Opinnäytetyö (AMK)

Tietotekniikan koulutusohjelma

Hyvinvointiteknologia

2010

Tomi Härkönen

WWW-SOVELLUKSEN TOTEUTUS MICROSOFTIN TEKNOLOGIOILLA

– Dokumenttienhallintajärjestelmä

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Tietotekniikan koulutusohjelma | Hyvinvointiteknologia

Helmikuu 2011 | Sivumäärä 22

Ohjaaja: KTT Reetta Raitoharju

Tomi Härkönen

WWW-sovelluksen toteutus Microsoftin teknologioilla

Opinnäytteen aiheena oli toteuttaa Elomatic Oy:lle heidän EloDoc dokumentinhallintajärjestelmään www-käyttöliittymä eli www-sovellus. EloDoc on Visual Basicillä toteutettu työpöytäsovellus.

Työn alkuvaiheessa tehtiin nopea selvitys mahdollisista teknologioista, ja näiden soveltuvuudesta www-sovelluksen toteutukseen. Pääpaino oli jo tässä vaiheessa Microsoftin tarjoamilla teknologioilla, kuten ASP.NET ja Silverlight.

Www-sovelluksen käyttöliittymästä haluttiin mahdollisimman paljon samanlainen kuin työpöytäsovelluksen käyttöliittymä. Työpöytäsovelluksessa olivat selvät pääelementit, jotka oli helppo siirtää lähes sellaisinaan www-sovellukseen. Käyttöliittymän toteutuksessa käytettiin jQueryn liitännäistä, Layout.UI:tä. CSS:n avulla määritettiin tyylit eri elementeille.

Tekniseltä toteutukselta www-sovellus eroaa työpöytäsovelluksesta suuresti. Www suunnittelussa joudutaan huomioimaan, että käyttäjän tekemisiä ei voida seurata niin tarkasti. Kontrollien luonti taustakoodissa on myös haastavampaa www-sovelluksessa, koska ne joudutaan luomaan uusiksi jokaisen sivulatauksen yhteydessä.

Www-sovellusta ei saatu valmiiksi opinnäytetyön kirjoittamisen aikana. Www-sovellus kuitenkin saatiin hyvälle mallille ja kehitys jatkuu opinnäytetyön jälkeen.

ASIASANAT:

WWW, ASP.NET, C#, dokumenttienhallinta

BACHELOR'S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Information Technology | Health Informatics

February 2011 | Number of pages 22

Instructor: Reetta Raitoharju, D.Sc.

Tomi Härkönen

Designing internet application with Microsoft technologies

Subject of the thesis was to make internet application for Elomatic Ltd. based on their EloDoc document management software. EloDoc is a desktop application made with Visual Basic.

At the beginning of the thesis there were quick research about what technology to use and how that would work in internet environment. Even on this point main focus was on Microsofts technologies, like ASP.NET and Silverlight.

User interface of internet application was supposed to look much like the one on desktop application. There were few main components that were easy to transfer into internet application. Layout was done with jQuery UI.Layout plugin and styles with CSS.

From technical aspect there are big differences when comparing internet application to desktop application. When designing intenet application, you need to take account that you cant follow users actions so closely. Also creating controls on code-behind is much more challenging than in desktop application. In internet application you need to create every controller again on every postback.

Internet application wasn't finished during the thesis. Internet application is on good stage, and development continues.

KEYWORDS:

Internet application, ASP.NET, C#, Document management

SISÄLTÖ

KUVAT

KUVIOT

KOODIT

SYMBOLI- JA LYHENNELUETTELO

AJAX	(Asynchronous JavaScript And XML) Nimitys joukolle www-sovelluskehitys tekniikoita
ASP	(Active Server Pages) Dynaaminen palvelinpuolen
CSS	(Cascading Style Sheet) Www-sivuille kehitetty tyylikieli
HTML	(Hypertext markup language) Www-sivujen kuvauskieli
RIA	(Rich Internet Application) Työpöytäsovelluksen kaltainen www-sovellus
.NET	Microsoftin kehittämä ohjelmistokomponenttikirjasto

1 Johdanto

Elomatic kuuluu Skandinavian johtaviin suunnittelutoimistoihin ja on merkittävä ohjelmistotalo maailmanlaajuisesti. EloDoc on Elomaticin kehittämä dokumenttienhallintajärjestelmä. Opinnäytteessä perehdytään mikä on EloDoc ja miten lähdettiin kehittämään sen rinnalle www-sovellusta.

Työn tarkoituksena on toteuttaa www-sovellus jo olemassa olevan työpöytäsovelluksen rinnalle. EloDoc dokumenttienhallintajärjestelmän tarkoituksena on koota jonkin projektin kaikki dokumentit yhteen paikkaan, josta ne ovat helposti löydettävissä. Sovelluksen toiminnallisuus olisi tarkoitus siirtää selaimesta käytettäväksi. Www-sovelluksen on tarkoitus tukea työpöytäsovellusta.

Sovellus toteutetaan Microsoftin teknologioilla, kuten ASP.NET ja C#. Kehitysympäristönä käytetään Visual Studio 2010:ä. Koska kehitysympäristö on täysin erillainen verrattuna EloDoc työpöytäsovellukseen, kaikkia ominaisuuksia ei pyritä edes siirtämään www-sovellukseen. Ainakin alkuun pyritään tukemaan tärkeimmät toiminnot ja varmistamaan, että nämä toimivat moitteetta. Www-sovelluksen kehityksessä pystytään käyttämään hyödyksi työpöytäsovelluksen koodeja jossain määrin. Suurin osa toiminnoista tapahtuu EloDoc palvelimella, joten sovelluksesta lähetetään paljon pyyntöjä palvelimelle, joka palauttaa vastauksen.

2 Teknologiat

Toteutus teknologioiksi valittiin hyvinkin nopeasti Microsoftin tarjoamat teknologiat. Sovellus jonka rinnalle www-sovellusta oltiin tekemässä, oli toteutettu Visual Basicilla ja uudistettiin myöhemmin VB.NET kieleen. Muut kuin Microsoftin tarjoamat vaihtoehdot hylättiin, koska niillä ei olisi saanut yhtä helposti hyödynnettyä jo olemassa olevia rajapintoja. Olisi myös ollut turhaa sekoittaa asioita entisestään eri valmistajien tarjoamilla ratkaisuilla.

Microsoftilla oli tarjolla kaksi eri ohjelmointikieltä, jotka soveltuivat tähän tarkoitukseen, VB.NET ja C#. [1] Näiden lisäksi oli syytä miettiä millainen www-sovelluksesta halutaan

eli millä arkkitehtuurilla sitä lähdetään totteuttamaan. Vaihtoehtoina olivat ASP.NET, ASP.NET MVC ja Silverlight. Näiden eduista ja haitoista tehtiin lyhyt kartoitus.

2.1 .NET

.NET Framework on Microsoftin kehittämä alusta, ohjelmistokomponenttikirjasto, sovellusten kehittämiseen. Sen avulla voidaan kehittää sovelluksia moniin eri ympäristöihin, mm. työpöytä, www ja mobiili. Se koostuu pääasiassa kolmesta eri komponentista: [2]

- Common Language Runtime (CLR) tuotetun koodin kääntäminen binäärikoodiksi
- Base Class Libraries esikäännetyt koodit useimmin käytetyille toiminnoille
- Development frameworks and technologies uudelleen käytettävät ratkaisut ohjelmointiin

.Net ohjelmointikieliä ovat mm. C#, VB.NET, C++.NET ja F#. Näistä käytetyimmät ovat C# ja VB.NET. [3]

2.1.1 C#

Tähän projektiin valittiin ohjelmointikieleksi C#. Valinnalle ei oikeastaan löydy mitään yksiselitteistä syytä. Aiemmin oli jo päätetty, että pysytään .NET ohjelmointikielissä. VB.NET ja C# eivät eroa ominaisuuksiensa puolesta juuri ollenkaan.[4] Molemmat kielet on julkaistu 2001, vaikkakin VB.NET on kehitetty Visual Basicista, joka julkaistiin jo vuonna 1991. C# on oikeastaan Microsoftin kehittämä vastine Javalle, ainakin syntaksin osalta. Syntaksissa ilmenee suurimmat erot VB.NETin ja C#:n välillä. VB.NET käyttää syntaksissaan paljon sanoja ja on helpommin luettava. C# sen sijaan käyttää erilaisia merkkejä ja termistö on vaikeampaa, esimerkkinä ehtolauseke:

```
If condition Then
    ' condition is true
End If
```

Yllä on esimerkki VB.NET syntaksista.

```
if (condition)
{
    // condition is true
}
```

Yllä on esimerkki C# syntaksista.

