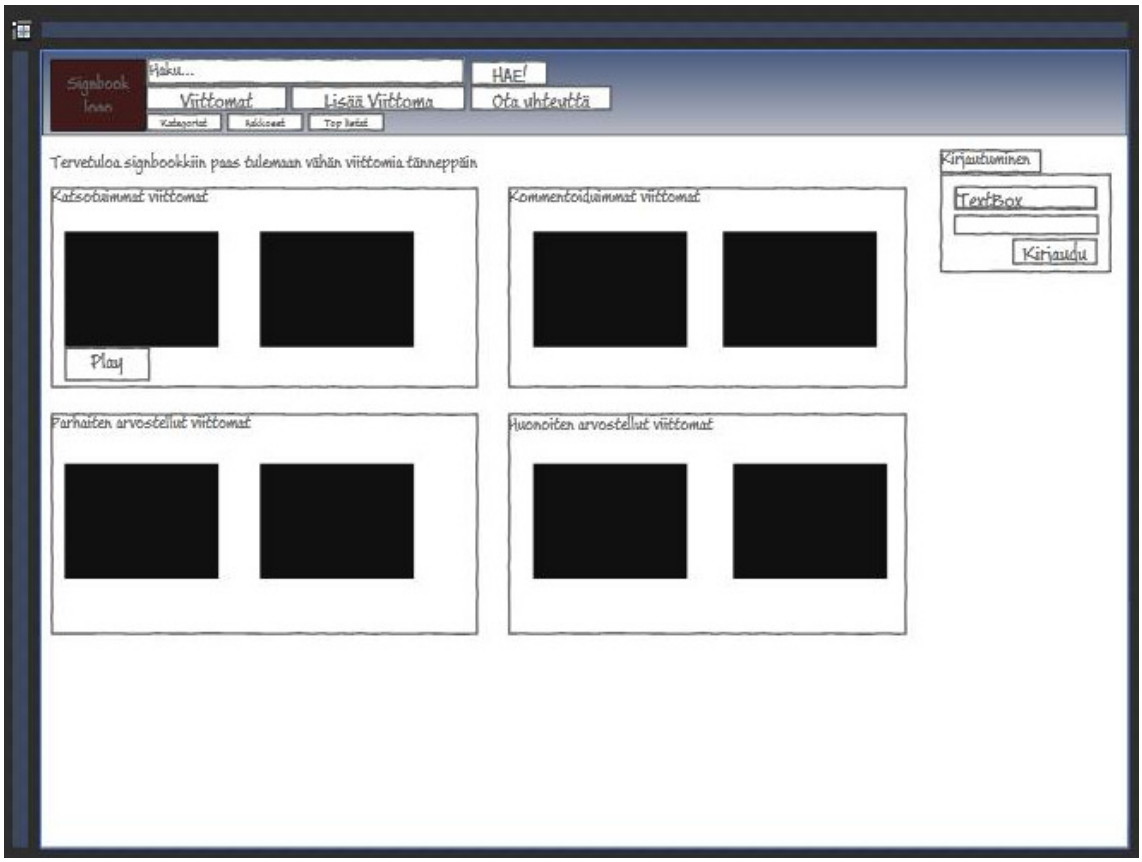
DomainDataSource -kontrollin avulla voi vuorovaikuttaa DomainServicestä saatuun dataan. DomainDataSourcen avulla voi käyttää määritettävää syntaksia spesifioimaan sivutusta, lajittelua, ryhmittelyä ja suodatusta

Jos halutaan muokata RIA Servicesin generoimaa koodia ei kuulu koskea Generated_Code-kansioon, koska nämä tiedostot korvataan, kun asiakasprojekti käännetään uudelleen. Sen sijaan voi mukauttaa koodia, joka luodaan asiakkaan projektiin kirjoitamalla osittaisia metodeita DomainContextissa. Nämä osittaisen metodit tehdään lisäämään laskettuja ominaisuuksia asiakasprojektissa tai lisäämän mukautettua logiikkaa. Generoidut osittaiset metodit herätetään suorituksen aikana vain, jos osittaisia metodeita on implementoitu. [11]

6. Signbook käyttöliittymä

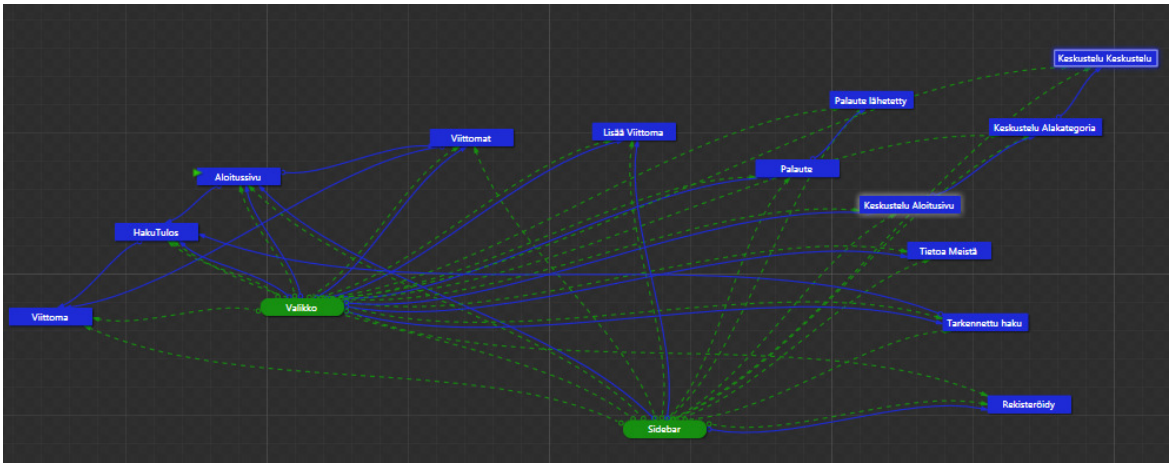
6.1 Prototyypitys

Projektin suunnittelussa ja sen esittämisessä asiakkaalle on käytetty Microsoft Expression Blend +SketchFlow -ohjelmistoa. SketchFlow on Microsoft Expression -tuotesarjaan kuuluva teknologia. Se tuli uutena ominaisuutena Expression Blend 3:n mukana. Sen tarkoitus on mahdollistaa prototyypitystä Silverlight-sivuilla. Eritoten puhutaan käyttäjien käyttökokemuksien prototyyppaamisesta.



Kuva 6.1: Termipankin Sketchflow -prototyyppi

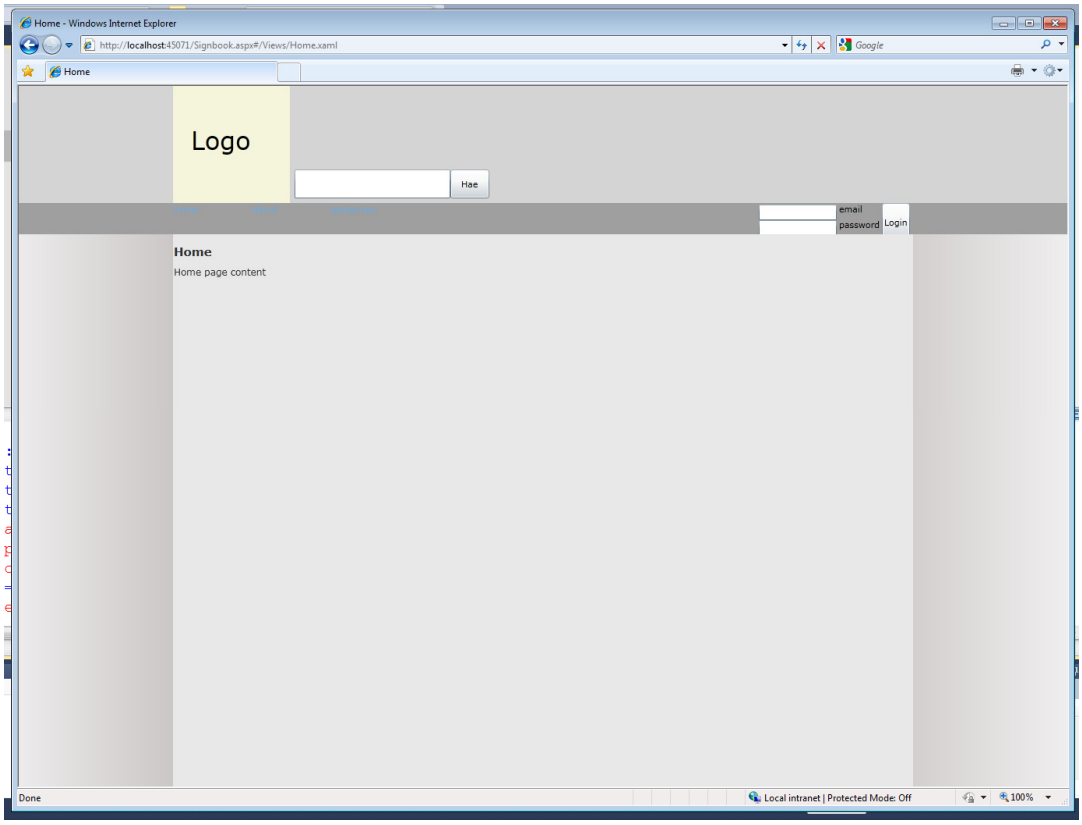
SketchFlow'n idea perustuu erilaisten sivujen (screenien) tekemiseen. Screenit edustavat yksittäisiä sivuja ja niitä voi yhdistellä näppärästi SketchFlow'n mukana tulevassa kartassa. Sivuja voi olla myös komponentteina, joka tarkoittaa sivua, joka voidaan yhdistää toisen sivun ”päälle”. Nämä ovat hyödyllisiä, kun prototyypitetään vaikka kirjautumisruutua tai valikkoa, joka voidaan sitten yhdistää moneen sivuun.



