Asko Huuki

SOVELLUKSEN KÄYTTÖLIITTYMÄN UUDISTUS CASE C-CARE

Tietojenkäsittelyn koulutusohjelma
2012

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

SOVELLUKSEN KÄYTTÖLIITTYMÄN UUDISTUS CASE C-CARE

Huuki, Asko
Satakunnan ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma
Helmikuu 2012
Ohjaaja: Nieminen, Hans
Sivumäärä: 55
Liitteitä: 0

Asiasanat: verkko-ohjelmointi, käyttöliittymät, Microsoft Silverlight

---

Opinnäytetyön aiheena oli sovelluksen käyttöliittymän uudistus Cieltum Oy:n tarpeisiin. Yrityksen C-Care-sovellusta oli jo jonkin aikaa kehitetty, mutta ulkoasu ei ollut vielä miellyttävä ja sitä on vaikea muokata. Sovellusta piti saada yhdenmukaisemmaksi, helppokäyttöisemmäksi, ylläpidettävämmäksi ja viimeistellymmäksi. Tätä varten yritys antoi minulle projektiksi sovelluksen ulkoasun uusimisen.

Alussa tutustuin yrityksen tarpeisiin. Cieltum Oy:llä oli jo asiakkailta, joilta sai arvokasta palautetta. Lisäksi muilla työntekijöillä oli omia kehitysehdotuksiaan. Tämän lisäksi opiskelin käyttöliittymien suunnittelun teoriaa muutaman kirjan avulla. Tutustuin myös tarkemmin käytössä olleeseen tekniikkaan.

Saatuani tarpeeksi materiaalia tein muutoksia sovellukseen. Ensiksi tein isoja muutoksia muuttaen sivuja samanlaisiksi kooltaan ja tyyliltään sekä muuttaen värityksiä ja mittasuhteita. Tavoitteena oli saada myös käytetyimmät sivut muutettua mahdollisimman aikaisessa vaiheessa. Myöhemmin siirryin vähemmän käytettyjen osuuksien muokkaamiseen, pienten asioiden korjaamiseen ja lapsi-ikkunoiden suunnitteluun. Lopuksi keskityin vielä viimeistelemään sovelluksen käyttöliittymän

Käyttöliittymän uudistus oli valmis vuoden 2011 loppuun mennessä. Siinä vaiheessa olin saanut työyhteisön avulla tehtyä halutut muutokset. Sovellus muuttuu myöhemmin, kun uusia tarpeita ja ominaisuuksia ilmaantuu. Suuret muutokset on kuitenkin jo tehty ja sovellusta tarjotaan nyt asiakkaille.

RENEWAL OF WEB USER INTERFACE CASE C-CARE

Huuki, Asko
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in business information systems
February 2012
Supervisor: Nieminen, Hans
Number of pages: 55
Appendices: 0

Keywords: web programming, user interfaces, Microsoft Silverlight

---

The purpose of this thesis was to renew the web user interface for Cieltum Oy's reguirements. Cieltum's C-Care-application was already developed for some time but the layout wasn't enjoyable and it was difficult to edit. Application was needed to change easier to use, easier to maintain and to look more finished. Because of this reason my company gave me renewal of application's user interface as a project.

At first I familiarized myself with the requirements of company. Cieltum Oy already had some clients which gave some valuable feedback. The other employees also had some suggestions. I studied theory of designing user interfaces from couple of books as well. Also I had a closer look to the available technology.

When I got enough material I made changes to application. At first I made big changes including editing pages similar in size and style and changing colors and dimensions. I tried to edit pages which are often used as early as possible. Later I moved into less used parts of the application and focused on fixing small things and planning child windows. Finally I focused on giving finishing touches to user interface.

Renewal of user interface was ready by the end of the year 2011. At that point I had finished the desired changes with the help of other employees. Application will change later on when new needs and features appears. But the big changes are already done and application is now being offered to customers.

SISÄLLYS

1 JOHDANTO

Olen ollut Cieltum Oy:llä töissä työharjoittelun jälkeisen ajan. Se on vaatinut luonnollisesti ajan jakamista opintojen ja työn kesken, mutta toisaalta se on myös antanut paljon käytännön kokemusta ohjelmoinnista, erilaisista tekniikoista ja työskentelystä yrityksessä.

Työpaikkanani toimiva yritys on kuitenkin pieni ja sillä on vain vähän erilaisia projekteja. Olemme kehittäneet pitkään sovellusta nimeltä C-Care, johon ohjelmointipuolen resurssit liki täysin keskittyivät. Opinnäytetyö oli tarkoitus aloittaa hyvissä ajoin, mutta projektin omat tarpeet säätelivät tilannetta. Kun yrityksellä ei ollut tarjota mitään luonnollista projektia, oli mahdotonta aloittaa opinnäytetyötä.

Lopulta kuitenkin aihe löytyi, kun sovelluksen oleellisimmat osat oli tehty. Sovelluksessa oli paljon toimintoja, mutta ulkonäkö ei ollut silmiä hivelevää. Siksi minä sain projektiksi uudistaa C-Caren käyttöliittymän. Projekti oli laaja ja se sisälsi paljon sivun värimaailman muutoksia, toimintojen selkeyttämistä ja turhan sivujen välillä hyppimisen poistamista.

Olin aiheesta hyvin innoissani. Alun perin olin jo tehnyt sivujen pohjat, mutta tuolloin minulla oli malli apunani, jota parhaani mukaan aloittelevana ohjelmoijana yritin toistaa. Nyt tilanne oli erilainen, sillä minulla oli paljon vapaammat kädet toimia. Aiemmin olin tehnyt, kuten minua käskettiin, mutta nyt sain itse päättää monesta asiasta.

Siksi ei olekaan ihme, että tutkin heti aluksi paljon erilaisia teoksia ja annoin monta kehitysideaa. Kaikkea ei yrityksen johto luonnollisesti hyväksynyt, mutta moni asia oli toteuttamiskelpoinen. Sen jälkeen tehtävänä olikin vielä ohjelmoida ideat sovellukseen.

2 CIELTUM OY JA C-CARE

Cieltum Oy on ulvilalainen yritys, joka on perustettu joulukuussa 2009. Yritys on pieni ja sillä on vain kuusi työntekijää. Näistä neljä on vanhempia ihmisiä, joista vain yksi keskittyy ohjelmointityöhön. Yrityksen loput työntekijät koostuvat minusta ja toisesta henkilöstä, jotka tulimme työharjoitteluun toukokuussa 2010. Sen jälkeen olemme koulun mahdollistamissa puitteissa työskennelleet yrityksessä.

Yrityksen perusperiaatteena on toimittaa asiakkaalle laadukkaita sovelluksia, joilla voidaan hallita tietoa. Tietoa on aiemmin voitu hallita paperilapuilla, lomakkeilla, puhelimilla ja sen sellaisella, mutta Cieltum halusi tarjota näihin asioihin IT-ratkaisun. Tätä varten Microsoftilta löytyi sopivia tuotteita ja Cieltum päätti käyttää niitä. Pääasialliset tuotteet Azure, Silverlight, Microsoft Online ja Sharepoint.

Erilaisena piirteenä moneen yritykseen Cieltum on panostanut pilviteknologiaan. Pilven avulla käyttäjän ei tarvitse asentaa mitään tiettyä sovellusta koneelle, vaan tietokonekapasiteetti ja ohjelma ovat käytettävissä Internetin välityksellä. Näin säästytään kalliiden palvelimien ostolta, sillä pilven kautta voi vuokrata sen mitä tarvitsee ja maksaa sen mukaan. Microsoftin oma pilvialusta on Azure ja sitä käyttää myös Cieltum omissa ohjelmissaan.

Käytännössä Cieltum tarjoaa kahta eri tuotetta. C-Share toimii Sharepoint-ympäristössä ja se on tarkoitettu parantamaan dokumentinhallintaa, tiedonvälitystä ja tiedon keräämistä. C-Share toimii pilven kautta, jolloin on tarkoitus saada karsittua ylläpitotehtäviä ja kustannuksia. Lopulta C-Share on kuitenkin vain valmis lähtökohta, jonka pohjalta voidaan tehdä jokaiselle yritykselle omiin tarpeisiin sopiva ratkaisu. Toinen tuote onkin minulle paljon tutumpi C-Care.

C-Caren ajatus on luoda ”tuotetiedon Facebook” käyttäen hyväkseen pilviteknologiaa ja Microsoft Silverlightia. Ohjelma on tarkoitettu lähinnä yksilöiden elinkaaritiedon hallintaa varten ja sen on tarkoitus yhdistää jokin tietty kohde yhteisöpalvelutavoin kohteeseen liittyville sidosryhmille. Sen sisältö on tarkoitus myös esittää avoimesti, tehokkaasti ja taloudellisesti järkevällä tavalla.

C-Caressa on paljon erilaisia toimintoja. Sillä voi luoda erilaisia kohteita, kohteille voi luoda tehtäviä, liittää dokumentteja ja lisätä tietoa. Kohde on myös mahdollista päivittää, mikäli sille on tehty muutoksia. Lisäksi ohjelmassa kykenee hallitsemaan käyttäjien työtunteja ja tehtäviä. Ohjelma tarjoaa myös kustannustietoa erilaisten kaavioiden ja raporttien avulla.

Käytännössä käyttäjä pääsee C-Careen kutsukoodin ja Microsoftin livetunnusten avulla, kunhan jokin toinen käyttäjä on jakanut uudelle käyttäjälle jonkin kohteen. Tuo kohde voi vaikka olla jokin tietty traktori. Kohteessa taas on useampia pienempiä osia, kuten vaikka rengas. Rengas taas menee pienempiin osiin. Toisin sanoen kohteessa on hierarkkinen rakenne, jossa voi mennä helposti alaspäin tai palata hierarkiassa ylöspäin.

Uudelle käyttäjälle, joka voi olla vaikkapa korjaaja, on saatettu myös antaa tehtäväksi vaikka käydä tutkimassa traktorin rengas. Kun korjaaja on todennut, että rengas on puhki, hän voi kirjoittaa siitä muistiinpanon C-Careen ja ilmoittaa työtehtävän kohdalle, että rengas on puhki ja uusi rengas asetettu. Tämän korjauksen taas näkevät kaikki, joille kyseinen traktori on jaettu. Työtehtävien hallinnassa on myös mahdollista määritellä työn hinta, laatu ja ilmoittaa tehdyn työn määrä. Näin sovellusta voi käyttää myös työtuntien seurantaan ja laskutukseen.

3 WEB-KÄYTTÖLIITTYMÄ JA SEN SUUNNITTELU

3.1 Perusteita

Kotisivuilla on lukemattomia vaatimuksia, jotka sieltä tulisi löytyä. Sivustolta tulisi löytyä sivuston tunnus ja tehtävä, kotisivulla tulisi näkyä sivuston hierarkia, käyttäjän tulisi löytää paikka etsinnälle, sisällön tulisi olla ajanmukaista, sivustolla tulisi näkyä yhteistyökumppanit, sivuston tulisi tarjota käyttäjälle oikoteitä ja sinne pitäisi kyetä rekisteröitymään. Tämän lisäksi sivuston tulisi näyttää käyttäjälle mitä hän etsii ja mitä hän etsi, näyttää aloituskohta ja vieläpä näyttää luotettavalta ja uskottavalta.

Tällaisia vaatimuksia yritetään täyttää haastavissa olosuhteissa, mikä ei tietenkään ole helppoa. (Krug 2006, 95-97.)

Vaatimuksia ei kuitenkaan ikinä pysty täysin täyttämään. Jostain tulee aina luopua, jotta jokin asia kyetään tekemään. Tärkein seikka josta ei saa luopua on kuitenkin yleiskuvan antaminen. Sivun tulisi kyetä vastaamaan siihen, että mitä siellä on, mitä siellä voi tehdä, mikä se on ja miksi juuri sillä sivulla tulisi olla. Sivuston tulisi toisin sanoen kyetä olla yksiselitteinen vähällä vaivalla. (Krug 2006, 98-99.)

Jos sivusto rakentuu jonkin vaiheittaisen prosessin ympärille, alkukohdan tulee olla todella selkeä. Siksi ei kannata viljellä koko sivua täyteen toteamuksia, kuten ”aloita tästä”. Sama koskee myös rekisteröitymistä. Sotkun välttämiseksi kannattaakin muuttaa aloituskohdat aloituskohtien näköiseksi, etsintäruudun sen näköiseksi ja luettelot näyttämään juuri luetteloilta. Niihin kannattaa myös liittää selkeät otsikot. (Krug 2006, 107.)

3.2 Hyviä periaatteita

Ensiksi kannattaa miettiä, miksi on tekemässä web-sivuja. Sivujen tarkoituksena on lähes aina jonkinlainen viestintä. Siksi onkin oleellista miettiä se, että mitä haluaa viestiä ja kenelle. Toisaalta sivut kannattaa rakentaa webin vahvuuksien varaan. Internet on maailmanlaajuinen kanava, jossa on yhteisiä teknisiä sopimuksia. Hyvä sivu puhuttelee laajaa yleisöä ja noudattaa vakiintuneita teknisiä käytäntöjä. Lisäksi on hyvä huomata, että linkkejä tulee hyödyntää eikä laittaa kaikkea yhdelle sivulle. Yksi sivu ei saa ikinä olla liian täynnä. (Korpela, Linjama 2005, 67.)

Varsinkin sosiaalisen median, kuten Facebookin ja Twitterin esimerkki on mielestäni osoittanut sen, että sivulla olisi mahdollisimman vähän mitään ylimääräistä. Sosiaalisessa mediassa lähes kaikki on kirjoitettu lyhyesti, kun taas pitkiä tekstejä harva jaksaa lukea. Mielestäni se pätee muuallakin, joten siksi sivuilla ei pitäisi olla mitään ylimääräisiä, turhan hienoja lauseita. Tietenkin se on sitten eri asia, mikäli sivuston on tarkoitus tarjota tietoa laajasti ja kattavasti.

Web-sivut mukautuvat luonnostaan erikokoisiksi ja eri toimintaympäristöön sopiviksi. Siksi onkin hyvä antaa web-sivun toimia niin, eikä kahlita sitä yhteen esitystilanteeseen, kuten määräkokoiseen kuvaruutuun sopivaksi. Myös vanhoja, pitkäaikaisia periaatteita on hyvä noudattaa. Kielenkäyttö, typografia ja graafinen ilmaisu ovat muodostuneet vuosisatojen kuluessa ja on parempi osata ne, kuin keksiä samat asiat uudestaan. Kaikissa tilanteissa kirjoitetun tekstin säännöt eivät enää kuitenkaan päde, eivätkä kaikki painojulkaisujen säännöt ole sellaisenaan käyttökelpoisia. Nämä kuitenkin antavat hyvää suuntaa. (Korpela, Linjama 2005, 68.)

Web-sivuilla voidaan kertoa sen rakenne ja kuvata yksi tai kaksi tapaa esittää se. Määräily ei kuitenkaan ole suotavaa. Toisaalta web-sivuilla yksinkertainen versio on syytä tehdä ensin. Se toimii, se on nopea tehdä ja se toimii hyvin muun kehittämisen pohjana. Se toimii myös eräänlaisena varmuuskopiona, jos jokin menee vikaan. Se tarkoittaa myös sitä, että ensiksi tulee esittää asia yksinkertaisesti ja vasta sitten tarkentaa. Suurin osa ihmisistä lukee enintään otsikon ja ensimmäisen kappaleen. Siksi ei tule säästää tärkeintä asiaa loppuun. (Korpela, Linjama 2005, 68.)

Huomioitava seikka on myös se, että hyvin tehtyyn asiaan kuuluu myös viimeistely. Se voi koskea ulkonäön muokkaamista, kieliasua, teknisiä muutoksia ja sen sellaista. On hyvää viimeistellä työ, mutta on syytä myös lopettaa silloin, kun virittelystä ei ole enää kenellekään hyötyä. (Korpela, Linjama 2005, 68.)

Hyvä esimerkki viimeistelyn merkityksestä tuli aloittaessani käyttöliittymän uudistamista. Mielestäni olin tehnyt vain pieniä viilauksia ulkonäköön ja sen lisäksi lisännyt sivulle murupolun. Jo se riitti tuomaan sivustolle paljon vakuuttavamman vaikutelman.