C# on kehitetty yksinkertaiseksi, moderniksi ja yleispäteväksi olio-ohjelmointikieleksi. Kielen oppiminen on suunniteltu helpoksi, jos on jo aiempaa kokemusta C-, C++- tai Java-kielistä. Ennen julkaisua C#:sta käytettiin nimitystä ”Cool” (C-like Object Oriented Language), eli C:n kaltainen olio-ohjelmointikieli. Kieli soveltuu käytettäväksi monenlaisissa ympäristöissä, esim. Windows työpöytäkäyttöjärjestelmissä tai sitten Windows Phone 7 - puhelinkäyttöjärjestelmissä. [4]

C#-ohjelmointikieleen päädyttiin oikeastaan aikaisemman kokemuksen vuoksi. Vaikka sovellus, jonka pohjalta www-sovellusta lähdetään tekemään, onkin tehty VB.NET ohjelmointikielellä, niin tästä ei koidu juurikaan ongelmia. Www-sovellusta kehitettäessä on jopa mahdollista käyttää saman projektin sisässä yhtä aikaa C#- ja VB.NET-ohjelmointikieliä.

2.1.2 ASP.NET

ASP.NET on Microsoftin ratkaisu www-sisällön tuottamiseen, kuten dynaamisia internetsivuja, www-sovelluksia ja www-palveluita. Nykyisin käytössä oleva ASP.NET pohjautuu ASP:iin, joka on julkaistu jo vuonna 1998. Uusin versio pohjautuu pitkälti ASP.NET versioon 2.0. Se toi mukanaan suuren määrän uudistuksia, ja se julkaistiin yhdessä Visual Studio 2005:n kanssa. Keväällä 2011 uusin versio on 4.0, joka on pitkälti samaa tekniikkaa kuin 2.0, mutta lukuisia korjauksia ja päivityksiä sisältävä. [5]

ASP.NET:in pääpiirteitä ovat www-lomakkeet, taustakoodi, direktiivit, käyttäjän luomat kontrollit ja tilan hallinta.

Www-lomakkeet (eng. web forms) ovat sivuja. Nimitys juontaa juurensa Windows-sovelluskehityksestä, jossa sovellus luodaan aina lomakkeelle. ASP.NET:ssä sivujen luonti on lähes vastaavaa kuin sovellusten luonti. On mahdollista käyttää Drag&Drop –menetelmää, tai sitten voi perinteiseen tapaan kirjoittaa kaiken. Myös komponenttien

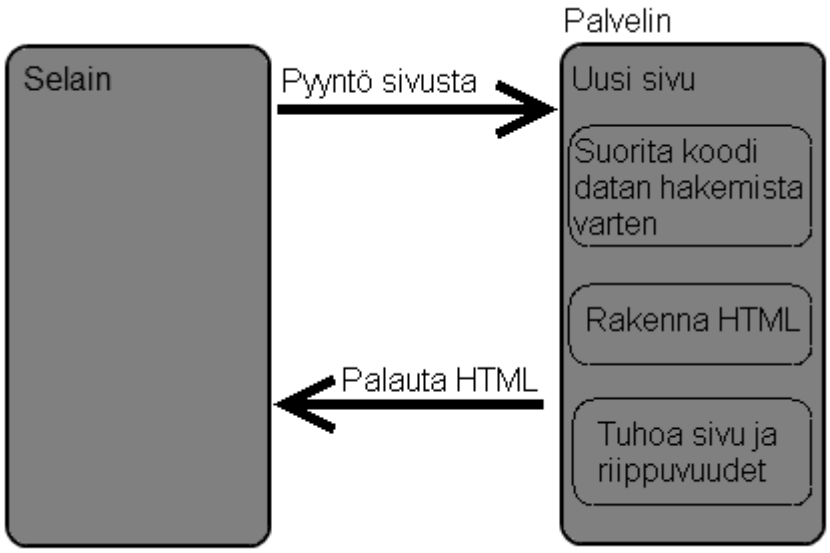
konfigurointi on mahdollista graafisen käyttöliittymän avulla. Sivujen tiedostopääte on ASPX.

Taustakoodi (eng. code-behind model) mahdollistaa ulkoasun ja toiminnallisuuden erottamisen. Eli kun käyttöliittymä luodaan, vaikka Default.ASPX tiedostoon, tähän tiedostoon on liitettynä joko Default.ASPX.CS tai Default.ASPX.VB tiedosto. CS ja VB viittaavat käytettyyn ohjelmointikieleen, C# ja VB.NET. Näin taustakoodissa voidaan sitten reagoida eri tapahtumiin, esim. kun sivu ladataan tai käyttäjä painaa jotain nappia.

Direktiivit mahdollistavat erilaisten asetusten määrittämisen sivukohtaisesti. Kuten sallitaanko datan lähettäminen sivulta palvelimelle tai mitä ohjelmointikieltä käytetään taustakoodissa.

Käyttäjien luoma kontrolli (eng. user control) on oikeastaan vain jokin jo olemassa oleva kontrolli tai kontrolleja, jotka rekisteröidään yhdeksi paketiksi. Tätä pakettia voidaan kustomoida tarpeen mukaan ja käyttää useita kertoja eri sivuilla. Esimerkiksi jos kehittäjä tarvitsee usein tekstikenttää, joka sallii syötettävän vain päivämääriä, hän voi luoda erillisen kontrollin sitä varten.

Www-sovelluksia kehittäessä ongelmana on usein, että ei tiedetä mitä käyttäjä tekee sivulla. Www on niin sanotustit tilaton, eli käyttäjä pyytää palvelimelta sivun ja se toimitetaan. Tämän jälkeen käyttäjä voi tehdä sivulla monia eri toimintoja, joista palvelimella ei tiedetä. Seuraava tieto välittyy palvelimelle vasta, kun käyttäjä painaa lähetä-nappia tai siirtyy toiselle sivulle. [Kuvio 1.] Tällöin on mahdollista tarkistaa, mitä sivulla on tapahtunut. Tähän seurantaan ASP.NET tarjoaa hyvät työkalut, kuten ViewState. ViewState säilöö automaattisesti käyttäjän tekemät muutokset, ja lähettää ne palvelimelle. Tässä on haittana siirrettävän datan lisääntyminen, varsinkin jos kyseessä on sivu, jolla on paljon eri kenttiä.



Kuvio 1. ASP.NET sivun elinkaari.

