WWW.JAMK.FI

RIA-SOVELLUSKEHITYS

Case: Asiakashallintajärjestelmä

Joni Mikkonen
Mikko Savolainen

Opinnäytetyö
Maaliskuu 2010

Tietojenkäsittely
Luonnontieteiden ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä(t) MIKKONEN, Joni SAVOLAINEN, Mikko	Julkaisun laji Opinnäytetyö	Päivämäärä 17.03.2010
	Sivumäärä 78	Julkaisun kieli Suomi
	Luottamuksellisuus ( ) saakka	Verkkojulkaisulupa myönnetty (X)
Työn nimi RIA-SOVELLUSKEHITYS Case: Asiakashallintajärjestelmä		
Koulutusohjelma Tietojenkäsittelyn koulutusohjelma		
Työn ohjaaja(t) TUIKKA, Tommi		
Toimeksiantaja(t) PYNNÖNEN, Topi SALMI, Tero		

Tiivistelmä

Opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa web-pohjainen asiakashallintajärjestelmä, parhaiten tarkoitukseen soveltuvaa RIA-teknologiaa hyödyntäen. Järjestelmän tuli olla käyttöjärjestelmäriippumaton ja toimia www-selaimella. Näin ollen tietoihin pääsee käsiksi useimmilta tietokoneilta tai mobiililaitteilta internet-yhteyden yli. Toteutettavan järjestelmän odotetaan korvaavan asiakkaiden vanhat paperille kirjoitetut yhteystiedot ja käyntikortit.

Tutkimusosio jakautui kolmeen vaiheeseen, joista ensimmäisessä tutkittiin yleisesti rikkaita internet-sovelluksia, vertailtiin niiden eri ominaisuuksia sekä selvitettiin mitä annettavaa RIA-teknologioilla on sähköiseen liiketoimintaan. Toisessa vaiheessa tutkittiin yleisesti web-sovellusten käytettävyyttä ja RIA-sovellusten mukanaan tuomia, käytettävyyttä parantavia ominaisuuksia. Tutkimusosion viimeinen osa oli varsinainen asiakashallintajärjestelmä-projekti, jossa käytiin läpi projektin eri vaiheet ensiaskelista, suunnittelusta, toteutuksesta ja testauksesta aina valmiiseen järjestelmään asti.

Työn tuloksista kävi ilmi, että RIA-sovelluksilla on paljonkin annettavaa sähköiseen liiketoimintaan ja sen kehittämiseen. Ne mahdollistavat käyttäjälleen entistä rikkaamman käyttökokemuksen ja tehostavat suuren tietomäärän hallintaa. Lisäksi niitä on nopea kehittää ja helppo jakaa eteenpäin. RIA-sovellukset nostavat myös käytettävyyden uudelle, entistä korkeammalle tasolle, tuomalla työpöytäsovellusten hyvän ja tehokkaan käytettävyyden myös web-sovelluksiin.

Projektiosion tuloksena syntyi onnistunut, toimeksiantajan vaatimukset ja toiveet täyttävä asiakashallintajärjestelmä ja projektin tuottamat dokumentit, joista tärkeimmät liitettiin myös opinnäytetyöhön. Hyvä jatkotutkimus aihe voisi olla RIA-sovellusten soveltuvuus mobiililaitteisiin. Kyseinen aihe on erittäin ajankohtainen nyt ja varsinkin lähitulevaisuudessa.

Avainsanat (asiasanat)
RIA, asiakashallintajärjestelmä, käytettävyys, web-sovellus

Muut tiedot

Author(s)
MIKKONEN, Joni
SAVOLAINEN, Mikko

Type of publication
Bachelor´s Thesis

Date
17032010

Pages
78

Language
Finnish

Confidential

( ) Until

Permission for web publication
(X)

Title
RIA-DEVELOPMENT
Case: Customer management system

Degree Programme
Business Information Systems

Tutor(s)
TUIKKA, Tommi

Assigned by
PYNNÖNEN, Topi
SALMI, Tero

Abstract

The objective of this thesis was to design and create a web-based customer management system with the help of one rich internet application (RIA) technology most suitable for the purpose. The system had to be operating system independent as it was to operate on a web-browser. Thus, the data would be accessible from most computers or mobile devices over an internet connection. The system was expected to replace any old customer related notes or business cards.

The study section of the thesis is divided into three sections. The first one focuses on the study over RIA in general as well as comparing its various features and benefits of RIA in e-business. The second section is a study over web application usability in general and also RIA usability. The last part of the study focuses on the actual customer management system project in which the system was planned, designed, created and finally tested.

The study section revealed that RIA applications have a lot to give for e-business and its development. RIA enables users to have a richer experience with the application and enables more efficient management over large amounts of information. RIAs are also quick to develop and easy to share forward. By using some RIA technology the level of usability can be increased by bringing the good and more efficient usability of desktop application to the web-based application.

The result of the project section was a successful customer management system that met with the requirements set in the requirements analysis and was approved by the customer. In addition, a great number of project related documents were made during the actual project and it was decided to append the most important documents to the thesis. As a good follow-up study based on this thesis could be RIAs suitability for mobile devices. This subject is very current at the moment and especially in the near future.

Keywords
RIA, customer management system, usability, web-application

Miscellaneous

SISÄLTÖ

KUVIOT

TAULUKOT

KÄSITTEISTÖ

Dialogi tarkoittaa työssämme ihmisen ja järjestelmän vuoropuhelua eli käyttöliittymässä esiintyvää tekstiä, joka on kirjoitettu esittävään muotoon.

Flex on Adobe Systems Incorporated-yhtiön kehittämä sovelluskehitysratkaisu RIA-sovellusten luontiin.

HTML (Hyper Text Markup Language) on yleisin web-sivujen luontiin käytetty kuvauskieli.

Intranet on yrityksen sisäinen lähiverkko, joka on eristetty ulkopuolisilta.

Kustannus-hyötyanalyysi on investointien suunnittelua, jossa asioita tarkastellaan laajemmin kuin yksittäisen yrityksen näkökulmasta. Sen avulla määritetään esimerkiksi tietyn projektin yhteiskunnallinen kannattavuus eli se, ylittävätkö projektista saadut hyödyt sen kustannukset.

Md5 (Message-Digest algorithm 5) on viestitiivistealgoritmi. Se tuottaa annetusta merkkijonosta 128-bittisen tiivisteen, joka esitetään 32-merkkisenä heksadesimaalilukuna.

MySQL on avoimeen lähdekoodiin perustuva SQL-tietokannan hallintajärjestelmä.

PHP (Hypertext Preprocessor) on web-palvelinympäristössä yleisesti käytetty dynaamisten web-sivujen luontiin käytetty ohjelmointikieli.

RIA (Rich Internet Application) tarkoittaa internet-sovelluksia, jotka toimivat kuten perinteiset työpöytäsovellukset.

TCP (Transmision Control Protocol) on tietoliikenneprotokolla, jonka avulla tietokoneet kommunikoivat verkossa.

1 JOHDANTO

Tämän opinnäytetyön tarkoituksena on suunnitella ja toteuttaa asiakashallintajärjestelmä, jonka avulla yritys pystyy hoitamaan asiakkuudenhallinnan helposti ja vaivattomasti internetissä. Järjestelmä on käyttöjärjestelmä-riippumaton, koska se toimii www-selaimella. Tästä syystä tietoihin pääsee käsiksi useimmilta tietokoneilta tai mobiililaitteilta internet-yhteyden yli. Hyvällä asiakashallinnalla on suuri merkitys nuoren yrityksen ottaessa ensiaskeliaan yritysmaailmassa.

Opinnäytetyön toimeksiantajan kaltaisilla nuorilla yrityksillä on jo ennen virallista rekisteröintiä runsaasti potentiaalisia ja varsinaisia asiakkaita, kontakteja, sekä muita tärkeitä tallennettavia yhteystietoja. Toimeksiantajan toiveena on teettää web-pohjainen asiakashallintajärjestelmä tulevan yrityksen verkkosivujen yhteyteen. Web-sovellus tulisi korvaamaan vanhat paperille kirjoitetut yhteystiedot ja käyntikortit kätevällä, itse päivitettävällä ja muokattavalla tavalla. Järjestelmä toteutetaan jollakin RIA-teknologialla, mikä takaa toimeksiantajalle nykyaikaisen ja viimeisintä tekniikkaa edustavan asiakashallintajärjestelmän.

RIA-teknologiat tarjoavat tänä päivänä ylivertaiset mahdollisuudet interaktiivisten ja käytettävyydeltään tehokkaiden järjestelmien luontiin. RIA on myös aiheena erittäin ajankohtainen, sillä internet-teknologiat ovat kovaa vauhtia siirtymässä seuraavaan sukupolveen, joita RIA käsitteenä edustaa. Työssä RIA-teknologioita tarkastellaan myös sähköisen liiketoiminnan ja käytettävyyden näkökulmasta, sillä niiden tuoma etu verrattuna vanhemmilla teknologioilla toteutettuihin sovelluksiin on merkittävä.

Ammatillisen kehittymisen kannalta RIA-teknologioilla on jo nyt ja varsinkin tulevaisuudessa suuri painoarvo työmarkkinoilla. Lisäksi projektin suunnittelu, toteutus ja testaus antavat tekijöilleen arvokasta kokemusta niin ohjelmoinnista, tietokannoista kuin ohjelmistokehityksestäkin.

2 TUTKIMUSASETELMA

Tässä luvussa esitellään toimeksiantaja, tutkimuksen tavoitteet ja rajaukset. Lisäksi kuvataan käytettävät tutkimusmenetelmät sekä esitellään tutkimuskysymykset ja opinnäytetyön aikataulu. Yhdessä nämä kaikki muodostavat opinnäytetyön tutkimusasetelman, jota työssä lähdetään purkamaan.

2.1 Toimeksiantaja

Toimeksiantajana on vielä rekisteröimätön jyväskyläläisyritys. Yrityksen tavoitteena on rekisteröityä kevään 2010 aikana. Yrityksen toiminta keskittyy erilaisiin koulutus-, valmennus-, konsultointi- ja myyntipalveluihin. Myyntipalvelut on suunnattu sekä yksityis- että yritysasiakkaille ja lähtökohtana on tarjota asiakkaiden tarpeisiin räätälöityjä palveluita.

2.2 Tavoitteet ja rajaukset

Opinnäytetyön tavoitteena on tutkia, suunnitella ja toteuttaa web-pohjainen asiakashallintajärjestelmä toimeksiantajalta kerättyjen vaatimusten pohjalta. Toteutettavan järjestelmän odotetaan korvaavan asiakkaiden vanhat paperille kirjoitetut yhteystiedot ja käyntikortit. Toimeksiantajalla ei ole aiempaa asiakashallintaa, joten toteutettava järjestelmä luodaan niin sanotusti puhtaalta pöydältä.

Tutkimusvaiheessa käydään läpi järjestelmän toteutustapoja ja selvitetään miten toimeksiantajayrityksen kaltaisen nuoren yrityksen liiketoimintaa voidaan tietotekniikan avulla parhaiten tehostaa. Suunnittelu- ja toteutusvaiheessa järjestelmää tarkastellaan käytettävyyden näkökulmasta ja otetaan huomioon yrityksen eri työntekijöiden erilaiset tarpeet.

Tärkeimmät toiminnalliset vaatimukset järjestelmän puolelta ovat käyttöjärjestelmäriippumattomuus- ja se, että varsinainen asiakashallinta on suojassa kirjautumisen takana, ettei kuka tahansa pääsisi sisälle järjestelmään. Lisäksi asiakastietoja tulee pystyä lisäämään, muokkaamaan, poistamaan ja tulostamaan. Vähemmän tärkeinä

vaatimuksina ovat mahdollisuus lisätä alemman tason käyttäjiä, sekä muistiinpano työkalu.

2.3 Tutkimusmenetelmät

Tutkimus on kehittämisprojekti, sillä projektiluontoisella hankkeella pyritään luomaan asiakkaalle heidän vaatimustensa mukainen asiakashallintajärjestelmä. Yrityksen nykyistä toimintaa kartoitetaan haastattelemalla työntekijöitä ja luomalla heistä käyttäjäkohtaiset toimintamallit.

Opinnäytetyössä ilmenevät kehitysprojektille tunnusomaiset piirteet, jotka ovat: ongelmalähtöisyys, tavoitteellisuus, osallistuvuus, suunnitelmallisuus, kertaluonteisuus sekä ennalta määritetyt resurssit, joita ovat esimerkiksi projektin aikataulu ja budjetti.

2.4 Tutkimuskysymykset

Tutkimuskysymysten tavoitteena on ilmaista tutkimusongelma kysymyksen muodossa. Tutkimuskysymykset pyritään esittämään mahdollisimman selkeinä ja yksiselitteisinä, sillä ne luovat perustan opinnäytetyön tutkimusosiolle. Työhön laadittiin seuraavat tutkimuskysymykset:

1. Mitä annettavaa RIA-sovelluksilla on sähköiseen liiketoimintaan?

2. Mitä tulee ottaa huomioon web-sovelluksen käytettävyyttä suunniteltaessa?

3. Kuinka kehitysprojekti käytännössä toteutetaan?

    - Mitä suunnittelussa tulee ottaa huomioon?

    - Kuinka sovellus toteutetaan?

    - Kuinka sovelluksen toimintaa testataan?

2.5 Aikataulu

Opinnäytetyö aloitettiin syksyllä 2009 ja tavoitteena oli saada työ valmiiksi keväällä 2010. Aiheen valinta ja varsinkin toimeksiantajan etsiminen vei suurimman osan ajasta työn alkuvaiheessa. Alkuun päästyämme työ edistyi hyvin ja opinnäytetyö sekä varsinainen projekti toteutui ennalta laaditun aikataulun puitteissa.

Aikataulussa eniten aikaa vieneet osiot olivat sovelluksen toteutus eli koodaus sekä tietoperustan rakentaminen, jolloin projektiryhmä perehtyi aiheeseen liittyvään kirjallisuuteen ja tutustui ensimmäisen kerran Adobe Flex -sovelluskehitysympäristöön.

TAULUKKO 1. Aikataulu

Opinnäytetyön aikataulu	Aloitus pvm	Lopetus pvm	Kesto (tuntia)
Aiheen valinta ja aineiston hankinta	**19.10.2009**	**25.11.2009**	**25 h**
Tietoperustan rakentaminen	**20.10.2009**	**31.1.2010**	**175 h**
Käsitteistö	**3.3.2010**	**4.3.2010**	**2 h**
1. Johdanto	**23.1.2010**	**5.3.2010**	**9 h**
2. Tutkimusasetelma	**20.10.2009**	**3.2.2010**	**25 h**
Toimeksiantaja	3.2.2010	3.2.2010	2 h
Tavoitteet ja rajaukset	1.11.2009	15.12.2009	3 h
Tutkimusmenetelmät	15.12.2009	15.12.2009	1 h
Tutkimuskysymykset	31.10.2009	28.1.2010	15 h
Aikataulu	20.10.2009	31.10.2009	4 h
3. RIA- Sovellukset	**4.1.2010**	**28.2.2010**	**35 h**
RIA- teknologiat	4.1.2010	22.2.2010	12 h
Oikean RIA- teknologian valinta	27.1.2010	3.3.2010	5 h
RIA- sovellusten hyödyntäminen sähköisessä liiketoiminnassa	15.2.2010	28.2.2010	10 h
RIA- sovellusten tuomat edut ja ongelmat	15.2.2010	7.3.2010	8 h
4. Web-sovellusten käytettävyys	**4.1.2010**	**28.1.2010**	**19 h**
Käyttäjien toiminnan ja tarpeiden ymmärtäminen	4.1.2010	6.1.2010	8 h
Tuotteen käyttäminen	5.1.2010	17.1.2010	5 h
Heuristinen arviointi	17.1.2010	19.1.2010	2 h
RIA- sovellusten käytettävyys	25.1.2010	28.1.2010	4 h
5. Projektin toteutus	**25.11.2009**	**15.2.2010**	**408 h**
Suunnitteluvaihe	25.11.2009	17.12.2009	70 h
Toteutusvaihe	26.11.2009	8.2.2010	254 h
Testausvaihe	15.1.2010	15.2.2010	49 h
Käyttöönottovaihe	8.2.2010	15.2.2010	25 h
Yhteenveto projektista	15.2.2010	20.2.2010	10 h
6. Tutkimustulosten analysointi	**8.2.2010**	**28.2.2010**	**102 h**
7. Pohdinta	**15.2.2010**	**15.3.2010**	**62 h**
Työn viimeistely	**3.3.2010**	**16.3.2010**	**8 h**
		Yhteensä:	**870 h**

Opinnäytetyön aloitusseminaari pidettiin 20.1.2010 työn ollessa jo hyvässä vauhdissa. Tarkoituksena on palauttaa opinnäytetyö tarkastukseen maaliskuun 2010 alussa ja pitää lopetusseminaari huhtikuun aikana.