3.3 Testaaminen

Testaaminen on äärimmäisen tärkeää, mikäli haluaa toimivan ja hyvän näköisen sivuston. Tekijä itse tietää mitä missäkin pitää tehdä, joten siksi sivuston toimivuus on pakko testata jollakin ulkopuolisella henkilöllä. Jo yhden käyttäjän testaaminen on

parempi kuin kokonaan testaamatta jättäminen. Joka ikinen testi tuo ilmi asioita, joiden parantaminen on tärkeää. (Krug 2006, 133-134.)

Asian kykeni helposti havainnoimaan testaamisen yhteydessä. Ulkopuoliset ihmiset toimivat vaistonvaraisesti ilman aiempaa kokemusta. Usein testikäyttäjä toimi eri tavalla kuin mitä ohjelmaa tehdessä oli suunniteltu. Tekijä on sokea työlleen ja toimii aina oikein, mutta satunnainen testaaja ei ole.

Kannattaa myös ottaa huomioon se, että testaus heti alussa vähäisellä määrällä on parempi kuin lukuisat testaukset lopussa. Alussa on aikaa hyödyntää opittuja asioita, joka on arvokkaampaa kuin lopussa tehdyt suuret ja korjaavat ponnistukset. Testaukseen voi osallistua myös kuka tahansa ja vain arviointiperusteita täytyy muuttaa. Testaamisen tavoitteena ei ole myöskään saada vastausta siitä, että onko jokin asia väärin vai oikein. Sen sijaan tarkoituksena saada palautetta, jonka avulla voi tehdä perusteltuja päätöksiä. (Krug 2006, 134-135.)

Testaajia ei tarvita lukuisia, vaan kolme tai neljä riittää hyvin. Tämä riittää todennäköisesti merkittävimpien ongelmien havaitsemiseen. Koehenkilöiksikin voi valita kenet tahansa. Koehenkilöiden laadulla ei ole paljon merkitystä. Yleisesti riittää vain se, että valituiksi on tullut ihmisiä, jotka ovat käyttäneet jonkin verran Internetiä ja tuntevat perusasiat. (Krug 2006, 138-139.)

Tärkeintä on kuitenkin aloittaa testaaminen varhain ja toistaa sitä jatkuvasti. Eli ensiksi tehdään jotain ja se taas testataan. Testin jälkeen on luvassa virheiden korjaus, jonka jälkeen aloitetaan uusi testaus. Testauksia on monenlaisia. Käytettävyystestissä testataan sivuston ymmärrettävyyttä. Silloin voi katsoa, että ymmärtääkö koehenkilö sivuston tarkoituksen, tavoitteen, rakenteen, toimintatavan ja niin edelleen. Avaintehtävätestissä taas koehenkilöä pyydetään tekemään jotain ja katsoa, miten hän tehtävästä suoriutuu. (Krug 2006, 135, 144.)

Kannattaa kuitenkin käyttää vertailusivuja, ennen kuin ryhtyy itse hahmottamaan omaa sivuaan. Niistä voi tutkia erilaisia tyylejä, rakenteita ja ominaisuuksia. Nämä vertailusivujen ratkaisut taas voi testata koehenkilöillä ja katsoa mikä toimii ja mikä ei. (Krug 2006, 144.)

Testaaminen jää kuitenkin usein turhan vähälle huomiolle, kun asiakkaat haluavat uusia ominaisuuksia ja korjauksia tulisi saada nopeasti tehtyä. Samalla nopeuden mukana useimmiten uudet tai korjatut ominaisuudet ja ratkaisut jäävät usein kunnolla testaamatta. Usein myöhemmässä vaiheessa tuleekin tarpeen korjata korjaus, vaikka se olisi voitu välttää heti alussa kunnollisella testaamisella.

3.4 Navigointi

On selvä asia, että navigointi on tärkeää. Jos käyttäjä avaa jonkun sivuston, eikä löydä hakemaansa tai kykene hahmottamaan sivua, hän ei myöskään vietä aikaansa sivulla. Silloin käyttäjä ei myöskään tule sivulle takaisin. Pitää siis olla selkeä, yksinkertainen ja johdonmukainen navigointi. (Krug 2006, 51.)

Web-sivustossa on hyvä huomioida kolme erikoista piirrettä. Ensinnäkin mittakaavaa on vaikea hahmottaa, sillä sivustolla voi olla todella paljon sivuja ja kokonaisia alueita, joita ei käytä. Toisaalta web-sivustossa ei ole ikinä selvää suuntaa, kuten fyysisessä tilassa. Ei siis voi mennä vasemmalle tai oikealle. Ja toisaalta web-sivustolla ei ole mitään kiinteää paikkaa. Web-sivustolla joutuu painamaan erilaisia linkkejä, mutta kukaan ei voi luottaa suuntavaistoonsa. Edellinen-painikkeen osuus kaikista painalluksista onkin noin 30–40%. Siksi kotisivun merkitys on tärkeä, sillä se luo selkeää pysyvyyttä sivulle. (Krug 2006, 57-58.)

Navigointivälineillä on vakiintuneita käytäntöjä. Kun ne ovat tutussa paikassa, ne löytyvät helposti ja nopeasti. Ja kun ne ovat samankaltaisia, ne on helppo erottaa. Näin syntyy tiettyjä oletuksia. Sivustot kehittyvät koko ajan, mutta perusasiat on hyvä tiedostaa. (Krug 2006, 60-61).

Web-sivuston tunnus tai logo toimii maamerkkinä, joka pitää näkyä kaikkialla ja se on totuttu näkemään vasemmassa yläkulmassa. Tämä selittyy sivun visuaalisella hierarkialla. Tunnus ikään kuin kehystää sivun, mutta yläkulmassa se ei ole silmiinpistävin osa. Ensisijaiset välineet puolestaan ovat linkkejä sivuston pääosastoihin. Ne edustavat sivuston ylintä tasoa. Linkeissä on hyvä ottaa myös huomioon se, että yksi

navigoinnin tärkeimmistä osista on painike kotisivulle. Se on oltava olemassa, vaikka olisi mennyt syvälle sivustoon, sillä se rauhoittaa käyttäjää. Mahdollisuutena linkittää kotisivu on vaikkapa sisällyttää kotisivulinkki osastoihin ja antaa mahdollisuus, jolla tunnusta napsauttamalla pääsee kotisivulle. (Krug 2006, 63-67.)

Käyttäjällä täytyy olla käsitys sijainnistaan. Siksi jokaisella sivulla tulisi ylhäällä olla iso otsikko. Otsikossa tulee lukea sivun nimi ja sen pitää vastata valittua sivua. Otsikoissa voi myös käyttää tehokeinoja, kuten lihavointia, poikkeavaa väritystä ja leveämpää ylätilaa. Helpotusta sijaintiin tuottaa myös nykyisen sijainnin korostaminen navigointipalkissa, listoissa ja valikoissa. Korostuksesta kannattaa tehdä sellainen, että sen huomaa helposti. Siinä voi käyttää esimerkiksi sekä lihavointia että eri väriä. Yksi tapa on myös käyttää välilehtiä. Välilehtien käyttö on perusteltua, koska ne ovat itsestään selviä, niitä on vaikea olla huomaamatta, ne näyttävät hyviltä ja ne tuovat mieleen fyysisen tilan. Kaikkein tehokkaimmillaan välilehdet ovat, kun ne ovat värikoodattuja. Tuolloin sekä korostettu välilehti että sivun sisältö ovat samalla värillä, jolloin sijainti on helppo huomata. (Krug 2006, 71-75, 80-83.)

Lisäksi voi käyttää murupolkua. Se ilmaisee, että missä olet. Niiden käyttö on yleistynyt koko ajan, mutta yksinään ne eivät ole hyvä ratkaisu. Murupolku ei tarjoa itsessään tarpeeksi selvää tietoa, mutta osana toteutusta ne ovat erittäin hyviä. Murupolku kannattaa ehdottomasti sijoittaa sivun yläosaan, sillä silloin ne muuttuvat samankaltaiseksi apuvälineeksi kuin vaikka sivunumerot. Lisäksi tasojen nimien väliin kannattaa kirjoittaa ”>” erottamaan sivut toisistaan. Tekstin tulisi mielellään olla pientä, joka osoittaa kyseessä olevan vain apuväline. Viimeinen nimi kuitenkin tulee lihavoida, sillä se korostaa nykyistä sivua. (Krug 2006, 76-79, 80-83.)

Sijainnin korostaminen on osoittautunut myös tärkeäksi vaatimukseksi C-Carea tehtäessä. Aloittaessani käyttöliittymän uudistamisen sijainnin määrittäminen oli asiakkailta saadun palautteen mukaan hankalaa. Sijainti oli kyllä korostettu navigointipalkissa, mutta se ei itsestään riittänyt hahmottamaan omaa sijaintia sovelluksen sisällä. Sitä varten tarvittiin parempaa otsikointia, murupolkua ja navigointipalkin parempaa korostamista. Vasta kaikkien tapojen yhdistelmä osoittautui tarpeeksi hyväksi ratkaisuksi.

3.5 Ulkoasun suunnittelu

Ulkoasun suunnittelussa lähdetään tyypillisesti käyttäjäkunnasta. Käyttäjäkunta vaikuttaa niin käytettävään tekstityyppiin, tekstiin ja väreihin. Vasta sitten voidaan tarkastella sivustoon tulevia kuvia, miettiä sivuille yhtenäinen tyyli ja suunnitella muut tarvittavat asiat. (Korpela, Linjama 2005, 356.)

Web-sivu poikkeaa joiltain osin vanhemmista television ja painotuotteiden esitystavoista. Näistä selkein ero on muuttuva esitysmuoto. Selain, näyttö ja käyttäjät vaikuttavat aina sivuston ulkonäköön. Siksi sivua ei voi muotoilla pysyvästi tietyillä arvoilla ja tämä tulee muistaa ulkonäön suunnittelussa. Näyttävän grafiikan usein kuvitellaan olevan tärkein asia, mutta lopulta käyttäjät kokevat sivuston sisällön ja käytettävyyden grafiikkaa tärkeämmäksi asiaksi. Ulkoasun suunnittelussa päädytäänkin aina kompromisseihin, jossa lopputulos on aina jossain eri asioiden välissä. (Korpela, Linjama 2005, 356-357.)

Tarkkana kannattaa olla erityisesti sivujen koon kanssa. Käyttäjillä on lukuisia eri näyttöjen kokoja ja se aina vaikuttaa siihen miltä sivusto näyttää. Sama sivu voi toisella koneella näyttää erittäin hyvältä, kun taas toisella sivulla sivu voi olla käyttökelvoton.

Yhtenäinen ulkoasu sivustolla on tärkeää ja sitä varten onkin syytä käyttää samanlaisia elementtejä sivulta toiselle tuomaan yhtenäisyyttä. Samoin sivuilla tulee olla samanlainen tapa liikkua, taustakuvan tulee olla aina samanlainen ja sisällön pitäisi olla eri sivuilla samalla tavalla. Myös yhtenäisten fonttien ja värien käyttö lisää yhtenäisyyden tunnetta. Yhtenäisyyden kannalta erillinen tyylitiedosto onkin suotavaa. Sitä kautta voi linkittää eri tyylejä eri sivuille ja liittää sama taustakuva vaikka jokaiselle sivulle. (Korpela, Linjama 2005, 357-358.)

Olen havainnut, että kaikki muutokset siirtyessä sivulta toiseen rikkovat kuvan viimeistellystä sivustosta. Silverlight tukee tyylitiedostoa ja siihen on helppo linkittää eri elementit. Yhtenäisyyden kannalta tyylitiedosto onkin loistava työkalu.

Kannattaa keskittyä siihen, mitä etusivu kertoo. Etusivun pitäisi kertoa jotain sivustosta, niiden tyylistä ja sisällöstä ja ulkoasulla tulisi korostaa haettua viestiä. Etusivulla tulisikin heti ilmentyä mistä sivustossa on kysymys. (Korpela, Linjama 2005, 358.)

Teksti Web-sivuilla ovat käytännössä aina suorakulmioissa. Selaimet asettavat nämä lohkot useimmiten suhteellisen järkevästi, mutta leveyden asettaminen voi silti olla fiksua. Kuvat puolestaan kannattaa asemoida sen tekstin viereen, mihin kuva liittyy. Tavallisinta asettelussa on asettaa kaikki yhdelle palstalle. Se on myös käyttäjille tyypillisesti paras ratkaisu. Toisin sanoen Internetissä ei ole suositeltavaa käyttää sanomalehtien kaltaista useamman palstan sisältävää ratkaisua. Myöskään oikean reunan tasaus ei ole suositeltavaa. (Korpela, Linjama 2005, 358-359.)

Sivujen tausta luo sivuille koko sivun kattavan väripinnan, joten taustaväriä tulee harkita tarkasti. Tausta luo suuren osan sivun tunnelmasta, joten väriä voi miettiä sivun tarkoituksen mukaan. Tekstin väri tulee puolestaan aina valita taustan pohjalta, eli tummassa taustassa vaaditaan valkeaa tekstiä ja vaaleassa taustassa tummaa tekstiä. Linkkien pitää myös erottua erikseen. Taustakuvaan ei sovi mikään vahva kuvallinen ilmaisu, sillä taustan värivaihtelut eivät saa olla suuria. Muuten on mahdollisuus, että tausta häiritsee itse tekstiä. (Korpela, Linjama 2005, 362.)

3.6 Sommittelu

Sommittelu tarkoittaa sivujen tiettyjen osien järjestämistä rajattuun tilaan. Sillä voidaan ilmaista asioita, ohjata katseita ja välittää tunnelmia. Laittamalle sivuille viivoja, värejä, kuvia ja pintoja sivulle syntyy muun muassa rytmiä ja syvyyttä. (Korpela, Linjama 2005, 363.)

Sommittelu on jatkuvaa tasapainottelua tuttujen ja johdonmukaisten tapojen sekä virkistävän vaihtelevuuden välillä. Johdonmukaisuus väreissä ja tekstissä auttaa ja opastaa käyttäjää, mutta toisaalta jokin hallittu poikkeama sivustolla voi piristää ja luoda mielenkiintoa. Poikkeamien määrä puolestaan riippuu sivuston tavoitteista, käyttäjäryhmästä ja sisällöstä. (Korpela, Linjama 2005, 364-365.)

Miellyttävä sommittelu edistää sivuston viestin perillemenoa. Jotkut perusasiat pätevät aina. Värit aiheuttavat tietynlaisia tuntemuksia, pystysuora viiva tuntuu vaikeakulkuiselta ja vaakasuora viiva taas tuntuu helposti edettävältä. Tekstisisällön lisäksi web-sivuilla käytetään staattisia elementtejä, kuten kuvia, tekstejä ja eri muotoja. Käyttäjän katse puolestaan hakeutuu väreihin, niiden kontrasteihin, kokoon ja liikkeeseen. Katse puolestaan pyrkii luonnostaan hakeutumaan liikkeen suuntaan. Siksi kuvan liike kannattaa kohdistaa kohti sivun sisältöä, sillä muuten käyttäjä katsoo helposti väärään suuntaan ja se puolestaan aiheuttaa sekavuutta. Tyhjä tila jonkin kohteen ympärillä puolestaan kiinnittää huomion kohteeseen ja erottaa sen muusta sivusta. Se onkin hyvä, sillä ihminen pyrkii aina erottamaan eri kuvioita tausta. (Korpela, Linjama 2005, 365-366.)

Kuvapinnalla on yksikkönä pikseli. Peräkkäin ja kiinni toisissaan olevat pikselit muodostavat yhtenäisen viivan. Viivoilla on muoto ja suunta, jotka antavat sille ominaisen luonteen. Vaakasuora viiva luo rauhallisen olon, kun taas pystysuora luo jännitettä. Vinot viivat puolestaan tuovat sellaisen olon, että kuvapinta on liikkeessä tai kaatumaisillaan. Nämä ovatkin tehokkaita pinnan jakajia ja siksi web-sivuilla käytetään usein elementtien kehystämistä, eli erilaisten elementtien ympäröimistä viivojen avulla. (Korpela, Linjama 2005, 363.)

Erilaiset viivat ovat todella käteviä jakamaan sivustoa. C-Care käyttää jokaisella sivulla jakajana reunuksia ja se ainakin omasta mielestäni on erittäin hyvä tapa. Samalla kannattaa ottaa ehdottomasti huomioon se, ettei jonkin toisen reunuksen sisällä oleva asia vaikuta toisen reunuksen sisällä olevaan asiaan. Toisiinsa vaikuttavien asioiden tulisi olla siis lähellä toisiaan.