2.2 Ajax

Ajax tulee sanoista Asynchronous JavaScript And XML. Kuten nimestäkin voi jo päätellä, niin tekniikkaan liittyy asynkroniset siirrot ja javascripti. Xml ei ole nykyisin enää ainut vaihtoehto. Ajax tarkoittaa siis datan siirtämistä asynkronisesti käyttäen XMLHttpRequest-objektia ja DOMia (Document Object Model). Normaalisti kun www-sivulle halutaan päivittää jotain tietoa, koko sivu pitää ladata uudestaan. Ajax mahdollistaa, että jokin tietty objekti sivulla päivitetään. Tähän käytetään yhdistelmää edellä mainituista tekniikoista. XMLHttpRequest hoitaa datan siirtämisen ja JavaScript käsittelee DOMia, jotta uusi tieto saadaan näkyviin sivulle haluttuun kohtaan. [5]

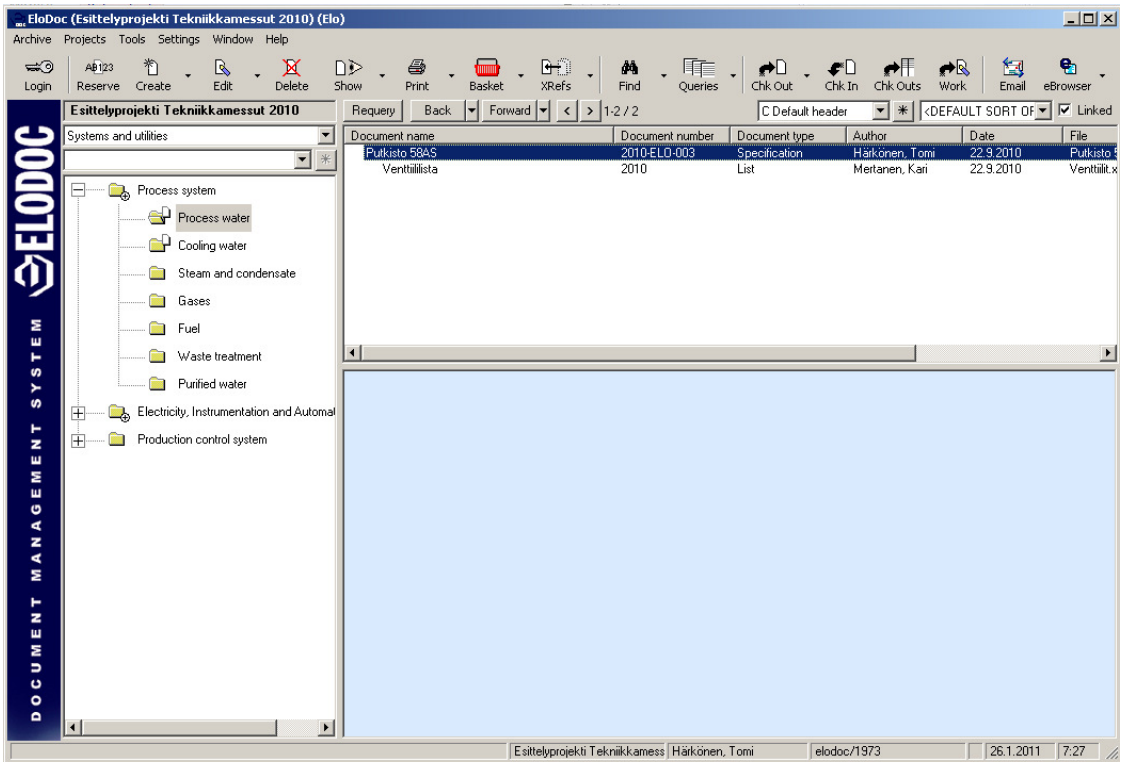
Ajax on nykyisin lähes pakollinen implementaatio www-sivuille ja –sovelluksille, varsinkin jos halutaan sujuva käyttökokemus. Hyvänä esimerkkinä voidaan mainita Facebook, joka käyttää Ajaxia erittäin tehokkaasti. Tilapäivitykset, uusien viestien hakeminen, chat ja monet muut toiminnot pohjautuvat Ajaxiin. Kaikki nämä toiminnot tapahtuvat ilman, että käyttäjän pitäisi ladata koko sivua uudestaan. Datan lähetys on myös mahdollista, esim. edellä mainittu tilapäivitys. Käyttäjä klikkaa lähetys-nappia, ja data siirtyy palvelimelle. Edelleen voidaan käyttää Ajaxia, jos halutaan näyttää tämä juuri lähetetty viesti käyttäjälle. Vieläkään ei tarvita koko sivun uudelleen latausta.

Javascriptiä käytetään datan esittämiseen, mutta siihen voidaan käyttää mitä tahansa asiakaspuolen skriptikieltä, mutta Javascript on tuettu lähes kaikissa selaimissa ja

tämän vuoksi käytetyin. Ajaxista monelle tulee mieleen animoidut sivustot, tämä on kuitenkin hieman pelkistetty ajatusmalli. Ajax ei vaadi minkäänlaista animointia toimiakseen, sen sijaan Javascript mahdollistaa animoinnit ja koska javascript vaaditaan Ajaxin käyttöön, niin animointia käytetään usein Ajax sivuilla. Javascriptillä voidaan siis toteuttaa helposti liukuvia tai laajentuvia paneeleita, katoavaa tekstiä, ominaisuuksien poistamista käytöstä ja näppäinpainaluksiin reagointi.

3 EloDoc-dokumenttienhallintajärjestelmä

EloDoc on Elomaticin kehittämä dokumenttienhallintajärjestelmä, tarkoituksena parantaa dokumenttien käsittelyä, hallintaa ja arkistointia. Sovellus on tarkoitettu ylläpitämään jonkin projektin dokumentit eri versioineen koko projektin elinkaaren ajan, aina esisuunnittelusta ylläpitöön asti. Tavoitteena on myös, että dokumenttien hallinta paranee. Tarvittavat tiedot ja oikea dokumentti löytyy helpolla, lukuisten eri luokittelujen ja järjestelyjen avulla. Parhaimman hyödyn saa, kun kaikki projektin dokumentit talletetaan EloDociin, josta ne on helposti hyödynnettävissä. [7]



Kuva 1. EloDoc:in päänäkymä

EloDocin päänäkymässä näkyy, kirjautumisen jälkeen, haluttu projekti, projektiin kuuluva hierarkia, hierarkian dokumentit ja valitun dokumentin dokumenttikortti, ks. kuva 1. Näiden lisäksi löytyy työkalurivi, tilarivi ja päämenu.

Halutun dokumentin etsimiseen on monia tapoja, kenttäarvon perusteella, sekalaisten kenttien perusteella, dokumentin SysID:n perusteella tai dokumenttikortin avulla. Lisäksia hakua voidaan rajata edelleen hakemalla haun tuloksista toisella hakuehdolla. Lisäksi on mahdollista tallentaa haku, jotta se on helposti käytössä seuraavalla kerralla. Kaikilta käyttäjiltä löytyy tallennetuista hauista Omat yksityiset dokument sekä Kaikki dokumentit.

Dokumentin luonti tapahtuu Create-napista, josta aukeaa ensimmäiseksi tiedoston valintaikkuna. Vaihtoehtoisesti tämä voidaan tehdä raahaamalla tiedosto hierarkian kansioon. Seuraavaksi dokumentille määritetään dokumenttinumero. Numero tulee joko automaattisesti määrätyltä alueelta tai sitten voidaan käyttää aiemmin varattua numeroa. On myös mahdollista antaa poikkeuksellinen dokumenttinumero, joka ei vastaa mitään ennaltamääriteltyjä sääntöjä.

Kun dokumentin dokumenttinumero on valittu, valitaan dokumenttikortti. Dokumenttikortteja voidaan luoda erillaisia sen mukaan, millaisia tietoja tarvitsee tallentaa. Jos tiedosto raahattiin hierarkian kansioon, niin dokumenttikortissa saattaa olla jo jotain arvoja valittuna, muussa tapauksessa dokumenttikortti on tyhjä. Dokumenttikorttii on tarkoitus tallentaa tiedot jotka ovat ollennaisia siihen liittyviin tiedostoihin.