3 RIA-SOVELLUKSET

Terminä RIA (Rich Internet Application) syntyi jo maaliskuussa 2002, kun Macromedia keksi käyttää termiä kuvaamaan seuraavan sukupolven internet-sovelluksia. Termillä kuvataan internet-sovelluksia, jotka toimivat kuten perinteiset työpöytäsovellukset. Nykyään nämä sovellukset ovat todellisuutta ja kehittyvät koko ajan. (Tapper, Labriola, Boles & Talbot 2008, xvi.)

Perinteisessä asiakas-palvelin-mallissa toiminta tapahtuu niin, että käyttäjä klikkaa painiketta, joka lataa palvelimelta sivun ja näyttää sen käyttäjälle. Tästä syystä perinteiset sivut ovat hitaita, käytettävyydeltään huonoja, ja jokaisen napin painalluksen jälkeen koko sivu joudutaan lataamaan uudestaan. RIA-sovellus tuo suurimman hyödyn juuri tähän, sillä se ei päivitä koko sivua vaan minimoi siirrettävän datan määrän ja siirtää vain tarpeellisen. RIA-sovellusten tavoite on, että käyttäjällä on mahdollisuus hallita tiedon näyttämistä, ja palvelimen kanssa kommunikointi tapahtuu vain uutta tietoa noudettaessa tai sitä lisättäessä.
(Tapper ym. 2008, 6–7.)

Sittemmin yleistyneet RIA-sovellukset toimivat useimmiten selaimessa, mutta ovat kuitenkin käytettävyydeltään sekä visuaalisuudeltaan hyvin lähellä perinteisiä työpöytäsovelluksia. Selaimessa toimivia RIA-sovelluksia ei tarvitse asentaa, vaan ne ajetaan suoraan selaimessa toimivissa ajoympäristöissä. Tästä johtuen käyttöjärjestelmä ei vaikuta siihen, kuinka sovellus näkyy. Selaimeen saatetaan kuitenkin joutua asentamaan erillinen lisäosa, esimerkiksi Adoben Flash Player, jotta sovellus toimisi. Suomessa kyseisiä sovelluksia voidaan kutsua rikkaiksi internet-sovelluksiksi, mutta vakiintuneempi tapa on puhua niistä RIA-sovelluksina.

Vaikka silloinen Macromedia, nykyinen Adobe, lanseerasikin RIA-termin, nykyisin se on laaja käsite ja kaikkien käytettävissä. Monet eri tahot ovat julkaisseet omat teknologiansa RIA-sovellusten kehittämiseen. Seuraavassa luvussa käsitellään sitä, mitä vaatimuksia RIA-sovelluksen tulee täyttää.

3.1 RIA-teknologiat

RIA-teknologiat mahdollistavat rikkaiden internet-sovellusten luomisen, kehittämisen ja suorittamisen. Allairen (2002) mukaan RIA-teknologian tulisi:

- Tarjota suorituskykyinen ajoympäristö koodin, sisällön ja kommunikoinnin suorittamiseen.
- Tarjota tehokas ja laajennettava oliomalli vuorovaikutukseen.
- Mahdollistaa nopea ohjelmistonkehittäminen komponenttien ja uudelleen käytettävyyden avulla.
- Mahdollistaa ohjelmien toimiminen niin offline- kuin online-tiloissakin.
- Mahdollistaa web- ja datapalveluiden käyttö sovelluspalvelimen kautta.
- Mahdollistaa sovellusten kehittäminen eri alustoille ja laitteille.

Tällaisia teknologioita on olemassa useita ja esiteltäväksi on koottu niistä yleisimmin käytetyt. Teknologioiden esittely on rajattu vain kunkin teknologian uusimpiin, yleisesti käytettävissä oleviin versioihin. Mahdollisia beta-versioita ei oteta huomioon, eikä myöskään aiempia versioita käsitellä kovin tarkasti.

3.1.1 Ajax

Ajax ei ole varsinaisesti teknologia, vaan itse asiassa kokoelma jo olemassa olevia teknologioita ja uusi tapa käyttää niitä yhdessä. Ajax on akronyymi sanoista Asynchronous JavaScript And XML. Ajax ei myöskään ole mikään tuote, jonka joku omistaa tai jonka voi jostain ladata. Se on lähinnä ajattelumalli – tapa ajatella web-sovellusten arkkitehtuuria käyttäen tiettyjä teknologioita. (Garret 2005.)

Ajax koostuu seuraavista teknologioista (Garret 2005.):

- **XHTML ja CSS:** käytetään sovelluksen pohjan luomiseen.

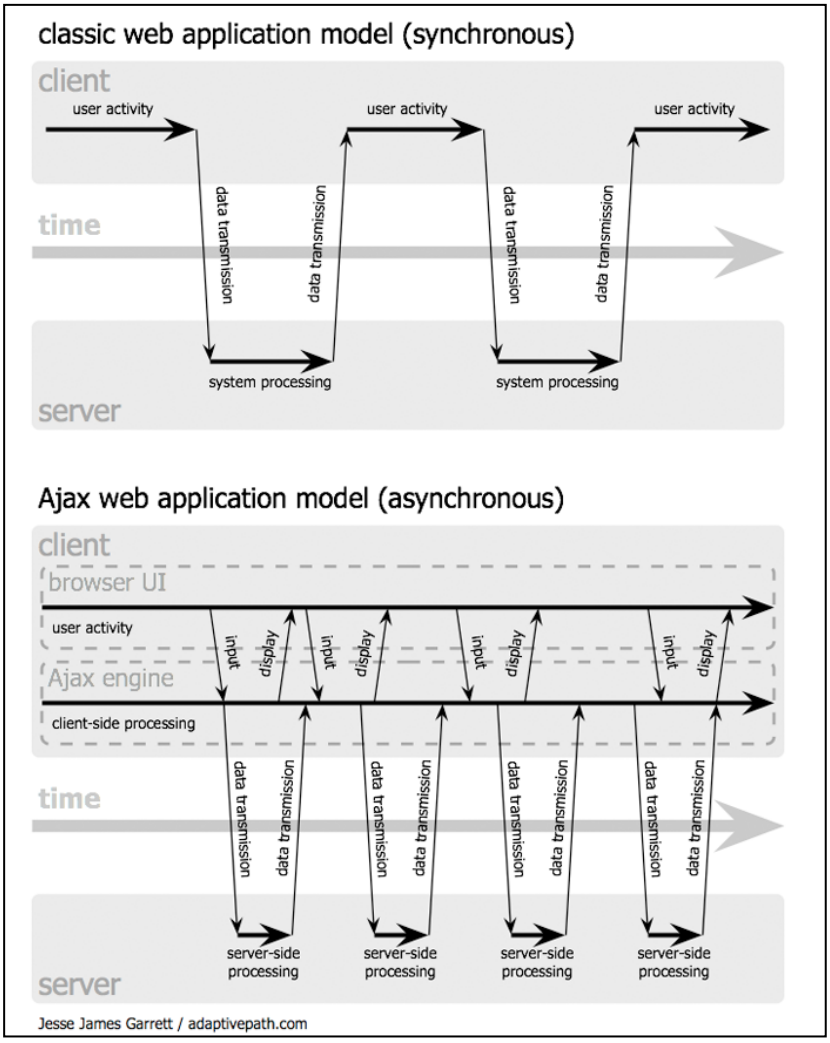
- **DOM:** toteutetaan dynaaminen esillepano ja vuorovaikutus sovelluksen ja käyttäjän välille.

- **XML:** käytetään tiedonvälitykseen ja -käsittelyyn.

- **XMLHTTPRequest:** mahdollistaa asynkronisen tiedonvälityksen.

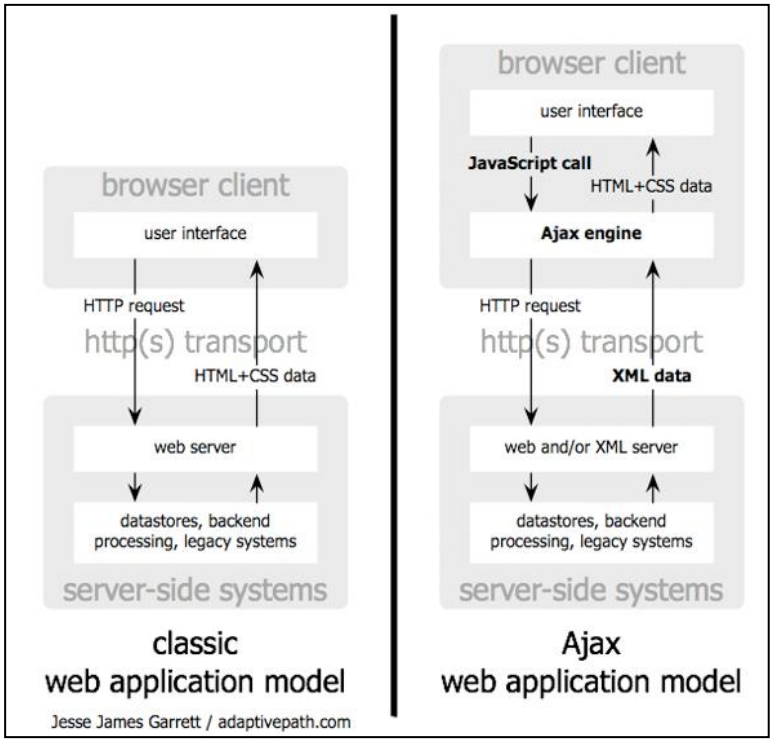
- **JavaScript:** käytetään sitomaan edellä mainitut teknologiat yhteen.

Kommunikointiin palvelimen kanssa Ajax käyttää JavaScriptin XMLHttpRequest-objektia, joka mahdollistaa tiedon välityksen ilman sivun uudelleenlatausta. XMLHttpRequest-objekti on tuettu kaikissa nykyaikaisissa selaimissa, kuten Internet Explorerissa, Firefoxissa, Chromessa, Operassa ja Safarissa. (AJAX XMLHttpRequest n.d.)



KUVIO 1. Perinteinen malli vs. Ajaxin asynkroninen malli (Garret 2005)

Kuten kuvio 1 osoittaa, perinteiset web-sovellukset toimivat seuraavasti: Käyttäjä tekee sivuilla jotain, joka lähettää HTTP-pyynnön palvelimelle (esimerkiksi klikkaa linkkiä). Tämän jälkeen palvelin tekee oman osuutensa, muun muassa prosessoi datan ja sitten lähettää HTML-sivun takaisin asiakkaalle eli käyttäjän selaimelle. Koska joka kerta palvelimelta palautetaan kokonaan uusi sivu, perinteiset web-sovellukset ovat hitaita ja käytettävyydeltään huonoja. Tähän Ajax pyrkii tuomaan ratkaisun, sillä sen avulla sovellukset voivat lähettää ja vastaanottaa dataa lataamatta koko sivua uudestaan. Käytännössä tämä tapahtuu lähettämällä HTTP-pyyntö palvelimelle ja muokkaamalla vain tarvittava osa sivusta JavaScriptin avulla, kuten kuvio 2 osoittaa. (Garret 2005.) Yleisin tapa viestiä palvelimen kanssa on välittää data XML-muodossa, mutta myös muita tapoja on mahdollista käyttää (Peltomäki & Nykänen 2006, 296).



KUVIO 2. Perinteisen web-sovellusmallin ja Ajax-mallin erot (Garret 2005)

Positiivista Ajaxissa on sen toiminta kaikissa nykyaikaisissa selaimissa ilman erillisen liitännäisen asennusta, kunhan selain tukee JavaScriptiä. Tämä on myös osaltaan sen heikkous, sillä eri selaimissa Ajaxilla toteutetut sivut voivat näyttää erilaisilta, koska ne tukevat JavaScriptiä eri tavalla. Lisäksi JavaScript voidaan kytkeä pois päältä selaimesta, jolloin sillä toteutetut sivut eivät välttämättä toimi oikein.

Ajaxin vahvuuksina voidaan pitää sen oppimisen helppoutta, yhteisön laajuutta, ilmaisten frameworkien määrää ja sitä, että internet tarjoaa runsaasti materiaalia aiheesta. Ajaxin heikkoutena voidaan pitää frameworkien määrää siinä mielessä, että on vaikea valita sopivin framework omiin tarkoituksiin. Lisäksi testaaminen on monimutkaista ja hankalaa.

3.1.2 Adobe Flex

Flex on Adoben kehittämä sovelluskehitysratkaisu RIA-sovellusten luontiin. Ensimmäinen versio Flexistä julkaistiin vuonna 2004, ja nykyään on päästy kolmanteen kokonaiseen versioon. Myöhemmin tänä vuonna tullaan julkaisemaan Flex 4, mutta tässä käsitellään kolmatta, tällä hetkellä viimeisintä vakaata ja julkista versiota.

Flex on avoimen lähdekoodin sovelluskehitysratkaisu RIA-sovellusten kehittämiseen. Flexillä luodut sovellukset koostuvat MXML-merkkauskielestä ja ActionScript 3 -ohjelmointikielestä. MXML on XML -pohjainen kieli ja sitä käytetään kuvaamaan sovelluksen ulkoasua ja käyttäytymistä. ActionScript 3 on taas tehokas olio-ohjelmointikieli, jolla sovelluksen toiminnallisuus toteutetaan. Flex sisältää itsessään paljon hyödyllisiä komponenttikirjastoja, joita voi käyttää vapaasti. Flexillä kehitetyt sovellukset toimivat selaimessa Adoben Flash Player -ajoympäristössä tai työpöydällä Adobe Air -ajoympäristössä. Tämä mahdollistaa sen, että Flex-sovelluksia voidaan käyttää kaikilla yleisimmillä käyttöjärjestelmillä ja selaimilla. (Flex overview n.d.)

Käytännössä Flex-sovellus käännetään SWF-tiedostoksi ennen ajoa ja tämä tiedosto toistetaan Flash Playerillä. Flexin vahvuuksina voidaankin pitää sitä, että Flash Player on asennettu 98 %:iin tietokoneista, jotka ovat yhteydessä internetiin. Muita Flexin vahvuuksia ovat vahva dokumentaatio, laajat valmiit kirjastot, Flex Builder, suuri yhteisö, CSS-tuki ja hyvä integroitavuus useimpien ohjelmointikielien kanssa. Parhaiten Flex soveltuu vähän isompien sovellusten kehittämiseen esimerkiksi bisnes-käyttöön.