Kultainen leikkaus on yksi tunnetuimmista ja tärkeimmistä mittasuhteista. Siinä jaetaan jana kahteen osaan, joiden jakosuhteen arvo on 1: 1,618, joka karkeasti ilmaistuna on 2: 3. Koska Web-sivu ei näytä samanlaiselta jokaisella näytöllä, on suunnittelu kultaisen leikkauksen kanssa vaivalloista. Sen voi silti ottaa huomioon vaikkapa kuvaa tehtäessä. Mittasuhteen lisäksi on olemassa tasapainolinja. Se voidaan piirtää tekemällä lyhyen sivun levyinen kaarre pitkälle sivulle ja piirtämällä vastasuuntainen lävistäjä esitysalueen poikki. Näiden risteyskohta on optinen piste. Tämän lisäksi

huomiota kannattaa kiinnittää symmetriaan. Ei ole sattumaa, että useimmiten sisältö on keskellä, sillä se on symmetrisesti turvallinen ratkaisu. Asymmetriassa kohteet asetellaan vapaasti, mutta silloin tulee ottaa huomioon kultainen leikkaus, sivujen linjat ja liikesuunnat. (Korpela, Linjama 2005, 367-369.)

Ihmissilmä etsii poikkeavia värejä. Jos poikkeama on suuri, sanotaan sillä olevan suuri kontrasti. Kontrasti voi olla vahvuudessa, eli esimerkiksi teksti on voitu liha-voida. Värikontrastissa käytetään kahden ääripään värejä, kuten vaikkapa tummaa tekstiä vaalealla taustalla. Muotokontrastissa käytetään vierekkäin erilaisia muotoja, kuten vaikka pyöreää ja nelikulmaista elementtiä. Sitten on vielä pintakontrasti, jonka tavallisin esimerkki on tasainen pinta rosoisella pinnalla. (Korpela, Linjama 2005, 369.)

Kontrasti on tehokas tapa erottaa teksti taustasta. Se on myös tehokas tapa erottaa valittu välilehti toisista välilehdistä, joita ei ole valittu. Jos valittu välilehti on valittu auki, se on taustan värinen. Muut välilehdet taas ovat kontrastin ansiosta erinäköisiä, jolloin käyttäjät harvoin kuvittelevat niiden olevan valittuja.

3.7 Teksti ja fontit

Fonttikoko, fonttityyli ja fonttilaji yhdessä määräävät fontin. Yleensä fontilla tarkoitetaan fonttilajia. Fontteja puolestaan voi olla kahta eri tyyliä, groteskia ja antiikvaa. Antiikvassa useimpien kirjainten viivat päättyvät pieneen pääteviivaan, kun taas groteskit ovat pääteviivattomia. Tämän vuoksi näiden fonttien käyttötarkoitukset ovat erilaisia. Antiikvafontit ovat hyviä painettavissa tuotteissa helpompia lukea kuin groteskit, kun taas tietokone heikentää antiikvafontin luettavuutta. Groteskin sietävät paljon enemmän pienentämistä, jonka vuoksi useimmiten ilman suuria tekstialueita olevat web-sivut ovat helpompia luettavia groteskien kanssa. (Korpela, Linjama 2005, 372-373.)

Siksi ei olekaan ihme, että groteskeja fontteja käytetään usein oletuksena web-sivuilla. Lukemisen helppous pitää käyttäjän sivuilla paremmin kuin vaikeasti luettava teksti. Myös C-Caressa käytettään juuri tämän asian vuoksi groteskeja fontteja.

Kannattaa kiinnittää huomiota siihen, että ei käytä tekstissä erilaisia fontteja, sillä ne tekevät sivusta levottoman. Otsikot voivat olla eri fonttia kuin teksti ja samoin tehokeinona erilainen fontti voi toimia. Useimmiten riittää kuitenkin kolme erilaista fonttilajia, joita voi käyttää vaikkapa otsikossa, leipätekstissä ja kuvatekstissä. Jos tämä määrä ylittyy, tulee kokonaisuuden hahmottamisesta haastavaa. (Korpela, Linjama 2005, 373-374.)

Fonttilajien toimivuus riippuu suuresti siitä, että miten laajasti kukin fonttilaji on käytettävissä selaustilanteessa. Fonttilajit vaihtelevat valmistajien mukaan ja myös Microsoft toimittaa erilaisia valikoimia. Suurin osa johonkin Windowsiin kuuluvista fonteista ei toimi jollain muulla Windowsilla. Toinen asia on ulkonäkö. Näytöllä toimivat parhaiten yksinkertaiset ja selkeät fontit. Lisäksi tutut ja tutunkaltaiset fontit toimivat yleensä parhaiten. Lyhyissä ja isokokoisissa teksteissä sekä otsikoissa voi käyttää koristeellisempia ja tyyliteltyjä, muista poikkeavia fontteja. Leipäteksteissä on sen sijaan syytä luottaa groteskeihin. (Korpela, Linjama 2005, 377-378.)

Tekstityyleistä lihavoitu teksti toimii hyvin korostuksena, mikäli sitä käyttää rajoitetusti. Jo virkkeen mittaisina pätkinä korostus kuitenkin heikentää luettavuutta. Linkit ovat puolestaan yleensä erivärisiä ja alleviivattuja. Ne luovat kontrastia niin muotonsa kuin värinsä vuoksi, minkä takia linkit erottuvat selvästi. Linkkien tuleekin erottua muusta tekstistä, samoin kuin käydyt linkit. (Korpela, Linjama 2005, 378.)

Teksteissä kannattaa käyttää yleistä kirjoitustapaa, jossa käytetään sekä isoja että pieniä kirjaimia. Pienet kirjaimet erottuvat toisistaan paremmin ja miellyttävät silmää enemmän kuin isot kirjaimet. Isojen kirjaimien kanssa kannattaakin olla maltillinen, sillä pelkästään isojen kirjaimien käyttö käsitetään huutamisena. Samaan aikaan myös virkkeiden alut, sanakuvat ja erisnimet erottuvat todella huonosti. (Korpela, Linjama 2005, 379.)

Tekstit voidaan tyylitiedostoissa määritellä pikseleinä, millimetreinä tai typografisina pisteinä. Kannattaa kuitenkin huomioida, että eri ympäristöissä olevat selaimet valitsevat siihen sopivat ja säädetyt fontit, jotka valitsevat esityskoon tilanteeseen sopivaksi. Perinteisesti leipätekstin koko on noin 12 pistettä, kolmannen tason otsikon 14

pistettä, toisen tason 18 pistettä ja pääotsikon 24 pistettä. Tällainen tapa sopii laajahkoille dokumenteille. Jos sivulla on vain yksi otsikko, se voi olla pienempikin. Kokoa voi usein määritellä myös prosenttimitoituksella. (Korpela, Linjama 2005, 383.)

Myös Silverlight tarjoaa useita eri vaihtoehtoa. Pikselit ovat kuitenkin osoittautuneet hyväksi ratkaisuksi. Samalla tein useita eri määrityksiä, jolla erottelin erilaisia tekstejä toisistaan. Otsikon erottaa selvästi paremmin suurentamalla fonttikokoa, kuin vaikkapa isoilla kirjaimilla kirjoitettuna, mutta tavallisen kokoisena tekstinä.

3.8 Värien käyttö

Värien käyttö vaatii tarkkaa harkintaa, sillä heikolla ratkaisulla voi pilata koko sivun jopa koko käyttöliittymän osalta. Värien valinta on omalla tavallaan pitkän oppimisen tulos, mutta yleissäännöilläkin pääsee jo pitkälle. Esimerkiksi värien hillitty käyttö käyttäen sateenkaarien värien sijaan vain paria väriä ja muita korostuksen keinoja on tehokasta ja suositeltavaa. (Korpela, Linjama 2005, 392, 394.)

Väri on itsessään todella voimakas viesti. Se erottaa, osoittaa ja järjestelee sekä voi toimia tunnistautumisen keinona. Väreillä on myös symbolisia ja vakiintuneita merkityksiä. Taivaan väri on sininen, ruohon vihreä ja niin edelleen. Sininen tuntuu etäännyttävältä, punainen kiihdyttävältä ja oranssi lämpimänä. Siksi väreillä voikin korostaa sivuja hyvin voimakkaasti esimerkiksi laittamalla punainen reunus jollekin sivun osalle. Punainen väri nimittäin kiinnittää katseen tehokkaasti. Punainen ei voimakkuuden vuoksi sovikaan leipätekstin tai taustan väriksi, vaan varsinkin taustoissa kannattaa käyttää himmeämpiä värejä. (Korpela, Linjama 2005, 392-393.)

Cieltumin nimen osa ”ciel” on ranskaa ja tarkoittaa taivasta ja taivaansinistä. Tämä oli harkittu osa yrityksen ilmettä. Siksi myös käyttöliittymän tulisi olla taivaan värinen, eli sininen. Leipätekstien taustat puolestaan eivät saa olla liian räikeitä, joten noille taustoille sopii hyvin todella neutraali väri.

Värisuunnittelussa voidaan käyttää apuna värien keskinäisiä suhteita. Suhteita on väriharmonia käyttäen lähellä toisiaan olevia värejä, vastavärejä (esimerkiksi punainen

ja vihreä), yhtä tiettyä väriä tai harmaan eri sävyjä. Sävy taas tarkoittaa valon aallonpituutta, joka yleensä ilmaistaan käyttämällä värin nimiä (punainen, sininen, vihreä). Web-sivulla tätä määrittelee rgb-arvo. Kontrasteissa taas tehdään erilaisia rinnastuksia joko päävärien sävyjen kesken, vastavärien mukaan, kylmien ja lämpimien värien mukaan, valovoiman perusteella tai kylläisyysasteen mukaan. (Korpela, Linjama 2005, 394-395.)

Mitkä tahansa värit eivät sovi yhteen ja sopivan väriyhdistelmän lisäksi kannattaa käyttää värejä hillitysti. Yleensä kannattaa valita vain kolme eri väriä, joista yksi on taustan väri, yksi tekstin väri ja kolmas korostuksien, otsikoiden ja tehostusten väri. Värin valinta kannattaa aloittaa suurimmasta ja hallitsevimmasta. (Korpela, Linjama 2005, 395-396.)

Yksi hyvä nyrkkisääntö on käyttää tummaa tekstiä vaalealla pohjalla. Musta tai tumman harmaa voivat joskus sopia taustaksi, mutta useimmiten ne eivät toimi. Se johtuu siitä, että valkea teksti mustalla taustalla on vaikeampaa lukea kuin musta teksti vaalealla taustalla. Vaalea teksti vaatiikin usein fonttikoon suurentamista, mutta erityisesti lyhyissä esittelyissä tämä ei toimi. Keskenään tasavahvat värit tai vastavärit tekevät tekstistä myös vaikean luettavan. Tekstin ja taustan tulisikin muodostaa tarpeeksi suuri kontrasti, jotta teksti erottuu. Silti vaaleaa tekstiä on useimmiten vaikeaa luettavaa. (Korpela, Linjama 2005, 400.)

Harva suosittu sivusto nykyään käyttää vaaleaa tekstiä tummalla taustalla. Uskon, että se vieroksuttaa kävijöitä. Sen sijaan tumma teksti vaalealla taustalla miellyttää lähes kaikkia ja siksi se onkin hyvä ratkaisu.

Sininen väri on useimmiten käytetty linkkeinä. Useat selaimet käyttävät automaattisesti tummansinistä vierailemattoman linkin värinä, joten sininen mielletään useimmiten muuksi kuin tekstiksi. Samasta syystä myös violettia väriä kannattaa välttää, sillä se on usein vierailtujen linkkien väri. (Korpela, Linjama 2005, 401.)

4 KÄYTETTÄVÄT TEKNIIKAT

4.1 Silverlight

4.1.1 Yleistä

Silverlight on ilmainen, Aboden Flash Playerin kaltainen ohjelmointiympäristö, joka toimii useassa eri käyttöjärjestelmässä ja Internet-selaimessa. Silverlight tukee Microsoftin käyttöjärjestelmiä, Mac Os X:ää sekä Linuxia. Silverlightia voi käyttää myös miltei jokaisella selaimella, kuten Internet Explorerilla, Mozilla Firefoxilla, Google Chromella, Safarilla ja Operalla. Tuki löytyy myös Microsoft Phone 7:lle. (Microsoft 2011.)

Silverlight perustuu Microsoftin omaan WPF-tekniikkaan (Windows Presentation Foundation). Voisi sanoa, että Silverlight on kevennetty versio WPF:stä. Microsoftin Visual Studio ja Experssion Blend tarjoavat toimivaa välineistöä Silverlightin toteutukselle ja Silverlightilla kykenee luomaan käyttökelpoisia ohjelmia. Se pyrkii parantamaan sovelluskehityksen tehokkuutta, tehostamaan nykyaikaisten laitteistojen hyödyntämistä sekä mahdollistaa graafisten sovellusten ajamisen selaimessa ja sen ulkopuolella. (Heikniemi 2010, 68.)

Käyttöä helpottaa myös se seikka, että Silverlight tukee .NET-ympäristöä. Näin ollen esimerkiksi ASP.NET-ohjelmointimenetelmää käyttäneet tottuvat nopeasti Silverlightiin. Lisäksi Silverlight kykenee ymmärtämään useita tuttuja ohjelmointikieliä. Siksi ei ole väliä, että käyttääkö Visual Basicia vai C#-kieltä. (Heikniemi 2010, 68.)

4.1.2 XAML

XAML on lyhenne sanoista Extensible Application Markup Language. Sillä voidaan lisätä objekteja ja sen avulla voi tehdä määritelmiä lisätyille objekteille. Se käyttää tekniikkaa, joka näyttää hierarkkisesti objektien suhteet. XAML:n avulla voi tehdä myös ulkonäöltään valmiita käyttöliittymiä. Metodeja ja tapahtumia voidaan taas

tehdä "code behindissa" eli erillisessä ohjelmakooditiedostossa, josta tulee lisää myöhemmin. XAML-kieltä kykenee myös käyttämään esimerkiksi Visual Studion ja Expression Blendin välillä ilman tiedon häviämistä. Yksinkertaistettuna XAML on kuitenkin XML-tiedosto, jolla on tiedostopäätteenä .xaml. (MSDN A.)

.NET Framework 3.0 teknologiassa käytetään usein XAML:ia. Pääasialliset käyttötarkoitukset ovat kuitenkin WPF ja Silverlight, joissa sillä kyetään tekemään ulkoasu, datasidontaa sekä käsittelemään erilaisia tapahtumia. Silverlightissa XAML on upotettu HTML-sivun sisälle, jota HTML kykenee ymmärtämään erillisen liitännäisen avulla. Tässä asiassa XAML muistuttaa hieman Javascriptiä. (Anatharam 2008.)

Silverlightissa XAML:lla on monia tärkeitä rooleja. Ylivoimaisesti tärkein on XAML:n toimiminen sivun käyttöliittymänä. Tyypillisesti vähintään yksi XAML toimii Silverlightissa pääsivuna ja sen sisälle voi taas lisätä pienempiä XAML-tiedostoja, jotka taas ovat erilaisia kontrolleja ja elementtejä. XAML:lla määritellään myös käyttöliittymän tyylit, sekä valmiiksi määritellyt objektit. XAML toimii myös reittinä Silverlightin ja WPF:n välillä, sillä kummallakin on sama nimiavaruus. Käyttöliittymän suunnittelu tehdään erillisesti XAML:ssa ja kaikki logiikka on erillisessä luokassa. Tämän ansiosta samaa sivua voi tehdä kaksi henkilöä, joista toinen tekee käyttöliittymää ja toinen taustalla koodia. (MSDN A.)

XAML:ssa objektien tekoon pätee tietyt säännöt. Objektin alkupisteessä laitetaan objektin nimi kulmasulkeisiin, ja lopussa kulmasulkeiden sisälle laitetaan objektin nimen eteen vinoviiva. Tässä sukulaisuus XML:ään näkyy selvästi. Esimerkiksi Canvas-niminen objekti luodaan näin:

```
<Canvas></Canvas>.
```

Monen objektin sisälle voi myös laittaa toisia objekteja. Se onnistuu esimerkiksi näin:

```
<Canvas>
 <Rectangle></Rectangle>
</Canvas>
```

Määrittelyt tehdään objektien nimien sisälle:

<Canvas Margin="200,4,4,4" Width="80" Height="100" Background="Blue"></Canvas>

Silverlight sisältää oletuksena suurimman osan XAML:iin tarvittavasta tiedoista. Monimutkaisempia objekteja voi olla kuitenkin pakko hankkia oletusarvojen ulkopuolelta. Näitä varten täytyy hakea ja määritellä uusia nimiavaruuksia. Näitä voivat olla esimerkiksi oletusten ulkopuoliset Silverlight-objektit, Silverlightin kehittäjätyökalujen avulla tehdyt objektit, kolmannen osapuolen tekemät objektit, tai itse tehdyt objektit. Nämä nimiavaruudet voidaan määritellä sivujen alussa, jolloin ne tulevat XAML:ssa käyttöön. Niille täytyy kuitenkin määritellä omat etuliitteensä ja näitä objekteja kutsutaan näiden etuliitteiden avulla. Esimerkiksi jos määritellään luokka x, jossa on objekti y, niin objekti haetaan muodossa <x:y></x:y> (MSDN A.)