Dokumentissa on myös versiointi ominaisuus. Dokumenttia voidaan päivittää, ja aiemmat tiedostot säilyvät, mutta eivät oletuksena avaudu, mutta tarvittaessa niihin pääsee käsiksi. Versiointiin on mahdollista myös liittää paperiviitteitä, nämä ovat vähän kuin fyysinen paperiviite. Myös aliversiointi on mahdollista.

Dokumentti voidaan merkitä ulos, jolloin sitä ei voi muut työstää. Tämä tapahtuu valitsemalla halutun dokumentin kohdalla Check Out –nappi. Aukeaa versioikkuna, josta valitaan haluttu tiedosto. Ulos merkattu tiedosto talletetaan oletuksena EloDocCheckOuts-kansioon, joka luodaan käyttäjän Omat dokumentit –kansioon.

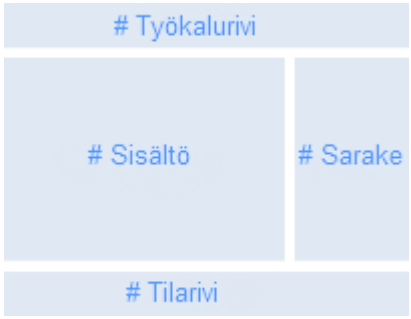
EloDocista löytyy lukuisia muitakin toimintoja, kuten koko projektin kopiointi paikalliseksi, ulkoiset ja sisäiset linkit, linkitys CADMATIC eBROWSER:n kanssa, XReffit ja tulostustaminen.

4 Www-kehittäminen

Www-sovelluksen kehittämisen haastavuus piilee lukuisissa muuttujissa. Kun kehitetään sovellus työpöytäkäyttöön, voidaan sen käyttöympäristö rajata melko tarkkaan. Esimerkiksi voidaan määrittää missä käyttöjärjestelmässä ohjelma toimii, määrittää vähimmäisresoluutio, jolla ohjelma käynnistyy tai seurata käyttäjän jokaista painallusta ohjelmassa. Nämä on mahdollista toteuttaa www-sovelluksessa, mutta saattavat hankaloittaa käyttökokemusta ja voivat aiheuttaa turhia esteitä sovelluksen käytölle.

Www-sovellusta voi nykyisin käyttää hyvin monelta erilaiselta alustalta, mm. tietokoneet, tablet-laitteet ja kännykät. Näillä jokaisella on oma käyttöjärjestelmänsä ja selaimensa. Parhaiten yhteensopivuuden voi varautua seuraamalla standardeja käyttöliittymä suunnittelussa. W3C järjestö määrittää mm. html ja css standardit.[8] Näitä seuraamalla saa usein monella alustalla toimivan käyttöliittymän, mutta ei kuitenkaan aina ja olisi syytä testata mahdollisimman moni alusta ennen www-sovelluksen julkistamista. Lisäksi on hyvä pohtia, että onko tarpeen tarjota tukea lainkaan mobiilialustoille.

Asettelu saattaa koitua yllättävänkin haastavaksi vaiheeksi, varsinkin jos aiempaa kokemusta www-sivun asettelusta ei ole. Www-sovelluksesta halutaan usein koko selainikkunan täyttävä. Yläreunasta löytyy työkalurivi, vasemmalta mahdollisesti sarake, jossa lisää työkaluja ja alareunasta tilarivi. Näin keskelle jää tila, jossa esitetään suurin osa halutusta datasta. Tämän keskellä olevan tilan pitää mukautua selainikkunan koon muutoksiin, työkalurivin, vasemman sarakkeen ja tilarivin pysyessä saman kokoisina. Ongelmaksi muodostuu usein tilarivi, ja sen asettaminen sivun alareunaan.[9]



Kuvio 2. Www-sovelluksen asettelu

Käyttöliittymän suunnitteluun liittyy nykyisin vahvasti JavaScript ja tämän mahdollistama Ajax. Koska JavaScript on asiakaspuolen skriptikieli, niin se saattaa toimia eri ympäristöissä hieman eri tavalla. Hyvä lähtökohta on tehdä käyttöliittymä toiminta kuntoiseksi ilman javascriptiä, kun kaikki toimii, niin voidaan käyttöliittymää parantaa javascriptin avulla. Näin ollen jos käyttäjällä on estetty javascriptin suorittaminen tai hänen selaimensa ei tue käytettyjä toimintoja, pystyy www-sovellusta käyttämään silti. Myös kosketusnäytölliset laitteet on syytä huomioida tässä vaiheessa, koska niillä ei usein pysty käyttämään toimintoja, jotka vaativat kursorin viemistä jonkin elementin päälle. [10]

5 EloDoc Web

EloDoc Web pohjautuu siis EloDoc-sovellukseen. Tarkoituksena tarjota lähes samat ominaisuudet webkäyttöliittymän kautta. Alkuvaiheessa tärkeintä on saada perustoiminnat kuntoon, kuten dokumenttien selaus ja haku, dokumenttikortin muokkaus ja tallentaminen, tiedostojen lataus ja tallentaminen. Hallintaominaisuuksia ei luultavasti tuoda ollenkaan webympäristöön.

5.1 Alkukartoitus

Ensimmäisenä tuli valita teknologia, jolla lähdettäisiin toteuttamaan webkäyttöliittymää. Tämä siksi, että valittu teknologia tulisi vaikuttamaan vahvasti millainen käyttöliittymästä tulisi. Seuraavista teknologioista tehtiin edut ja haitat listaus.

ASP.NET

Määriteltiin teknologiaksi, jolla luultavasti lähdettäisiin toteuttamaan webkäyttöliittymää. Mahdollistaa jo olemassa olevien VB.NET koodien hyödyntämisen. Etuina listattiin mm. skaalautuvuus, turvallinen, vastaa jossain määrin työpöytäsovelluksen kehittämistä, tarjolla paljon valmiita komponentteja. Haittoina mm. vaatii Windows palvelimen, ilman Ajaxia toteutuksesta ei saisi käyttäjäystävällistä.

ASP.NET MVC

Microsoftin uusi lisäys ASP.NET:iin, tulee sanoista Models, Views ja Controllers. [11] Ideana että toteutus jaetaan osa-alueisiin, jotka ovat yhteydessä toisiinsa. Etuina oli mm. samat ominaisuudet kuin ASP.NET, tehokas URL mappingin hyödyntäminen,

helppo testata. Haitoiksi listattiin mm. uusi teknologia, MVC tekniikan hyödyntäminen EloDocissa, vaikea oppia.

Silverlight

Silverlightia pidettiin hyvin soveltuvaksi tähän projektiin. Silverlight on Microsoftin kehittämä selainlaajennus, joka mahdollistaa RIA toteutukset helposti. [12] Epäilyksiä kuitenkin herätti teknologian uutuus ja onko tarjolla kaikkia tarvittavia komponentteja. Etuina listattiin mm. RIA:n helppous, sama näkymä kaikilla selaimilla, C#. Haittoina mm. vaatii liitännäisen asentamisen, uusi teknologia, vaatii koneelta hieman tehoa.