Kuva 6.2: Termipankin SketchFlow Map, Vihreät näkymät tarkoittavat komponentteja, jotka sisältyvät kaikille sivuille.

Erikoista SketchFlow'sta tekee sen tapa esittää sovelluksen eri tilat (statet). Jokaisen sivun voi yhdistää toiseen sivuun. Tämä tarkoittaa sitä, että sieltä on pääsy toiseen sivuun (esim. kirjaudu sivusta aloitusvalikkoon). Sivuille itselleen voi myös tehdä omia tiloja, mikä tarkoittaa tilan vaihtumista sivussa itsessään, esim. kirjautuminen epäonnistui tai kirjautuminen onnistui. (Kuva 6.2)

SketchFlow on tarkoitettu eritoten prototyypittämiseen, eikä sillä pidä yrittää tehdä valmiita sovelluksia. Tätä varten siinä on valmiiksi luotuja piirretyn näköisiä kontrolleja. Asiakkaan näkökulmasta tämä tavoite toteutui hyvin eikä prototyyppivaiheessa jääty kiinni yksityiskohtiin kuten väreihin tai laatikoiden kokoon. Sketchflow luonnoksesta siirryttiin tekemään varsinaista projektia johon rakennettiin monia uusia toiminnallisuuksia, jotka käydään läpi seuraavaksi.



Kuva 6.3: Ensimmäinen ulkoasu ilman grafiikoita.

6.2 Ominaisuudet

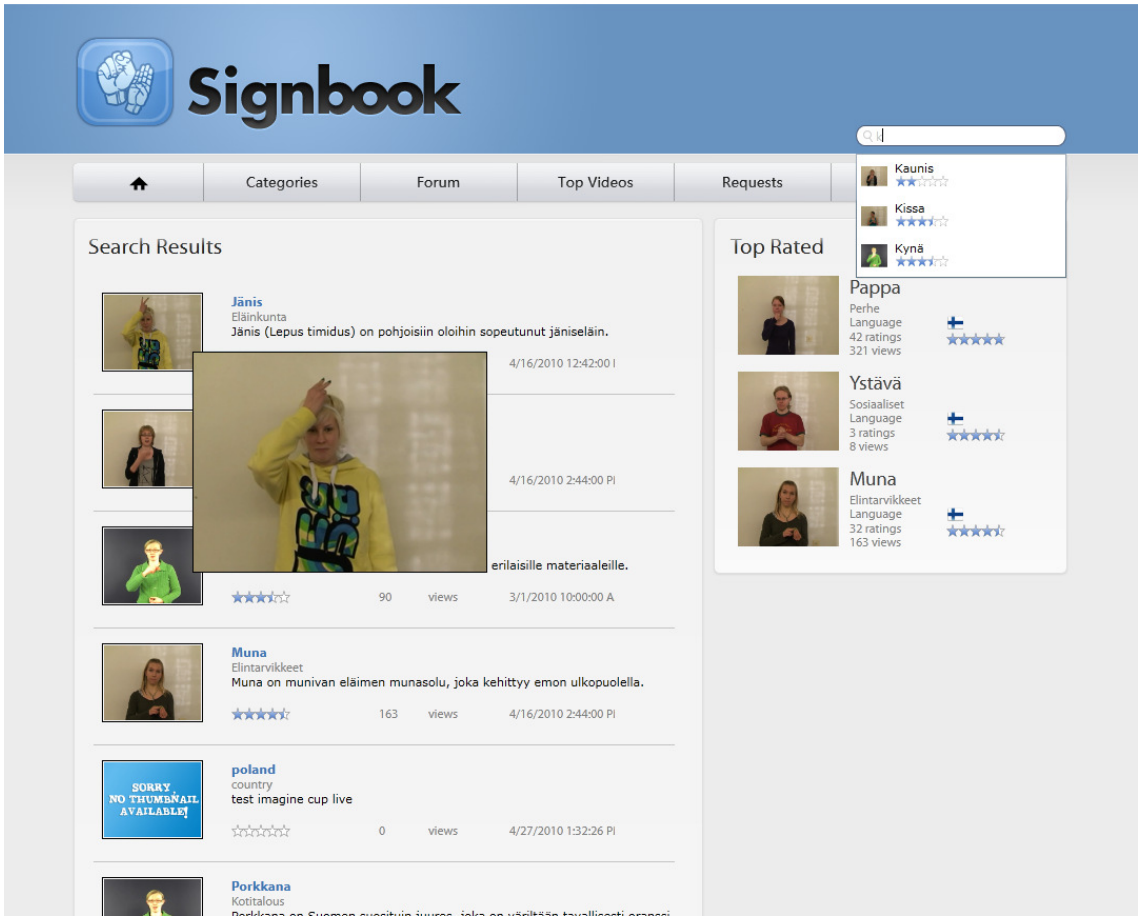
Signbookin keskeiset ominaisuudet ovat:

- Viittomien haku
  - Ennustava haku
  - Esikatselu
- Vittomien lisäys
  - Verkkokamera tuki
  - "Vedä ja pudota" valmiiksi nauhoitettu video
- Sosiaaliset ominaisuudet
  - Keskustelu
  - Kommentointi
- Opiskelumuoto
- Offline -ominaisuus

Seuraavissa luvuissa perehdytään näihin ominaisuuksiin syvällisemmin.

6.2.1 Viitomien haku

Hakutoiminto tulee varmasti olemaan Signbookin käytetyin ominaisuus. Siksi siihen onkin panostettu paljon ja sillä on todella helppoa löytää viittomia. Viittomien haku on ennustava ja visuaalinen ja itse hakukenttä on sijoitettu näkyvälle paikalle sivun yläosaan. Hakutulos sivulla näytetään kaikki viittoman perustiedot ja jo siitä voi esikatsella videota viemällä kursorin videon päälle.

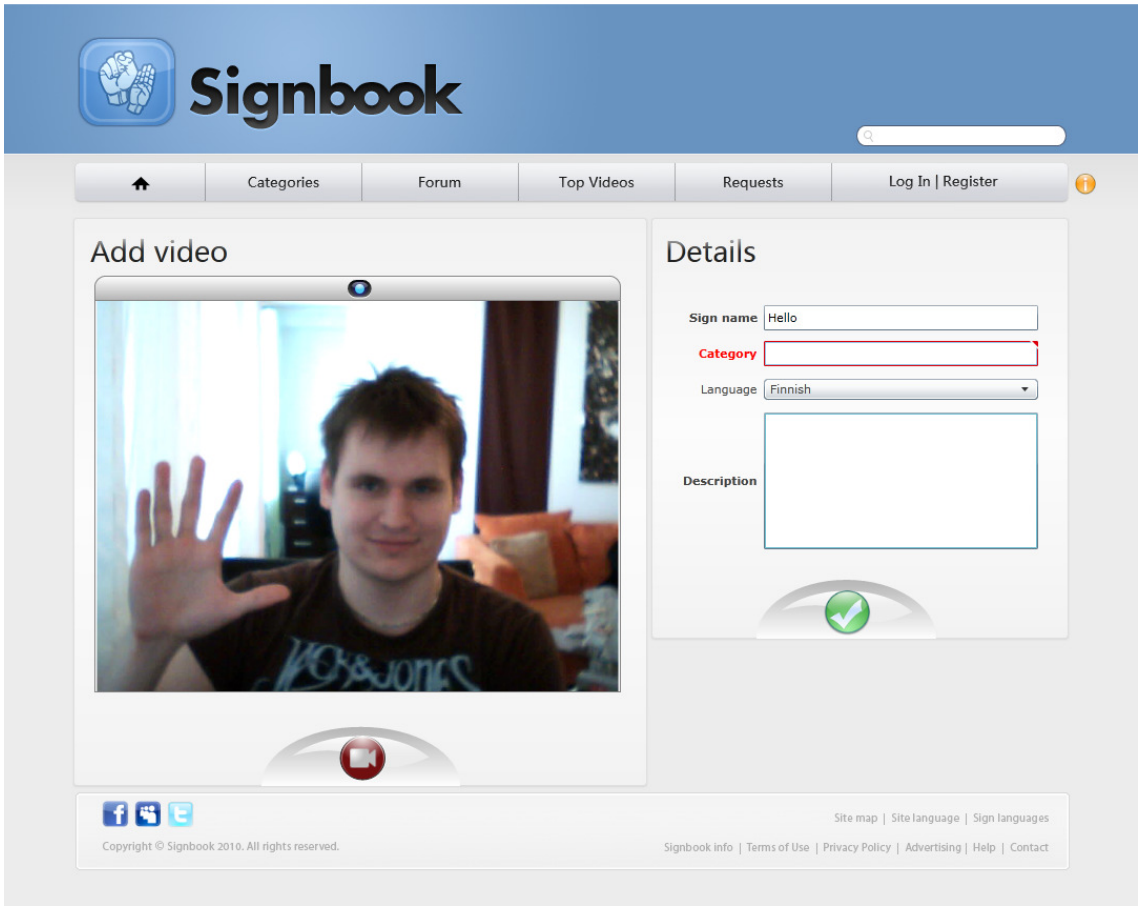


Kuva 6.4: Ennustava haku, hakutulokset ja esikatselu.