XAML:n suurin heikkous lienee kuitenkin se, että se vaatii erillisen liitännäisen selaimeen toimiakseen. Silverlightin yleistyessä ongelma on kuitenkin huomattavasti pienentynyt. Itse kielessä on myös useita pieniä erikoisuuksia. Esimerkiksi ajoittain hierarkiassa alempana oleva objekti valuu ylempänä olevan objektin päälle, vaikkei sen pitäisi edes olla mahdollista. (Anatharam 2008.)

4.1.3 Ohjelmakooditiedosto

Ohjelmakooditiedostolla tarkoitetaan luokkaa, joka yhdistää koodin ja XAML:n. Nämä sitten yhdistetään, kun sovellusta luodaan. tiedosto ja XAML toimivat toisin sanoen toisiaan tukien. (MSDN B.)

Tähän luokkaan voidaan tehdä tavallista koodia, jolla saadaan toiminnallisuuksia XAML:iin. .NET-tuen ansiosta koodia voi tehdä useammalla ohjelmointikielellä, minkä takia minäkin olen käyttänyt tuttua C#-kieltä. Luokassa voi tehdä normaalin ohjelmoinnin tapaan erilaisia metodeja, tapahtumia, olioita sekä hallita tietokantoja.

Ohjelmakooditiedosto on siis toiminnallinen puoli Silverlightissa, kun taas XAML on graafinen puoli.

4.2 Azure

Windows Azure on Microsoftin käyttöjärjestelmä pilvipalveluille. Azurea ajetaan maailman eri kolkissa sijaitsevissa, toisiinsa yhdistetyistä Microsoft Windows Server -pohjaisista tietokoneresursseista. Näiden resurssien kautta käyttäjä voi hallinnoida sovelluksia sekä ylläpitää tietokantaa Internetissä ilman omia palvelimia. Windows Azure voidaan jakaa kahteen osaan. Nämä ovat laskenta- ja tallennusympäristö. Käyttöjärjestelmän pohjalle on puolestaan rakennettu oma alustansa Windows Azure Platform, mihin kuuluvat AppFabric, SQL Azure ja itse Windows Azure. (Betts 2010, 1-2.)

Windows Azuren isännöimät resurssit sisältävät kahdenlaisia rooleja:

- työskentelijän rooli
- web-rooli

Nämä rooli-instanssit toimivat rajapintana Windows Azuren ja käytettävän sovelluksen välillä. Työskentelijänä roolit isännöivät koodia ajaen jatkuvasti pitkäkestoisia tehtäviä. Web-rooli puolestaan hyväksyy sovelluksesta tulevia HTTP- tai salattuja HTTPS-kutsuja. Web-roolissa on se hyvä puoli, että se toimii minkä tahansa sovelluskehitykseen tarkoitetun alustan kanssa, missä on tuki IIS:lle. Tällaisia alustoja ovat .NET-alustat, kuten ASP.NET, Microsoft Silverlight, Windows Communication Foundation sekä PHP. (Betts 2010, 1-5.)

Windows Azuressa tieto kulkee kahteen suuntaan. Toisaalta molemmat roolit voivat tehdä yhteyksiä ulospäin ja avata avoimia päätepisteitä sisäänpäin tuleville kutsuille. Käytännössä Azuressa pyörivässä sovelluksessa voi olla joko työskentelijä- tai web-rooli, mutta mikään ei estä sitä, etteikö Azure voisi sisältää molempia. Web-roolin käskyt voi esimerkiksi siirtää WCF-palvelun välittämänä työskentelijän rooliin. (Betts 2010, 3-6.)

4.3 MVVM-arkkitehtuuri

MVVM (Model – View – ViewModel) on malli, jossa ulkonäkö on view, model on itse data ja viewmodel on näiden kahden välissä oleva osa. Tämänkaltaisia malleja ohjelmoijat ovat käyttäneet jo pidemmän aikaa helpottaakseen omaa työtään ja tehdäkseen koodista helpommin hallinnoitavan. Josh Gossman julkaisi MVVM:n vuonna 2005, mikä taas on arkkitehtuurinen suunnitelma nimenomaan Windows Presentation Foundationille ja Silverlightille. (Smith.)

Syitä MVVM:n käytölle on useita. Pääsyy on helppo datasidonta, jossa näyttöön voi helposti sitoa koodissa olevia elementtejä ja komentoja. MVVM mahdollistaa myös datan muokkauksen näytöstä tulevan tiedon avulla. Toisaalta se myös mahdollistaa datapohjien käytön. Lisäksi MVVM mahdollistaa helpon rakenteen, joissa tietyt asiat tapahtuvat aina jossain tietyssä luokassa. Tällöin koodia on helppo testata ja ylläpitää. Kaiken lisäksi data ei muutu, vaikka näyttö muutettaisiin ja poistettaisiin kokonaan, sillä data on näytöistä täysin erillään. (Smith).

MVVM:n näyttö tarkoittaa käytännössä XAML:ia, jossa voidaan päättää sovelluksen ulkonäkö. Näyttöön voi myös sitoa dataa. Viewmodelia käytetään puolestaan useimmiten näytöstä tulevien komentojen toteuttamiseen ja muuhun datan käsittelyyn. Model puolestaan esittelee datan Viewmodelille, josta se etenee kohti näkymää. (Smith.)

4.4 Kieli- ja tyylitiedostot

4.4.1 Kielitiedostot

C-Care on suunniteltu käytettäväksi useammalla kielellä. Sitä varten sovelluksesta löytyy kielitiedostoja. Nämä ovat sovelluksen Languages-kansiossa. Tällä hetkellä kielet ovat suomi ja englanti, jotka löytyvät language.fi ja language -nimillä. Kielitiedostot ovat .resx-tyylisiä ja ne saadaan luomalla uusi resurssitiedosto. Näistä language on englanninkielinen ja se on pääasiallinen kielitiedosto, joka valitsee, että onko kielitiedosto julkinen vai sisäinen.

Kielitiedoston käsittelyssä on kuitenkin bugi. Aina, kun englanninkielistä kielitiedostoa muokkaa, muuttuu julkinen luokka kielitiedoston Designer-puolella sisäiseksi. Se täytyy aina muistaa muokata julkiseksi, tai muuten uudet muutokset eivät näy muualle sovellukseen.

Tiedoston käyttö on helppoa. Siellä on kohtina nimi, arvo ja kommentti. Näistä vain nimi ja arvo ovat tarpeellisia. Nimeen laitetaan jokin kuvaava nimi (esim. AboutLinkText) ja sille annetaan arvo. Samanniminen nimi tulee antaa kumpaankin tyylitiedostoon. Arvoksi sitten laitetaan englanninkieliseen tiedostoon englanniksi haluttu teksti ja vastaavasti suomenkieliseen laitetaan suomeksi. Tämän jälkeen tämä nimi voidaan sitoa mille tahansa sivulle, joka käyttää nimiavaruutena kielien kansiota (xmlns:Languages="clr-namespace:PilotSilverlight.Languages"). Lisäksi sivujen resursseissa täytyy määritellä jokin nimi tälle nimiavaruudelle (C-Carella LocalLang). Tämän jälkeen tekstit voidaan sitoa määrittelemällä eri kontrollien teksti- tai sisältökenttiin arvoksi vaikkapa Content="{Binding Source={StaticResource LocalLang}, Path=ResultPgOpenText}".

4.4.2 Tyylitiedostot

Silverlight mahdollistaa tyylitiedostojen tekemisen. C-Caressa on kaksi tyylitiedostoa ja nämä ovat Assets-kansiossa nimillä StdStyles ja Styles. Ne on tehty tekemällä uusi Silverlight Resource Dictionary -sivu, joilla on tiedostopääte .xaml. Normaalista sivusta se eroaa siten, että kaikki koodi on laitettu ResourceDictionary-luokan sisälle. Silverlight tukisi todennäköisesti .NET-tuoteperheen kuuluvana myös tavallisempaa .css-tiedostoa, mutta käyttöön on otettu Silverlightin oma kontrolli.

Tyylitiedostoissa tyylin tekeminen on todella yksinkertaista. Hierarkiassa ylimmäksi tulee Style, jolle tulee antaa yksilöllinen nimi (jos on samanniminen tyyli, niin siitä tulee virhe) ja määritellä sen kohdetyyppi. Kuvassa yksi tyylin nimi on ViewBorderStyle ja se vaikuttaa juuri reunuksiin. Tyylille on tarvetta, sillä samanlaisia reunuksia on jokaisella sivulle useampia ja jokaisen reunus on musta ja se kääntyy kul-

missa viiden pikselin verran. Tämän voisi laittaa jokaiselle reunalle yksitellen, mutta yksinkertaisempaa on tehdä sille oma tyylitiedostonsa ja kutsua sitä datasidonnan avulla. Aina kun kyseistä tyyliä kaivataan reunukseen, voidaan kirjoittaa Style="{Binding Source={StaticResource ViewBorderStyle}}". Näin säästyy aikaa. Ja jos haluaa jossain tilanteessa muuttaa kaikkia reunuksia, voi mennä suoraan tyylitiedostoon, jolloin ei tarvitse muuttaa kaikkien XAML:ien tietoja.

```
<!--Views border style-->
<Style x:Key="ViewBorderStyle" TargetType="Border">
    <Setter Property="BorderThickness" Value="1"/>
    <Setter Property="BorderBrush" Value="Black"/>
    <Setter Property="CornerRadius" Value="5"/>
    <!--Setter Property="VerticalAlignment" Value="Top"></Setter!-->
    <!--Setter Property="HorizontalAlignment" Value="Left"></Setter!-->
</Style>
```

Kuva 1. Reunuksen määrittely tyylitiedostossa.

Tyylitiedosto ei kuitenkaan lähde vielä toimimaan automaattisesti, eikä sitä voi sitoa heti muille sivuille. Sen olemassaolo pitää ensiksi ilmoittaa. Silverlight tekee automaattisesi App-nimisen xaml-tiedoston. Se on oletuksena tyhjä, mutta sille voidaan ilmoittaa sovelluksen käyttämät resurssit. Se ei ole onneksi monimutkainen prosessi, vaan XAML on todella yksinkertainen. Resursseihin pitää vain käytännössä kirjoittaa resurssikirjaston olemassaolo. Kun tämä on tehty, datasidonta jokaisesta saman kansion sivusta on mahdollisuus tehdä tyylitiedostojen tyyleihin.

```
<Application.Resources>
    <ResourceDictionary>
        <ResourceDictionary.MergedDictionaries>
            <ResourceDictionary Source="Assets/Styles.xaml"/>
            <ResourceDictionary Source="Assets/StdStyles.xaml"/>
        </ResourceDictionary.MergedDictionaries>
    </ResourceDictionary>
</Application.Resources>

/Application>
```

Kuva 2. Resurssikirjaston määrittäminen

Tyylitiedoston käyttäminen on suositeltavaa sen takia, että sitä kautta voi saada helposti useasti toistuvia tyylejä, joita ei tarvitse aina koodata XAML:iin. Samalla se mahdollistaa helpon muokattavuuden, sillä tyyliä tarvitsee muokata vain tyylitiedos-

tossa. Lisäksi se tuo selkeää yhdenmukaisuutta, sillä tyylitiedoston kautta määrittelyt kohteet ovat varmasti samanlaiset jokaisella kerralla.

5 KÄYTTÖLIITTYMÄN UUDISTAMINEN

5.1 Alkutilanne

Cieltumilla on oma sovellus nimeltä C-Care. Siihen on alun perin jo suunniteltu käyttöliittymä, jonka pohjalle sovellus on rakennettu. Alkuperäistilanteessa oli pyritty yhtenevään värimaailmaan sekä yhdenmukaisuuteen. Muun muassa samaa fonttia, samankaltaisia ikoneita ja kooltaan samanlaisia kuvia oli pyritty käyttämään.

Kuitenkin sovellus kaipasi selvää kasvojenkohotusta. Käyttöliittymä ei ole ollut C-Caren kehittäjien mielessä etusijalla, vaan taustakoodi oli ajanut edelle. Se näkyi siinä, että kukin sivu oli hivenen erilainen kuin toinen. Havaittavissa oli myös se seikka, että kukin kehittäjä oli laittanut kuhunkin koodiin sopivia kuvia, tekstejä, ikoneita ja sen sellaisia tavalla, joka itselle parhaiten sopi.

Siksi ongelmaksi tulikin yhtenäisyys. Käyttäjälle ei voi antaa tuotetta, jossa toiminnot eivät ole loogisesti samassa paikassa. Sovelluksen rakenne oli heikko ja sivujen koot vaihtelivat, eli jotkin sivut olivat leveämpiä tai pidempiä kuin toiset. Toisaalta sisällönkin rakenne vaihteli. Sovelluksessa on yläpalkki, jossa on usein navigaatioon liittyviä nappuloita. Tämä yläpalkki kuitenkin oli aina välillä toisaalla korkeampi kuin toisaalla. Sisäosien reunukset eivät myöskään olleet aina yhtä leveitä.

Cieltum Oy käy tiivistä yhteistyötä eri yritysten ja asiakkaiden kanssa. Käyttöliittymä sai siis palautetta toisaalta. Usein toistuvia asioita oli yhtenäisyyden lisäksi muun muassa se, että sovellusta oli vaikea käyttää. Uudet käyttäjät eivät kunnolla tienneet missä päin sovellusta he olivat, eivätkä tienneet mitä mikäkin painike tekee, eivätkä saaneet välttämättä selvää, mihin sivu oli tarkoitettu. Navigointi oli monen mukaan liian vaikea ja turhia napin painalluksia oli liikaa.

Myös ulkonäkö sai kritiikkiä. Sovellus oli ”insinöörimäinen” sekä keskeneräisen oloinen. Värimaailma ei ollut huono, mutta sille kaivattiin uusia ideoita. Ja ennen kaikkea ilme piti saada valmiiksi. Tähän eivät sopineet esimerkiksi reunojen alle osittain jäävät elementit, irrallaan muusta sivusta olevat painikkeet tai reunusten päälle menevä sisältö. Myös kuvakkeiden ja tekstien suhdetta ja ulkonäköä piti saada paranneltua visuaalisesti miellyttävämmäksi.

Kuten liitteessä yksi oleva etusivun alkutila osoittaa, ulkonäkö kaipasi kasvojenkohotusta. Parannusta kaivattiin niin käytettävyyteen, itseohjautuvuuteen, yhtenäisyyteen kuin ylläpidettävyyteen kaivattiin. Muun muassa näitä edellä mainittuja virheitä lähdin korjaamaan tämän opinnäytetyön merkeissä.

5.2 Testaaminen

Sovelluksen testaaminen on oleellinen asia, sillä testaaminen on ainoa tapa tutkia sivuston toimivuus. Mutta ennen kaikkea se tuo aina uusia ajatuksia kehittämistä varten. Siksi se on prosessi, jota tulisi tehdä jatkuvasti. Testaaminen heti alussa on hyödyllisempää kuin lopussa, mutta tähän ei nyt kyetty, sillä sovellusta oli jo kehitetty pitkälle. Kuitenkin se on parempi kuin ei mitään ja toisaalta yhdenkin käyttäjän testaaminen on parempi kuin se, ettei testattaisi ollenkaan. Testaamalla saa arvokasta palautetta, jonka avulla voi yhdessä omien kokemusten, ammattitaidon ja terveen järjen kanssa voi viedä sovellusta eteenpäin ja tehdä tarvittavia korjauksia.
(Krug 2006, 133-135.)

Cieltumin testaus on ollut pitkään yhden henkilön harteilla. Yrityksessä on yksi lähinnä SharePointiin keskittynyt henkilö, joka on kyllä käyttöliittymää miettinyt, mutta joka ei ole tehnyt riviäkään koodia. Hän on sitten testannut aina kulloistakin versiota ja etsinyt virheitä. Cieltumin asiakkaat myös luonnollisesti antavat palautetta niin erilaisten palaverien kuin käytön aikana.