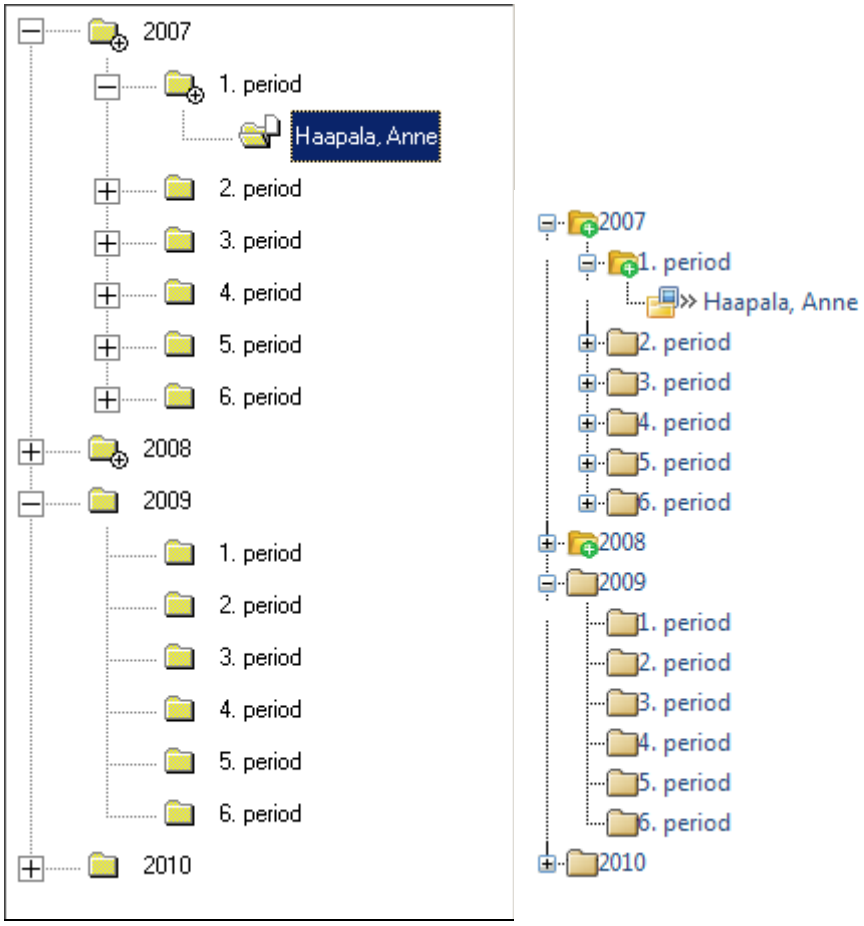
Myös PHP ja JAVA käytiin läpi, mutta näistä luovutiin koska yhteensopivuus olemassa olevien komponenttien kanssa olisi luultavasti tuottanut ongelmia.

5.2 Työpöydältä selaimeen

Ensimmäinen tärkeä asia oli vertailla näitä kahta ympäristöä keskenään. Perinteisellä työpöytäsovelluksella on mahdollista toteuttaa paljon monimutkaisempia asioita huomattavasti helpommin. Koska työpöytäsovellus oli jo olemassa, oli tästä helppoa lähteä käymään läpi siinä käytettyjä komponentteja. Pikaisella läpikäynnillä, mikään komponentti ei olisi mahdoton toteuttaa www-sovelluksessa.

Testausta varten loin Visual Studioon projektin, joka ei ollu millään tavalla yhteydessä EloDocin palvelimiin. Projektin lisäksi loin pienen tietokannan, jonka oli tarkoitus tarjota samankaltaista dataa kuin EloDoc. Näin oli helppo tehdä yksinkertaisia testiversioita.

Suurta osaa EloDocin käytöstä näyttelee puurakenne, taulukko ja dokumenttikortti. Puurakenteessa esitetään projektin hierarkia. Näitä hierarkioita voi olla yhdessä projektissa useita, joten niiden vaihto pitää olla helppoa. EloDocissa oli käytetty pudotusvalikkoa, ja samaan päädyttiin www-sovelluksessa. Valitsemalla hierarkian pudotusvalikosta, puunäkymä päivittyy vastaamaan valittua hierarkiaa. Puunäkymää ei rakenneta missään vaiheessa kokonaan, vaan haaroja lisätään tarvittaessa. Aluksi ladataan vain päätasot, ja merkataan onko niitä mahdollista laajentaa. Mikäli käyttäjä haluaa laajentaa tason, niin ladataan sinne sisältö sisältö klikkauksen yhteydessä. Lisäksi puunäkymässä näkyy pienet ikonit, joista selviää löytyykö tasolta tiedostoja, tai onko sen alitasoilla tiedostoja. Puunäkymän toteutus onnistui hyvin ASP.NET:n TreeView komponenttilla.



Kuva 2. Treeview EloDoc sovelluksesta ja www-sovelluksesta.

Seuraavana suuri rooli on taulukkonäkymällä, johon listataan dokumentit. Sisältö tulee sillä perusteella, mikä taso on valittu puunäkymästä. Testiympäristössä käytin datan liittämiseen ADO.NET komponentteja, joka toimi hyvin. Ongelmaksi muodostui myöhemmin se, että EloDocin data ei tule sopivassa formaatissa. Tämä ratkaistiin luomalla EloDocin tarjoamasta datasta DataTable taustakoodissa ja liittämällä tämä taulukkonäkymään. Taulukkossa oli tärkeää saada toimimaan klikkaus rivikohtaisesti. Klikkauksella avattiin dokumentti näkymään dokumenttikorttiin. Tämä toteutettiin aktivoimalla taulukon valintaominaisuus, joka normaalisti luo uuden solun, jossa on Valitse-nappi. Sitten tämä toiminallisuus kaapataan taulukon luonnin yhteydessä ja liitetään koskemaan koko riviä.

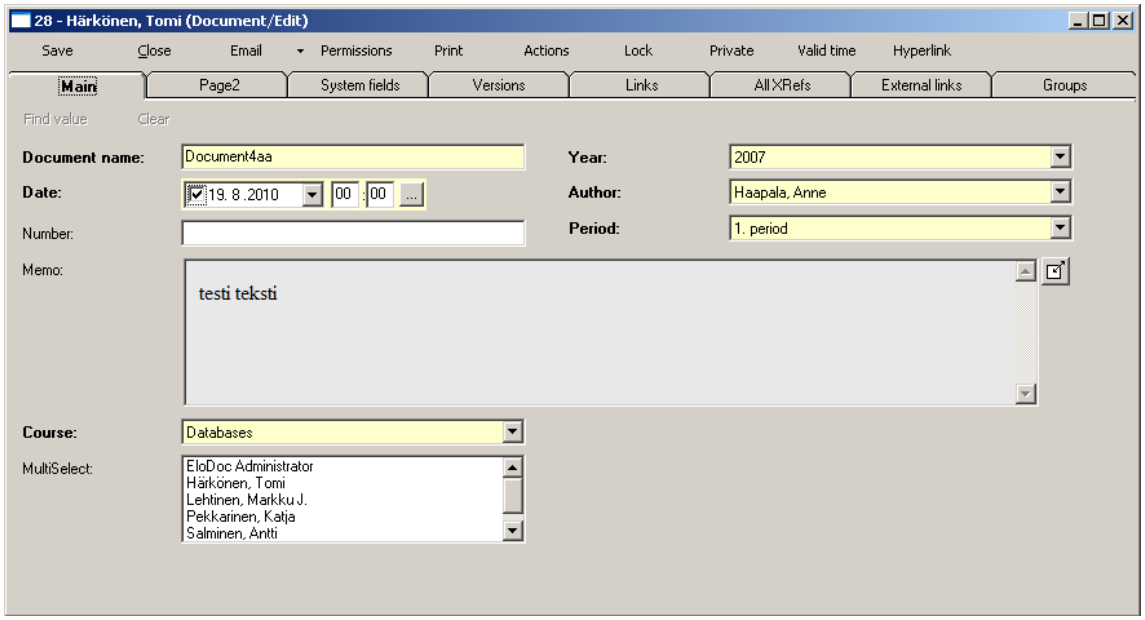
Document name	Date	Period	Last update date
Document4Edit	19.8.2010 20:33:00	1. period	20.1.2011 9:51:33
Document4aa	19.8.2010	1. period	31.1.2011 9:36:19
Document4aa	19.8.2010	1. period	31.1.2011 9:34:58
Document2bb	19.8.2010	1. period	31.1.2011 9:35:08
Document2cc	19.8.2010	1. period	31.1.2011 9:35:19
Document2dd	19.8.2010	1. period	31.1.2011 9:35:40
Document2ee	19.8.2010	1. period	31.1.2011 9:35:52
Document2ff	19.8.2010	1. period	31.1.2011 9:36:07
Document4gg	19.8.2010	1. period	31.1.2011 9:36:38
Document4rr	19.8.2010	1. period	13.10.2010 12:42:27

Kuva 3. Taulukko EloDoc sovelluksesta.