6.2.2 Sisällön tuottaminen

Käyttäjät pystyvät lisäämään viittomia Signbookin tietokantaan rekisteröitymisen jälkeen. Viittomien lisäys on helppoa ja se voidaan tehdä suoraan käyttäjän webcamistä. Jos video on tallennettu jo muualla käyttäjä voi vain vetää ja pudottaa (drag&drop) videotiedoston sovellukseen.

Tietojen eheys tarkistetaan RIA Servicesin metadaa hyödyntämällä, joka tulee Signbookin Domain Servicestä. Validointi on käyttäjäystävällistä ja se toimii ennustavasti, joka tarkoittaa, että tiedot validoidaan oikeiksi ennen niiden lähettämistä palvelimelle. Käyttäjän yrittäessä syöttää väärää informaatiota hänelle annetaan punaiset varoitusliput, jotka kertovat mikä on väärässä.



Kuva 6.5: Viittoman lisäys

6.2.3 Sosiaaliset ominaisuudet

Suurin osa sisällöstä Signbookiin tulee käyttäjiltä. Sosiaalinen media on suuri tekijä Signbookissa. Ihmisten tulee pystyä valitsemaan ystäviä muista käytäjistä ja keskustelemaan heidän kanssaan. Tällä hetkellä keskustelu toimii tekstimuodossa ja se on toteutettu RIA Servicessillä. Tulevaisuudessa tulisi lisätä myös videokeskustelu mahdollisuus, jotta käyttäjät pystyisivät käyttämään viittomankieltä keskusteluissa. Signbook tulisi myös integroida muihin sosiaalisen median sivustoihin, kuten

Facebookiin. Tämän jälkeen käyttäjät voisivat suoraan saada ystävänsä Facebookista ja voisivat myös nähdä, jos ystävät ovat paikalla Facebookissa ja lähettämään heille viestejä suoraan Signbookista.

Sisällön laatua valvotaan myös käyttäjien toiminnalla. Käyttäjät pystyvät kommentoimaan ja arvostelemaan viittomia ja jos he näkevät epäasiallisia videoita tulee myös tarjota mahdollisuus niiden ilmiantamiseen. Tarpeeksi monen ilmiannon jälkeen voimme piilottaa viittoman kokonaan. Sosiaalinen media on Signbookin ominaisuus, joka tekee siitä ainutlaatuisen, sillä kaikki nykyiset viittomakielen palvelut/järjestelmät ovat siinä mielessä puutteellisia. Signbookissa on myös toiveet osio, jossa ihmiset voivat toivoa tietyn aihealueen viittomia. Tämän ominaisuuden tavoitteena on antaa käyttäjille enemmän mahdollisuuksia sisällön kontrollointiin. Esimerkiksi, jos tulkki on menossa tilaisuuten, jonka aihealue on hänelle entuudestaan vieras, hän voi valmistautua sanastoon pyytämällä jotain tiettyä termiä tai yleisesti alan termistöä Signbookissa. Muut viittomakielen käyttäjät – tulkit tai viittomakieliset – jotka tietävät sanastoa, voivat auttaa ja lisätä sitä nähdessään pyynnön.

Kuva 6.6: Omavalikko ja keskustelu.

6.2.4 Opiskelumuoto

Vaikka Signbook sivustoa voi käyttää opiskelutarkoituksiin ihan hyvin sellaisenaankin, siitä haluttiin tehdä vielä parempi. Signbookiin tuli idea keskitetystä paikasta, jossa voi suorittaa viittomakielen opiskelua. Tätä osiota kutsutaan Signroomiksi ja sen ideana on tarjota käyttäjälle "augmented reality" tyyppinen kokemus, jossa itse on peilin (webcamin) ääressä ja siihen rinnalle saa tarkasteltavaksi viittoman. Vertailemalla omaa viittoman suoritustapaa Signbookissa olevaan, voi hyvin nähdä mahdolliset erot ja korjata omat virheensä päätyen lopulta täsmälliseen oikeaan suoritustapaan.

Kuva 6.7: Opettelumoodi.

6.2.5 Offline-ominaisuus

Signbookista löytyy ominaisuus, jonka avulla on mahdollista ladata viittomia käyttäjän omalle koneelle, jotta niitä voidaan katsella ilman internetyhteyttä. Tämä ominaisuus on hyödyllistä ihmisille, jotka haluavat opetella viittomia esimerkiksi matkustaessaan.

Offline videoita voidaan katsella Applen laitteista tutusta Coverflow-kontrollista. Offline-videoiden toiminta tulee toteuttaa käyttäen Silverlightin Isolated Storage ominaisuutta. Tässä esiselvitystyössä sitä ei ole vielä toteutettu.

Kuva 6.8: Offline viittomien katselu.

6.2.6 Signbook-käyttökokemus

Signbook tarjoaa rikkaan käyttökokemuksen, joka lisää käyttäjätyytyväisyyttä ja parantaa tuottavuutta. Jotkin Signbookin toiminnoista ovat uniikkeja, eikä niitä ole ennen internet -sovelluksissa nähty. Rekisteröitymisen jälkeen käyttäjän kaikki toiminnallisuudet löytyvät omasta valikosta, josta näkyvät myös käyttävän ystävät ja

heidän tilansa. Ystävien kanssa keskustelu on helppoa ja chat ikkunan voi sijoittaa minne tahansa sivulla ja tehdä niistä sen kokosia, mitä itse haluaa.

Normaalin www-sivuston navigaation lisäksi Signbookissa on nopeampi hiiren eleitä hyväksikäyttävä navigaatio. Tämä toimii näyttämällä visuaaliset valinnat Signbookin toiminnoista varsinaisen sisällön edessä. Tämän tyylinen navigaatio on helppo ja nopea käyttää verrattuna normaaliin navigaatioon.

Kuva 6.9: Visuaalinen navigointi.

6.3 Verkkokamera

Projektin yksi pääominaisuus on verkkokameran käyttö. Kameran avulla käyttäjä pystyy viittomaan itse termin suoraan sen lisäyksen aikana. Tämä mahdollistaa videon tallentamisen oikean kokoisena ja käyttäen yhtenäistä formaattia, joka taas auttaa videoiden optimoinnissa. Kameran nauhoitusta tullaan käyttämään myös kommentoinnissa ja keskustelualueella keskustellessa.

Silverlight 4 toi mukanaan ominaisuuden videon ja äänen paikalliseen taltiointiin, joka toimii ilman palvelinta. Tämän avulla pystytään jo toistamaan kameran kuvaa sovelluksessa, mutta ei toistaiseksi ole videon taltiointi mahdollisuutta.

Kameran toiminnan aktivointi on hyvin yksinkertaista. Aluksi lisätään sovellukselle lähde käytettävälle laitteelle sivun ladatessa:

```
public MainPage()
{
  InitializeComponent();
  Loaded += new RoutedEventHandler(MainPage_Loaded);
}
void MainPage_Loaded(object sender, RoutedEventArgs e)
{
  _captureSource = new CaptureSource();
}
```

Lähteen esittelyn jälkeen voidaan konfiguroida sille laitteen. Sovellus voidaan laittaa suoraan käyttämään koneen oletus -kameraa tai kaikki kamerat voidaan listata esimerkiksi listakenttään, josta käyttäjä voi valita haluamansa kameran käyttöön nauhoitukseen. Jälkimmäinen tapa on parempi siihen nähden, että monella käyttäjällä voi huonompi kamera olla koneen oletuksena. Kaikki käytössä olevat laitteet voidaan hakea koodilla MainPage_Loaded vaiheessa listboxiin seuraavasti:

```
// Listataan kaikki kamera laitteet listboxiin
VideoSources.ItemsSource =
CaptureDeviceConfiguration.GetAvailableVideoCaptureDevices(
);
```