Minä olen myös muokkauksen aikana pyrkinyt miettimään, että miten mitäkin sivua voisi saada paremmaksi. Samaten olen miettinyt, että kertooko jokin asia tarpeeksi paljon ensi kertaa sovellukseen tulevalle käyttäjälle. Olen myös pyrkinyt kokeile-

maan ohjelmaa mahdollisimman paljon ja pyrkinyt tekemään erilaisia asioita, jotka ovat sovelluksen logiikan ulkopuolella. Täytyy kuitenkin muistaa, että havaittavissa on aina sokeus omalle työlle sekä tottuminen sovellukseen, jonka vuoksi tiettyjä asioita ei tule ajatelleeksi.

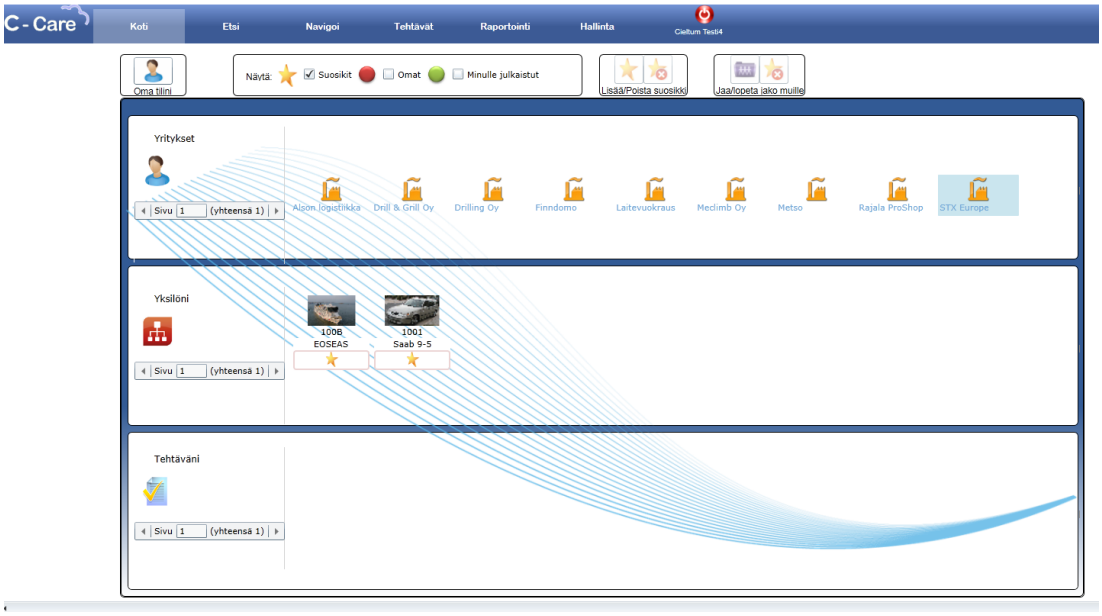
Halusin testata C-Caren henkilöllä, joka ei ollut käyttänyt sovellusta aiemmin. Ja koska ystäviä ei tarvitse pelätä testaamisessa, niin pyysin ystävääni testaamaan sovelluksen (Krug 2006, 141.). Tässä testissä laitoin sovelluksen kutsukoodin sähköpostiin, jonka avulla kykenee rekisteröitymään C-Careen. Kun rekisteröityminen oli onnistunut, annoin ystäväni hetken aikaa pyöritellä sovellusta. Vähän ajan kuluttua annoin hänelle vielä joitain tehtäviä, joita tulisi tehdä. Testi oli varsin hyödyllinen, sillä se paljasti sovelluksesta kohtia, joihin ei kiinnitetty huomiota ollenkaan. Lisäksi samalla vaivalla sovelluksesta löytyi useampia bugeja.

5.3 Ulkonäön muokkaaminen

5.3.1 Väritys

C-Caren väritys alkuperäisessä tilassa (Kuva 3.) oli varsin hyvä. Taustalla oli tummansinistä väriä, tekstien taustat olivat vaaleita ja teksti mustia. Nämä olivat hyviä asioita, mutta viimeistely ontui. Siksi tein jotain pieniä muutoksia väritykseen.

Taustavärin tummemman sininen säilytettiin, mutta laitoin sivuille liukuvärin. Tässä tapauksessa sekä sivun ylä- että alaosassa väritys on tumman sininen, mutta liukuvärit vaihtavat taustan värityksen puoliväliin mennessä. Puolivälissä sävy on edelleen melko tumma, mutta keskellä käytetty sininen on kuitenkin selvästi vaaleampi, kuin sivun ulkoreunassa käytetty sävy.



Kuva 3. Alkuperäinen etusivu

Reunuksissa oli selkeää parantamisen varaa. Mustat reunukset selvittävät selvästi, mihin elementti loppuu. Mutta ainakin omaan silmääni noin räikeä muutos oli huono asia, sillä musta väri korostui turhan paljon. Siksi tein reunuksille uuden taustavärin. Taustaväri on suurimmalta osalta valkoinen, mutta loppuun laitoin pienen efektin. Reunuksissa on liukuväri, mutta lopuksi se muuttuu vaalean siniseksi. Tämä tuo kivan pikku lisän reunukseen.

Tämän lisäksi pyrin laittamaan monien reunusten oikealle puolelle pienen varjostuksen. Tämä näyttää varsin hyvältä pieneltä efektiltä, joka tuo heti viimeistellymmän kuvan. Varjostus Silverlightilla onnistuu valitsemalla XAML:ssa kulloisellekin elementille efekti (<elementti.Effect>) ja sijoittamalla sen sisäpuolelle DropShadowEffectin. Varjostus ei ole iso, mutta kuitenkin kohtuullisen näyttävä. Mutta jottei efekti dominoisi liikaa, niin määrittelin sen arvoksi seuraavat:

Opacity="0.25" BlurRadius="4" Direction="2" Color="#FF365799"

5.3.2 Efektit

Efektien käytön avulla voi luoda erilaisia liukuvärejä. Nämä voidaan tehdä LinearGradientBrush-toiminnolla, joka puolestaan voidaan liittää eri elementien taustaan kiinni. Nämä pystyy puolestaan liittämään suoraan XAML:iin, tai sitten ne voi sitoa tyylitiedostoihin.

Huomasin myös, että XAML:iin tehty määritys korvaa tyylitiedoston kautta tehdyn määrityksen, mikäli samaa elementtiä on käytetty molemmissa. Siten on mahdollista vielä tarpeen tullen tehdä muutoksia XAML:ssa, mikäli sivu vaatii joitain pieniä tavallisuudesta poikkeavia asioita. Näin voi esimerkiksi olla, mikäli reunuksen taustan tulisi olla läpinäkyvä, vaikka se pääosin olisikin valkoinen.

Esimerkkinä reunuksen taustalla oleva liukuväri voidaan asettaa näin:

```
<Border.Background>
    <LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
        <GradientStop Color="#FF325A96" Offset="0.248" />
        <GradientStop Color="White" Offset="0.969" />
    </LinearGradientBrush>
</Border.Background>
```

Silverlightilla on mahdollisuus tehdä eri elementeille myös varjostuksia. Nämä näyttävät hyviltä ja luovat kuvaa viimeistellystä ohjelmasta. Varjostus toimii DropShadowEffect-nimisellä luokalla, jolle voidaan määritellä varjostuksen väri, pituus, suunta, syvyys ja läpinäkyvyys. Tämän luokan voi liittää miltei jokaiseen elementtiin, mutta sen käyttöä kannattaa harkita. Esimerkiksi tekstistä tulee todella helposti epäselvä, mikäli siihen lisää varjostuksen. Siksi tyydyin vähäisiin varjostuksiin. Laitoin pienet kahden pikselin kokoiset varjostukset sovelluksesta löytyvien reunusten oikealle puolelle reunusten väriä käyttäen 25 % läpinäkyvyydellä. Näin sovelluksessa on pienet ja tyylikkäät varjostukset, jotka eivät kuitenkaan hypi turhaan silmille.

Efektejä käyttäessä tulee ottaa huomioon, että niiden käsittely vie aina aikaa. Taustalla on aina matemaattisia laskutoimituksia. Siksi tein asian tiimoilta testin ja kellotin ajan, mikä kuluu sovelluksen ajamisessa selaimen aukeamisesta sivun valmistumi-

seen. Alkutilanteessa oli jo näytön taustalla liukuvärit, mutta lisäsin testimielessä liukuefektejä myös muiden reunusten sisäpuolelle, jolloin liukuvärien määrä lisääntyi kuudella. Keskimäärin tämä hidasti ajoa noin sekunnin. Selaimessa ero lienee pienempi, mutta tämä asia on syytä ottaa huomioon efektejä tehtäessä. Aina, kun efekti tehdään, se tulee hidastamaan sovellusta. Siksi koko sovellus ei voi olla täynnä efektejä.

5.3.3 Skaalautuvuus

C-Caressa oli selkeä ongelmana se, etteivät eri sivut muokkautuneet näytön koon mukaan. Tämä taas aiheutti sen tilanteen, että isommissa näytöissä sivut jäivät vajaiksi, kun taas pienemmän resoluution näytöillä kaikki sisältö ei mahtunutkaan enää sivuille. Tätä ongelmaa ei koodatessa osattu ajatella, sillä ohjelmoijien näytöillä oli suuret resoluutiot ohjelmointityön helpottamiseksi. Testatessa toisella koneella ongelma kuitenkin tuli selkeästi esille.

Ongelma johtui siitä, että sivujen uloimmassa reunassa olevien Gridien palstojen leveydet olivat kovakoodattuja joihinkin tiettyihin arvoihin. Samoin jotkin sivujen elementit (kuten vaikkapa tietoja näyttävät ListBoxit) olivat kovakoodattu leveiksi. Joissain tilanteissa toki on pakko laittaa joitain leveysarvoja, mutta nämä eivät saa olla oikeassa reunassa.

Ratkaisuna ulkoasua säätelevän Gridin sarakkeiden arvoja tuli muuttaa. Aivan vasemmassa reunassa voi olla arvoja (vasemmalla oleva reunus on 440 pikseliä leveä), mutta tämän jälkeen ei ole suotavaa olla tarkkoja arvoja. Viimeiseksi laitoin useimmiten sarakkeen leveyden arvoksi ”*” ja minimileveydeksi 200. Tämä tarkoittaa sitä, että sarakkeen leveys on ainakin 200 pikseliä, mutta kykenee tarpeen tullen täyttämään näyttöön jäävän alueen. Otin myös Gridin sisäpuolisista elementeistä paljon leveysarvoja pois.

Yllätykseksi huomasin, että Gridin jälkeisen lapsen tuli olla reunus (eli Border), jotta lapsi mukautuisi Gridin arvojen mukaisesti. C-Caressa useimmiten käytetty ratkaisu oli StackPanel, mutta syystä tai toisesta se ei toiminut kunnolla. StackPanelien

käyttö normaali olosuhteissa on toimivaa, sillä sen sisälle voi laittaa useampia lapsia. Näin ei voi tehdä reunuksessa, sillä reunukseen mahtuu vain yksi lapsi. Niinpä tein lukuisia reunuksia, kiinnitin ne Gridiin ja laitoin StackPanelit reunusten sisälle.

```
</Border.Background>
 <Grid>
        <Grid.ColumnDefinitions>

        <ColumnDefinition Width="440" ></ColumnDefinition>
        <ColumnDefinition Width="3"></ColumnDefinition>
        <ColumnDefinition Width="736*" ></ColumnDefinition>
    </Grid.ColumnDefinitions>
        <Grid.RowDefinitions>
```

Kuva 4. Esimerkki leveyden määrittämisestä

Lisäksi reunuksissa täytyi vaihtaa HorizontalAlignment ja VerticalAlignment-arvot muotoon ”Stretch”, jotta reunus täyttyi kokonaan. Valittuna pitää olla myös määritellyn Gridin arvojen mukaan Gridin rivi, sarake ja näiden kesto.

<Border x:Name="contentBorder" VerticalAlignment="Stretch" HorizontalAlignment="Stretch" Grid.Row="1" Grid.RowSpan="1" Grid.Column="1" Grid.ColumnSpan="1" Style="{Binding Source={StaticResource ViewBorderStyle}}">

Tämän jälkeen sivulle jäi oikeaan laitaan tilaa halutun marginaalin verran. Toki tämäkään ratkaisu ei aina toimi täydellisesti, mikäli sivuilla on paljon täytettä, tai joitain kohtia, mitkä vaativat staattiset arvot. Näin on esimerkiksi kuvanavigoinnissa, jossa kuvan lähentäminen ja loitontaminen vaativat selkeät arvot leveyteen ja korkeuteen. Lisäksi Gridien sarakkeita ja rivejä ei voi muokata tyylitiedostoissa, vaan työ täytyy tehdä jokaisen sivun kohdalla erikseen.

5.3.4 Kuvakkeiden ja tekstin parantaminen

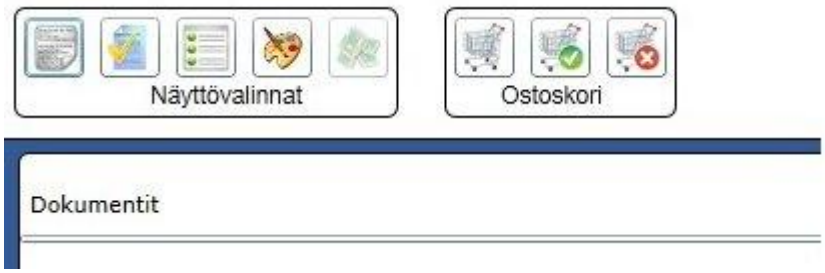
Alkutilanteessa C-Caren navigointi ja toiminnallisuus oli laitettu kuvakkeiden taakse. Tämä näytti hyvältä ulkonäöllisesti, erityisesti koska kuvakkeet olivat samalla tyylillä toteutettuja. Ongelma tuli siinä, että kuvakkeiden toiminnallisuutta oli harvoin se-

litetty tekstimuodossa. Tämä aiheutti sen, että kaikkialla Microsoftin tuotteista löytyvien peruskuvien (tallenna, leikkaa, liitä) ulkopuoliset kuvat olivat monille hämärän peitossa. Minäkään en aina tajunnut, mitä kuvakkeiden pitäisi tehdä.

Selatessani Internetin suosittuja sivuja (mm. Facebook, Wikipedia ja Twitter) tein havainnon, että kuvakkeet ovat hyvin harvassa paikassa pääasiallinen tapa navigoida. Enemmän ratkaisuina käytettiin navigointia tekstin avulla. Tämän vuoksi päätin tehdä muutoksia sovellukseen ja lisätä tekstiselityksiä. Tässä tilanteessa tein aluksi kahdenlaisia muutoksia.

Joissain kohdissa tyydyin vain lisäämään kuvakkeiden viereen tai yläpuolelle tekstiselityksen. Nämä olivat yleensä tilanteita, joissa oli lähekkäin vain yksi kuvake tai kaksi kuvaketta. Näihin kohtiin siis mahtui helposti lisäselvitys. Tällainen oli esimerkiksi tehtävien hakuun suunnitellulla sivulla. Tässä nimenomaisessa tilanteessa levensin reunuksen, laitoin kuvakkeet allekkain ja lisäsin kuvauksen siitä, mitä linkki tekee. Linkin kuvaukseen liitin toiminnallisuudeksi saman komennon, mikä oli jo kuvakkeissa. Näin käyttäjä ei joudu väkisin napsauttamaan kuvaketta. Lisäksi laitoin hakupalkin yläpuolelle kuvauksen siitä, mitä tämä sivu hakee.

Samankaltainen ongelma vaivasi yläpalkissa olevia navigointivalintoja. Monella sivulla oli yläpalkissa ikoneita, joissa ei ollut edes kuvausta siitä, että ne olivat suunniteltu näyttövalinnoiksi. Navigointisivu oli tästä hyvä esimerkki. Yläpalkista pystyi valitsemaan sen, että näytetäänkö sivuilla jonkin kohteen ominaisuudet, tehtävät, dokumentit, kuvanavigointi vai kustannukset. Nämä kaikki valinnat oli tehty kuvakkeiden taakse ja nämä kuvakkeet taas eivät olleet selkeitä.



Kuva 5. Navigointisivun yläosa ja kuvakkeet alkuperäisessä tilanteessa.