Document name	Date	Period	Last update date
Document4Edit	19.8.2010 20:33:00	1. period	20.1.2011 9:51:33
Document4aa	19.8.2010	1. period	31.1.2011 9:36:19
Document4aa	19.8.2010	1. period	31.1.2011 9:34:58
Document2bb	19.8.2010	1. period	31.1.2011 9:35:08
Document2cc	19.8.2010	1. period	31.1.2011 9:35:19
Document2dd	19.8.2010	1. period	31.1.2011 9:35:40
Document2ee	19.8.2010	1. period	31.1.2011 9:35:52
Document2ff	19.8.2010	1. period	31.1.2011 9:36:07

Kuva 4. Taulukko www-sovelluksesta.

Dokumenttikortti ei ole perinteinen komponentti, kuten edelläkäydyt puunäkymä ja taulukkonäkymä. Dokumenttikortti koostuu oikeastaan useista peruskomponenteista, kuten tekstikentistä, pudotusvalikoista ja valintakentistä. Nämä sitten näytetään ennaltamääritetyssä järjestyksessä paneelilla. EloDoc palvelin antaa listan komponenteista, jotka ovat dokumenttikortilla, ja näiden arvot. Lisäksi tulee tieto, sijoitetaanko komponentti ensimmäiseen vai toiseen sarakkeeseen.

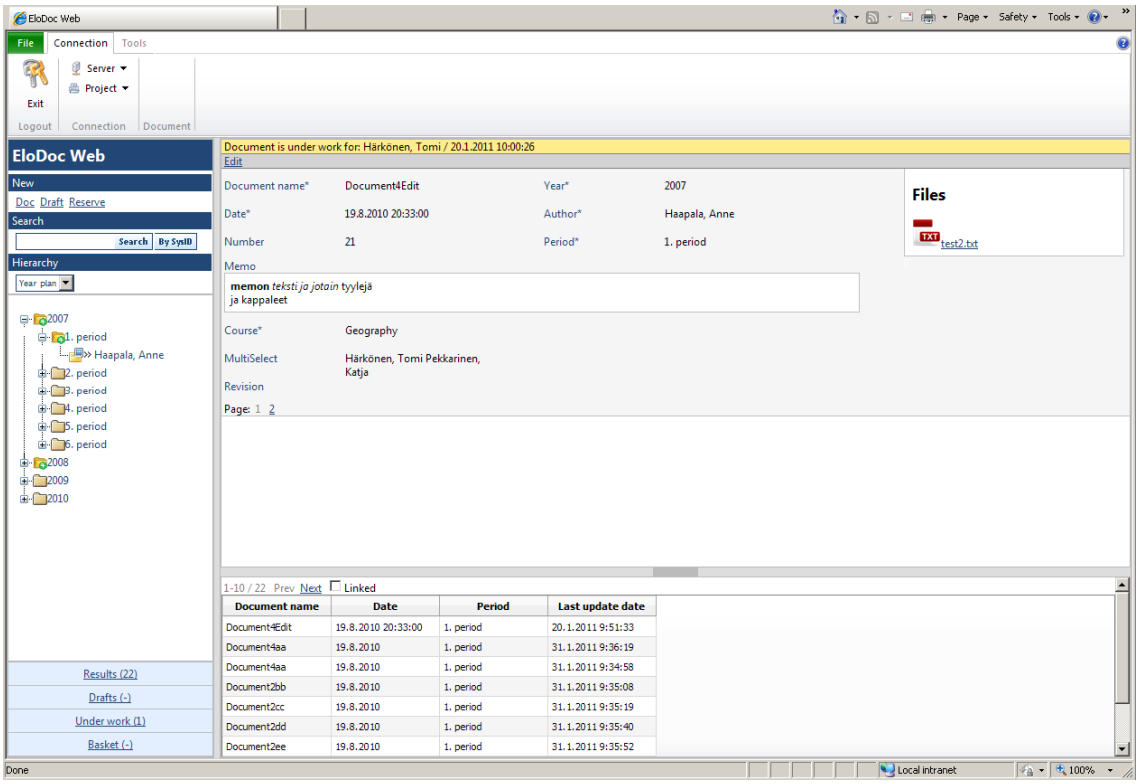


Kuva 5. Dokumenttikortti EloDoc sovelluksesta.

5.3 ASP.NET User Controls

ASP.NET tarjoaa kattavan valikoiman erillaisia kontrolleja, aina peruskontrolleista hieman erikoisempiin, mm. tekstikenttä, pudotusvalikko, taulu ja erikoisemmista esim. tiedoston lähetys, mainoselementti ja kalenteri. Näillä pääsee jo erittäin pitkälle www-sovelluksen luonnissa. Joskus saattaa kuitenkin tulla vastaan tilanne, että näillä valmiilla kontrolleilla ei saada aikaan haluttua toteutusta. User Control mahdollistaa valmiiden kontrollien muokkaamisen omaan käyttöön soveltuviksi. EloDocissa tätä ominaisuutta käytettiin dokumenttikortin kontrolleissa, jotka ovat teksti, aika, numero, valinta, lista, monivalinta ja muistio. Dokumenttikortissa käytettiin näitä kustomoituja kontrollereita.

5.4 WWW-sovelluksen luonti



Kuva 6. EloDoc Web.

5.3.1 Kirjautuminen

Kirjautuminen tapahtuu, kun käyttäjä ensi kertaa avaa sivun, jolloin hänelle esitetään kirjautumisikkuna. Käyttäjänimi ja salasana riittää, myöhemmin tulee luultavasti ominaisuudeksi, että voidaan käyttää Windows-tunnuksia kirjautumiseen. Ilman kirjautumista, ei ole mahdollista päästä käsiksi www-sovellukseen.

Käyttäjän syötettyä kirjautumistiedot ja painettua Kirjaudu-painiketta. Tarkistetaan että ne ovat oikeat ja mihin projekteihin löytyy oikeudet. Mikäli tiedot ovat oikein ja käyttäjältä löytyy joitain projekteja, niin luodaan istunto, johon talletetaan tarpeelliset tiedot. Istuntoon talletetaan EloDocin käyttämä EloDocClient.Session_Client luokka. Käyttäjä siirretään automaattisesti www-sovellukseen. Kirjautumissivua ei näytetä, jos käyttäjä on jo aiemmin kirjautunut ja aiempi istuntoi ei ole vielä vanhentunut.

5.3.2 Aloitusnäkymä

Www-sovelluksen päänäkymä on Default.aspx–sivulla. Ensimmäiseksi taustakoodissa tarkistetaan onko käyttäjä kirjautunut. Tämä katsotaan istunnosta, jos koko istuntoa ei löydy, käyttäjä ohjataan kirjautumiseen. Istunto talletetaan tämän jälkeen muuttujaan, tässä tapauksessa pSession nimiseen. Ensimmäisellä avauskerralla katsotaan löytyykö istuntoa, josta selviää viimeksi avoinna ollut projekti ja hierarkia. Jos istunto löytyy, avataan edellinen projekti ja hierarkia. Muussa tapauksessa avataan ensimmäinen projekti, jota tarjotaan. Automaattinen projektin avaaminen todettiin pakolliseksi kirjautumisen yhteydessä. Aluksi tämän tilalla oli mahdollisuus valita listasta projekti, joka halutaan avata, mutta ongelmana oli sovelluksen kaatuminen, jos käyttäjä jäi miettimään hetkeksikin valintaansa. Tähän ei löytynyt syytä, mutta ongelman pystyi kiertämään yksinkertaisella ratkaisulla, eikä käyttäjälle koidu ylimääräisiä klikkauksia.