Kun kamera on valittu, voidaan kuvan tuottaminen sovellukseen aloittaa. Kun taltiointi -nappia painetaan pitää valittu kamera määritellä taltiointilähteelle. Kameran määrittelyn jälkeen luodaan VideoBrush, joka näyttää videokuvaa. VideoBrushin lähteeksi annetaan taltiointilähde, jonka jälkeen sillä täytetään rectangle. Ennen kuin sovellus aloittaa kuvan piirtämisen, kysytään käyttäjältä lupaa kameran käyttämiseen. Tällä tavalla suljetaan pois mahdollisuus, jossa käyttäisimme käyttäjän kameraa hänen tietämättään. Taltioinnin aktivointi on ohjelmoitu nappiin seuraavalla tavalla:

```
private void CaptureButton_Click(object sender,
RoutedEventArgs e)
```

```
{
  if (_captureSource != null)
  {
    // Lopetetaan kaikki nykyinen taltiointi.
    _captureSource.Stop();

    // Asetetaan taltiointilähteelle valittu kameralaite.
    _captureSource.VideoCaptureDevice =
(VideoCaptureDevice)VideoSources.SelectedItem;

    // Luodaan VideoBrush, ja kuva piirretään Rectangleen.
    VideoBrush vidBrush = new VideoBrush();
    vidBrush.SetSource(_captureSource);
    // Täytetään Rectangle videolla.
    WebcamCapture.Fill = vidBrush;

    // Pyydetään käyttäjältä lupa kameran käyttöön.
    if (CaptureDeviceConfiguration.AllowedDeviceAccess ||
    CaptureDeviceConfiguration.RequestDeviceAccess())
    {
      _captureSource.Start();
    }
  }
}
```

Kaappaus voidaan pysäyttää yksinkertaisesti `captureSource.Stop();` -komennolla, joka voidaan ohjelmoida omaan painikkeeseen.

6.4 Navigointi

Silverlight-projektiin pitää rakentaa navigointi, koska itsellään selaimessa hostattu Silverlight sovellus ei anna mitään informaatiota selaimelle, millä sivuilla mennään. Ilman lisättyä navigointia ei käyttäjällä siis toimi selaimessaan esim back/forward napit ja myöskään linkitys johonkin ohjelman sisäiseen sivuun ei toimi.

Navigointia ei sisällytetä Silverlightiin oletuksena, vaan siihen pitää lisätä referenssit, jotka sitten lähetetään käyttäjälle Silverlight sovellusn mukana. Navigaatio löytyy System.Controls.Navigation komponentista, jonka lisäksi XAMLiin pitää lisätä namespace:

```
xmlns:navigation="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Contro
ls.Navigation"
```

Navigointi rakennetaan kahdesta kontrollista: Framesta ja pagesta. Frame on pääkontrolli, johon page ladataan. Framea voi ajatella sivun pääkehyksenä, johon sitten luodaan sisältö erilaisten Viewien avulla:

Tällä XAML rivillä luodaan Frame, johon aloitussivuna ladataan Homepage:

```
<navigation:Frame x:Name="MainFrame"
Margin="10,80,30,50 Source="/Views/HomePage.xaml" />
```

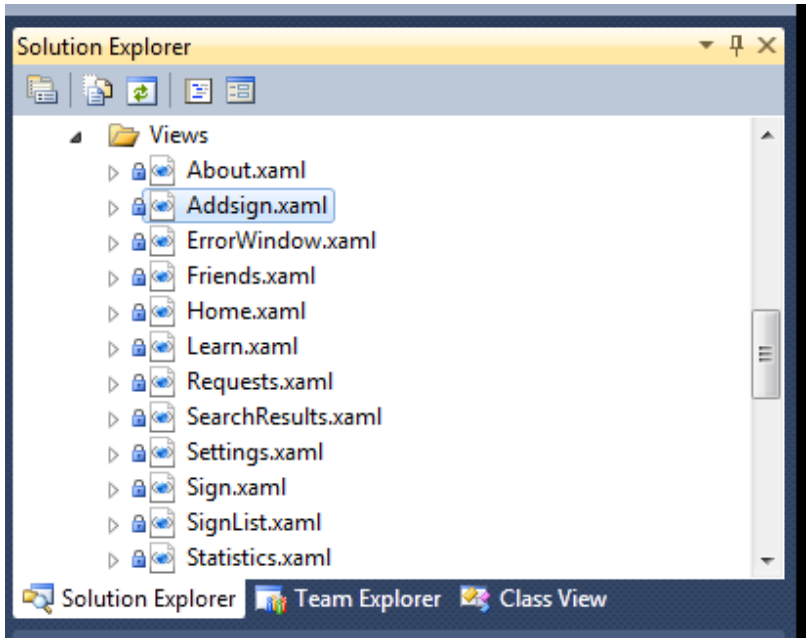
Itse navigointi voidaan toteuttaa vaikka napin painalluksesta:

```
<Button x:Name="Signs" Click="NavigateButton_Click"
Tag="/Views/Signs.xaml" />
```

Napin painallusta vastaava taustakoodi:

```
private void NavigateButton_Click(object sender, RoutedEventArgs
e)
{
    //napataan urli napin tagista ja navigoidaan sinne
    Button btn = sender as Button;
    string url = btn.Tag.ToString();
    this.MainFrame.Navigate(new Uri(url, UriKind.Relative));
}
```

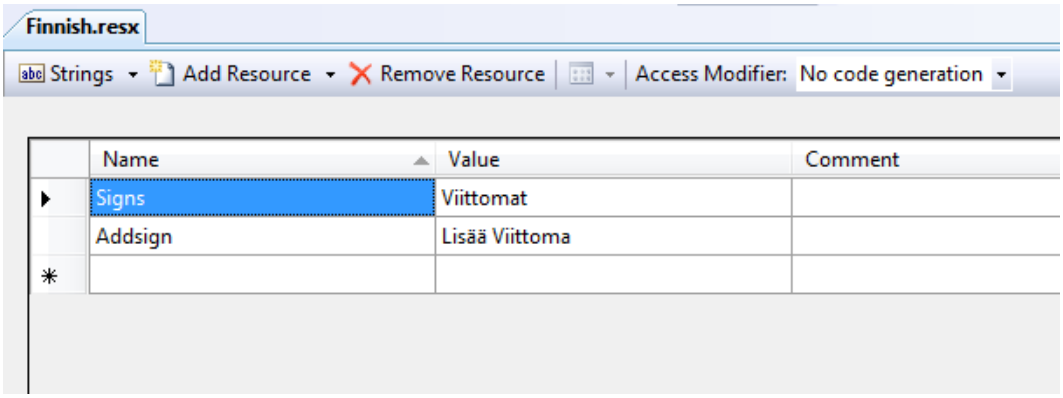
Silverlight navigaation avulla saadaan eriytettyä sivujen koodit ja käyttöliittymäkuvaukset omiin tiedostoihinsa, tämä selkeyttää koko ohjelmiston rakennetta. Mainframeen ladattu sisältösivu löytyy siis Views kansiosta:



Kuva 6.10: Silverlight sivut Views kansiossa.

6.5 Lokalisointi

Lokalisointi Silverlight -sovellusssa on parasta hoitaa resurssitiedostojen kanssa. Resurssitiedostot pystyvät pitämään sisällään informaatiota, kuten tekstiä, kuvia, tiedostoja, ja ääniä. Resurssitiedoston tunnistaa sen tiedostopäätteestä, joka on *.resx. Termipankki tulee rakentaa siten, että se on helppo kääntää muille kielille, mutta dynaamisesti kieltä ei todennäköisesti tarvitse pystyä vaihtamaan. Erikieliset versiot tullaan siis hostaamaan omissa domaineissaan, jonka kielen tunnistaa domainin päätteestä. Resurssitiedostojen muokkaaminen on helppoa, eikä se vaadi ohjelmointikokemusta, joten lokalisaatiotyöntekijän ei tarvitse olla tekninen osaaja. Standardi .NET lokalisaation nimeämistyyli on (<RESOURCE_NAME>.resx neutraaleille resursselle ja <RESOURCE_NAME>.<LANGUAGE>.resx kielikohtaisille resurssitiedostoille.



Kuva 6.11. Suomenkielinen resurssitiedosto, johon on käännetty muutaman napin tekstit

Resurssitiedoston Access Modifier pitää vaihtaa publiciksi, jolloin siihen päästään XAML:n kautta käsiksi

Käännettyjen tekstien hakeminen resurssitiedostosta on hyvinkin helppoa. Kääntäjälle pitää kertoa, mistä resurssitiedosto löytyy, joka tässä tapauksessa on ohjelman juuri:

```
xmlns:local="clr-namespace:SignbookTechnologyDemo"
```

jonka jälkeen kerrotaan minkä nimistä resurssitiedostoa käytetään:

```
<UserControl.Resources>
   <local:Finnish x:Key="Finnish"/>
</UserControl.Resources>
```

Tämän jälkeen ollaan valmiita hakemaan itse lokalisoitu teksti esim. nappulaan:

```
<Button Content="{Binding Signs, Source={StaticResource Finnish}}" />
```

6.6 XAML -resurssit

Silverlight sisältää resurssisysteemin, mikä toimii XAMLin avulla. Resursseja käyttämällä voidaan uudelleenkäyttää tyylejä, minkä avulla saadaan helpommin esim. nappeihin kaikkiin samanlainen tyyli. Lisäksi se helpottaa päivitystä, kun informaatiota pitää muuttaa tarvittaessa vain yhteen paikkaan. Tyylit toimivat hyvin samaan tapaan kuin CSS:n tyylit.