Siksi tein muutoksia yläpalkkiin. Pienensin kuvia, jotta niitä reunuksen sisälle mahtuisi kaksi allekkain. Mutta ennen kaikkea lisäsin kuvien jälkeen korostetun tekstin, joka kertoi mitä painikkeet tekevät. Lisäksi lisäsin teksteihin samat komennot, mitä ikoneissa oli. Tämä selkeytti selvästi toiminnallisuutta.

Tästä muutoksesta tuli nopeasti hyvää palautetta. Kommentit olivat yleensä sen suuntaisia, että kuvakkeet olivat selvästi aiempaa selkeämpiä. Tämän lisäksi tekstien lisääminen kuvakkeen viereen aiheutti nopeasti työpaikan sisällä uuden ajattelutuokion. Tämän perusteella päädyttiin siihen, että moni haluaisi tekstin ilmoittavan kaiken toiminnallisuuden jopa näyttävyyden kustannuksella. Siksi Microsoftin paljon käyttämien kuvakkeiden ulkopuoliset kuvat päädyttiin suunnittelemaan uudestaan siten, että niissä on mukana myös teksti.

Painike haluttiin luoda siten, että olisi yksi tietty muotti, jonka pohjalta isot painikkeet tehtäisiin. Tämän painikkeen olisi keskellä kuva ja sen alapuolella taas teksti. Reunoille ja yläosaan haluttiin liukuvärit ja tämän halusin lisätä tähän vielä varjostuksen.

Tätä varten taas tein oman StackPanelin, jonka sisällä taas on reunukset. StackPanelin sisällä taas on painikkeet. Painikkeilla on reunus, jotta painike saadaan pyöristettyä. Reunus taas kostuu hyperlinkkipainikkeesta, jonka sisällä taas on StackPanel. Tähän StackPaneliin lopulta laitetaan kuva ja kuvateksti.

Tavallista painiketta (Button) en halunnut käyttää, sillä siinä on oletusmuotoilu. Sen muokkaaminen on puolestaan hankalaa. Siksi käytin ennemmin hyperlinkkipainiketta (HyperlinkButton). (Neel, haettu 1.11.2011)

Huomioitavaa oli, että liukuvärin saa vain sivu tai pystysuunnassa. Päädyin laittamaan tässä tapauksessa reunuksen vasempaan ja oikeaan reunaan liukuvärin. Yläosaan yltävään hyperlinkkipainikkeeseen saa myös liukuvärin. Siksi laitoin sille liukuvärin, jossa yläosassa on väritys ja alaosa taas on läpinäkyvä.

Kuva 6. Painikkeet käyttöliittymän uudistamisen jälkeen.

Luonnollisesti näiden muutosten jälkeen koodi on pitkä. Ongelmaa voidaan lievittää selkeästi määrittelemällä tyylit tyylitiedostossa. Omassa koodissani olen määritellyt tyylitiedostoissa hyperlinkkipainikkeelle ja reunuksille oman tyylin, jonka taas olen liittänyt ulkonäköön vaikuttavaan XAML:iin. Tämän jälkeen koodi näyttää melko lyhyeltä ja samalla selkeältä.

Nappuloissa tuli kuitenkin ongelmaksi se, että sen tilasta ei saanut selvää. Toisin sanoen käyttäjä ei tiennyt, kykenikö painiketta painamaan vai ei. Onneksi tyylitiedostoihin sai linkstaten avulla oman määreensä silloin, kun linkki ei ole käytössä. Tätä varten tarvittiin template, jossa oli paljon koodia ja lukuisia eri arvoja, joista jokainen vaikutti johonkin. Koodin määrään nähden onkin huvittavaa, että käytännössä suurin muutos oli yhdessä kohdassa läpinäkyvyys arvon muokkaaminen arvoon 0.6.

5.3.5 Välilehdet

Päätimme tehdä alasivujen navigoinnin välilehdillä. Tähän oli muutama hyvä syy. Ne nimittäin ovat selviä kaikille käyttäjille, ne näyttävät hyviltä, niitä on vaikea olla huomaamatta ja ennen kaikkea ne luovat illuusion siitä, että aktiivinen välilehti siirtyy etummaiseksi. Välilehdistä syntyy tavallista parempi vaikutelma siitä, että sivusto jakautuu osiin ja olet nyt yhdessä niistä. Välilehti pitää vain tehdä kunnolla, eli sen tulee olla yhteydessä alatilaan. Toisin sanoen välilehti tuli tehdä samalla värillä aktiivisena olevan alatilan kanssa ja yhden välilehden piti olla oletuksena auki. (Krug 2006, 80-84)

Sovelluksessa navigointi oli toteutettu napeilla, jotka olivat omassa pienessä UserControllissa. Käytännössä välilehtien teko vaati nappien tyylimäärittelyn muuttamista. Tyyli valitulle painikkeelle piti olla esimerkiksi taustaväriltään sama, kuin siihen

liittyvän alueen tausta. Lisäksi kulmien pyöristystä ei voinut enää alareunaan laittaa. Lisäksi painikkeet sisältävää sivua jouduttiin laskemaan ja alareunan marginaaliksi jouduin laittamaan negatiivinen arvo, eikä itse sivuilla siinä kohdassa, johon on määritelty painikkeen sisältävän sivun paikka saanut laittaa mitään marginaalia.

Valittu välilehti on hyvä korostaa. Siksi tein korostukselle oman tyylinsä, jossa alkuperäisen reunuksen tummempi sävy muuttuu selkeästi vaaleammaksi. Tämä on taas tyylitiedostossa omana tyylinään. Tyylin sain vaihtumaan tekemällä oman Border_GotFocus-metodin, joka muuttaa valitun reunuksen tyylin ja toisaalta taas laittaa oletustyylin muille yläosan reunuksille.

Taustalle kuitenkin vaaditaan pari staattista arvoa, joka varmistaa sen, ettei eri sivulla käymällä tule väärää korostusta. Tätä tehdessäni huomasin, että vanha reunus tulee pitää staattisena. Samoin tein boolin, joka aktivoitui muutoksien yhteydessä. Tällöin mahdollistuu se, että sivujen latautuessa tulee jokin tietty sivu esille, mutta muutoin se ei ota tuota oletusarvoa, vaan hakee staattisesta arvosta oikean viimeksi käytetyn reunuksen.

Suurin ongelma tuli tyylimäärittelystä. Nappuloiden tausta oli helppo vaihtaa valinnan ohella, sillä nappuloiden ulkonäön suurimmalta osalta määritteli ylimmän tason objektin tausta, eli tässä tapauksessa reunuksen tausta. Mutta tekstin väriin se ei vaikuttanut, eikä staattinen tekstiväri yksinkertaisesti näyttänyt hyvältä eri värisessä taustoissa. Mietin pitkään ratkaisua, mutta lopulta keksin etsiä aina vain alempia lapsia, kunnes teksti löytyy ja vaihtaa sen lapsen tyyliä. Näin koodi oli lopulta muodossa:

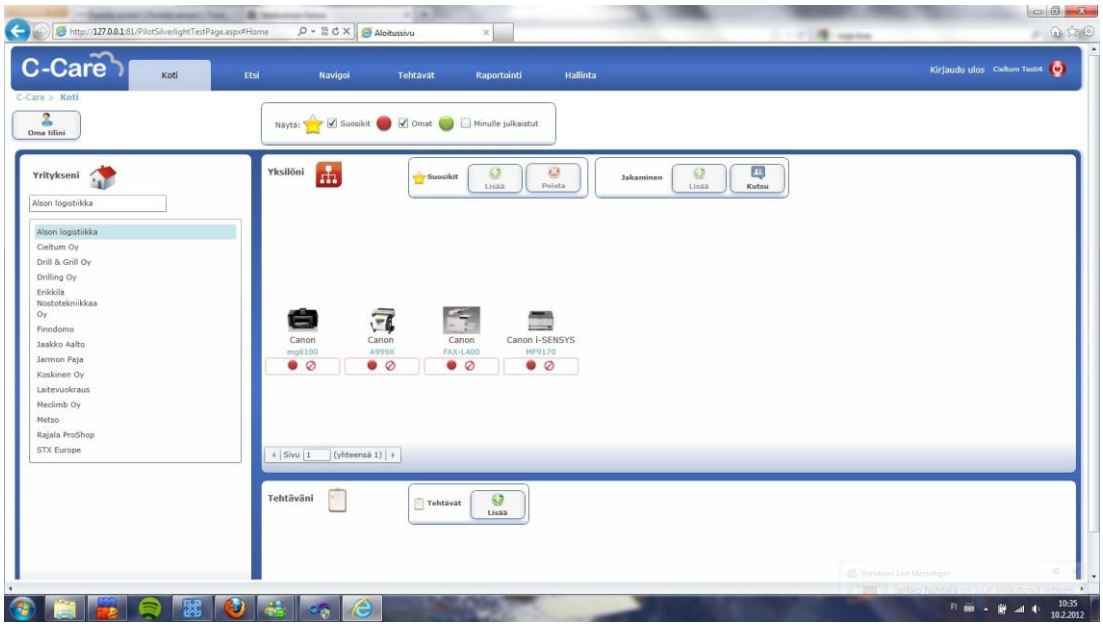
```
 HyperlinkButton hb = (HyperlinkButton)border.Child;
StackPanel s = (StackPanel)hb.Content;
foreach (var t in s.Children.OfType<TextBlock>().Select(child => child))
 {
t.Style = (Style)Application.Current.Resources["navigationSelectedTextStyle"];
  }
```

5.3.6 Etusivu

Sovelluksen etusivulla oli oma toimintalogiikkansa. Jotta näki omat kohteet, piti ensiksi valita yritys, jonka kohteet sovellus esitti. Kohdetta napsauttamalla taas sovellus esitti käyttäjälle asetetut tehtävät, jotka liittyivät kohteeseen. Etusivulla oli kolme isoa aluetta allekkain, joista ylimmässä olivat yritykset, keskimmäisessä kohteet ja alimmaisessa tehtävät. Lisäksi näiden yläpuolella oli erilaisia nappuloita eri tarkoituksia varten.

Ongelma tuli siinä, että muilla sivuilla kaikki navigointi tapahtui vasemman puoleisissa laatikoissa, minkä vuoksi yrityksen napsauttaminen etusivulla ei ollut luonnollista. Toisaalta etusivulla tapahtuva kohteiden jako, suosikeiksi laittaminen ja muiden ihmisten kutsuminen olivat erikoisessa paikassa. Ne eivät olleet kohteiden vieressä, mikä olisi ollut loogista. Sen sijaan ne olivat ylhäällä, josta niitä oli vaikeaa löytää.

Siksi muutin etusivun muistuttamaan enemmän muuta sovellusta. Koska etusivulla navigointi tapahtuu yrityksen kautta, siirsin yrityksen valinnan vasempaan reunaan ja tein yrityksistä selvemmin listamaisen. Oikealle puolelle suuremmalle alueelle sitten jäi enemmän tilaa kohteille ja tehtäville. Samalla siirsin yläpalkista kutsumisen, suosikkien lisäämisen ja poistamisen sekä jakamisen kohteiden kanssa saman reunuksen sisälle (Kuva 7). Tällöin käyttäjälle tulee voimakkaammin tunne, että ne painikkeet liittyvät juuri kohteisiin, eivätkä koko näytettävään sivuun.



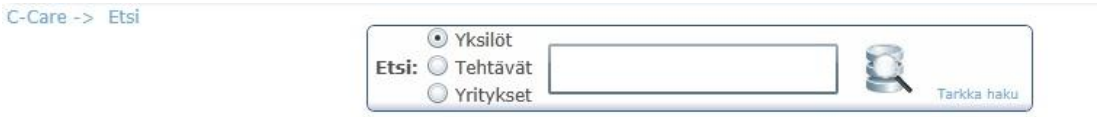
Kuva 7. Uusittu etusivu

5.3.7 Sivujen toimintojen selkeyttäminen

5.3.7.1 Hakusivun muokkaaminen

Sovelluksen hakusivu oli alkuperäisessä muodossaan hivenen sekava ja oli hyvä esimerkki sivun selkeyttämisen tarpeesta. Sivulla oli kolme painiketta, jotka esittivät tehtäviä, dokumentteja ja objekteja. Sen sijaan mitään suoraa hakukenttää tai painiketta ei sivulla ollut. Syystä tai toisesta hakukentät tulivat esille vasta painallusten jälkeen ja tuolloin olikin jo laaja haku käynnissä. Tämä ei ole yksinkertaisin ratkaisu. Toisaalta suuri ongelma tuli myös siinä, että sivu piti ladata kokonaan uudestaan, mikäli valitsi jonkin painikkeen. Tehtävähausta ei ollut esimerkiksi ollenkaan mahdollisuutta siirtyä objektihakuun.

Tätä varten ajatuksena oli tehdä hakusivulle ensimmäiseksi yksinkertainen haku. Tähän kuului hakulaatikko ja radiopainikkeet, joista pystyi valitsemaan sen, että haetaanko objektia, tehtäviä vai dokumentteja. Oletusvalinnaksi määrittelin objektit, sillä sovellus perustuu pitkälti tuotteisiin ja niiden hallintaan. Näin hakusivulle tultaessa huomaa heti, että tämä sivu on tarkoitettu hakuihin.



Kuva 8. Yksinkertainen haku

Tämän lisäksi laitoin sivulle kohdan ”tarkka haku”. Kyseinen kohta on tehty linkkivärillä ja sitä painamalla avautuu yksityiskohtaisemmat haut. Tässäkin paikassa laitoin oletukseksi objektit. Nämä hakupalkit olivat jo olemassa ja ainoa asia mitä niille tein, oli niiden siirtäminen keskemmälle. Vasemmalle reunalle loin tehtävien, objektien ja dokumenttien valintaa varten oma paikkansa. Valinnan korostin harmaalla värillä.

Tämä taas vaati selkeitä muutoksia koodiin. Sivu oli rakennettu visibility-arvojen taakse, jotka taas vastasivat näkyvyydestä. Nämä taas olivat taustakoodissa boolien kanssa. Näin esimerkiksi tehtävät oli liitetty datatiedostoon, ja sitä taas vastasi oma bool-arvo, joka oli oletuksena epätosi. Mutta kun tehtävähaku oli valittu, niin datassa tehtäviin liittyvä bool-arvo muuttui todeksi. Näin tehtävät tulivat sivulla näkyviin. Tässä oli vain se huono puoli, että alun perin tehtävähaun yhteydessä esille tuli sekä yläpalkki että tulokset. Yläpalkkia ei kuitenkaan saanut tulla esille, mikäli tehtiin yksinkertainen haku. Yläpalkeissa oleville osille piti näin ollen tehdä omat bool-arvot ja liittää ne yläosassa oleviin valintanappuloihin, jottei nämä häiritsisi yksinkertaisia hakuja.

Toisaalta moneen hakuun liittyi myös tiettyjen laatikoiden arvot, joita datatiedosto sitten käsitteli. Siksi näihin vanhoihin hakumetodeihin tuli asettaa mahdollisuus sille, että haku onkin tehty perushaun kautta ja että se katsoisi kyseisen hakulaatikon arvon. Samalla piti huomioida, että näissä tapauksissa tietokantahaut eivät etsisi yksinkertaisen haun lisäksi muita arvoja. Tätä varten täytyi koodiin asettaa muutama yksinkertaisiin hakuihin liittyvä if-lause, joissa tutkittiin, että onko jokin tietty bool tosi vai ei.

5.3.7.2 Pakolliset kentät

Testaamisessa tuli ilmi, etteivät käyttäjät tiedä mihin kenttään on pakko laittaa tekstiä. Joissain tapauksissa pyritään menemään helpointa mahdollisinta reittiä, jolloin tietoa ei aina ole riittävästi sovellukseen tallentamiseksi. Tietokanta vaatii aina joitain arvoja ja ne on pakko täyttää, jotta tallennus onnistuu.. Tämä aiheuttaa selvää sekavuutta, joten pakolliset kentät on hyvä laittaa. Varsinkin, kun sovelluksessa ei varsinaisesti ole mitään syytä olla merkkaamatta pakollisia kenttiä.