Adobe Air

Adobe Air (Adobe Integrated Runtime) on käyttöjärjestelmäriippumaton, ja sen tarkoituksena on mahdollistaa rikkaiden internet-sovellusten kehittäminen ja käyttäminen työpöytäsovelluksina. Mediatyyppinä Air edustaa uutta sukupolvea, joka tuo yhteen web- ja työpöytäsovellusten parhaat puolet. Sitä voikin kuvailla puuttuvana siltana työpöytäsovellusten ja web-maailman välillä. (Adoben uusi alusta internet-sovelluksille on nyt valmis 2008.)

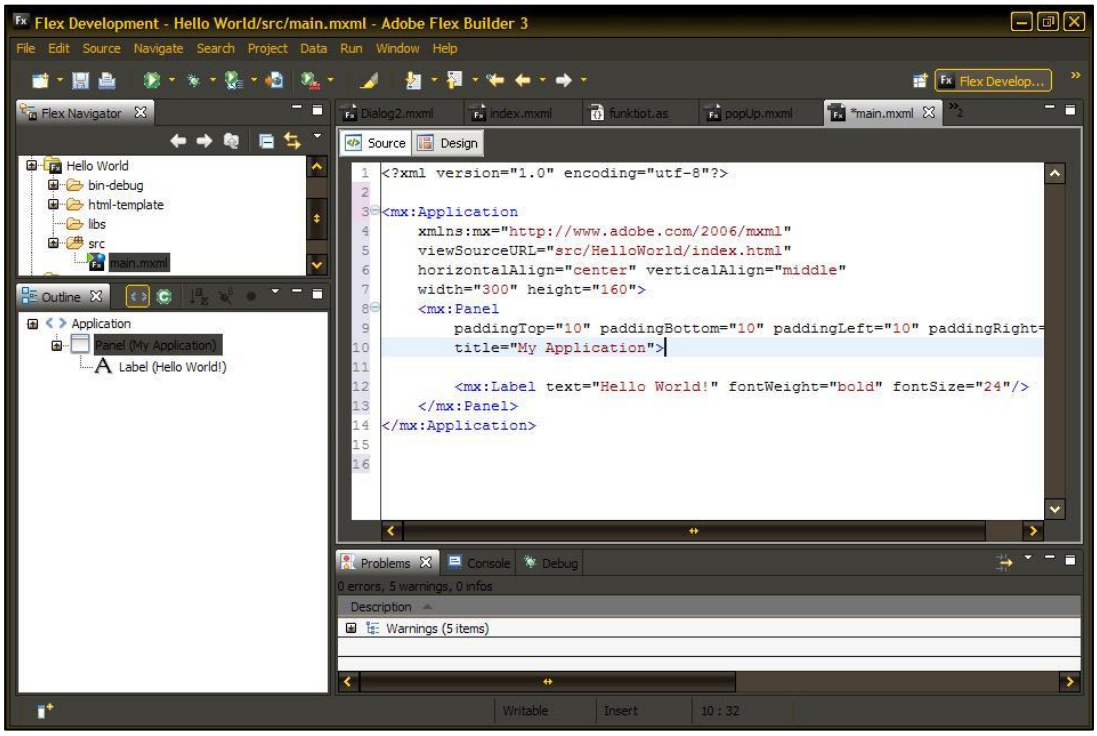
Käytännössä Adobe Air siis tuo RIA-sovellukset työpöydälle. RIA-työpöytäsovelluksilla on joitain merkittäviä hyötyjä selainsovelluksiin nähden. Esimerkiksi ne voivat pyöriä taustalla ja tarjota ilmoituksia kuten perinteiset työpöytäsovelluksetkin. Ne voivat myös tallentaa tietoa paikallisesti ja toimia offline-tilassa. Lisäksi käyttäjä voi vaikuttaa sovelluksen käyttöliittymään monipuolisemmin kuin selaimessa toimivilla sovelluksilla. (Browser vs. desktop n.d.)

Pääosin RIA-sovellukset sekä työpöydällä että selaimessa tarjoavat samat mahdollisuudet ja molemmissa on omat hyötynsä ja haittansa toisiinsa nähden. Se, kannattaako

sovellus kehittää työpöydälle vai selaimeen, riippuu paljon sovelluksen käyttötarkoituksesta. Tarvittaessa on mahdollista kehittää molemmat.

Flex Builder 3

Flex Builder 3 on Eclipse-pohjainen, alustariippumaton kehitystyökalu Flex-sovelluksien tekoon. Flex Builder helpottaa sovellusten koodaamista tarjoamalla muun muassa syntaksin korostuksen, debug-työkalun, graafisen ulkoasun suunnittelun, lauseiden ennakoinnin, virheiden paikantamisen sekä paljon muita hyödyllisiä toimintoja ja ominaisuuksia. (Flex FAQ n.d.)



KUVIO 3. Screenshot Flex Builder 3-ohjelmasta

Flex Builder on maksullinen, mutta siitä on olemassa ilmainen kokeiluversio, ja lisäksi opiskelijoilla on mahdollisuus saada se käyttöönsä ilmaiseksi Adoben sivuilta. Flex-sovelluksia voidaan tehdä myös ilman Flex Builderia, sillä Adobe on julkaissut Flex SDK:n avoimella lisenssillä. Käytännössä tämä on monimutkaisempaa kuin sovellusten kehittäminen käyttäen Flex Builderia.

3.1.3 Microsoft Silverlight

Silverlight on Microsoftin kehittämä käyttöjärjestelmä-, selain- ja laiteriippumaton teknologia rikkaiden internet-sovellusten kehittämiseen ja jakamiseen. Silverlight pohjautuu Microsoftin .NET framework -sovelluskehykseen. Käytännössä Silverlight on selaimeen asennettava laajennus, joka tarjoaa ajoympäristön sitä käyttäville ohjelmille. Tämä mahdollistaa osaltaan sen, että ohjelmista voidaan tehdä käyttöjärjestelmä- ja selainriippumattomia. Silverlight-sovelluksia voidaan kehittää muun muassa Visual Studio -ohjelmalla. (Ghoda & Scanlon 2009, 1–8.)

Microsoft on selvästi uudempi tekijä RIA-markkinoilla ja voidaankin olettaa sen olevan vielä hieman jäljessä Adobea. Uutuudestaan huolimatta se on kehittynyt nopeasti ja on varteenotettava vaihtoehto RIA-sovellusten kehittämiseen. Varsinkin kun monet yritykset valmiiksi luottavat Microsoftin tekniikoihin, on heille luonnollista lähteä myös tarvittaessa RIA-kehitykseen Microsoftin tekniikalla. Haasteena Microsoftilla on saada käyttäjät asentamaan Silverlight-laajennus selaimeensa, jotta Silverlight-sovellukset toimivat. (The Battle for the RIA Throne: Flex vs. Silverlight 2008.)

Nykyään Silverlight on kehittynyt kolmanteen versioonsa ja onkin ehkä merkittävin kilpailija Adobe Flexille. Silverlightista on tulossa myös 4. versio, mutta se on vielä beta-vaiheessa.

3.1.4 JavaFX

JavaFX on Sun Microsystemsin kehittämä teknologia RIA-sovellusten kehittämiseen ja jakamiseen. Nykyinen versio on 1.2 ja se julkaistiin kesäkuussa 2009. JavaFX mahdollistaa sovellusten kehittämisen työpöydille, selaimiin ja mobiililaitteisiin. JavaFX-sovellukset käyttävät suosittua Sunin Java-ajoympäristöä, joten sillä toteutetut sovellukset toimivat miljardeissa laitteissa ympäri maailmaa ja tämä onkin sen suurin vahvuus. JavaFX tarjoaa myös oman skriptikielen, JavaFX Scriptin, jolla sovellukset toteutetaan. JavaFX Script on staattisesti tyypitetty, deklaratiivinen kieli ja se myös mahdollistaa perinteisten Java -luokkien käytön sovelluksissa. (JavaFX FAQ n.d.)

3.2 Oikean RIA-teknologian valinta

Nykyisin on olemassa lukuisia RIA-teknologioita ja oikean valitseminen on koko ajan hankalampaa. Jokaisella on omat vahvuutensa ja heikkoutensa kuin myös ominaisuutensa tietoturvallisuudessa, käytettävyydessä ja integroinnissa. Teknologiaa valittaessa kannattaa ottaa huomioon esimerkiksi seuraavat asiat:

- **Kehittämisen helppous:** jotkin teknologiat on helpommin oppia kuin toiset ja niillä sovellusten toteuttaminen on yksinkertaisempaa kuin toisilla.

- **Integroitavuus:** mitä parempi integroitavuus on, sitä helpommin vältytään tulevaisuudessa odottamattomilta integrointi ongelmilta.

- **Tietoturva:** hyvä ratkaisu on tietoturvallinen.

- **Selain- ja käyttöjärjestelmäriippumattomuus:** Suurin osa internetin käyttäjistä käyttää Internet Explorer -selainta, mutta myös esimerkiksi Firefox-selain on suosittu. On suotavaa, että sovellus toimii samalla tavalla selaimesta riippumatta.

- **Teknologian tulevaisuuden näkymät:** Noudattaako ratkaisu standardeja ja onko se esimerkiksi avoimen lähdekoodin projekti? Tällä voi olla tekemistä siirrettävyyden kannalta tulevaisuuden infrastruktuureissa.

- **Koulutus kustannukset:** uuden teknologian oppiminen voi olla hidasta ja kallista.

- **Liitännäisten riippuvuus:** Täytyykö käyttäjän asentaa uusia ohjelmia tai liitännäisiä tietokoneelleen, jos sovelluksia kehitetään tietyllä teknologialla? Kaikki käyttäjät eivät pidä tällaisesta.

- **Kehitystyökalut:** Hyvät kehitystyökalut nopeuttavat kehittämisen elinkaarta ja helpottavat kehittäjien työtä.

- **Teknologian tausta:** Kuka on teknologian takana? Korjataanko bugeja ja tuleeko uusia versioita? On hyvä jos teknologialla on aktiivinen kehitysyhteisö ja dokumentaatio.

Yhteenvetona voitaisiin sanoa, että paras RIA-teknologia riippuu kulloinkin vaatimuksista ja tarpeista. Mikään ratkaisu ei varmasti miellytä kaikkia käyttäjiä kaikissa tapauksissa. (Cheng 2008.) Lisäksi tulee ottaa huomioon, että teknologiat kehittyvät nopeasti ja kunkin teknologian seuraava versio voi sisältää ominaisuuksia, jotka kallistavat vaakaa taas toiseen suuntaan.

Voidaan kuitenkin todeta, että tällä hetkellä Adobe Flex, Ajax ja Microsoft Silverlight ovat merkittävimmät keinot kehittää rikkaita internet-sovelluksia. Vaikka teknologioiden kehitys IT-alalla on todella nopeaa ja trendit herkästi muuttuvia, on todennäköistä, että edellä mainitut teknologiat säilyttävät paikkansa RIA-teknologioiden huipulla. Kilpailu RIA-teknologioiden välillä on kovaa ja sen vuoksi ne kehittyvätkin vauhdikkaasti.

Ajoympäristöt

Sovellusten saavutettavuus on syytä ottaa huomioon. Tällä tarkoitetaan sitä, kuinka helposti käyttäjät pääsevät käyttämään sovellusta, mielellään ilman uuden liitännäisen tai ohjelman asennusta. Ajaxia lukuun ottamatta edellä esitellyillä teknologioilla toteutetut web-sovellukset toimivat jossain ajoympäristössä. Adoben teknologiat käyttävä Flash Playeria, Microsoft käyttää Silverlight Playeria ja JavaFX Sunin Java-ajoympäristöä. Tämä on hyvä ottaa huomioon pohdittaessa sitä, millä teknologialla sovelluksia lähdetään kehittämään.

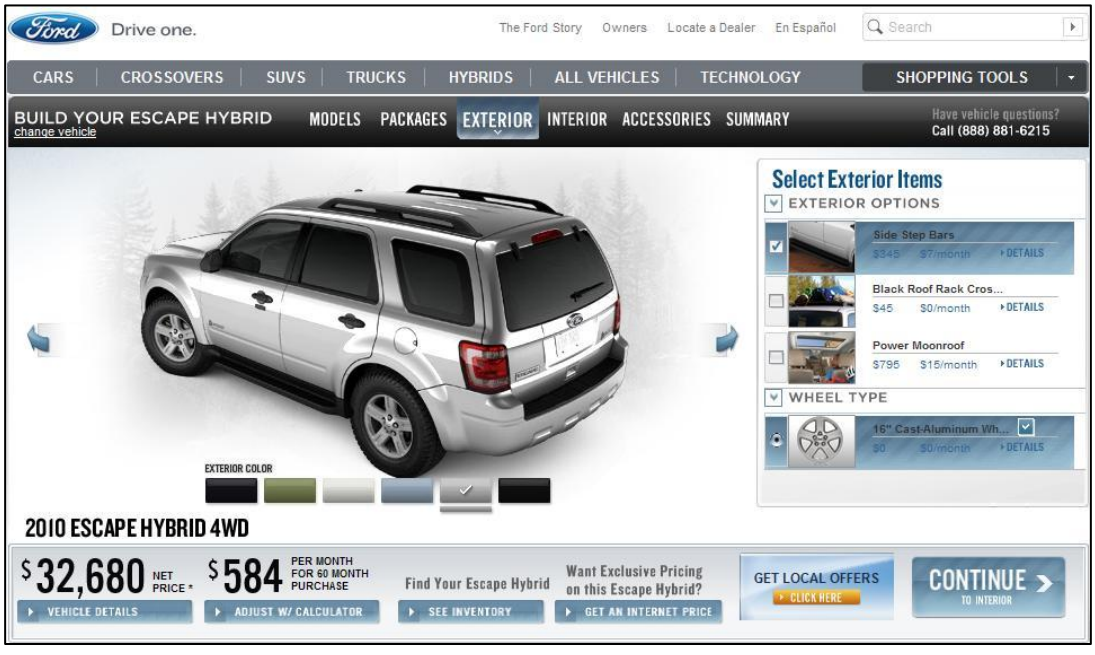
Riastats.com on sivusto, joka kerää tietoa kolmen yleisemmän RIA-ajoympäristön käytöstä ja esittää ne helposti luettavina kaavioina. Sivut vertailevat sitä, kuinka moneen selaimeen on asennettu Adobe Flash Player-, Microsoft Silverlight Player- ja Sun Java -liitännäiset. Dataa kerätään julkisille web-sivustoille (95 sivustoa) upotetun JavaScript-koodin avulla. Sivuston tarkasteluhetkellä (23.2.2010) tilastot oli kerätty 15 012 230 selaimesta edellisen 30 päivän aikana. Sivuston mukaan näistä selaimista 97,22 %:iin oli asennettu jokin Adoben Flash Playerin versio. Microsoft Silverlight Playerin kohdalla sama prosenttiosuus oli 54,38 % ja Sunin Java-ajoympäristön koh-

dalla osuus oli 73,04 %. Tuloksista voidaan päätellä, että Flash Player on suosituin selaimiin asennettu ajoympäristö, seuraavaksi tulee Java ja kolmanneksi Silverlight Player, joka on asennettu vain noin 55 %:iin selaimista. (Riastats n.d.)

3.3 RIA-sovellusten hyödyntäminen sähköisessä liiketoiminnassa

Nykypäivän globaalissa bisnes-maailmassa kuluttajat ovat vaativampia, ja merkkiuskollisuus on vaikeammin saavutettavissa ja säilytettävissä kuin ennen. RIA-sovellukset voivat tehdä kuluttajan kokemuksesta vaikuttavamman, dynaamisemman ja tehokkaamman. Tyytyväiset käyttäjät ovat myös tärkeitä merkin tai tuotteen puolestapuhujia. (Benefits of Rich Internet Applications n.d.)