Jokaisella elementillä on omat resurssinsa. Usein kuitenkin on tapana tehdä resurssit sivukohtaisiksi, eikä esim. napissa. Tässä esimerkiksi luodaan UserControlliin tekstityyli, jota voidaan sivulla uudelleen käyttää missä tahansa visuaalisessa elementissä.

```
<UserControl.Resources>
        <Style x:Key="HeaderTextStyle" TargetType="TextBlock">
            <Setter Property="Foreground">
               <Setter.Value>
                  <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                     <GradientStop Color="#FF232323" Offset="0.5"/>
                     <GradientStop Color="White" Offset="0"/>
                  </LinearGradientBrush>
               </Setter.Value>
            </Setter>
            <Setter Property="FontSize" Value="26.667"/>
            <Setter Property="FontWeight" Value="Normal"/>
            <Setter Property="TextWrapping" Value="Wrap"/>
            <Setter Property="Margin" Value="10,5,0,5"/>
            <Setter Property="HorizontalAlignment" Value="Left"/>
            <Setter Property="FontFamily"
            Value="/Signbook;Component/Fonts/Fonts.zip#Microsoft YaHei"/>
        </Style>
</UserControl.Resources>
```

Tämä tekstityyli voidaan helposti implementoida tekstikontrolliin, jolla saadaan seuraava lopputulos:

```
<TextBlock x:Name="txtComments" Text="Sign details" Style="{StaticResource
HeaderTextStyle}"/>
```

Sign details

Silverlight hakee resurssit hierarkkisesti, joka tarkoittaa sitä, että se käy elementti elementiltä läpi resurssit ja lopettaa, jos löytää kyseisen resurssin. Esimerkiksi tässä tapauksessa katsotaan ensin napin resurssit, kun sieltä ei mitään löydy siirrytään UserControllin resursseihin, joista resurssi löytyy.

Resurssit voidaan määritellä myös applikaatiokohtaisiksi, jolloin resurssit löytyvät applikaation kaikilta sivuilta. Tällöin resurssit määritellään App.xamliin Applikaation alle tällä tavoin:

```
<Application xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
x:Class="SilverlightApplication1.App">
    <Application.Resources>
        <Style x:Key="HeaderTextStyle" TargetType="TextBlock">
             <Setter Property="Foreground">
                <Setter.Value>
                   <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
                      <GradientStop Color="#FF232323" Offset="0.5"/>
                      <GradientStop Color="White" Offset="0"/>
                   </LinearGradientBrush>
                </Setter.Value>
             </Setter>
             <Setter Property="FontSize" Value="26.667"/>
             <Setter Property="FontWeight" Value="Normal"/>
             <Setter Property="TextWrapping" Value="Wrap"/>
             <Setter Property="Margin" Value="10,5,0,5"/>
             <Setter Property="HorizontalAlignment" Value="Left"/>
             <Setter Property="FontFamily"
             Value="/Signbook;Component/Fonts/Fonts.zip#Microsoft YaHei"/>
         </Style>
    </Application.Resources>
</Application>
```

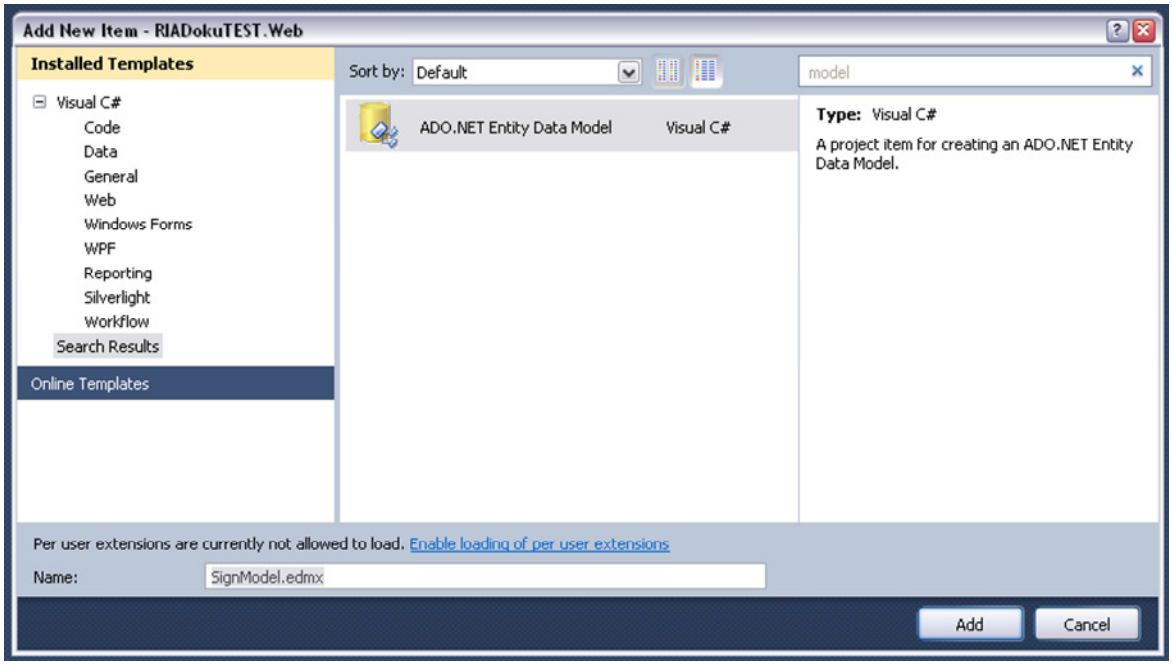
Lopputulos on täysin sama molemmin tavoin. On kuitenkin ehkä suositeltavampaa sijoittaa resurssit App.xamliin applikaatiotasolle, jolloin sivukohtaiset XAML:t ovat siistimpiä ja kaikki tyylit löytyvät samasta paikasta

7. Signbook keskikerros

7.1 Data access

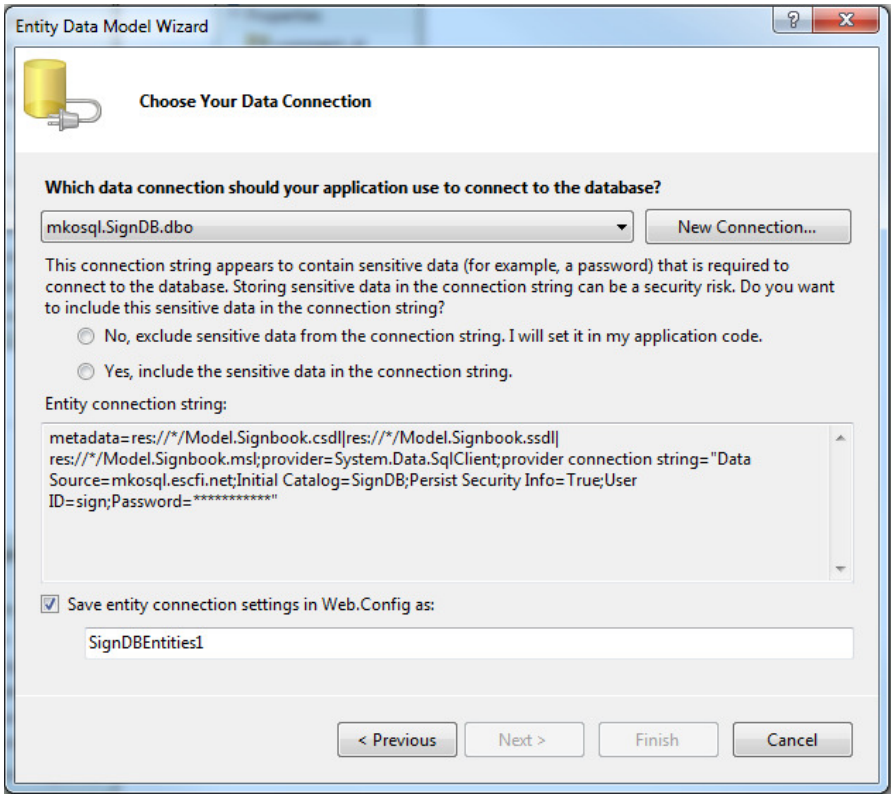
Kaikki tietokantayhteydet Signbookissa on hoidettu WCF RIA Services:llä. Palvelu mahdollistaa tietokantakyselyjen ajamisen ilman sivuston päivittämistä. Tämä tapa tuo mukanaan paljon käytettävyyttä sivulle, koska kyselyt voidaan suorittaa päivittämättä sivua ja käyttöliittymä pysyy koko ajan käytettävissä.

Kun tietokantayhteyttä lähdetään rakentamaan sovellukseen pitää aluksi lisätä ADO.NET Entity Data Model. Tietomalli pitää sisällään tietokantakuvauksen halutusta tietokannasta.