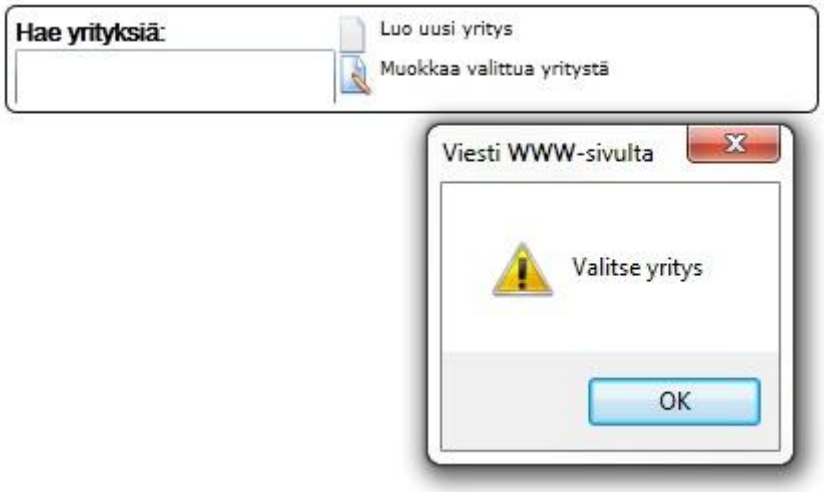
Ratkaisu oli yksinkertainen. Laitoin pakollisten kenttien kuvaustekstien perään tähden. Tämä ratkaisu on varsin yleinen eri web-sivustoissa, joten siksi se ei aiheuta suurta päänvaivaa käyttäjissä. Korostaakseni tähteä asetin sen vielä tumman punaiseksi. Täytettävien kenttien yhteyteen laitoin myös ilmoituksen, että tähti merkkaa pakollista kenttää. Yleensä laitoin kuvaustekstin TextBlockiin, joka taas oli suoraan määritelty johonkin tiettyyn Gridin riviin ja palstaan kiinni. Tähti ei tällaisissa tilanteissa mennyt tekstin perään. Siksi pakollisten kenttien kohdalla piti tekstit laittaa StackPaneliin. StackPanel taas kiinnitettiin Gridiin ja sen sisällä olevat elementit taas määritin asettumaan vaakasuuntaisesti. Näin tähdet tulivat automaattisesti tekstien jälkeen.

5.3.8 Virheilmoitukset

Käytin ajoittain aikaani sivujen testaamiseen ja tein aina välillä tahallaan asioita väärin. Samalla huomasin, että sovelluksessa oli melko hyvin virheilmoituksia sekä muitakin ilmoituksia. Muun muassa tallennuksen onnistuminen ilmoitettiin poikkeuksetta. Aina virheen sattuessa virheilmoitusta ei kuitenkaan tullut.

C-Care käyttää virheilmoituksissaan HtmlPagen luomaa ikkunaa. Tähän päädyttiin siksi, että yleisesti käytetty MessageBox-ikkuna ei jostain syystä toimi kunnolla kaikilla selaimilla. Siksi jo aiemmin päädyttiin käyttämään HtmlPagea ja sen Alert-metodin tuomaa ikkunaa. Alertin jälkeen voidaan määritellä kielitiedostosta tilanteelle sopiva teksti, joka toimii virheilmoituksena. Virheilmoitusta varten tein monia uusia rivejä kielitiedostoon, kuten rivin TaskPgSelectObjectText. Se taas tuo ilmoituk-

seen lauseet "Select object" tai "Valitse kohde" riippuen käyttöjärjestelmän käyttämästä kielestä.



Kuva 9. Virheilmoitus yrityshaussa.

Lisäksi ajoittain sovelluksessa löytyi tilanne, ettei nappuloita painamalla tapahtunut mitään. Näissä tilanteissa syynä oli se, että jotain piti olla jo valittuna. Käyttäjän on kuitenkin mahdotonta tietää, että miksi mitään ei tapahdu, vaikka painiketta painetaankin. Painikkeilla on usein joko koodin kautta määriteltyjä arvoja, tai sitten click-tapahtuma. Näille voidaan helposti tehdä if-lauseita, jotka tarkistavat jonkin tietyn asian valinnan. Jos vaikkapa tehtäväsivulla yritys on valittu, niin sovellus valitsee yrityksen. Jos taas ei, niin if-lauseen ansiosta sovellus ilmoittaa, että yritystä ei ole valittu (kuva 9).

5.3.9 Sijainnin parempi ilmoittaminen

Asiakkailta oli tullut kommentteja, että heidän on vaikeaa hahmottaa heidän sijaintinsa C-Caressa. Alkuperäisessä tilanteessa oli jo korostettu pääsivun yläosassa valittu alasivu. Ratkaisu ei kuitenkaan ollut vielä tarpeeksi selkeä, eikä se kertonut, että millä alasivuilla kulloinkin ollaan. Siksi ajattelin tehdä ns. murupolun. Eli laitoin sivun vasempaan yläkulmaan valikkorivin alapuolelle StackPanelin, joka kertoo missä milloinkin ollaan. StackPanelin sisälle laitoin HyperlinkButtonin ja sen sisällöksi taas TextBlockin, jonka vuoksi HyperlinkButton näkyi tekstinä. Tekstit taas sidoin

kielitiedostoista löytyviin teksteihin. Viimeisenä olevan HyperlinkButtonin puolestaan korostin selkeyttääkseni valitun sivun hahmottamista. Nämä taas sijoitin Gridin mukaisille riveille ja kolumneille, jotta paikka olisi kullakin hikijäljellä sama. Tein samankaltaisen xaml-tiedoston jokaiselle sivulle, jotka käyttivät oikeaa yläkulmaa. Yksinkertaisimmillaan tämä XAML näytti tältä:

```
<StackPanel Grid.Column="1" Grid.Row="0" Grid.RowSpan="2" Margin="0,-10,0,0" Style="{StaticResource NavPathStackPanelStyle}">
    <HyperlinkButton NavigateUri="/Views/UserPage.xaml" >
        <HyperlinkButton.Content>
            <TextBlock x:Name="naviLink" Text="C-Care -> " Style="{StaticResource NavPathLinkStyle}"></TextBlock>
        </HyperlinkButton.Content>
    </HyperlinkButton>
    <HyperlinkButton NavigateUri="/Views/UserPage.xaml">
        <HyperlinkButton.Content>
            <TextBlock x:Name="naviLink2" Text="{Binding Source={StaticResource LocalLang}, Path=HomeLinkText}" Style="{StaticResource NavPathLinkStyle}" ></TextBlock>
        </HyperlinkButton.Content>
    </HyperlinkButton>
</StackPanel>
```

HypelinkButtoniin päädyin sen vuoksi, että on käytännöllistä päästä aiemmille tasoille murupolkua painamalla. TextBlockilla tähän ei kykene ilman komennon tekemistä. Se taas on tuntuvasti vaikeampaa, kuin käyttää HyperlinkButtonin NavigateUri-toimintoa. NavigateUriin kun voi suoraan määrittää kyseinen painikkeen kohteen. Näin pääsin suurimmaksi osaksi pienellä vaivalla. Vain Navigointisivu ja yrityksen muokkaamiseen tarkoitettu sivu aiheuttivat suurempaa vaivaa, sillä ne tarvitsivat muita tietoja. Navigointisivu tarvitsee objektin ID:n toimiakseen, kun taas yritystä muokatessa tarvitaan yrityksen tunnus. Tätä varten tein kyseisten sivujen luokkiin NavigateUri-nimisen Uri-luokan. Sitä kautta pystyin normaalin Urin jälkeen laittamaan tietokannasta haetun ID:n. Tämän jälkeen tieto säilyi, vaikka murupolussa painoikin uudestaan vaikkapa kohtaa "navigointi".

Yläpalkin taustaväri oli aiemmin ollut sinertävän sävyinen ja se näytti jo alun perin hyvältä. Sitä kuitenkin kykeni parantamaan vielä muutamalla pienellä muutoksella. Taustaväri päätettiin pitää sinertävänä, mutta siihen tehtiin LinearGradientBrush-efekti. Toisin sanoen taustalle laitettiin liukuväri. Tämä väritys lähti tummemman sinisestä muuttuen pikkuhiljaa hivenen vaaleammaksi aina puoliväliin asti, jonka jälkeen väritys muuttui taas takaisin samaan värisävyyn yläosan kanssa. Myös reunus sai korjausta. Aiemmin reunus oli vain musta, mutta nyt se muutettiin tumman siniseksi. Näin yläpalkin koodiksi tuli seuraava:

```
<Border Style="{StaticResource ViewBorderStyle}" Margin="10,0" BorderBrush="#FF365799">
<Controls1:WrapPanel x:Name="GridPanel" ItemWidth="120" ItemHeight="65" Margin="0">

<Controls1:WrapPanel.Background>
<LinearGradientBrush EndPoint="0.5,1" StartPoint="0.5,0">
<GradientStop Color="#FF365799"/>
<GradientStop Color="#FF4773CC" Offset="0.5"/>
<GradientStop Color="#FF365799" Offset="1.0"/>
</LinearGradientBrush>
</Controls1:WrapPanel.Background>
```

Toisaalta yläpalkin korostus ei aiemmin ollut tarpeeksi tehokas. Yläosa muuttui hivenen vaaleammaksi, mutta ei tarpeeksi. Näin ollen taustan oli hyvä olla aiempaa vaaleampi ja tekstin värin tulisi muuttua. Ja jottei korostus olisi viimeistelemättömän oloinen, korostukseen haluttiin yläkulmaan pyöristys. Alaosalla ei ollut samaa tarvetta, sillä vaalean korostuksen alaosa osuu valkoiseen taustaan.

Aiemmassa toteutuksessa korostus oli tehty Gridin taustaväriä muuttamalla. Värimaailman muuttaminen ei olisi ollut tässäkään tapauksessa vaikeaa, mutta Gridiin ei saa pyöristystä. Tämän ongelman ratkaisin laittamalla Gridin sisälle reunuksen, jossa voidaan määritellä CornerRadius-niminen arvo, jolla taas saadaan pyöristys aikaan. Tämän reunuksen sisälle taas laitoin sivunavigointiin tarkoitetun TextBlockin. Reu-

nuksen CornerRadius-arvoksi laitoin 5,5,0,0, jolla saatiin vain yläkulmiin pyöristykset.

Määrittelin reunukselle ja TextBlockille omat arvot taustakoodissa. Nämä taas muuttuivat valitun sivun mukaan. Eli aina sivun vaihtuessa yläpalkin arvo vaihtui myös ja sen mukaan tehtiin korostus sivua vastaavan linkin ympärille. Yläpalkissa taas Background-arvo määritteli taustan värin ja Foreground-arvo taas määritteli fontin värin. Taustakoodi myös samalla poisti niiden sivujen Background- ja Foreground-arvot, joita ei ole valittu.

```
border.Background = new SolidColorBrush(Color.FromArgb(180, 255, 255, 255));
block.Foreground = new SolidColorBrush(Colors.Black);
```

Kuva 10. Yläosan sijainnin ilmoittavat korostukset.

Testauksessa kävi ilmi, ettei murupolun ja yläpalkin korostamisenkaan jälkeen käyttäjällä ole aina selvää, että mitä sivua käytetään. Sivun voi toki päätellä, mutta se on aikaa vievää. Tämä ongelma oli erityisesti raportointisivuilla ja tehtävien sekä yrityksen muokkaamisessa. Ratkaisuna oli laittaa pääasiallisen sisällön yläpuolelle isolla ja lihavoidulla fontilla sivun nimi. Tekstin perään laitoin vielä sopivan kuvan tuomaan visuaalista ilmettä. Tällä tavalla ohjelman käyttämät kuvat tulevat aiempaa paremmin tutuiksi käyttäjille.

Sivujen yläosaa muokkasin vielä siten, että C-Caren logon c-kirjain on kutakuinkin saman matkan päässä ylä- ja alareunasta sekä vasemmasta sivusta. Saman tein myös uloskirjautumiseen tarkoitetulle painikkeelle.

5.3.10 Yleinen käytettävyyden parantaminen

Alussa C-Caren etusivulla oli toiminnallisuus, josta asiakkaat valittivat. Samainen ominaisuus oli aiheuttanut myös minulle useita huonoja muistikuvia. Todennäköisesti syynä oli se, että toiminnallisuutta oli pikkuhiljaa muutettu, jonka vuoksi vanha, muttei enää toimivat toiminnallisuus oli jäänyt sovellukseen.

C-Caren etusivulla on kolme reunusta, jonka sisälle tulee tietoja. Nämä reunukset näyttävät omat yritykset, suosikit ja yksilöt. Mutta aluksi sivulla näkyy pelkästään yrityksiä. Vasta yrityksen valinnan jälkeen sovellus näyttää suosikit ja yksilöt.

Alussa sovelluksessa näkyi kaikki kerralla, mutta tällä pyrittiin vähentämään etusivulle tuotavan tavaran määrää. Kuitenkin alkuperäisessä suunnitelmassa yrityksen tietoihin tuli päästä yrityksen nimeä napsauttamalla. Tätä varten tarvittiin hyperlinkki. Mutta luonnollisesti tilanne oli se, että yrityksen valinnassa usein tuli painettua yrityksen nimeä. Yrityksen nimestä painamalla taas joutui yrityksen tietoihin, joka ei ollut useimmiten toivottua.

Siksi poistin hyperlinkin yrityksen nimestä. Sille ei ollut niin suurta tilausta, että sitä täytyisi pitää etusivulla häiritsemässä. Yrityksen tietoihin oli kuitenkin tapa päästä niin yleiseltä hakusivulta, kuin tehtävien haun kautta. Linkin poiston jälkeen etusivulta ei joutunut palaamaan enää turhaan yrityksen sivuille ja näin ollen käytettävyys ja käyttökokemus paranivat.

5.4 Yhteneväisyyden lisääminen

5.4.1 Fonttien käyttö

Web-sivujen fonttina on hyvä käyttää päätteetöntä fonttia. Tällainen on esimerkiksi Arial, jota sovelluksessa käytettiin. Kuitenkin parempia ja vähemmän käytettyjä vaihtoehtoja on olemassa. Tällainen on esimerkiksi Helvetica Regular, joka korvasi Arialin. Se tuntuu toimivan hyvin lyhyissä teksteissä, sillä se erottuu selvästi taustasta sekä toimii kaikilla selaimilla.

Fonttivärin puolestaan tulisi olla tumma. En kuitenkaan halunnut käyttää aivan mustaa tekstiä, sillä se ei ole omaperäinen eikä erityisen tyylikäs. Siksi muutin mustan hivenen harmaampaan suuntaan. Nyt fonttiväri on #FF555151, joka hivenen hopeanvivahteinen.

Hyvänä puolena fonttien muokkauksessa on se, että suurin osa fonteista ja fonttiväreistä on määritelty tyylitiedostoissa. Loput löytävät taas helposti hakutoiminnolla. Jos Visual Studiolla hakee tekstiä ” #FF555151”, voi olla varma, että kyseessä on fontti. Jouduin kerran toteutuksen aikana muuttamaan kaikkien fonttien väriä. Muokkauksessa kesti noin vartti, sillä haun ja tyylitiedoston yhdistelmällä kaikki käyttötarkoitukset löytyi nopeasti. Loppu olikin arvojen korvaamista tyylitiedostossa ja XAML:ssa.

5.4.2 Otsikot

C-Caren suuria ongelmia oli muuttuva käytäntö otsikoissa. Otsikoiden fonttikoko oli välillä 12, välillä 16 ja välillä 18. Toisaalta välillä otsikot eivät edes näyttäneet otsikoilta. Näin oli esimerkiksi navigointisivulla ja raportoinnissa. Sitten taas joillain sivuilla oli kaikkien muiden sivujen vastaisesti suuraakkosia. Tämän vuoksi tarvittiin jonkinlainen yhteinen tapa, jolla otsikointi tehdään.

Luin hieman teoriaa, minkä mukaan otsikon tulisi olla iso ja kaiken yläpuolella. Lisäksi jokaisella sivulla pitää olla nimi ja sen piti vastata valittua sivua. Luonnollisesti otsikon tulee myös erottua selkeästi. Myös tehokeinoja, kuten lihavointia, poikkeavaa väritystä tai leveämpää ylätilaa voi käyttää. (Krug 2006, 31, 71-73).

Lopulta oma ratkaisu (kuva 11) oli laittaa sivun yläosaan lihavoidulla fontilla sivun nimi. Tällä oli myös hivenen muuta tekstiä isompi fontti. Tekstin perässä on vielä kuva, sillä niitä käytetään sovelluksessa paljon. Lisäksi ne toimivat hyvänä visuaalisena lisänä.

Kuva 11. Otsikko ja sivun sisältöä.