Perinteinen keino hyödyntää RIA-teknologioita sähköisessä liiketoiminnassa on auttaa kuluttajaa päätöksenteossa. Kansainvälinen autojätti Ford on onnistunut hyvin tuomaan RIA-sovelluksen lähelle kuluttajaa. He tarjoavat verkkosivuillaan (kuvio 4) kuluttajalle mahdollisuuden koostaa autosta omaan makuun sopiva. Kuluttaja voi hiirellä valita muun muassa haluamansa auton mallin, värin ja lisävarusteet samalla kuin sovellus päivittää ulkonäön, hinnan ja muut tiedot ruudulle ilman sivujen uudelleen latausta. Tämän jos minkä luulisi auttavan ostajaa tekemään valintoja.



KUVIO 4. RIA:n hyödyntäminen fordvehicles.com-sivustolla (Fordvehicles n.d.)

Nykyään RIA-sovelluksia ei käytetä vain kuluttajille suunnatuissa sovelluksissa tai sivustoissa, vaan organisaatiot hyödyntävät niitä sisäisesti mahdollistamalla työntekijöille tehokkaamman tavan työskennellä. Usein yritykset ovat joutuneet päättämään hyvin saavutettavissa olevien web-sovellusten ja paremman toiminnallisuuden omaavien työpöytäsovellusten välillä. RIA yhdistää näiden parhaat puolet. RIA-sovelluksia voidaan hyödyntää yrityksissä monin tavoin. Niiden kehittäminen ja ylläpito on nopeaa ja niiden jakelu eteenpäin on helppoa ja tehokasta. Niitä voidaan välittää yrityksen lähiverkon, internetin tai vaikka cd:n/dvd:n välityksellä. RIA-sovelluksia voidaan käyttää yrityksessä helpottamaan työntekijöiden päätöksentekoa, koska ne tarjoavat interaktiivisen ja helpon pääsyn yrityksen tietolähteisiin. Lisäksi niitä voidaan käyttää tehostamaan tavallisia työsuorituksia. (The business benefits of rich Internet application for enterprises 2008.)

3.4 RIA-sovellusten tuomat edut ja ongelmat

Käytettävyys näyttelee suurta roolia web-sovelluksen toimivuuden kannalta. RIA-teknologioilla toteutetut sovellukset voivatkin parantaa käytettävyyttä merkittävästi ja näin ollen parantaa yrityksen imagoa varsinkin verkossa. Hyvät web-sivut tai toimiva web-sovellus antaa yrityksestä asiantuntevamman ja positiivisemman kuvan kuin vanhat staattiset sivut tai käytettävyydeltään heikompi sovellus.

Tutkija Nicoline Van Eltenin (2008) mukaan RIA-sovellusten suurimmat hyödyt käytettävyyden kannalta ovat seuraavat:

1. **Nopeus**: RIA-sovellukset vastaavat nopeammin käyttäjän tekemiin toimintoihin kuin perinteiset web-sivut. Tämä siksi, että vain tarvittava osa sivusta päivitetään eikä koko sivua tarvitse ladata uudestaan. Näin ollen tämä vähentää tietoliikennettä käyttäjän ja palvelimen välillä, joten aikaa säästyy ja käyttäjäkokemus on miellyttävämpi.

2. **Tiedon hakeminen:** Nykyään web-sivut ovat suuressa roolissa kommunikoinnissa ja bisneksessä. Tästä johtuen yritysten sivut sisältävät usein paljon sisältöä. RIA voi auttaa tässä tapauksessa käyttäjää löytämään etsimänsä helpommin, ilman että hänet ohjataan koko ajan uudelle sivulle. Tämä parantaa tiedon saavutettavuutta sekä selkeyttä.

3. **Helppokäyttöisyys**: RIA auttaa käyttäjiä esimerkiksi lomakkeiden täytössä tarjoamalla aiemmin käytettyä informaatiota tarvittaviin kohtiin. Näin ollen käyttäjän ei tarvitse syöttää samoja tietoja uudestaan, ja tämä tapahtuu reaaliajassa, ilman sivunlatauksia.

4. **Multimedian hyödyntäminen:** RIA-sovelluksia voidaan hyödyntää usein eri tavoin monissa eri käyttötarpeissa. Ne voivat yhdistää kaupankäyntiä, kommunikointia, videoita ja animaatioita uniikilla tavalla. Siksi RIA tekee verkkosivuilla käynnistä kokemuksen itsessään.

RIA-sovellusten hyödyt eivät rajoitu pelkkään käytettävyyteen vaan voidaan katsoa, että niistä on myös muun muassa seuraavia hyötyjä:

- Ei tarvitse asentaa, jos käytetään selaimessa.
- Pääosin käyttöjärjestelmäriippumattomia.
- Mahdollistaa alustariippumattomuuden, voidaan kehittää niin työpöydälle, selaimeen kun mobiililaitteisiinkin.
- Mahdollistaa välittömän ja dynaamisen palautteen annon käyttäjälle.
- Mahdollistaa reaaliaikaisen tiedonhallinnan. (Benefits of RIA’s n.d.)
- Päivittää vain tarvittavan tiedon, ei tarvitse ladata koko sivua uudestaan, mikä vähentää tietoliikennettä asiakkaan ja palvelimen välillä.
- Datan visualisointi ja käsittely tehokkaampaa
- Mobiili saavutettavuus. (5 Benefits Of Rich Internet Applications For Manufacturing ROI 2009.)

RIA-sovelluksista näyttää olevan paljon hyötyä, mutta jottei asia olisi niin mustavalkoinen, on niissä jotain ongelmiakin sovelluskehityksen kannalta. Merkittävin ongelma on, että RIA-sovellukset, Ajax/JavaScript-toteutuksia lukuun ottamatta, pyörivät pääosin jossakin ajoympäristössä. Tästä johtuen käyttäjä joutuu yleensä lataamaan

jonkin selainlaajennuksen, ennen kuin sovellusta voidaan käyttää. Pitää ottaa huomioon, ettei jokainen käyttäjä halua tai edes voi asentaa ylimääräistä liitännäistä.

RIA-sovelluksilla on myös ongelma hakukoneoptimoinnin kannalta. Hakukoneoptimointi parantaa verkkosivujen näkyvyyttä hakukoneiden (kuten Google tai Yahoo!) tuloksissa, ja yrityksille on usein tärkeää löytyä hakutulosten kärkipäästä. Perinteisesti hakukoneet eivät ole indeksoineet kunnolla esimerkiksi flash-sivustoja. RIA-sivustot sisältävät paljon dataa, joka haetaan vasta vastauksena käyttäjän toimeen. Tästä johtuen hakukoneet eivät ole voineet löytää ja indeksoida tätä dataa, kun sitä ei oletuksena sivuilla ole. Tämä on ollut ongelmana RIA-sovellusten hakukoneoptimoinnissa. Nykyään on erilaisia ratkaisuja kiertää tätä ongelmaa, mutta ongelma on kuitenkin olemassa, ja hakukoneoptimointi on hankalampaa tai vaatii enemmän perehtymistä kuin hakukoneoptimointi perinteisillä web-sivustoilla. (Search optimization techniques for RIAs n.d.)

4 WEB-SOVELLUSTEN KÄYTETTÄVYYS

Tässä luvussa käsitellään yleisesti web-sovellusten käytettävyyttä ja lopuksi tutkitaan mitä annettavaa RIA-sovelluksilla on käytettävyyteen. Luvussa käsitellyt seikat ovat tärkeitä ottaa huomioon myös järjestelmän suunnittelu- ja toteutusvaiheessa, jotta järjestelmä olisi onnistunut.

Käytettävyys tarkoittaa menetelmä- ja teoriakenttää, jonka avulla laitteen ja käyttäjän yhteistoimintaa pyritään tehostamaan ja muokkaamaan käyttäjän kannalta miellyttävämmäksi. Käytettävyyden määrittelyssä käytetään hyväksi kognitiivisen psykologian sekä ihmisen ja koneen vuorovaikutuksen tutkimusta. Käytettävyydelle on useita määritelmiä, mutta seuraavaksi esiteltävät Jakob Nielsenin ja ISO 9241–11-standardin määritelmät ovat niistä eniten käytettyjä. (Sinkkonen, Kuoppala, Parkkinen & Vastamäki 2006, 17–18.)

Käytettävyyden tutkimuksen uranuurtaja Jakob Nielsen (1993) määrittelee käytettävyyden osaksi tuotteen käyttökelpoisuutta. Käyttökelpoisuuteen vaikuttavat seuraavat viisi tekijää:

1. **Opittavuus**: Käyttäjän on helppo oppia käyttämään tuotetta ja käyttöönotto on nopeaa ja vaivatonta.

2. **Tehokkuus**: Tuotteen on oltava tehokas käyttää ja sillä on saavutettavissa korkea tuottavuustaso.

3. **Muistettavuus**: Tuotteen käytön on oltava helposti käyttäjän muistettavissa, myös pitkän käyttämättömän ajanjakson jälkeen.

4. **Virheet**: Tuotteessa tulee olla mahdollisimman vähän virheitä. Tuotteen on pystyttävä toipumaan käyttäjän tekemistä mahdollisista virheistä, eikä tuotteen kannalta vakavia virheitä saa esiintyä.

5. **Tyytyväisyys**: Tuotteen tulee olla käyttäjäystävällinen ja käyttäjien mielestä mukava käyttää.

Käyttökelpoisuuden määritelmä on vaikea, sillä käytettävyydenkin on oltava kunnossa, jotta tuote olisi käyttökelpoinen. Lisäksi käyttötilanteet vaihtelevat käyttäjän tarpeen mukaan. (Nielsen 1993, 26–37.)

ISO 9241–11-standardi näyttöpäätetyön ergonomiasta – ohjeita käytettävyydestä – määrittelee käytettävyyden riippuvaiseksi käyttötilanteesta. Sen mukaan tuotteen käytettävyys kertoo, kuinka hyvin käyttäjät voivat käyttää tuotetta tehokkaasti, miellyttävästi ja tuottavasti määritettyjen tavoitteiden saavuttamiseksi tietyssä käyttöympäristössä. (Sinkkonen ym. 2006, 17–18.)

Useasti ihmisen ja koneen vuorovaikutus (Human-Computer Interaction) ja käytettävyys nähdään samana asiana. Ihmisen ja koneen vuorovaikutus ei kuitenkaan ajattele ihmistä organisaation osana, tahtovana toimijana ja työntekijänä. Käytettävyys sen sijaan ottaa huomioon nämäkin ihmisen ja laitteen vuorovaikutukseen kuuluvat osat. (Sinkkonen ym. 2006, 17–18.)

4.1 Käyttäjien toiminnan ja tarpeiden ymmärtäminen

Käyttäjien tehtävien ja työn kautta oppimat toimintatavat poikkeavat usein merkittävästi suunnittelijoiden oppimasta. Tästä syystä jokaiseen hyvään tuotekehitysprojektiin kuuluu käyttäjien toiminnan havainnointi niin, että tiedetään, mitä käyttäjät oikeasti tuotteella tekevät ja mitkä ovat potentiaaliset virhetilanteet. Käyttäjien toimintaa havainnoidaan ja tutkitaan jo ennen tuotteen suunnittelua, suunnittelun aikana ja sen jälkeen. Ennen suunnittelua rakennetaan käyttäjistä kuvaukset sen perusteella, millaisia käyttäjiä he ovat, ja jaetaan heidät käyttäjäryhmiin. Käyttäjäryhmän muodostavat ne käyttäjät, jotka toimivat tuotteen käyttöliittymän kannalta katsottuna samalla tavoin keskenään. Käyttäjäryhmittelyn perustana ovat useimmiten henkilöiden rooli ja tarpeet organisaatiossa sekä kokemus tuotteesta tai ylipäätään tietotekniikasta. Lisäksi koulutus, ikä, toiminnan rajoitteet ja käyttötilanteet on hyvä ottaa huomioon eri käyttäjäryhmiä suunniteltaessa. Suunnittelussa käytetään hyväksi näitä käyttäjäryhmiä, tai mieluummin käyttäjäryhmiä edustavia persoonia, joiden kuvaus on tiivistelmä käyttäjäryhmän merkityksellisistä piirteistä. (Sinkkonen ym. 2006, 17–18.)

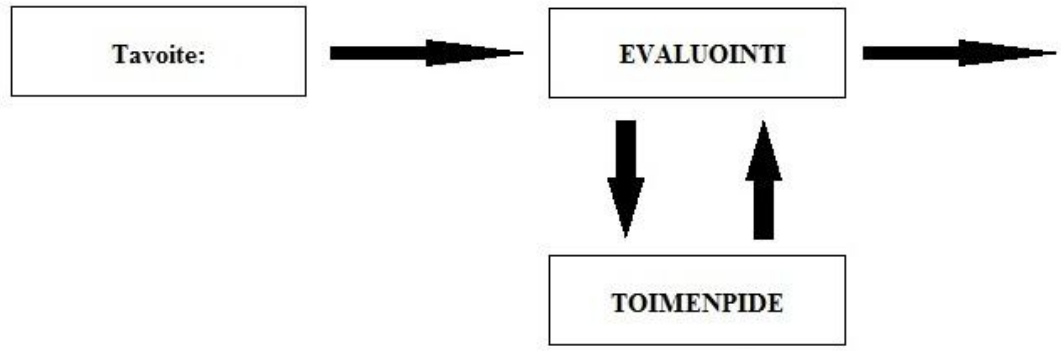
Persoonien toiminta ja tekemiset tuotteen kanssa kuvataan vastaavasti skenaarioissa. Toimintaskenaario kuvaa persoonan toiminnan ilman tuotetta ja käyttöskenaario ku-

vaa sen, miten persoona tulisi toimimaan uudella, suunniteltavalla tuotteella. Persoonaa käytetään suunnittelun lisäksi myös projektin loppuvaiheessa toiminnan simuloinnissa ja tuotteen käyttöliittymän testauksessa. Testaus edellyttää, että käyttäjillä on käytössään tuote tai prototyyppi, jolla voidaan suorittaa joitain testattavia tehtäviä. Samalla suoritettava käyttäjien toiminnan hahmottaminen tarkoittaa käyttäjän toiminnan seuraamista ja mallintamista niin, että tuotetta voidaan edelleen suunnitella ja kehittää käyttäjien toimintaa tukevaksi. (Sinkkonen ym. 2006, 17–18.)

4.2 Tuotteen käyttäminen

Ihmisen toimiessa hän joko asettaa itselleen jonkin päämäärän, jota kohti hän pyrkii, tai sitten ulkomaailmassa ilmenee jotain, joka käynnistää toiminnan. Toiminta voi olla esimerkiksi sisäsyntyisen tavoitteen ”olla kylläinen” tai tarpeen ”nälkä” täyttämistä tai ulkoisen pakon ”liikennemerkki STOP” tai virikkeen ”herkullinen keksi” käynnistämää toimintaa. (Sinkkonen ym. 2006, 47.)

Kaikessa ihmisen tavoitteellisessa toiminnassa voi havaita kolme perusvaihetta: tavoitteen asettaminen, toiminnon tai toimenpiteen tekeminen ja vaikutuksen tarkastaminen eli toiminnan evaluointi palautetta apuna käyttäen. (Sinkkonen ym. 2006, 47.)