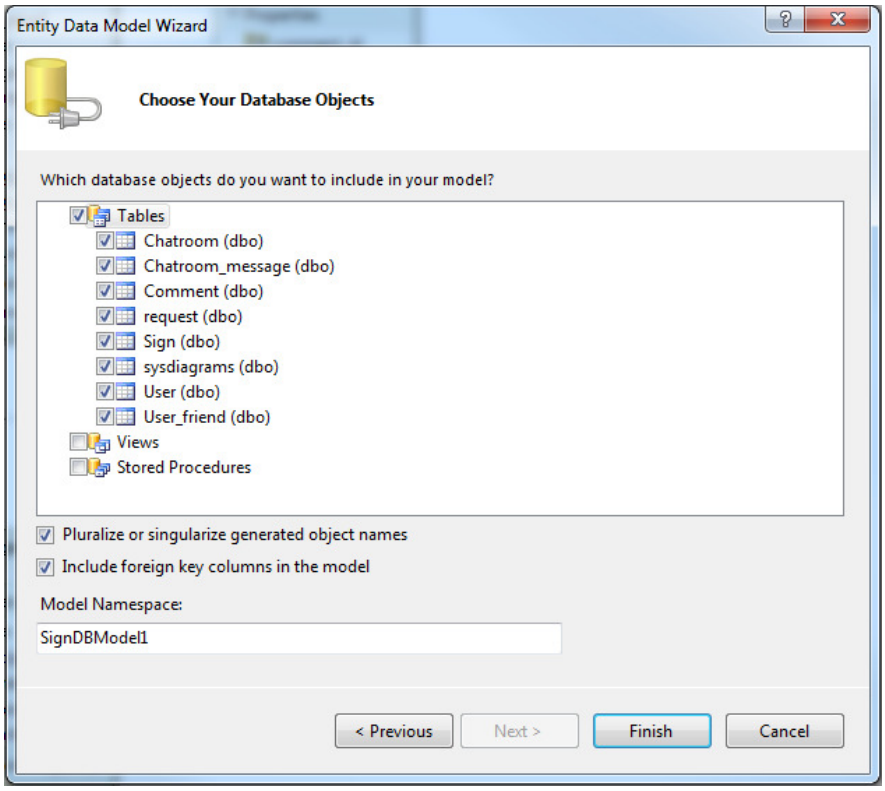


Kuva 6.12. Lisätään ADO.NET Entity Data Model sovellukseen

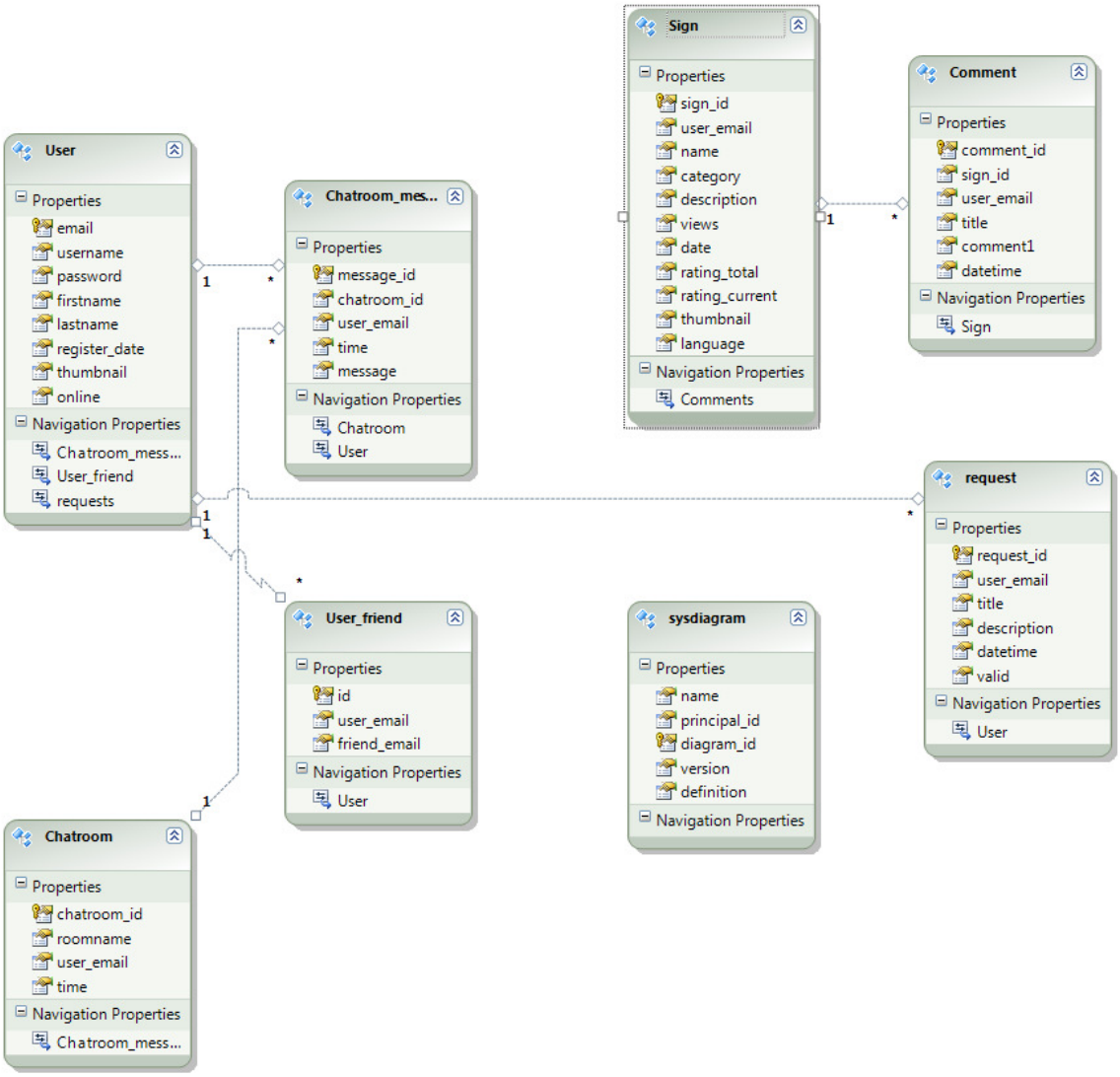
Kun tietomalli on valittu, siihen määrittellään tietokantayhteys haluttuun tietokantaan. Yhteysasetukset tallennetaan automaattisesti Web.Config –tiedostoon, josta sovellus osaa hakea sen aina kun sitä tarvitaan. Kun yhteys on luotu, määritetään halutut tietokannan taulut, joita halutaan käyttää sovelluksessa. Lopuksi kun tietomalli on luotu, se näyttää graafisella näkymällä tietokantamallin, jota tullaan käyttämään kyselyjen luontiin. Kun kyseiseen malliin lisätään esimerkiksi koko tietokanta, se tulee näyttämään kaikki siihen määritellyt suhteet ym.



Kuva 6.13: Tietokantayhteyden luontivaihe

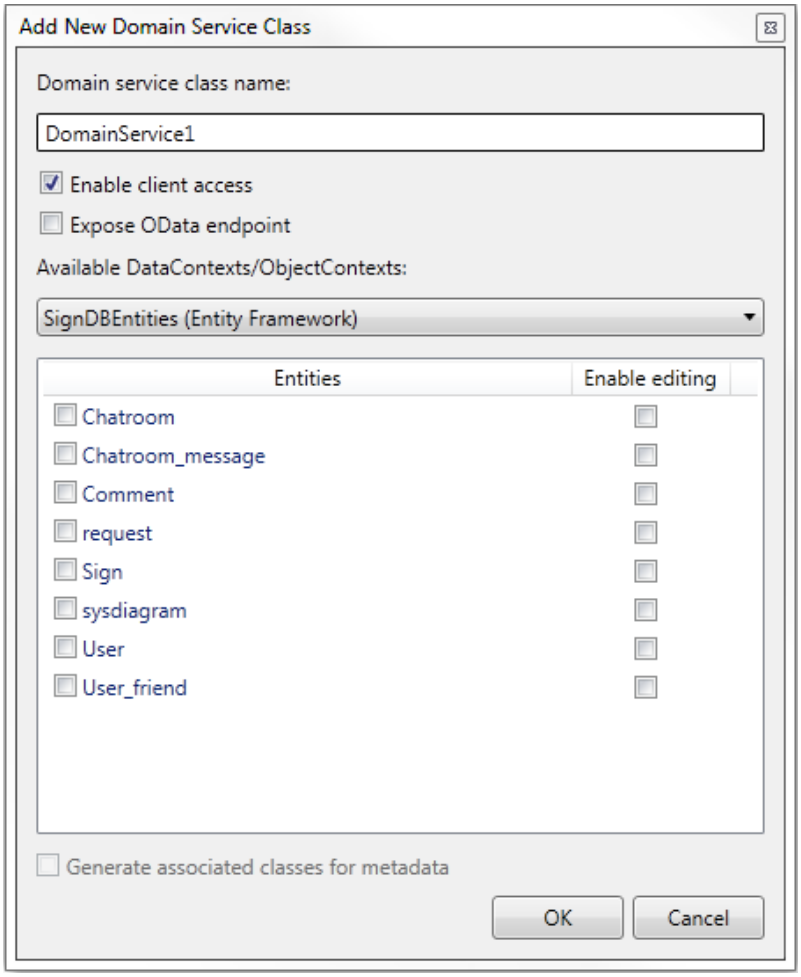


Kuva 6.14. Tietokantaobjektien valintavaihe



Kuva 6.15: Valmis Signbook -tietomalli

Kun tietomalli on luotu haluttavalla tavalla, tarvitaan ominaisuus, jonka avulla voidaan suorittaa kyselyitä tietomallista. Kyseiseen tapaukseen tarvitaan Domain Service Class. Tämä palvelu muodostaa kyselyt LINQ:a apuna käyttäen suoraan tietomallista. Tämä palvelu mahdollistaa kyselyiden suorittamisen ilman, että joudutaan päivittämään sivustoa. Tämä toimintatapa on tärkeää Silverlightin kannalta, koska Silverlight sovellusta ei voida päivittää kesken ajon.