5.4.3 Sivujen muokkaus samankaltaiseksi

Ohjelmassa oli alkuperäisessä tilanteessa lukemattomia sivuja, joiden koot eivät olleet samankaltaisia kuin muualla sovelluksessa. Lähes kaikilla sivuilla navigointi tapahtui järjestelmän vasemmassa reunassa ja keskeltä oikeaan reunaan jatkuvalla alueella oli varsinainen sisältö. Monella sivulla oli myös samankaltaisia toteutuksia, joilla haettiin vaikkapa DataGridiin tietoja tietokannoista.

Sivuissa oli alun perin jo samankaltainen rakenne, mutta leveydet ja paikat vaihtelivat suuresti. Kaikki sivut oli toteutettu Grideillä, joihin kykeni sitomaan vaikkapa reunusten paikat. Yritin aluksi sijoittaa Gridien arvoja tyylitiedostoihin, mutta nopeasti havaitsin rivien ja palstojen määrittelyn sitä kautta mahdottomaksi.

Siksi jouduin määrittelemään yksitellen samoja arvoja eri sivuille, jotta sivut olisivat yhtä leveitä ja reunukset olisivat samassa paikassa. Kaikeksi onneksi rakenteen samankaltaisuus sivulta toiselle oli sen verran hyvässä kunnossa, että suurin osa työstä oli yhden sivun muokkaamisen jälkeen samojen arvojen kopioimista toisille sivuille. Sen jälkeen tarvitsi enää tehdä pieniä muokkauksia.

Reunuksien ja marginaalien määrittelyssä sain tehtyä loputkin osat sivuista samankaltaisiksi. Tämä työ onnistui tyylitiedoston kautta. Tyylitiedostossa kykenee tekemään erinimisiä tyylejä, nimeämään ne ja määritellä se, että millaisiin elementteihin tyyli vaikuttaa. Valmis tyyli puolestaan sidotaan XAML:iin elementin Style-kohtaan. Tämän jälkeen tarvitsikin vain valita resurssi ja etsiä paikallisesta hakemistosta sopiva tyyli. Tein samalla havainnon, että Silverlight ehdottaa vain sellaisia tyylejä, mitkä sopivat juuri käsiteltävälle elementille. Eli näin esimerkiksi reunukselle ehdote-

taan reunuksille suunnattua tyyliä, mutta ei kuvien tyylejä. Sidonnan jälkeen XAML:ssa elementin tyyli näyttää vaikkapa seuraavalta:
Style="{Binding Source={StaticResource ViewBorderStyle}}"

Pyrin myös tekemään kaikki marginaalit ja sitä kautta välit samankokoisiksi. Marginaaleiksi laitoin viisi mahdollisimman moneen paikkaan. Se tuntui olevan sopivan oloinen ero reunoihin. Muissa kohdissa pyrin myös käyttämään viiden pikselin väliä tai niiden kerrannaisia.

C-Caressa oli muutamalla sivulla alun perin kolme erillistä reunusta. Se aiheutti sen, että keskellä oli liian vähän tilaa itse pääsisällölle. Näin päädyin ratkaisuun, jossa olisi navigointi ja siihen suoraan liittyvät asiat vasemmalla ja itse pääsisältö keskellä. Tähän päästäkseni tein muutamalla sivulla muutoksia. Tehtävän ja yrityksen muokkaukseen tarkoitetuilla sivuilla oli vasemmalla puolella paljon tyhjää tilaa, mutta oikealla puolella oli paljon lisävalintoja. Tällä sivulla tarvitsi vain muuttaa elementtien paikkaa Gridissä, jolloin oikea puoli siirtyi vasempaan reunaan. Samalla kykenin venyttämään pääsisällön keskeltä oikeaan reunaan asti mahdollisuuksien mukaan. Navigointisivulla puolestaan oli alun perin kuva keskellä ja oikeassa reunassa kapeat kaistaleet dokumentteja, tehtäviä ja ominaisuuksia varten. Siirsin nämä kaikki keskelle, otin oikean puoleisen reunuksen pois, laitoin kuvan vain yhdeksi keskellä näytettävistä vaihtoehdoista ja levensin keskellä olevan reunuksen jatkumaan oikeaan reunaan saakka.

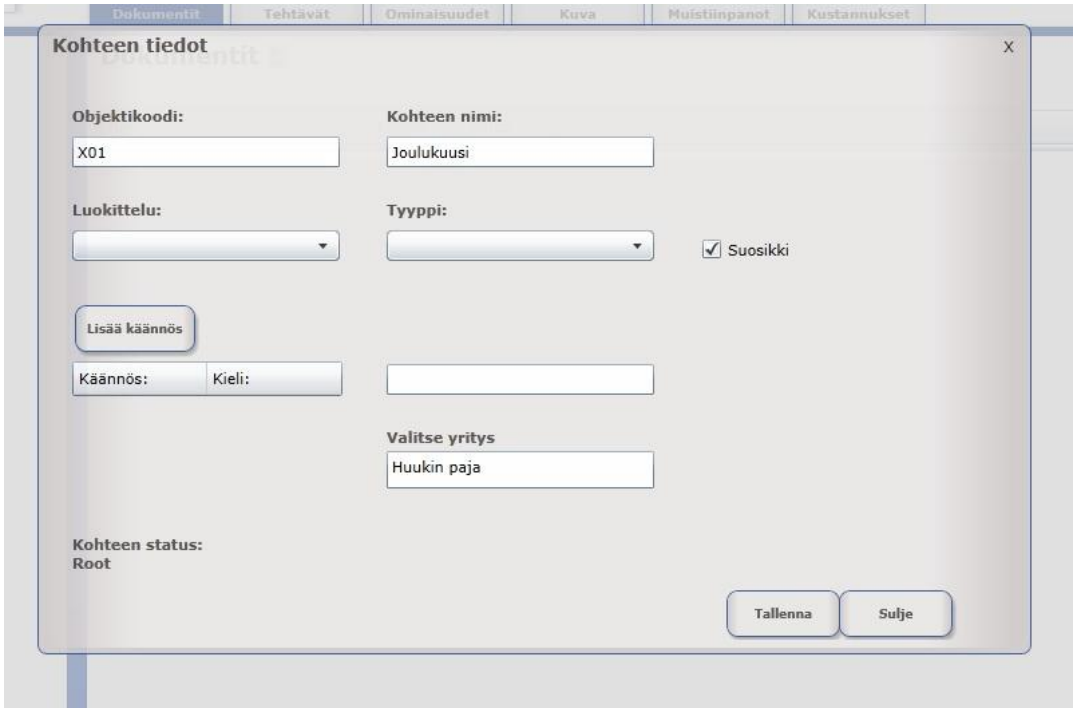
5.4.4 Lapsi-ikkunat

Monessa tilanteessa ei ole tarkoituksenmukaista ladata kokonaan uutta sivua. Tällaisia tilanteita ovat ne, joita ei saa laitettua jollekin sivulle enää valmiiksi, mutta joille ei kannata kokonaan täysimittaista sivua tehdä. Tällaisia tilanteita on esimerkiksi kuvan tai dokumentin lisääminen sovellukseen. Silloin käytetään lapsi-ikkunoita, jotka löytyivät Microsoftin omista kontrolleista.

Lapsi-ikkunat olivat kuitenkin vaiheessa, sillä niissä käytettiin oletusarvoja. Esimerkiksi taustassa ja otsikossa oli pelkät oletusvärit, fontit olivat sitä, mitä sovellus ole-

tuksena käytti ja painikkeetkin olivat alkuperäisessä tilassaan. En halunnut tehdä lapsi-ikkunoita liian häiritseväksi, mutta ne kaipasivat parannuksia värimaailmaan sekä viimeistellympää ilmettä. Ja koska sovellus oli täynnä pyöristyksiä kulmissa, niin mielestäni sellainen kannatti myös lapsi-ikkunoihin tehdä.

Lapsi-ikkunoissa voi normaalin XAML:n puolella muokata joitain asioita. Näitä oli kuitenkin varsin vähän. Oikeastaan ainoat asiat, mitkä sillä puolella pystyi kunnolla tekemään, olivat läpinäkyvyyksien asettaminen (itse lapsi-ikkunassa 0.9/1, taustassa 0.7/1) ja taustan väri, jonka laitoin tumman harmaaksi. Loput piti linkittää tyylitiedoston kautta, jota lapsi-ikkuna kaikeksi onneksi ymmärsi.



Kuva 12. Lapsi-ikkuna

Tyylitiedostossa taas piti määritellä arvoon ”template” muutoksia. Tätä kautta sain muun muassa lapsi-ikkunan reunukset piiloon laittamalla lapsi-ikkunan taustavärin arvon läpinäkyväksi. Itse ikkunaan (kuva 12) laitoin samanlaiset liukuvärit kuin sovelluksen nappuloissa oli. ContentRoot-nimiseen Gridiin laitoin taas reunukset yleisesti sovelluksessa olevan reunuksien värin mukaisesti. Tähän sain myös CornerRadius-arvot, joiden perusteella kulmat pyöristyivät itse ikkunassa. Muutin myös otsikkoa sellaiseksi, että fonttikoko on 14, fontti on sama kuin muualla sovelluksessa

(Helvetica Regular) ja fontti on korostettu. Nimesin tämän tyylin ChildWindowStyleksi, jota sitten käytin kaikissa lapsi-ikkunoissa.

Muina muutoksina lapsi-ikkunoissa määrittelin painikkeet samalla tavalla kuin muualla sovelluksessa. Tähän oli valmiit tyylitiedostot, jolloin vain lähinnä tekstit ja komennot piti vaihtaa kulloisellekin sivulle sopiviksi. Lapsi-ikkunoissa en kuitenkaan käyttänyt ikoneita toisin kuin täysikokoisilla sivuilla. Myös fontteihin tein pieniä muutoksia, jotta fontti ja sen tyyli ovat samat kuin koko muussakin sovelluksessa. Tämä luo sitä tunnetta, että myös lapsi-ikkunat kuuluvat samaan sovellukseen.

5.5 Ylläpidettävyyden parantaminen

Sovelluksessa oli alkutilanteessa lukuisia elementtejä, joissa on samankaltaiset arvot. Nämä arvot olivat kuitenkin syötetty suoraan XAML:iin. Mutta koska sovelluksessa on myös tyylitiedosto, tuollaista ratkaisua ei ole järkevää käyttää. Siksi käytinkin pariin otteeseen paljon aikaa, jossa määrittelin tyylitiedostoon tyylejä, jotka puolestaan sidoin tämän jälkeen XAML:iin.

Tyylitiedoston avulla erityisesti erilaiset isommat Stackpanelit ja reunukset, jotka toistuvat samankaltaisia sivusta toiseen, ovat paljon helpompia määritellä. Usein määrittely voisi muuten vaatia XAML:iin useita riviä koodia, mutta tyylitiedoston avulla riittää pelkästään Style-arvon määrittely.

5.6 Muut toteutettavat asiat

5.6.1 Yläpalkki

Jokaisen sivun yläreunassa on palkki, jossa on C-Caren logo, linkkirivi ja uloskirjautumisen mahdollistava painike. Tätä palkkia ei kuitenkaan tarvitse tehdä uudestaan sivujen latautuessa, sillä se on itse asiassa pääsivu. Sen alapuolella on sitten paljon tilaa, jonne upotetaan muut sivut.

Yläreuna näytti aluksi hieman keskeneräiseltä, sillä kulmia ei ollut pyöristetty. Värimaailma ja reunukset olisivat myös voineet olla parempia. Palkki oli alun pelin pelkästään WrapPanelin sisällä. WrapPaneliin taas ei voi tehdä pyöristystä millään helpolla tavalla. Siksi olikin syytä käyttää helpompaa keinoa, eli laittaa WrapPanelin ympärille reunus. Reunus puolestaan on tyylitiedostossa määritelty ja yksi sen arvo on CornerRadius arvo, jolla puolestaan saa todella helposti pyöristyksen.

Monella sivulla on yrityksen logon tai muun sivuston tunnisteen kohdalle tehty painike. Tämä painike vie takaisin kotisivulle ja se helpottaa navigointia (Krug 2006, 66). Siksi tein C-Caren tunnuksena toimivaan kuvaan MouseLeftButtonDown-metodin, joka toimii, kun hiiren vasen näppäin on painettu kuvan päällä. Tämän metodin ainoa toiminto on siirtyminen etusivulle.

5.6.2 Dokumentit

Testaus paljasti navigointisivun dokumentti-osiossa sen seikan, ettei käyttäjä välttämättä huomannut sivun oikeassa yläreunassa olevaa kuvaketta, josta pääsi tarkastelemaan dokumenttia. Tämän vuoksi päättelin, että dokumentin tarkastelun aloittava painike olisi syytä laittaa heti tietojen perään.

Kunkin valitun tuotteen dokumentit esitettiin datagridissä. Ikävä kyllä datagrid ei tarjonnut mahdollisuutta laittaa komentoa sen sisälle. Ilmeisesti tieto ei kulkenut datagridin lapsista ylemmille tasoille, joista tieto vasta menee dataan. Tätä varten piti löytää kiertotie. Onneksi samaisesta asiasta oli kärsitty muualtakin, joten löysin eräältä keskustelupalstalta viestin (http://forums.silverlight.net/post/601730.aspx), josta löytyi ratkaisu ongelmaan. Tarvittiin käytännössä luokka, joka löytää vanhemman elementin, laukaisimen sekä luokan liittämisen hyperlinkkipainikkeeseen, josta taas löytyi komento. Tämän jälkeen komennot toimivat myös datagridin sisältä.

6 LOPPUSANAT

Käyttöliittymän uudistamisesta oli minulle suurta hyötyä. Osaksi siksi, että jälkeen kykenen paremmin tekemään projektiluontoista työtä. Mutta ennen kaikkea tämän

työn ohella tutustuin käytännön tasolla käyttöliittymiin sekä niiden luomiseen ja muokkaamiseen.

Työn ohella käsitykseni käyttöliittymistä parani muutenkin huomattavasti. Luin paljon teoriaa ja sen perusteella sain paljon tietoa siitä, miksi jokin asia pitää tehdä jollain tietyllä tavalla hyvien perusteluiden kera. Siitä oli suuri hyöty, kun uudistin C-Caren käyttöliittymää. Toisaalta myös siitä oli suunnaton apu, kun oikeasti sai tehdä teorian perusteella tehtyjä muutoksia ihan käytännössä. Erot näkyivät konkreettisesti nenän edessä, minkä jälkeen usein huomasikin kirjoista saadun tiedon olleen ihan oikeassa.

Varmasti jatkossakin tulen käyttämään tämän perusteella tullutta tietoa hyväkseni ja olen jo nyt huomannut katsovani välillä eri sivuja tai ohjelmia käyttöliittymän näkökulmasta. Kaikki sivut eivät miellytä minua syystä tai toisesta ja olen miettinyt, että mikä siinä on pielessä.

C-Caren käyttöliittymän uudistaminen päättyi joulukuussa 2011, mutta sovellus muuttuu aina. Aina tulee uusia vaatimuksia, aina tulee korjattavaa tai sitten sovellukseen halutaan asiakkaan halun perusteella uusi ominaisuus. Siksi myös käyttöliittymä tulee aina muuttumaan. Suurin uudistaminen on nyt kuitenkin päättynyt ja sovellusta tarjotaankin nyt jo asiakkaille.

LÄHTEET

Anatharam Nandini 2008: XAML and Silverlight. Viitattu 13.10.2011. Saatavissa http://www.codeproject.com/KB/silverlight/xaml.aspx.

Betts, D., Densmore, S., Dunn, R., Narumoto, M., Pace, E. & Woloski, M. 2010. Moving Applications to the Cloud on the Microsoft Azure Platform (Patterns & Practices). Microsoft Press

Heikniemi Jouni 2010: Nykyaikaisilla ohjelmointivälineillä Facebookiin. Mikrobitti 7/2010.

Korpela Jukka K., Linjama Tero 2005. Web-suunnittelu. Jyväskylä. Docento Finland.

Krug Steve, P. 2006. Älä pakota minua ajattelemaan!: Tervettä järkeä verkkosuunnitteluun, 2. p.. Jyväskylä. Gummerus.

Microsoft 2011. Viitattu 13.10.2011. Saatavissa http://www.microsoft.com/silverlight/faq/.

MSDN A. Viitattu 13.10.2011. Saatavissa http://msdn.microsoft.com/en-us/library/cc189036(v=vs.95).aspx.

Neel Michael C. Viitattu 1.11.2011. Saatavissa http://www.vinull.com/Post/2009/06/09/silverlight-20-setting-the-background-of-a-button.aspx

Smith Josh, Viitattu 21.11.2011. Saatavissa http://msdn.microsoft.com/en-us/magazine/dd419663.aspx