KUVIO 5. Ihmisen tavoitteellinen toiminta (Sinkkonen ym. 2006, 47)

Ihmisen toiminta alkaa tavoitteen luonnista. Tällöin hän tarkistaa, ovatko asiat niin kuin hän toivoisi. Mikäli tavoitetta ei ole saavutettu, hän tekee toimenpiteen, jonka uskoo korjaavan asiantilan. Tämän jälkeen hän evaluoi tilanteen ja tekee uuden toimenpiteen tarpeen vaatiessa. Tämä kierre jatkuu niin kauan, kunnes tavoitteet täytty-

vät. Varsinainen tuotteen käyttäminen on harvoin kenenkään tavoite. Tuote on vain apuväline tavoitteen saavuttamiseksi. Esimerkiksi tavoitteen: saada käteistä toteuttaminen tapahtuu pankkiautomaatin avulla. Ihmisen tavoitteen ymmärtäminen on tärkeää ihmisen toiminnan ymmärtämisessä. Myös toiminnan apuvälineen eli käyttöliittymän suunnittelussa tulisi aina ottaa huomioon, mikä on minkin toiminnan tavoite ja mitä tavoitteita eri käyttäjillä on, kun he tulevat tiettyyn valikkoon tai internet-sivulle. (Sinkkonen ym. 2006, 47–48.)

4.2.1 Havainnointi osana tuotteen käyttöä

Pystyäkseen käyttämään tuotetta tehokkaasti käyttäjän pitää pystyä havaitsemaan kaikki tehtävän suorittamisen kannalta oleelliset tuotteen osat. Hänen pitää pystyä myös havaitsemaan toimenpiteidensä vaikutus tuotteen tilaan. Mikäli hän ei havaitse kaikkea, mitä pitäisi, syynä on yleensä se, että jokin väärä asia käyttöliittymässä vie hänen huomionsa, tai se, että asiat eivät hahmotu hänelle tai hahmottuvat väärin. (Sinkkonen ym. 2006, 67.)

Havaitseminen ei kuitenkaan ole pelkkää aistimista. Ei riitä, että toiminnot vain ovat käyttöliittymässä, vaan käyttäjän täytyy pystyä tunnistamaan ne ja mieltää ne joksikin, ennen kuin voi käyttää niitä. Jo ensi kertaa tuotetta käyttävällä käyttäjällä, joka joutuu luottamaan vahvasti itse tuotteessa näkemiinsä mahdollisuuksiin, on olemassa ennakkokäsitys tuotteesta. Ennakkokäsitykset syntyvät käytettäessä muita vastaavia tuotteita, mainosten ja median kautta, käyttöohjeesta tai omana päättelynä siitä, mikä tuotteen käyttötarkoitus on. Ihminen myös tulkitsee kuulemaansa tai näkemäänsä ja vertaa niitä ennakkokäsityksiinsä asiasta. Ihmismuistiin jää oma tulkinta, ei objektiivinen todellisuus. Tulkintaan vaikuttavat aiemmat oman toiminnan kokemukset, opit ja ennakkoluulot ja jopa mielentila. Tyypillisesti ihminen hakee selitysmalleja havaitsemilleen asioilleen, ja havainnot vääristyvät todennäköisempään suuntaan. (Sinkkonen ym. 2006, 67–68.)

Käyttöliittymäsuunnittelun kannalta ehkäpä tärkeimmät seikat, jotka suunnittelijan tulisi ymmärtää ihmisen havaintojärjestelmästä, ovat: ihminen ei havaitse kaikkia asioita, joita käyttöliittymässä on, eikä suunnittelija pysty näkemään tuotteensa käyttöliittymää, kuten aloittelija sen näkee. Tuotteen tekijöille sen jokaisella elementillä on merkitys, mutta käyttäjille näin ei ole ainakaan aluksi. Kun käyttäjä katsoo tuotetta,

koko hänen kokemusmaailmansa vaikuttaa siihen, miten ja miksi hän näkemänsä ymmärtää. Lisäksi ihmisen kyky tunnistaa tuttuja hahmoja ja elementtejä on äärimmäisen tarkka samoin kuin hänen kykynsä oppia tunnistamaan hahmoja, mikäli hänellä on näille hahmoille merkitys. (Sinkkonen ym. 2006, 69.)

4.2.2 Vuorovaikutus tuotteen kanssa

Hyvässä käyttöliittymässä ulkoasu tukee muuta sisältöä ja luo tuotteesta yhtenäisen kokonaisuuden. Tuotteen eri elementit, kuten painikkeet, otsikot, tekstit ja valikot, samoin kuin myös tuotteen käyttäjän muokattavissa olevat osat ovat merkittävä osa käyttöliittymää. Käyttäjän vuorovaikutus käyttöliittymän kanssa perustuu siihen, että käyttäjä pystyy lukemaan suunnittelijan merkkikieltä mahdollisimman helposti. Myös tuotteen visuaalisella suunnittelulla on tärkeä merkitys tuotteen käytettävyyden kannalta. Visuaalisessa suunnittelussa on syytä ottaa huomioon selkeys ja yksikäsitteisyys, joka auttaa aloittelijaa hahmottamaan kokonaisuuksia ja vastaa käyttäjän käsitystä todellisuudesta sekä helpottaa tärkeiden osien näkyvyyttä. (Sinkkonen ym. 2006, 109.)

Visuaalisella suunnittelulla voidaan myös vaikuttaa tuotteen käytön tehokkuuteen. Hyvä ja selkeä visuaalinen ilme voi parantaa työskentelynopeutta jopa 20–40 prosenttia. Tämä kaikki ei ole pelkästään tuotteen värien hallintaa tai kokonaisuuden suunnittelua tasapainoisen näköiseksi. Tärkeintä käyttöliittymässä on toimivuus ja sisältö. Tuotteen ulkonäköä suunniteltaessa täytyy pitää erityisesti huolta tiedon esitysmuodosta ja koodauksesta mukaan lukien käyttäjän ymmärtämä terminologia ja näihin liittyvä symbolien yhdistely ymmärrettäväksi kieleksi. Oleellista on myös eri ikkunoiden eli näkymien asettelu. Siinä apukeinoina voidaan käyttää värejä, typografiaa ja navigoinnin suunnittelua näkymän sisällä eli tiedon organisointia ja järjestämistä sekä taustan käyttöä. Suunnittelijan on lähdettävä liikkeelle siitä, mitä elementtejä tiettyyn ikkunaan tarvitaan, mikä on niiden järjestys, tärkeys, pituus ja muoto ja miten ne suunnitellaan sisällön mukaan järkevästi. (Sinkkonen ym. 2006, 155.)

Käyttöliittymän elementtien asettelun päällimmäinen tavoite on taata niin hyvä kommunikointi käyttäjän ja tuotteen välille kuin mahdollista. Käyttöliittymäkomponenttien asettelussa täytyy kaikesta huolimatta lähteä liikkeelle käyttäjän tavoitteesta ja tehtävistä. Käyttöä ohjaavista asioista etenkin vastaavuus luodaan visuaalisuuden

suunnittelun avulla. Suunnittelijan käytettävissä olevia keinoja ovat elementtien ryhmittely ja järjestys sekä asioiden hierarkian näyttäminen siten, että se vastaa elementtien takana olevia todellisia toimintoja. Asettelun tasapaino riippuu muun muassa elementtien sijoittelusta ja muodosta, tyhjän tilan käytöstä, värien käytöstä ja liikesuunnista. Tasapainoisessa esityksessä olennaiset osat on tuotu esille liioittelematta ja selkeästi. (Sinkkonen ym. 2006, 155.)

4.3 Heuristinen arviointi

Käytettävyys tutkimuksen tunnetuimmat heuristiikat ovat Jakob Nielsenin kymmenen heuristista sääntöä, joiden avulla tarkastellaan käyttöliittymän käytettävyyttä. Usein kehitysprojekteissa tutkijalla on liian suuri ongelma-avaruus hänen arvioidessaan käyttöliittymää. Heuristiikoilla hän rajaa avaruuden pienemmäksi ennalta havaittujen ohjeiden avulla käyttämättä hyväkseen itse ongelmaa eli käyttöliittymää. (Sinkkonen ym. 2006, 206.)

Heuristisen arvioinnin tavoite on löytää käytettävyysongelmia käyttöliittymästä ja sitä käytetään toistuvasti käyttöliittymää suunniteltaessa. Käytännössä heuristinen arviointi suoritetaan keräämällä joukko arvioijia, joista jokainen testaa käyttöliittymää yksinään ja vertaa sitä tunnettuihin käytettävyys sääntöihin eli heuristiikkoihin. Nämä Nielsenin kymmenen heuristista sääntöä ovat (Nielsen 1993, 20):

- **Yksinkertainen ja luonnollinen dialogi:** Dialogissa ei saa olla mitään ylimääräistä tai harvoin tarvittavaa tietoa. Jokainen turha elementti vie tilaa ja näkyvyyttä näkymän oleelliselta ja tärkeältä tiedolta. Lisäksi kaiken tiedon tulisi olla loogisessa järjestyksessä näkymässä.

- **Käyttöliittymä puhuu käyttäjän kieltä:** Dialogin kielen, termistön ja käsitteiden tulee olla selkeitä ja tuttuja käyttäjälle. Liian tekninen sanasto on kitkettävä pois.

- **Toimintojen tunnistaminen:** Tuotteen toimintojen ja eri toimintavaihtoehtojen tulee olla näkyviä. Käyttöliittymän osat ja niiden toiminnot tulee liittää toisiinsa loogisesti niin, että näiden yhteys on pääteltävissä tuotteesta.

- **Johdonmukaisuus:** Tuotteen toimintojen ja dialogien tulee olla yhteneviä, eivätkä esimerkiksi toiminnot saa vaihtaa merkitystään lennossa. Hyvän tuotteen tulee lisäksi tukea opitun siirtämistä, jotta tuotetta on helppo käyttää.

- **Tuotteen tila:** Tuotteen on pidettävä käyttäjä aina tietoisena siitä, mikä on tuotteen tila tai mikä toiminto on juuri nyt menossa.

- **Toimintojen peruutus:** Käyttäjät valitsevat usein eri toimintoja vahingossa, mistä johtuen toimintojen peruutus- ja palautuspainikkeiden tulee olla helposti havaittavissa. Lisäksi peruutuksen tai palautuksen lähtötilanteeseen tulee onnistua nopeasti ilman erillisiä vaiheita.

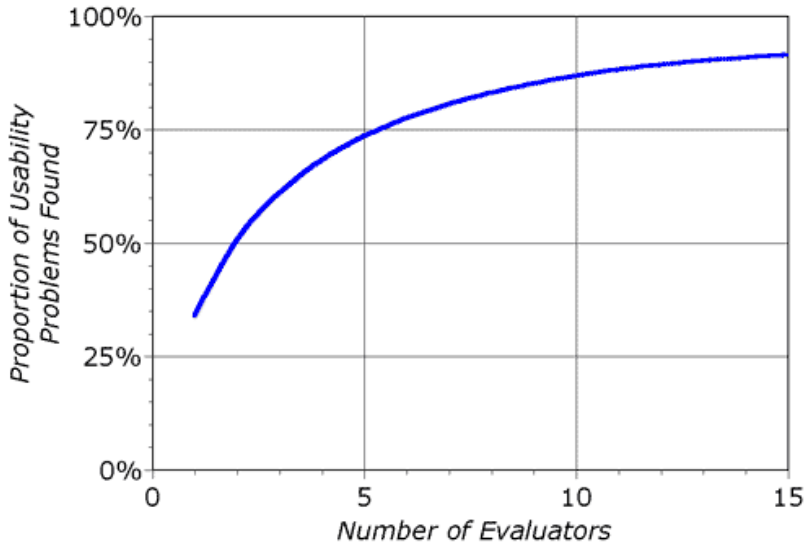
- **Käytön tehokkuus:** Tuotteen käytön tulee olla tehokasta ja joustavaa sekä uusille että kokeneemmille käyttäjille.

- **Virheilmoitukset:** Virheet tulee ilmoittaa selkeällä kielellä eikä esimerkiksi koodirivinä tai numerona. Lisäksi virhe on yksilöitävä selvästi ja siihen tulee ehdottaa ratkaisua.

- **Virhetilanteiden ehkäisy:** Mahdolliset vierhetilanteet on otettava huomioon ja ehkäistävä jo suunnittelussa. On muistettava, että kunnollinen käyttäjän ohjaus ja opastus ehkäisee virhetilanteita.

- **Ohjeet ja opastus:** Ohjeiden tulee olla selkeitä, tarpeeksi lyhyitä, helposti saatavilla sekä käyttäjän toimintaa opastavia ja tukevia.

Heuristista arviointia suoritettaessa on huomioitava, että yksittäinen arvioija löytää vain noin 35 % tuotteen käyttöliittymän kaikista virheistä ja ongelmista. Lisäämällä arvioijien määrää ja vertailemalla heidän tuloksiaan on mahdollista saavuttaa parempi ja tehokkaampi arvio käyttöliittymästä. Seuraavassa kuviossa on selkeästi nähtävillä arvioijien määrän ja löydettyjen käytettävyysongelmien välinen suhde. (Nielsen 1993, 156.)



KUVIO 6. Käytettävyysarvioijien määrän vaikutus löydettyihin virheisiin (Nielsen 1993, 156)

On suositeltavaa käyttää noin viittä arvioijaa ja ainakin kolmea, jotta tuotteen käytettävyys olisi hyvällä tasolla. Oikea arvioija määrä riippuu kustannus-hyötyanalyysistä ja tapauksissa, joissa tuotteen käytettävyys on kriittisessä tai isossa roolissa, on arvioijia tietysti lisättävä. (Nielsen 1993, 156.)

4.4 RIA-sovellusten käytettävyys

RIA-sovelluksia markkinoidaan web-sovelluksina, joilla on työpöytäsovellusten hyvä ja tehokas käytettävyys. Käytettävyyden kannalta suurin ero RIA-sovellusten ja edellisen sukupolven sovellusten välillä on käyttäjän vuorovaikutus sovelluksen kanssa. Vanhoilla HTML:ään rajoittuneilla sovelluksilla vuorovaikutus sovelluksen kanssa jäi perinteisiin elementteihin, painikkeisiin, valintaruutuihin ja valikoihin, mikä rajoitti huomattavasti sovellusten käytettävyyttä ja ulkoasua. Lisäksi useiden sovellusten web-versiot olivat huomattavasti kömpelömpiä ja vaikeakäyttöisempiä kuin vastaavan sovelluksen työpöytäversiot. RIA-sovellusten mukana tulee runsaampi valikoima erilaisia elementtejä parantamaan sovelluksen käyttöliittymää. Tämä lisä mahdollistaa kattavammat virheilmoitukset, tuo paremmin esille sovelluksen tilan sen eri vaiheissa sekä mahdollistaa kaiken kaikkiaan paremman käyttökokemuksen. (Maurer 2006.)

RIA-sovellukset ovat erittäin dynaamisia ja reagoivat huomattavasti nopeammin käyttäjän komentoihin kuin perinteiset HTML-pohjaiset sivustot. Tämä on mahdollista

siten, että sovellus kerää ainoastaan muuttuneen tiedon eikä koko sivua tarvitse ladata uudestaan kuten ennen. Uudistus vähentää tietoliikenteen määrää käyttäjän ja palvelimen välillä, mikä tekee sovelluksesta nopeamman, tehokkaamman ja käyttäjäystävällisemmän. (Van Elten 2008.)