5.3.3 Projektin avaus

Kirjautumisen yhteydessä listataan istuntoon projektit, joihin käyttäjällä on oikeus. Näistä luodaan sivulle pudotusvalikko, jossa on listattuna kaikki projektit. Piilotettuna arvona on projektin tunnus, jota käytetään projektin avamiseen. Valittuaan pudotusvalikosta projektin, taustakoodissa suoritetaan projektin avaus. Avaamisen yhteydessä listataan toiseen pudotusvalikkoon projektista löytyvät hierarkiat. Hierarkia pudotusvalikossa käytetään DataView muuttujaa datan liittämiseksi listaan. Tämä siksi, että saadaan kaksi arvoa yhteen kenttään. Tässä tapauksessa nimi ja tunnus. [Koodi 1.]

```
ICollection CreateDataSource2()
{
    // Create a table to store data for the DropDownList control.
    DataTable dt = new DataTable();
    // Define the columns of the table.
    dt.Columns.Add(new DataColumn("NameField", typeof(String)));
    dt.Columns.Add(new DataColumn("ValueField", typeof(String)));
    // Populate the table with values.
    int i = 0;
    foreach (int tempHierarchy_loopVariable in pSession.Hierarchies.HierarchyNames.Keys)
    {
        string name = pSession.Hierarchies.HierarchyNames[tempHierarchy_loopVariable];
        string value = tempHierarchy_loopVariable.ToString();
        dt.Rows.Add(CreateRow(name, value, dt));
        i++;
    }
    // Create a DataView from the DataTable to act as the data source
    // for the DropDownList control.
    DataView dv2 = new DataView(dt);
    return dv2;
}
```

Koodi 1. DataView:n luonti hierarkioita varten.

5.3.4 Hierarkian avaus

Projektin ollessa avattuna hierarkiat on listattuna pudotusvalikkoon. Hierarkian valitseminen rakentaa puunäkymän, jossa hierarkia esitetään. Puuta ei missään vaiheessa luoda uusiksi, se ainoastaan tyhjennetään hierarkian valinnan yhteydessä.[Koodi 2.]

```
private void AddNewNodes(string NodePath)
{
    TreeNode nodeName, nodeName2;
    int i = 0;

    // Go through nodes
    foreach (cItem_Hierarchy_Node tempNode in pSession.Hierarchies.CurrentHierarchy.NewNodes)
    {
        // Create new node
        nodeName = new TreeNode();
        nodeName.Text = pSession.Buffers.get_ListValue_WithNew(tempNode.ListValueID).ValueStr;
        nodeName.Value = tempNode.NodeID.ToString();
        nodeName.ImageUrl = "Img/folder_" + tempNode.ImageKey + ".png";

        //If ParentNodeID is 0 then we add it to root of tree
        if (tempNode.ParentNodeID != 0)
        {
            if (i == 0)
            {
                //this needs to be done only once, clears "Opening.."
                treeHierarchy.FindNode(NodePath).ChildNodes.Clear();
                i = 1;
            }
            treeHierarchy.FindNode(NodePath).ChildNodes.Add(nodeName);
        }
        else
        {
            treeHierarchy.Nodes.Add(nodeName);
        }
        // If the node isn't leaf, add "Opening..." -node
        if (!tempNode.IsLeaf)
        {
            nodeName2 = new TreeNode();
            nodeName2.Text = "Opening..";
            nodeName.ChildNodes.Add(nodeName2);
        }
    }
}
```

Koodi 2. Tasojen lisääminen puunäkymään.

AddNewNodes metodia kutsutaan hierarkian valinnan yhteydessä, mutta sitä käytetään myös, kun laajennetaan puun tasoa. Käy läpi muistissa olevat tasot.

5.3.5 Dokumenttien listaus

Mikäli puunäkymästä valitaan taso, joka sisältää dokumentteja, ne listataan taulukkonäkymään. Taulukkonäkymän täyttö tapahtuu Query-metodilla, eli suoritetaan kysely. Luodaan DataTable, johon talletetaan kyselyn tulokset oikeassa formaatissa, jotta liittäminen taulukkonäkymään on mahdollista. Kyselyn tulokset saadaan aluksi yhtenä merkkisarjana, joka pitää pilkkoa osiin ja asettaa taulun riviin. Tämä tapahtuu for-loopin sisässä. For-loop käydään läpi niin monta kertaa kuin dokumentteja palautui kyselystä. Ensimmäisen for-loopin sisässä on toinen for-loop, jossa pilkottu merkkijono talletetaan taulun riviin.

Samassa yhteydessä päivitetään myös sivutuksen napit aktiiviseksi tarvittaessa. Jos dokumentteja on enemmän, kuin käyttäjä on määrittänyt näytettäväksi kerrallaan, niin ne jaetaan eri sivuille. Taulukkonäkymän yläpuolelta löytyy napit sivutukset käyttöön. Lisäksi näytetään, miten paljon dokumentteja on yhteensä ja mitkä niistä ovat näkyvissä, esim. 1-20/45.

5.3.6 Dokumenttikortin luonti

Dokumenttikortti luodaan, kun taulukkonäkymästä valitaan jokin dokumentti. Dokumentin tunniste on valitun rivin ensimmäisessä solussa, joka on piilotettu. Tunnistetta tarvitaan, kun haetaan dokumenttikorttiin tiedot. Dokumenttikortin luonti on yksi vaikeimmista vaiheista www-sovelluksessa. Dokumenttikortteja on useita erilaisia, joten ei voitu luoda vain yhtä valmista pohjaa. Koko dokumenttikortti luodaan taustakoodissa. Ennakkoon on luotu ainoastaan yksi taulu, johon dokumenttikortti rakennetaan.

Ensimmäinen vaihe on luoda tauluun rivi ja solut, joihin sijoitetaan dokumenttikortin kontrollit. Tämän lisäksi luodaan kaksi paneelia, vasen ja oikea, joihin sijoitetaan kontrollit. Sijoitettavat kontrollit tulevat jompaankumpaan paneeliin ja niiden leveys on normaalisti paneelin leveys. Poikkeuksena on kuitenkin muistiokontrolli, joka on molempien paneelien levyinen. Muistion luonnin yhteydessä luodaan kokonaan uusi paneeli, johon sijoitetaan ainoastaan muistio. Tämän jälkeen luodaan uudestaan paneelit muita kontrolleja varten. [Koodi 3.]

```
//TEXT
if (tempFieldInGroup.FieldType == enFieldType.TEXT)
{
    Control c1 = LoadControl("UC/ucField_Text.ascx");
    ((ucField_Text)c1).Caption = tempFieldInGroup.Name;
    ((ucField_Text)c1).ID = "ucField_Text" + Page + i;
    ((ucField_Text)c1).Mode = edit;
    ((ucField_Text)c1).Compulsory = tempFieldInGroup.IsCompulsoryField;

    if (tempFieldValue.HasValues == true)
    {
        ((ucField_Text)c1).SetValues(tempFieldValue.Values);
    }
    if (tempFieldInGroup.Column == 2)
    {
        panel2.Controls.Add(c1);
        panel2.Controls.Add(new LiteralControl("<br />"));
    }
    else
    {
        panel1.Controls.Add(c1);
        panel1.Controls.Add(new LiteralControl("<br />"));
    }
}
```

Koodi 3. Tekstikontrollin luonti.