Kuva 6.16. Domain Service luokan luominen

Kun Domain Service Class luodaan, se tunnistaa aikaisemmin luodun tietomallin. Domain Service Classiin voidaan nyt valita tietomalliin lisätty taulu. Kun tauluun sallitaan muokkaaminen, palvelu luo automaattisesti peruskyselyt haulle, lisäämiselle, päivittämiselle ja poistamiselle seuraavaan tapaan:

```
public IQueryable<SIGN> GetSIGNs()
{
   return this.ObjectContext.SIGNs;
}

public void InsertSIGN(SIGN sIGN)
{
   if ((sIGN.EntityState != EntityState.Added))
   {
      if ((sIGN.EntityState != EntityState.Detached))
      {
         this.ObjectContext.ObjectStateManager.ChangeObjectState(sIGN,
EntityState.Added);
      }
```

```
      else
      {
         this.ObjectContext.AddToSIGNs(sIGN);
      }
   }
}

public void UpdateSIGN(SIGN currentSIGN)
{
   if ((currentSIGN.EntityState == EntityState.Detached))
   {
      this.ObjectContext.AttachAsModified(currentSIGN,
this.ChangeSet.GetOriginal(currentSIGN));
   }
}

public void DeleteSIGN(SIGN sIGN)
{
   if ((sIGN.EntityState == EntityState.Detached))
   {
      this.ObjectContext.Attach(sIGN);
   }
this.ObjectContext.DeleteObject(sIGN);
}
```

Kuten millä tahansa laajalla sovelluksella, tiedon haussa on pystyttävä antamaan parametrejä käyttöliittymästä, minkä mukaan tietoa haetaan:

```
public IQueryable<SIGN> GetSearchedSigns(String searchString)
{
   return this.ObjectContext.SIGNs.Where(n =>
n.name.Contains(searchString)).OrderBy(n=>n.name);
}
```

Luotuun kyselyyn viedään merkkijono sovelluksesta ja tarkastellaan, sisältääkö name –kenttä kyseistä arvoa.

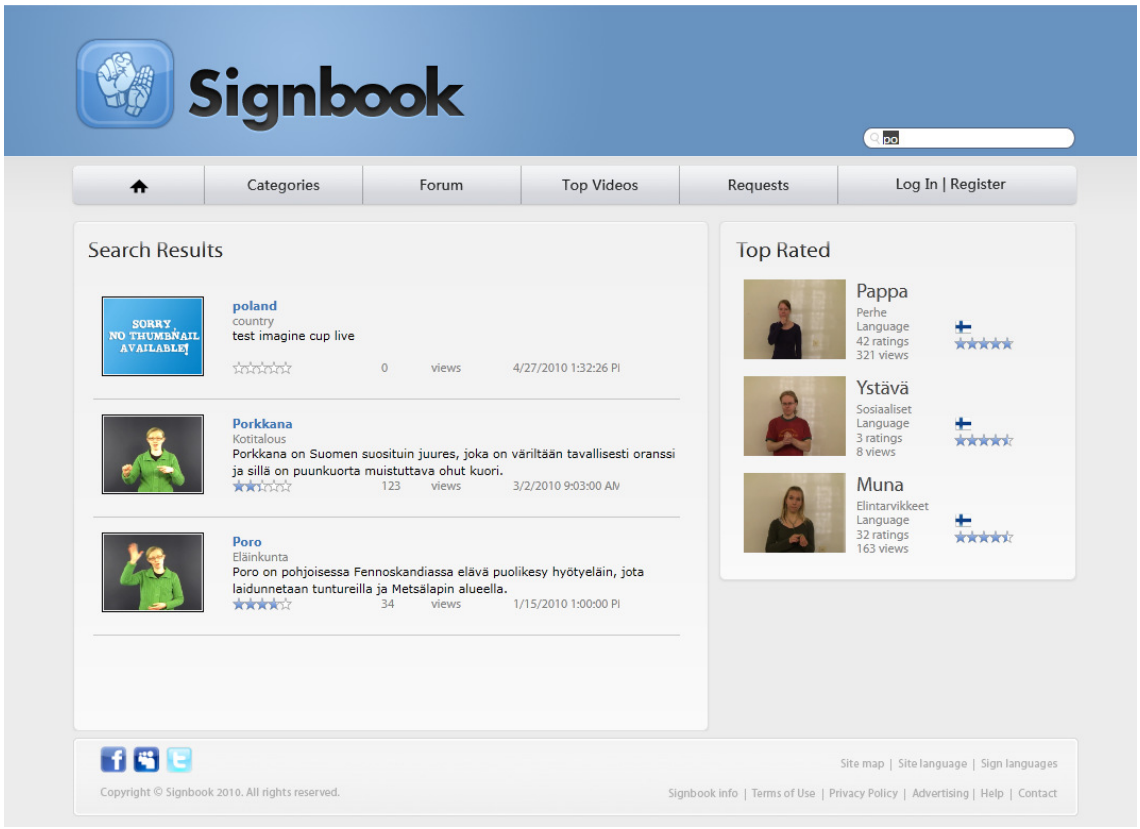
Kun tietomalli ja Domain Service Class on luotu, voidaan aloittaa kyselyn kutsuminen itse sovelluksesta. Domain Service Classin kutsuminen tapahtuu C# taustakoodilla seuraavalla tavalla:

```
public partial class About : Page
{
   SignDomainContext haku = new SignDomainContext();

   public About()
   {
      InitializeComponent();
      this.Title = ApplicationStrings.AboutPageTitle;
   }
```

```
   private void SearchButton_Click(object sender, System.Windows.RoutedEventArgs
e)
   {
      signSearchGrid.ItemsSource = haku.SIGNs;
      haku.Load(haku.GetSearchedSignsQuery(searchTextBox.Text));
   }
}
```

Aluksi sovellukselle pitää kertoa, mitä Domain Service Classia tulemme käyttämään. Kun Domain Serviceen luodaan uusi instanssi, pitää ottaa huomioon, että käyttöliittymässä Domain Service Class on muuttunut DomainContext:ksi. Tämän jälkeen luodaan koodi napille, jolla hakukysely tullaan suorittamaan. Aluksi datagridille ilmoitetaan lähde, jota se käyttää tiedonkeruuseen. Tässä tapauksessa lähde on koko SIGN taulu. Kun lähde on määritetty, voidaan kutsua itse kyselyä Load funktiolla. Funktiota luodessa huomataan, että Visual Studio ehdottaa automaattisesti luotuja kyselyitä ja ilmoittaa myös, tarvitseeko kyselyyn viedä arvoja. Tässä tapauksessa viedään tekstikentässä oleva teksti aiemmin luotuun kyselyyn, jonka jälkeen hakutulos tulee näkyviin datagridiin.



Kuva 6.17. Hakutulos sanalla ”po”

7.2 Tiedostonsiirto palvelimelle

Videotiedostot tulevevat alun perin Silverlight-sovellukseen, jotta ne saadaan sivuille käyttöön, pitää ne siirtää palvelimelle. Tiedostonsiirto palvelinpuolelle on toteutettu ASP.NET HTTP Handlerillä, joka on nimetty receiver.ashx:ksi. HTTP Handler:t ovat komponentteja, jotka impelementoivat System.Web.IHttpHandler rajapinnan. Toisin kuin ASP.NET-sivut, niillä ei ole ollenkaan HTML-kuvaustiedostoa, eventtejä tai muita tukia. Niillä on ainoastaan kooditiedosto, joka voidaan kirjoittaa millä tahansa .NET-yhteensopivalla kielellä, joka kirjoittaa jotain dataa palvelimen HTTP-vastaukseen. ASP.NET handlerilla on ".ashx" tiedostopääte, kun taas sivuilla on ".aspx" tiedostopääte.