RIA-sovellusten käytettävyyttä ja käyttökokemuksia suunniteltaessa on hyvä ottaa huomioon RIA-teknologioiden tarjoamat mahdollisuudet. Käyttäjälle kannattaa luoda mukava ja tehokas käyntikokemus antamalla hänen vaikuttaa sivun ulkoasuun ja tiedon esitykseen helposti klikkaamalla tai muuten kursoria hyväksi käyttäen. Tällaisen mahdollisuuden tarjoaa esimerkiksi RIA-sovellusten kaaviot ja muu grafiikka, jotka ovat vertaansa vailla tämän päivän web-sovellustenkehityksessä. Tärkeää on kuitenkin pitää sivusto selkeänä ja toiminnallisena. (Van Elten 2008.)

Sivuston selkeys on myös erittäin tärkeä osa käytettävyyttä. RIA-sovelluksissa suurenkin tieto määrän hallinta sujuu kätevästi ja vaivattomasti. Käyttäjälle esitettävä tieto on pidettävä mahdollisimman selkeänä ja helppolukuisena. Kaiken tiedon ei tarvitse olla näkyvillä samanaikaisesti, mutta tiedon saannin on oltava helppoa, korkeintaan klikkauksen päässä. Sivuston ohjeiden tulee olla helposti saatavilla sovelluksen syvimmistäkin lokeroista ja niiden tulee olla ns. käyttäjän kieltä eli ei liian teknistä tai ylimääräistä sanastoa. (Van Elten 2008.)

Yhteenvetona voidaan todeta, että RIA-sovellusten käytettävyydessä on ensinnäkin otettava huomioon, ketkä sovellusta tulevat käyttämään. Hyvässä sovelluksessa otetaan huomioon seikkailunhaluiset, mutta myös tavalliset käyttäjät, jotka eivät ole niin herkkiä testaamaan sovelluksen uusia ominaisuuksia. Tästä esimerkkinä kuviossa 7 kuvattu liukusäädin, jonka avulla voidaan kuviossa esitettyyn kysymykseen vastata, mutta käyttäjille on annettu mahdollisuus myös syöttää itse vastaus vanhaan malliin numeerisella valikolla kuvion oikeassa reunassa. (Van Elten 2008.)

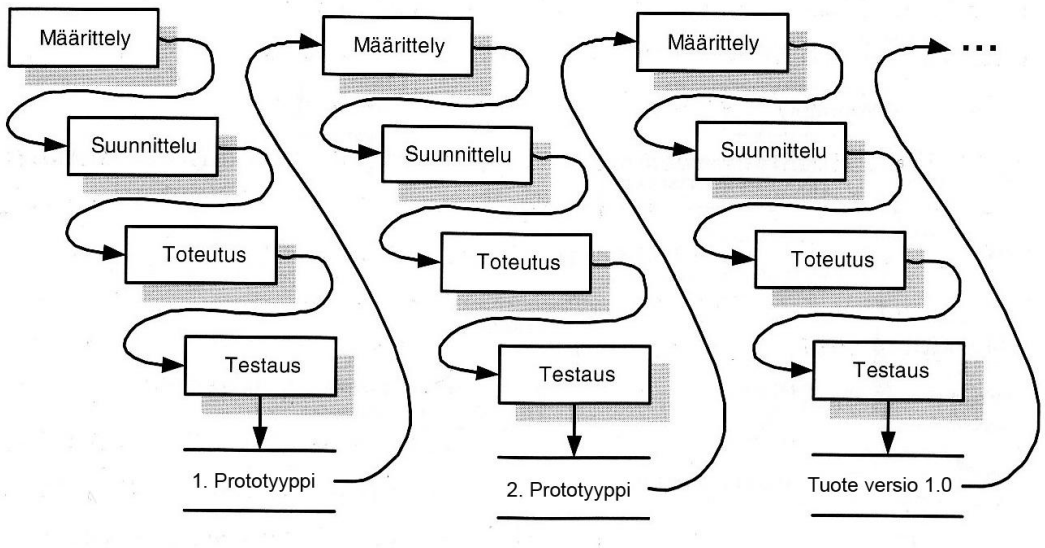
KUVIO 7. RIA-käytettävyys esimerkki - Adobe Flex

Vastaavat ratkaisut on otettava huomioon, jos sovelluksen tulevat käyttäjät ovat esimerkiksi vanhuksia tai kokemattomia tietotekniikan saralla. Lisäksi on tärkeää pitää RIA-sovellusten käyttäjä aina tilanteen tasalla. Kuviossa 7 on yksinkertainen esimerkki myös tästä, sillä liukusäädintä käytettäessä se ilmoittaa käyttäjälle tilanteen, eikä käyttäjän tarvitse nostaa hiirenpainiketta ja odottaa valitun luvun ilmestymistä numeeriseen valikkoon kuvion oikeaan reunaan. Erinäisillä RIA-kehitystyökaluilla saadaan helposti ulkoasultaan näyttävän näköinen sovellus, mutta varsinainen käytettävyys saattaa suunnittelijoilta unohtua. Tästä syystä myös perusteellisella testauksella on suuri merkitys toimivan sovelluksen kehityksessä.
(Van Elten 2008.)

5 PROJEKTIN TOTEUTUS

Projekti määritellään usein kertaluonteiseksi tehtäväksi, jolla on tietyt resurssit ja tavoitteet sekä organisaatio ja jonka toteutus tapahtuu suunnitellusti ennalta laaditun aikataulun mukaisesti. (Haikala & Märijärvi 2004, 225). Projektilla on aina alku ja loppu sekä selkeä tavoite. Tavoitteen saavuttamiseksi laaditaan projektisuunnitelma ja valitaan projektin toteutusmalli.

Projektin toteutusmalliksi valittiin prototyyppimalli, sillä toimeksiantaja antoi aikalailla ”vapaat kädet” asiakashallintajärjestelmän toteuttamiseksi. Tavoitteena oli luoda ensimmäisten vaatimusten pohjalta järjestelmän prototyyppi, jota testaamalla määriteltiin tarkemmat vaatimukset järjestelmälle.



KUVIO 8. Esimerkki prototyyppimallista (Haikala & Märijärvi 2004, 44)

Valmistuneen prototyypin jälkeen projektiryhmällä on yleensä kaksi vaihtoehtoa: joko prototyypin perusteella määritellään toteutettava järjestelmä, joka luodaan alusta alkaen uudelleen, tai prototyyppi kehitetään valmiiksi järjestelmäksi. Myös välimuodot ovat mahdollisia, kuten esimerkiksi kuviossa 8, jossa ensimmäinen prototyyppi ei vastannut vaatimuksia, mistä syystä luotiin toinen, josta sitten kehitettiin valmis tuote. (Haikala & Märijärvi 2004, 42–43.)

Käytännössä prototyyppi voi olla jo toiminnoiltaan toimiva sovellus, josta puuttuvat esimerkiksi virhetarkistukset erinäisistä syöte kentistä, opastukset tai viimeistelty tietokanta. Prototyyppimalli on osoittautunut käteväksi varsinkin käyttöliittymäsuunnittelussa ja -toteutuksessa, koska sillä voidaan varmistaa, että järjestelmästä löytyvät kaikki tarvittavat toiminnot ja että käyttöliittymä miellyttää asiakasta. Lisäksi sillä voidaan testata järjestelmän suorituskykyä.
(Haikala & Märijärvi 2004, 42–43.)

Prototyyppimallin ongelmaksi saattaa muodostua se, että asiakas luulee oikealta näyttävän järjestelmän olevan jo valmis, vaikka käytännössä suuri osa työstä on vielä tekemättä. Tästä syystä prototyypistä ei tehdä vielä valmiin näköistä ja kuvat sekä muu viimeistely jätetään viimeiseksi, jotta asiakas näkee työn keskeneräisyyden. Ongelmaksi saattaa usein myös muodostua juuttuminen silmukkaan, jossa prototyyppiä parannellaan loputtomasti. Tämä ongelma kannattaa ratkaista jo projektisopimuksessa, jossa varataan tietty määrä työviikkoja prototyypin kehittämiseen. (Haikala & Märijärvi 2004, 43–44.)

5.1 Suunnitteluvaihe

> *Suunnittelun tarkoituksena on muuntaa asiakkaan tarpeiden mukaan tehty määrittely tekniselle kielelle – järjestelmän toteutuksen kuvauksesi* (Haikala & Märijärvi 2004, 81).

Suunnitteluvaiheessa tarkastellaan projektin lähtötilannetta, tavoitteita ja keinoja tavoitteiden saavuttamiseksi. Lähtötilannetta ja tavoitteita kartoitetaan haastattelemalla ja havainnoimalla sovelluksen tulevia käyttäjiä. Havaittuja seikkoja laitetaan paperille, joista myöhemmin suodatetaan esiin tärkeimmät seikat esitutkimuksen muodossa. Esitutkimuksen perusteella arvioidaan, kannattaako kyseistä projektia toteuttaa. Esitutkimuksessamme järjestelmä arvioitiin tarpeelliseksi ja toteuttamiskelpoiseksi, joten projekti päätettiin yhdessä asiakkaan ja projektiryhmän kanssa toteuttaa.

5.1.1 Vaatimusmäärittely

Vaatimusmäärittely on yksi ohjelmiston toteutuksen kulmakivistä ja sen tarkoitus on kuvata sovelluksen sisältämät asiat ja toiminnot. Vaatimukset jaetaan kahteen eri kategoriaan: toiminnallisiin ja ei-toiminnallisiin vaatimuksiin. Toiminnalliset vaatimuk-

set määrittävät, mitä käyttäjä voi tehdä järjestelmällä ja miten järjestelmä reagoi käyttäjän tekemisiin. Ei-toiminnalliset vaatimukset kuvaavat, kuinka ohjelma tekee sen, mitä sen odotetaan tekevän.

Vaatimusmäärittely (liite 1) tehtiin yhdessä toimeksiantajan kanssa ja sitä muokattiin myöhemmin, kun aikaisempiin vaatimuksiin tuli uusia vaatimuksia tai korjauksia. Projektissamme ei-toiminnalliset vaatimukset ovat laadullisia vaatimuksia ja ne määrittävät, miten ohjelma tekee sen, mitä sen odotetaan tekevän. Vaatimukset määriteltiin joko pakollisiksi tai valinnaisiksi sen mukaan, mikä oli niiden prioriteetti. Suurin osa vaatimuksista oli pakollisia ja ne toteutettiin kokonaisuudessaan, mutta mukana oli myös muutama valinnainen vaatimus, jotka jäivät aikataulun takia toteuttamatta.

Vaatimusmäärittelyä voidaan käyttää hyväksi myös sovellusta testattaessa. Kun on toteutettu selkeät, yhdessä sovitut ja hyväksytyt vaatimukset, on helppo suunnitella ja toteuttaa testaus niin, että jokaisen vaatimuksen toiminta tarkistetaan myös käytännössä.

5.1.2 Käyttötapaukset

Käyttötapauksilla kuvataan järjestelmän toiminnallisuus, kun käyttäjä suorittaa järjestelmällä jonkin tietyn tapahtumaketjun. Jokaiseen käyttötapaukseen liittyy yksi tai useampia käyttäjäryhmiä, jotka suorittavat käyttötapauksen. Järjestelmän tärkeimmät käyttötapaukset kuvataan käyttötapauskaaviossa (liite 2). Käyttötapaus alkaa aina jonkin käyttäjäryhmän aloitteesta ja päättyy siihen, että käyttäjä on saanut jonkin tehtäväkokonaisuuden suoritettua. Hyvän käyttötapauksen tunnusominaisuuksia ovat:

- **Ymmärrettävyys:** käyttötapausten on oltava asiakkaan ja järjestelmän tulevien käyttäjien ymmärrettävissä.

- **Kuvaa asiakasvaatimuksia:** käyttötapauksia laadittaessa vältetään ottamasta tarpeettomasti kantaa järjestelmän toteutukseen.

- **Testattavuus:** käyttötapaukset luovat pohjan järjestelmän testaukselle. Tästä syystä on muodostettava kokonaisuus, joka voidaan ajaa testausvaiheessa yhtenä testitapauksena.

- **Koko ja tarkkuus:** käyttötapaus ei saa olla liian laaja. Käyttötapauksista kuvataan vain tärkeimmät, eikä kaikkia yksityiskohtia voida ottaa mukaan.

Käyttötapausten tärkein tarkoitus on toimia kommunikointivälineenä kartoitettaessa asiakasvaatimuksia ja kuvailtaessa niitä ohjelmistovaatimuksiksi.
(Haikala & Märijärvi 2004, 157–162.)

5.1.3 Käyttöliittymäsuunnittelu

Käyttöliittymäsuunnitelman (liite 3) perustana toimivat vaatimusmäärittely ja käyttötapaukset, ja niiden pohjalta rakennetaan myös järjestelmän ulkoasu. Järjestelmän ulkoasun suunnittelun lisäksi tavoitteena on selventää, miten ja millä komponenteilla järjestelmän toiminnot ovat toteutettavissa. Käyttöliittymäsuunnittelussa tärkeää oli ottaa huomioon jo aiemmin käytettävyysluvussa esitetyt hyvän käyttöliittymän ominaisuudet. Käyttöliittymäsuunnitelmaan valittiin esitystavaksi visuaalinen tapa, sillä se kertoo aiheesta tietämättömälle lukijalle enemmän kuin dialogikaavio tai yksityiskohtaiset käyttöliittymän operaatiot. Lisäksi suunnitelmassa esitetään ainoastaan järjestelmän tärkeimmät käyttöliittymänäkymät.

Kirjautumissivu päätettiin pitää sisällöltään varsin yksinkertaisena, sillä sen tarkoitus ei ole myydä tai mainostaa itseään kenellekään. Kirjautumisen tarkistus on toteutettu siten, että vaikka käyttäjä yrittää navigoida sisään jollekin järjestelmän sivulle ilman kirjautumista, ohjataan hänet poikkeuksetta takaisin kirjautumissivulle. Onnistuneen kirjautumisen jälkeen käyttäjä ohjataan automaattisesti järjestelmän etusivulle eli muistiinpanotyökaluun.

Muistiinpanotyökalu pidettiin toiminnoiltaan hyvin yksinkertaisena eli käytännössä muokattavana tekstikenttänä, joka tallentuu tietokantaan. Varsinaiset muistiinpanokentät jaettiin yhteisiin sekä molempien pääkäyttäjien omiin muistiinpanokenttiin. Yhteiset muistiinpanot näkyvät kaikille eritasoisille käyttäjille ja heillä kaikilla on oikeus muokata niitä haluamallaan tavalla. Sen sijaan pääkäyttäjien omat muistiinpanot näkyvät vain heille itselleen. Muistiinpanokentät on myös mahdollista tulostaa.

Yksityis- ja yritysasiakkaiden taulukkonäkymäsivuilla on paljon tietoja ja erinäisiä asiakashallintaan liittyviä toiminnallisuuksia. Tästä syystä kaikkien eri komponenttien

määrä haluttiin pitää mahdollisimman vähäisenä. Itse taulukkoa voidaan drag & drop -tyylisesti järjestellä halutulla tavalla, vaihtamalla esimerkiksi sarakkeiden paikkaa tai leveyttä. Lisäksi käyttäjä voi klikkaamalla sarakkeen otsikkoa järjestää sarakkeen tiedot pienimmästä suurimpaan tai aakkosjärjestykseen.

Asiakkaita lisätään Lisää-painikkeen takaa aukeavassa näkymässä. Näkymä sisältää joko yksityis- tai yritysasiakkaan lisäys-lomakkeen riippuen siitä, kummalla välilehdellä Lisää-painiketta painettiin. Lomakkeeseen syötetään halutut asiakastiedot ja painetaan Tallenna-painiketta. Tässä yhteydessä järjestelmä ilmoittaa, mikäli jokin pakolliseksi määritelty kenttä on jäänyt tyhjäksi. Kentän pakollisuus ilmoitetaan punaisella värillä kentän ympärillä, mikäli sen jättää tyhjäksi ja siirtyy syöttämään merkkejä seuraavaan kenttään. Lisäksi Kuvaus-kentässä on merkkilaskuri kertomassa käyttäjälle jäljellä olevien merkkien määrän, sillä kenttä on rajattu 1000 merkkiin.