Huomioitavaa dokumenttikortin luonnissa on se, että kortti pitää luoda aina uusiksi. Tätä varten talletetaan selaimen evästeeseen tieto, että dokumenttikortti on näkyvissä ja se tiedetään luoda uudestaan mahdollisen uudelleenlatauksen yhteydessä.

5.3.7 Dokumentin editointi

Dokumentin editoinnin mahdollistaminen on hyvin samanlainen toimenpide kuin dokumenttikortin luonti. Luonnin yhteydessä kontrolleille määritetään, että ne ovat editointi tilassa. On tärkeää muistaa luoda sivunlatauksen yhdessä kontrollit jälleen editointi tilaan, muuten muuttuneiden tietojen kerääminen ei onnistu.

5.3.8 Tiedostojen lataus

Dokumentti sisältää normaalisti tiedostoja, joita on mahdollista ladata ja tallentaa. Tallentaminen ei ollut vielä kirjoitushetkellä mahdollista. Dokumenttikortin avaamisen yhteydessä tulee lista tiedostoista, jotka liittyvät dokumenttiin. Tiedostot listataan dokumenttikortin viereen. Klikkaamalla tiedostoa aukeaa dokumenttikortin tilalle lisätietoja tiedostosta. Tiedoston lataaminen ei onnistunut aivan normaaliin tapaan, koska tiedosto ei sijaitse normaaliin tapaan palvelimella. Se pitää pyytää EloDoc

palvelimalta, josta se lähetetään binäärinä. Www-sovelluksessa sitten käsitellään tämä binääridata, ja tarjotaan tiedosto ladattavaksi käyttäjälle. Toinen ongelma muodostoi www-sovelluksessa käytettäväst UpdatePanel komponentista. UpdatePanel mahdollistaa ajax toiminnallisuuden. Tämän paneelin sisästä ei ole mahdollista tehdä pyyntöä tiedoston lataamiseksi. Ongelma ratkaistiin JavaScriptin avulla. Kun käyttäjä haluaa ladata tiedoston, JavaScript luo uuden iframen, joka on piilotettu. Tähän iframeen avataan uusi sivu, joka hoitaa tiedoston latauksen. Käyttäjälle tästä ei näy mitään merkkejä, ainoastaan normaali tiedostonlatausikkuna.

5.3.9 Muuta

Www-sovelluksen asettelu toteutettiin jQueryn UI.Layout liitännäisellä. Tämän avulla saatiin helposti luotua eri osioita sivulle. Osioille pystyttiin määrittämään kiinteät pituudet tai tarvittaessa ne joustavat jos selainikkunan kokoa muutetaan. Käyttäjän on myös mahdollista muuttaa osion kokoa. Näin käyttäjä voi antaa enemmän tilaa dokumenttilistaukselle dokumenttikortin kustannuksella.

Ulkoasun eri elementit pyrittiin toteuttamaan CSS-tyylimääritteillä. Näin ulkoasun muokkaus on tulevaisuudessa helppoa, kun ulkoasun muokkaus onnistuu yhdestä tiedostosta. Lisäksi CSS:llä toteutettiin hieman asetteluakin, mm. dokumenttikortti on toteutettu CSS:llä, vaikka taulukon käyttökin olisi ollut mahdollista.

Codeplexistä löytyvää ASP.NET Ribbon -komponenttia käytettiin www-sovelluksessa samaan tapaan kuin Microsoftin Officessa ja uudessa SharePoint 2010:ssa. Kyseisen komponentin tekijältä on ilmestynyt myös Office Web.UI, jonka mahdollista käyttöä www-sovelluksessa tullaan tutkimaan.

6 Yhteenveto

Opinnäytetyön tavoitteena oli toteuttaa www-sovellus, jo olemmassa olevan työpöytäsovelluksen rinnalle. Www-sovellusta ei saatu valmiiksi opinnäytetyön tekemisen aikana, mutta sen kehitys jatkuu opinnäytteen jälkeenkin. Tästä huolimatta opinnäytetyö oli erittäin hyödyllinen, erityisesti oppimisen kannalta. Www-kehittämisen rooli tulee taatusti kasvamaan ohjemistokehityksessä. Nykyisin monella on pääsy webbiin, ja täten mahdollista käyttää www-sovellusta.

Projektissa tehtiin kevyt kartoitus mahdollisista teknologioista, joilla lähteä toteuttamaan. Pääpainon ollessa Microsoftin tarjoamissa ratkaisuissa. Päädyttiin tekemään www-sovellus käyttäen ASP.NET ja C# yhdistelmää. Tekniikan valintaan vaikuttivat aiempi osaaminen ja se, että työpöytä sovellus oli tehty Visual Basicillä. Näin pystyttiin helposti hyödyntämään olemassaolevia koodeja.

Itse toteutus lähti liikkeelle varsin hyvin, kun tiedettiin melko tarkkaan, mitä ominaisuuksia tarvitaan. Vaikkakin projektin alkaessa, ei ollut vielä varmuutta ulkoasusta tai muutenkaan siitä, että millainen www-sovelluksesta halutaan. Siirrettiin tärkeimmät toiminnot toimimaan www-ympäristössä.

ASP.NET:n käyttäjäkontrollit osoittautuivat erittäin hyödyllisiksi ja helpoiksi hyödyntää. Ne myös selvensivät käyttöliittymän rakenneta, tehden siitä helpommin hahmotettavan. Ongelmia tuotti kuitenkin halutun asettelun aikaansaaminen. Jostain syystä ei ollut tarjolla yksinkertaista ratkaisua kokosivun asettelua varten, niin että vain tietyt elementit olisivat joustavia. Tähän kuitenkin löytyi ratkaisu jQueryn liitännäisestä, UI.Layoutista.

LÄHTEET

[1] Programming Languages [www-dokumentti] Saatavilla: http://msdn.microsoft.com/en-us/library/aa292164(v=vs.71).aspx. (Luettu 14.1.2011).

[2] Microsoft .NET Framework [www-dokumentti] Saatavilla: http://www.microsoft.com/net/. (Luettu 9.1.2011).

[3] TIOBE Software: Tiobe index [www-dokumentti] Saatavilla: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[4] Getting Started with Visual C# [www-dokumentti] Saatavilla: http://msdn.microsoft.com/fi-fi/vcsharp/dd919145.aspx. (Luettu 9.1.2011).

[5] Programming Microsoft® ASP.NET 3.5 (9780735625273) [Kirja]

[6] Ajax: A New Approach to Web Applications [www-dokumentti] Saatavilla: http://www.adaptivepath.com/ideas/essays/archives/000385.php. (Luettu 13.1.2011).

[7] EloDoc Dokumenttienhallintajärjestelmä [PDF] Saatavilla: http://www.elomatic.com/linked_files/pdf_brochures/elodoc_flyer_fin_low_09.pdf. (Luettu 4.1.2011).

[8] HTML & CSS - W3C [www-dokumentti] Saatavilla: http://www.w3.org/standards/webdesign/htmlcss (Luettu 1.3.2011).

[9] Structured process you must know to develop a web application [www-dokumentt] Saatavilla: http://woork.blogspot.com/2009/01/structured-process-you-must-know-to.html (Luettu 1.3.2011).

[10] JavaScript for Web Design - Advantages and Disadvantages [www-dokumentti] Saatavilla: http://ezinearticles.com/?JavaScript-for-Web-Design---Advantages-and-Disadvantages&id=645013 (Luettu 20.2.2011).

[11] ASP.NET MVC [www-dokumentti] Saatavilla: http://www.asp.net/mvc. (Luettu 9.1.2011).

[12] What Is Silverlight? [www-dokumentti] Saatavilla: http://www.microsoft.com/silverlight/what-is-silverlight/. (Luettu 22.1.2011).