Viittomien videot lisätään palvelimelle kaikki samaan kansioon, ja ne nimetään viittoman yksilöllisen id:n mukaan. Oheinen ohjelmakoodi ajetaan, kun viittoma lisätään drag&droppia käyttäen.

```
private void DropVideo(object sender, DragEventArgs e)
{
    FileInfo[] droppedFiles = e.Data.GetData(DataFormats.FileDrop) as FileInfo[];
    foreach (FileInfo droppedFile in droppedFiles)
    {
        //lastid comes from the sign id
        string filename = lastid.ToString()+".wmv";
        UploadFile(filename, droppedFile.OpenRead());
    }
}
```

Kun video pudotetaan ohjelmaan DropVideo-metodi kääntää sen bittivirraksi, jonka jälkeen se kutsuu seuraavaa UploadFile-metodia.

```
private void UploadFile(string fileName, Stream data)
{

    UriBuilder ub = new UriBuilder("http://localhost:7899/receiver.ashx");
    ub.Query = string.Format("filename={0}", fileName);

    WebClient c = new WebClient();
    c.OpenWriteCompleted += (sender, e) =>
    {
        PushData(data, e.Result);
        e.Result.Close();
        data.Close();
    };
    c.OpenWriteAsync(ub.Uri);
}
```

Uploadfile yhdistää palvelimelle löytyvään receiver handleriin ja kutsuu seuraavaa PushData metodia, kunnes tiedosto on kokonaan lähetetty. Palvelimeen yhdistämiseen käytetään WebClient luokkaa.

```
private void PushData(Stream input, Stream output)
{
    byte[] buffer = new byte[4096];
    int bytesRead;

    while ((bytesRead = input.Read(buffer, 0, buffer.Length)) != 0)
    {
        output.Write(buffer, 0, bytesRead);
    }
}
```

Palvelimelta löytyvän receiver.ashx:n ohjelmakoodi on yksinkertainen. Se ottaa tiedostoon haluttavan nimen käyttäen QueryString-metodia, jonka jälkeen se luo uuden tiedoston ja kirjoittaa siihen uuden tiedoston pala palalta.

```
public void ProcessRequest (HttpContext context) {
    string filename = context.Request.QueryString["filename"].ToString();

    using (FileStream fs = File.Create(context.Server.MapPath("~/SignVideos/" +
filename)))
    {
        SaveFile(context.Request.InputStream, fs);
    }
}

private void SaveFile(Stream stream, FileStream fs)
{
    byte[] buffer = new byte[4096];
    int bytesRead;
    while ((bytesRead = stream.Read(buffer, 0, buffer.Length)) != 0)
    {
        fs.Write(buffer, 0, bytesRead);
    }
}
```

Tiedoston lähettämisen jälkeen löytyy video Signbookin palvelimelta. Kun käyttäjät tulevat viittoman sivulle, haetaan oikea video sen id:n perusteella.

8. Jatkokehitys

Esiselvitystyön aikana projekti on kehittynyt suunnattomasti. Sen aikana sovellus on kehittynyt osiltaan jo lähelle valmista tuotantoversiota ja uusia ominaisuuksia on keksitty lisää samalla, kun prototyyppi on edennyt

Suurin ongelma, joka esiselvityksessä huomattiin, liittyi videon tallentamiseen. Pähkinänkuoressa ongelma on se, että käyttäessämme webcam tallennuskoodia, joka saatiin PDC09 keynotesta (http://www.silverlight.net/community/samples/silverlight-4/photobooth/) videon tiedoston koko on aivan liian iso (n. 100MB/10s). Se on siis raakavideota, jota pitäisi manipuloida ennen lähetystä palvelimelle.

Jotta videot saataisiin palvelimelle järkevästi, niitä pitäisi Silverlightissa ensin pakata jollain tavalla. Tämän kokoluokan tiedostojen siirtäminen palvelimelle pakattavaksi ei todennäköisesti tule toimimaan. Varsinkaan Azure-arkkitehtuurissa, jossa maksetaan siirretystä datasta. Palvelimella pakkauksen suorittaminen olisi siis sekä hidasta että kallista.

Microsoftilta on kysytty ratkaisua jo useampaan otteeseen niin Suomessa kuin ulkomaillakin. Paras vastaus tällä hetkellä on jonkin open source codecin implementointi sovellukseen. Tämä kuulostaa tosin hyvin hankalalta prosessilta ja varsinkaan C#:lla tehtyä ratkaisua on lähes mahdotonta löytää. Silverlight 4:ssa itsessään ei ole mitään toimintoja videon tallentamiseen webcamista. Ratkaisua on yritetty kysyä myös Silverlight forumeilta, mutta sieltä ei ole saatu minkäänlaista vastausta

Tämä on tällä hetkellä suurin tekninen haaste Signbookin tuotantoversion tekemisessä, joten aiheesta tulee kerätä niin paljon tietoa, kun mahdollista. Mahdollisten C++ codeccien kääntäminen Silverlightille kuulostaa kovin hankalalta prosessilta

Toinen tekninen haaste, joka esiselvityksen aikana ilmaantui liittyi Azure pilvipalveluun. Kirjoitushetkellä Azure ei vielä kykene suorittamaan .NET 4.0:lla tehtyjä sovelluksia. Signbook käyttää keskikerroksellaan .NET 4.0:n ominaisuuksia, eikä sen takia vielä toimi Azure alustalla. Microsoft on luvannut, että .NET 4 tulee toimimaan Windows Azuressa 90 päivän kuluttua sen julkaisusta.

Ongelmien huomaaminen ja miettiminen tässä vaiheessa on kuitenkin positiivista. Esiselvityksessä huomattuihin ongelmiin voidaan tulevaisuudessa reagoida paremmin, kun ne eivät pääse yllättämään tuotantoversion tekemisessä.

Kokonaisuutta katsottaessa oli esiselvitys todella onnistunut. Sillä saatiin paremmin kartoitettua ohjelmiston ominaisuuksia, sekä asiakkaan vaatimuksia. Onnistuneiden kilpailumenestyksen ja median avulla saatiin sovellukselle myös huomattavasti näkyvyyttä sen tulevassa käyttäjäkunnassa.

9. Yhteenveto

Työstä nähdään selkeästi. miten helposti ja tuottavasti voidaan luoda n-kerroksisia rikkaita internet -sovelluksia .NET RIA Servicesillä. Vaikka työssä käytettiinkin Silverlightia käyttöliittymän toteuttamiseen, pitää muistaa, että RIA Services ei ole pelkästään Silverlightiin suunnattu. Koska se kuuluu Windows Communication Foundation -tuoteperheeseen, sitä voidaan käyttää myös muissa toteutuksissa, kuten: ASP.NET, WPF ja MVC

Käyttökokemus on myös parantunut huimasti verrattuna normaaliin asiakas-palvelin verkkosivustoon, kun kaikki yhteydet tietokantaan tehdään asynkronisesti käyttöliittymän pysyessä koko ajan käytettävissä.

Rikkailla internet-sovelluksia voidaan myös toteuttaa sellaisia käyttöliittymiä, johon normaali html/javascript ei yksinkertaisesti pysty. Lisäksi .NET:n suorituskyky verrattuna javascriptiin on ylivoimainen.

Rajoituksena pitää kuitenkin muistaa, että käyttäjällä pitää olla koneellaan Silverlight-liitännäinen asennettuna. Tämä rajoittaa käyttöä esim. mobiililaitteilla, johon Silverlightia ei vielä saa ollenkaan. Tämä voidaan kiertää tekemällä sivustosta myös normaaliversio, joka tosin vaatii tietenkin ekstra määrän työtä.

Vaikka Signbook-sovellus tässä muodossaan onkin vielä vajaa, siitä lienee paljon hyötyä jatkossa. Moni sen toiminnoista on jo valmiita tuotantokäyttöön, mutta osa asiakkaan haluamista toiminnoista puuttuu vielä kokonaan.

LÄHTEET

[1] Rich internet Application Statistics, [www-dokumentti], Saatavilla: http://www.riastats.com/ (Luettu 03.2010)

[2] Moroney Laurence, Introducing Microsoft® Silverlight™ 3, Microsoft Press, 2009

[3] Microsoft Silverlight - Wikipedia [www-dokumentti], Saatavilla: http://en.wikipedia.org/wiki/Microsoft_Silverlight (Luettu 03.2010)

[4] Matthew MacDonald, Pro Silverlight 3 in C#, Apress, 2009

[5] .NET Framework sivusto [www-dokumentti], Saatavilla: http://www.microsoft.com/NET/ (Luettu 03.2010)

[6] Visual Studio - Wikipedia [www-dokumentti], Saatavilla: http://fi.wikipedia.org/wiki/Visual_Studio (Luettu 04.2010)

[7] Microsoft Developer Network, WCF RIA Services [www-dokumentti], Saatavilla: http://msdn.microsoft.com/en-us/library/ee781368.aspx (Luettu 04.2010)

[8] Heuer, Tim, Building Silverlight apps with RIA Services, TechEd Europe 2009

[9] Microsoft Developer Network, Working with RIA Services Solutions [www-dokumentti], Saatavilla: http://msdn.microsoft.com/en-us/library/ee707336(v=VS.91).aspx (Luettu 04.2010)

[10] Microsoft Developer Network, Middle Tier [www-dokumentti], Saatavilla: http://msdn.microsoft.com/en-us/library/ee707348(v=VS.91).aspx (Luettu 04.2010)

[11] Microsoft Developer Network, Silverlight Clients [www-dokumentti], Saatavilla: http://msdn.microsoft.com/en-us/library/ee707349(v=VS.91).aspx (Luettu 04.2010)