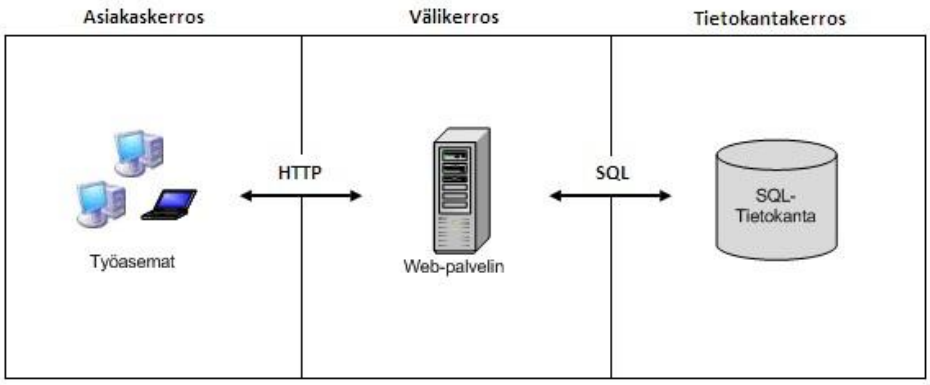
Esimerkkinä toimintojen yksinkertaistamisesta on asiakkaiden muokkaus, joka saatiin tuplaklikkaus-toiminnolla selkeästi helppokäyttöisemmäksi, kuin että käyttäjän tulisi ensin valita muokattava asiakas klikkaamalla, minkä jälkeen olisi klikattava erillistä Muokkaa-painiketta toisella puolella sivua. Tämä toimenpide selkeytti myös käyttöliittymää poistamalla yhden painikkeen käyttäjän havaintoalueelta. Tuplaklikkaus halutun asiakkaan rivillä avaa tiedot muokkausnäkymään, jossa käyttäjä voi muokata haluamiaan tietoja. Muokkauksen jälkeen käyttäjän painaessa Tallenna-painiketta suoritetaan samanlainen kenttien tarkistus kuin asiakkaita lisättäessä. Asiakastiedon poisto tulee aina vahvistaa erillisessä ikkunassa, jottei käyttäjä pysty poistamaan vahingossa tietoja. Lisäksi alemman tason käyttäjät eivät saa poistaa asiakastietoja, sillä heille Poisto-painike ei näy, kuten vaatimuksissa määriteltiin.

Järjestelmän tulosteista pyritään tekemään mahdollisimman helppolukuisia. Taulukkonäkymien tulosteeseen valittiin vain toimeksiantajan tarpeelliseksi kokemat sarakkeet, mikä tuo tulosteeseen selkeyttä. Tulosteista jätetään pois kaikki siihen kuulumaton, kuten sivun painonapit, taustakuva ja navigointivalikot. Yksittäisen asiakastiedon tulostus liitettiin muokkausnäkymään. Myös tässä jätettiin tulosteesta kaikki turha pois ja keskityttiin luomaan selkeä ja helposti luettava tuloste.

5.1.4 Arkkitehtuuri

Ohjelmiston arkkitehtuurilla kuvataan toteutettavan ohjelmiston erinäisiä rakenneosia ja näiden osien välisiä suhteita. Ohjelmiston suunnittelun aikaista rakennetta tärkeämpi seikka on kuitenkin ohjelmiston toteutusfilosofia. Ohjelmiston arkkitehtuuri kannattaa siis pelkkien rakenneosien ja näiden suhteiden lisäksi ymmärtää laajemmin. Arkkitehtuuri antaa yhteisen pohjan kaikelle järjestelmään liittyvälle suunnittelu- ja toteutustyölle. Järjestelmänkehittäjä saa arkkitehtuurin toteutusfilosofiasta mallin siitä, miten järjestelmä toteutetaan. Usein hyvälle arkkitehtuurille on ominaista se, että jos jotain asiaa ei tiedä, se on pääteltävissä toteutusfilosofian perusteella.
(Haikala & Märijärvi 2004, 317.)

Järjestelmän toteutusfilosofia perustuu kolmikerrosmalliin, joka on yleisin tietokantapohjaisten web-sovellusten suunnitteluun käytetty malli. Käytännössä web-selaimen ja tietokannan väliin luodaan välikerros, joka toimii asiakkaana tietokantapalvelimelle ja palvelimena www-selaimelle. Fyysisesti tietokanta voi sijaita web-palvelimen kanssa samassa palvelinkoneessa tai eri palvelimilla. Projektissa välikerroksen toiminnallisuus toteutetaan PHP-kielellä.



KUVIO 9. Kolmikerrosmallin esimerkki

Eri työasemien selaimet ja web-palvelin kommunikoivat HTTP-protokollalla, joka tarkoittaa lyhyesti sitä, että selain avaa TCP-yhdeyden web-palvelimelle ja lähettää pyynnön. Palvelin vastaa pyyntöön lähettämällä vastauksena esimerkiksi HTML-sivun. (Verkkopalveluarkkitehtuuri 2006.)

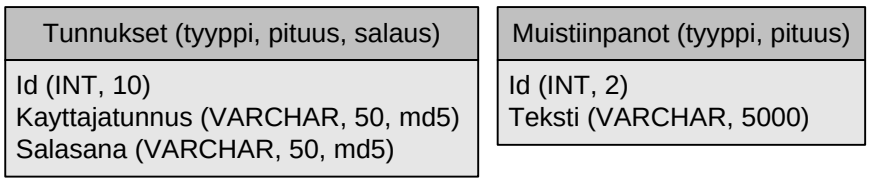
Toimiakseen järjestelmä vaatii Web-palvelimelta PHP-tulkin, joka käyttää tietokantapalvelinta asiakasohjelman roolissa funktiokirjaston avulla. PHP on varsin yleisesti käytetty ohjelmointikieli tietokantapohjaisten web-sivujen luontiin, sillä se tukee yleisimpien tietokantapalvelimien omia protokollia. (Verkkopalveluarkkitehtuuri 2006.)

Tietokantarakenne

Tietokantarakenne päätettiin pitää mahdollisimman yksinkertaisena, sillä kyseessä on suhteellisen pieni järjestelmä, jossa ei ole tarvetta monimutkaiselle tietokannalle. Yksinkertaisuus mahdollistaa keskittymisen järjestelmän kannalta olennaisiin seikkoihin, kuten tietojen ylläpitoon ja varmuuskopiointiin. Lisäksi se mahdollistaa paremman lähtökohdan järjestelmän kehittämiselle.

Tietokantaan luotiin neljä erillistä tietokantataulua, joilla ei ole keskenään minkäänlaisia suhteita. Käyttäjätunnukset ja salasanat sijoitettiin Tunnukset-tauluun ja suojattiin md5-salauksella. Käyttäjätunnuksen ja salasanan maksimipituus määriteltiin 50 merkkiin ja tyypiksi valittiin varchar, mikä tarkoittaa vaihtuvan mittaista merkkijonoa, joka sallii myös numeeriset sekä erikoismerkit ja näin ollen sopii erittäin hyvin tähän tarkoitukseen.

Id-sarakkeen tarkoitus on yksilöidä käyttäjätunnus ja salasana tietylle käyttäjälle. Sen tyypiksi valittiin INT eli INTEGER, joka tarkoittaa numeerista arvoa ja pituudeksi maksimissaan kymmenen merkkiä.



KUVIO 10. Tietokantarakenne, osa1

Muistiinpanot-tauluun riittää sarakkeiksi Id ja Teksti. Muistiinpanoja on käytössä kolmea tyyppiä eli molempien pääkäyttäjien omat muistiinpanot sekä yhteiset muistiinpanot. Näin ollen Id-sarakkeen pituus voidaan rajata kahteen merkkiin ja jättää tilaa mahdollisille uusille käyttäjille. Tekstiä muistiinpanoihin mahtuu maksimissaan 5000 merkkiä, sillä tarkoitus on pitää muistiinpanot lyhyinä ja ytimekkäinä.

Varsinaiset asiakastiedot jaettiin yksityis- ja yritysasiakkaisiin. Molemmille asiakastyypeille päätettiin rakentaa omat tietokantataulut, jotta asiakastiedot pysyvät selkeinä ja tietokantarakenne helposti muokattavana tai kehitettävänä.

Yksityiset (tyyppi, pituus)
Asiakasnumero (INT, 10)
Nimi (VARCHAR, 50)
Osoite (VACHAR, 150)
Puhelinnumero (VARCHAR, 50)
Email (VARCHAR, 50)
Kuvaus (VARCHAR, 1000)
Myynti (VARCHAR, 120)
Vastuuhenkilo (VARCHAR, 30)

Yritykset (tyypi, pituus)
Asiakasnumero (INT, 10)
Yritys (VARCHAR, 100)
Ytunnus (VARCHAR, 50)
Yhteyshenkilo (VARCHAR, 50)
Osoite (VARCHAR, 150)
Laskutusosoite (VARCHAR, 150)
Puhelinnumero (VARCHAR, 50)
Email (VARCHAR, 50)
WWW (VARCHAR, 50)
Toimiala (VARCHAR, 50)
Kuvaus (VARCHAR, 1000)
Myynti (VARCHAR, 120)
Vastuuhenkilo (VARCHAR, 50)

KUVIO 111. Tietokantarakenne, osa2

Asiakkaista kerättävät tiedot määriteltiin yhdessä toimeksiantajan kanssa. Asiakasnumero yksilöi asiakkaan ja lisäksi yksityis- sekä yritysasiakkaat erotellaan toimeksiantajan toiveiden mukaisesti toisistaan lisäämällä yksityisasiakkaan asiakasnumeroon A-kirjain ja yritysasiakkaan asiakasnumeroon B-kirjain, esimerkiksi A1001 tai B1001. Myös kenttiin mahtuvien merkkien maksimipituudet määriteltiin yhdessä toimeksiantajan kanssa.

5.2 Toteutusvaihe

Varsinaiseksi sovelluksen kehitysympäristöksi valittiin Adobe Flex, koska se on Suomessa toistaiseksi suhteellisen vähän käytetty sovelluskehitysratkaisu. Lisäksi uskomme Flex:n olevan ennen kaikkea tulevaisuuden sovelluskehitysmenetelmä. Flex:n lisäominaisuudet ja kehitysmahdollisuudet ovat tällä hetkellä vertaansa vailla muiden sovelluskehitysratkaisuiden joukossa. Toteutuskieliksi valikoitui Flex:n käyttämät MXML- ja ActionScript 3 -kielet sekä PHP- ja SQL-kielet.

PHP valittiin, koska se on projektityhmälle entuudestaan tuttu ja hyvin järjestelmän tarpeisiin sopiva ohjelmointikieli. Lisäksi se toimii hyvin yhteen Adobe Flex:n kanssa. Tietokannaksi valittiin MySQL-tietokanta, sillä se tukee kattavasti PHP-

ohjelmointikieltä. Useimmilta palvelimilta, joilta löytyy PHP-tuki, löytyy myös MySQL-tuki, varsinkin Suomessa, sillä MySQL on Suomessa ja Ruotsissa kehitetty SQL-tietokannan hallintajärjestelmä.

5.2.1 Toteutusympäristö

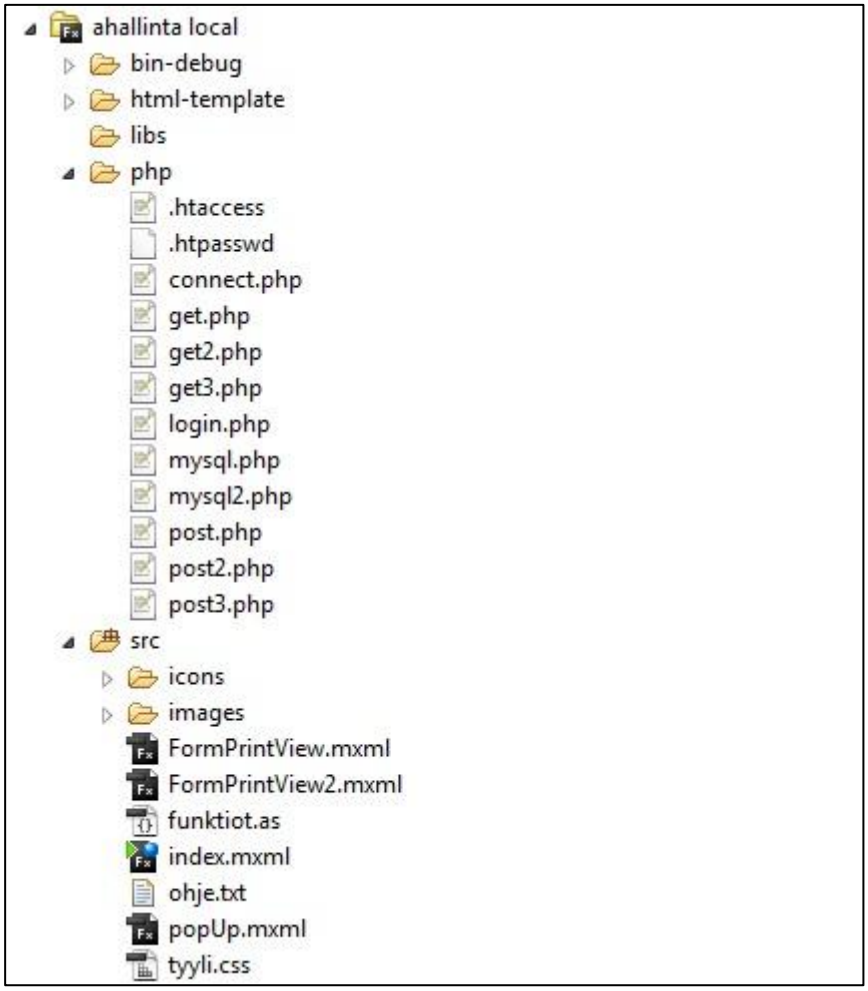
Projektiryhmällä oli käytössä molempien omat tietokoneet. Toisessa oli käyttöjärjestelmänä Windows 7 Ultimate, web-palvelimena WAMP-server 2.0i ja toisessa Windows XP Pro, web-palvelimena myös WAMP-server 2.0i. WAMP-server 2.0i:n mukana tulevat myös Apache 2.2.11 -, PHP 5.3.0 -, MySQL 5.1.36 - sekä Phpmyadmin-ohjelmat, joita myös käytettiin järjestelmän toteutuksessa ja testauksessa.

Järjestelmän prototyyppejä esiteltiin toimeksiantajalle palavereissa sekä käyttämällä hyväksi Jyväskylän ammattikorkeakoulun Batman-palvelinta, jolle prototyypit perustettiin. Näin ollen toimeksiantaja saattoi testata järjestelmää milloin ja mistä vain sekä tuoda esiin oman näkökulmansa järjestelmän kehitykseen.

5.2.2 Koodaus

Järjestelmän toteutusta eli koodausta ja kooditiedostojen sisältöä ei käydä kovinkaan tarkasti läpi, sillä emme näe järkevänä esitellä useita sivuja pitkiä ja useimmille asiasta tietämättömille mitään sanomattomia merkkijonoja. Koodausvaiheessa järjestelmää lähdettiin toteuttamaan ennalta suunniteltujen toimintojen ja ulkoasun mukaiseksi.

Järjestelmän perusta luotiin Adobe Flex Buider 3:lla ja sen käyttämillä MXML- ja ActionScript 3 -ohjelmointikielillä. MXML-ohjelmointikielellä luotiin pääasiassa järjestelmän rakenne ja ulkoasu. ActionScript 3 -kielellä lisättiin järjestelmään toiminnallisuudet. MySQL-tietokantaa käyttävissä toiminnallisuuksissa käytettiin lisäksi PHP-ohjelmointikieltä, jolla kommunikoitiin yhdessä MySQL-tietokannan kanssa. Kommunikaatio tapahtui eri tarkoituksiin yksilöidyillä SQL-kyselyillä. Seuraavassa kuviossa on esitetty järjestelmän kehityksen aikainen tiedostojen hakemistorakenne.



KUVIO 12. Kooditiedostot

Ahallinta local toimi järjestelmän työnimenä Adobe Flex Builder 3:ssa, jonka juureen järjestelmän muut tiedostot sijoitettiin. Aloitettaessa Flex-projekti Flex Builder 3 luo automaattisesti Bin-debug-, html-template-, libs- ja src-kansiot. Bin-debug-kansio sisältää automaattisesti luotuja tiedostoja, joita tarvitaan web-sovelluksen toteuttamiseen. Html-template-kansio sisältää automaattisesti luodun sovelluksen oletus sivupohjan, johon tämän järjestelmän puitteissa ei kajottu. Libs-kansio sisältää mahdolliset järjestelmän käyttämät ulkopuoliset funktiokirjastot.

Php-kansio suojattiin ulkopuolisilta muusta järjestelmästä erillisellä käyttäjätunnuksella ja salasanalla käyttäen .htaccess- ja .htpasswd-tiedostoja. Varsinaiset php-tiedostot päätettiin pitää toiminnallisuuksiltaan eri tiedostoissa, jotta järjestelmän jatkokehitys ja käyttöönotto sujuisivat mahdollisimman vaivattomasti. Connect.php luo yhteyden MySQL-tietokantaan, jota kaikki muut PHP-tiedostot käyttävät. Kirjauduttaessa sisään järjestelmään login.php tarkistaa käyttäjän syöttämien käyttäjätunnuksen ja sa-

lasanan oikeellisuuden tietokannasta. Mysql.php ja mysql2.php sisältävät varsinaisen asiakashallinnan toiminnot eli asiakkaiden lisäyksen ja muokkauksen tietokantaan sekä hakemisen ja poiston tietokannasta. Kaikki get- ja post-alkuiset PHP-tiedostot liittyvät muistiinpanotyökaluun. Get.php-tiedostot hakevat tekstieditoriin eli muistiinpanotyökaluun tietokannasta tietyn käyttäjän muistiinpanot. Post.php-tiedostot tarkistavat syötetyn tekstin oikeellisuuden ja tallentavat muutokset tietokantaan.

Src-kansiossa sijaitsee varsinainen Flex Builderilla tuotettu koodi sekä järjestelmän käyttämät kuvat ja ikonit (icons). Ikoneja käytettiin järjestelmän eri painikkeissa selkeyttämään painikkeiden toimintoja niitä kuvaavilla ja käyttäjälle entuudestaan tutuilla symboleilla. Käytössä oli ainoastaan taustakuva, joka säilöttiin images-kansioon.

FormPrintView.mxml ja FormPrintView2.mxml ovat kaikkien asiakkaiden tulostuksessa käytettyjä Flex-komponentteja, joihin kerättiin taulukkomuotoon kaikista asiakkaista tulostettavat tietueet ja valmisteltiin ne toimeksiantajalle räätälöityä tulostetta varten. Funktiot.as sisältää päätteensä mukaisesti kaikki järjestelmän käyttämät ActionScript 3 -kielellä ohjelmoidut toiminnot eli funktiot. Index.mxml-tiedosto sisältää MXML-kielellä toteutetun järjestelmän rungon eli rakenteen ja ulkoasun. Ohje.txt sisältää järjestelmän käyttöohjeet, jotka popup.mxml avaa nimensä mukaisesti popup-ikkunaan. Tyyli.css on järjestelmän tyylitiedosto, jossa määritellään esimerkiksi järjestelmän taustakuva.

Adobe Flex oli ennen projektia kehittäjille ennestään varsin tuntematon sovelluskehitysratkaisu, joten näin ollen myöskään koodausta ei voitu suorittaa niin sanotusti vanhasta muistista. Ohjeina käytettiin internetistä löytyviä niin Adoben kuin muidenkin julkaisijoiden Flex-oppaita ja -foorumeja. Toteutuksen yhteydessä valmistunut koodi kommentoitiin perusteellisesti, jotta järjestelmän mahdollinen jatkokehittäminen tai muu muutostyö olisi mahdollisimman helppoa, olivatpa jatkokehittäjinä sitten alkuperäiset tai täysin uudet kehittäjät.

5.2.3 Tietoturva

Tietoturvan huomioiminen on erittäin tärkeää web-sovelluksen kannalta. Tietoturva on otettu huomioon jo sovelluksen suunnittelussa, ja sitä on myös pyritty kehittämään

ja testaamaan projektin aikana. Projektin tietoturvan huomiointi oli tärkeää myös siksi, että järjestelmä tulisi sisältämään mahdollisesti yritykselle luottamuksellista tietoa.

Tietoturva huomioitiin muun muassa salaamalla käyttäjätunnukset ja salasanat tietokantaan md5-salausmenetelmää käyttäen. Tämä estää tunnusten ja salasanojen tarkastelemisen tietokannasta selkeäkielisenä, vaikka tietokantaan päästäisiin käsiksi.

Lisäksi sovellus ajettiin läpi HP:n kehittämällä SWFScan-ohjelmalla, joka tarkistaa SWF-tiedostojen tietoturvaa ja raportoi mahdollisista ongelmista. Tämä toteutettiin ja löytyneisiin ongelmiin reagoitiin. Flexillä toteutetuissa ohjelmissa voidaan haluttaessa julkaista lähdekoodit SWF-tiedoston mukana. Tämä on hyvä ominaisuus opetus- ja kehitystarkoituksessa, mutta tällaisessa tapauksessa lähdekoodit luonnollisesti poistettiin.

Tietoturvaa parantaaksemme otimme käyttöön myös Apachen hakemistokohtaisen asetustiedoston, htaccessin. Sen avulla hakemistossa olevat tiedostot voidaan muun muassa suojata salasanalla, jota tässä tapauksessa käytettiin. Järjestelmän tietoturvan kannalta oleelliset tiedostot sijoitettiin hakemistoon, jonne pääsy suojattiin erillisellä käyttäjätunnuksella ja salasanalla.

5.3 Testausvaihe

Järjestelmän testaustavaksi valittiin mustalaatikkotestaus (black box, functional testing), jossa testitapaukset valitaan järjestelmän toiminnallisten ja laadullisten vaatimusten perusteella tutustumatta järjestelmän toteutukseen. Tarvittavan testauksen määrää on ennalta hyvin vaikea arvioida, sillä testausta voidaan jatkaa, kunnes rahat tai aika loppuvat. Ohjelmistokehityksessä testauksen lopettaminen on yleensä kompromissi järjestelmässä olevien vikojen aiheuttamien kustannusten ja markkinoilta myöhästymisen aiheuttaman tuoton menetyksen välillä. Testauksen lopettamiselle tulisi asettaa hyväksymiskriteerit testaussuunnitelmassa. Yleisimpiä kriteerejä testauksen lopettamiselle ohjelmistotuotannossa ovat esimerkiksi kumulatiiviseen löydettyjen

virheiden määrä eli, kun löydettyjen virheiden määrä tasaantuu, voidaan testaus lopettaa. (Haikala & Märijärvi 2004, 293.)

5.3.1 Järjestelmätestaus

Järjestelmätestauksessa tarkastellaan koko järjestelmää ja verrataan tuloksia vaatimusmäärittelyyn ja asiakasdokumentaatioon. Järjestelmätestauksen suorittavat järjestelmän varsinaisesta kehitystyöstä mahdollisimman riippumattomat testaajat. Tässä vaiheessa valittujen testitapausten lisäksi testataan myös järjestelmän ei-toiminnalliset vaatimukset (liite 1). (Haikala & Märijärvi 2004, 290–291.)

Projektin aikana tehtiin myös järjestelmätestisuunnitelma, jossa määriteltiin, mitä, miten ja milloin testataan. Testauksen pääkohdat olivat:

- jatkuva testaus projektin läpi
- vaatimusten testaaminen
- tietoturvan testaaminen
- käytettyyden testaaminen
- käyttöönottotestaus.

5.3.2 Käytettävyystestaus

Käytettävyystestauksella pyritään varmistamaan, että käyttäjä selviytyy mahdollisimman hyvin tehtävistä, joiden suorittamiseksi järjestelmää ollaan kehittämässä. Käytettävyystestaus on usein pääasiassa käyttöliittymän testausta, jota tehdään yleensä koko järjestelmän kehityshistorian ajan käyttöliittymäprototyyppien avulla. (Haikala & Märijärvi 2004, 291.)

Käytännössä järjestelmän käytettävyyttä testattiin heuristisella arvioinnilla, jonka suorittivat järjestelmän tulevat pääkäyttäjät ja toinen kehittäjistä. He testasivat järjestelmän prototyyppejä ja kertoivat toiveensa ja havaintonsa kehittäjille. Testauksen jälkeen järjestelmässä havaitut virheet tai puutteet korjattiin ja testi toistettiin. Tällaista

testausta tapahtui koko järjestelmän kehityksen ajan eri prototyypeillä. Testauksen jälkeen järjestelmässä havaitut virheet tai puutteet korjattiin ja testi toistettiin.

Tärkeimpiä käytettävyystestauksessa löytyneitä puutteita olivat muistiinpanokenttien ja myös asiakkaiden kuvauskenttien merkkilaskurit. Kyseiset kentät ovat merkeiltään rajattuja tiettyyn maksimiarvoon, ja ennen testausta käyttäjä ei voinut havaita tämän arvon ylittymistä juuri mitenkään. Ongelmaksi muodostui se, että kenttiin kirjoitettu teksti katkesi ja jäi tallentumatta maksimiarvon ylittävältä osalta. Ongelma ratkaistiin lisäämällä kyseisiin kenttiin merkkilaskuri, joka pitää käyttäjän reaaliajassa tietoisena käytettävissä olevien merkkien määrästä.

Toinen käytettävyystestauksessa havaittu puute olivat järjestelmän ohjeet, joita ei ollut alun perin ollenkaan. Vaikka järjestelmä pyrittiin pitämään mahdollisimman yksinkertaisena ja helppokäyttöisenä, huomattiin pian, että ohjeet on hyvä olla olemassa auttamassa käyttäjää suorittamaan eri toimintoja. Ohjeita laadittaessa otettiin huomioon kielen selkeys, jotta kokemattomammatkin käyttäjät pystyisivät ne ymmärtämään. Ohjeet sijoitettiin näkymään järjestelmän jokaisessa näkymässä, jotta niihin pääsisi mahdollisimman helposti käsiksi syvimmistäkin järjestelmän syövereistä. Esitystavaksi valittiin popup-ikkuna, joka on helppo avata, selata läpi ja sulkea eli on käytettävyydeltään erinomainen tähän tarkoitukseen.

5.4 Käyttöönottovaihe

Alun perin tarkoitus oli toteuttaa ja raportoida myös järjestelmän käyttöönotto yrityksen palvelimella. Yrityksestä johtuen tämän suorittaminen jäi myöhemmin toteutettavaksi, joten siitä raportoiminen ei onnistu tämän opinnäytetyön puitteissa.

Teoriassa käyttöönottovaihe tullaan suorittamaan luomalla yrityksen käyttämälle palvelimelle tarvittava tietokanta ja taulut. Tämän jälkeen siirrettään sovellus palvelimelle ja määritellään tarvittavat asetukset, kuten käyttäjätunnukset ja salasanat. Lisäksi määritellään .htaccess-tiedostoon tarvittavat tunnukset ja salasanat lisäämään sovelluksen tietoturvaa. Kun järjestelmä on saatu toimintaan palvelimelle, se testataan vielä perusteellisesti, jotta voidaan olla varmoja sen toimivuudesta.

5.5 Yhteenveto projektista

Tavoitteena oli suunnitella ja toteuttaa web-pohjainen asiakashallintajärjestelmä auttamaan nuoren yrityksen liiketoimintaa kerättyjen vaatimusten pohjalta. Järjestelmän tuli olla käyttöjärjestelmäriippumaton ja suojassa ulkopuolisilta kirjautumisen takana. Varsinaisella asiakashallinnalla oli pystyttävä lisäämään, muokkaamaan, poistamaan ja tulostamaan asiakastietoja. Järjestelmän tarkoitus oli toimia yrityksen verkkosivujen yhteydessä ja sen odotettiin korvaavan mahdolliset edelliset asiakasrekisterit sekä käyntikortit.

Projektin resurssit

Asiakashallintajärjestelmä-projektiryhmä koostui projektin teknisestä johtajasta Joni Mikkosesta ja projektipäällikkö Mikko Savolaisesta. Asiakkaana toimi vielä rekisteröimätön jyväskyläläisyritys. Projektityön sisältämä raportointi, suunnittelu ja toteutus tapahtuivat tasapuolisesti projektiryhmäläisten kesken. Projektityhmä ja asiakas palaveerasivat kaikille osapuolille soveltuvina ajankohtina vähintään kerran kuussa. Lisäksi yhteydenpitoa tapahtui sähköpostitse sekä puhelimitse, ja myös raportointi asiakkaalle tapahtui pääosin sähköpostitse. Projektiryhmän sisäinen kommunikointi toteutettiin viikkopalavereilla, jotka pidettiin yleensä tiistaisin. Lisäksi projektia suunniteltiin ja toteutettiin yhdessä useita kertoja viikossa.

Projektin kulku

Projekti aloitettiin esitutkimuksella ja vaatimusmäärittelyllä 25.11.2009. Projekti päätettiin toteuttaa prototyyppimallilla, sillä projektiryhmä sai aika lailla ”vapaat kädet” järjestelmän toteuttamiseksi. Aiempaa asiakashallintaa toimeksiantajalla ei ollut, joten toteutettava järjestelmä luotiin niin sanotusti puhtaalta pöydältä. Toteutettavan järjestelmän kehitysympäristöksi valittiin Adobe Flex, joka on Suomessa toistaiseksi vielä varsin tuntematon ja vähän käytetty. Lisäksi uskomme Flex:n olevan tulevaisuudessa suuressa roolissa RIA-teknologioiden menestystä mitattaessa. Varsinaisiksi järjestelmän toteutuskieliksi valittiin Flex:n käyttämät MXML- ja ActionScript 3 -kielet sekä PHP- ja SQL-kielet.

Projektin suunnittelu vastasi hyvin toteutettua järjestelmää, vaikka muutoksia tulikin matkan varrella. Suunnittelu- ja kehitystyön aikana järjestelmän toimintoihin lisättiin muistiinpanotyökalu ja mahdollistettiin uudet, alemmantasoiset käyttäjät järjestelmään. Kaiken kaikkiaan projekti oli onnistunut. Toteutettu järjestelmä täytti toimeksiantajan alkuperäiset ja myöhemmin lisätyt vaatimukset. Projektin aikataulu oli jo alkuvaiheessa, toimeksiantajasta johtuen, varsin avoin, mutta silti sen puitteissa pysyttiin.

Projektin tulokset

Projektin tulokseksi saatiin toimeksiantajan asettamat tärkeimmät vaatimukset täyttävä asiakashallintajärjestelmä ja projektin tuottamat dokumentit, joista tärkeimmät liitettiin osaksi opinnäytetyötä. Toimeksiantajan alkuperäisistä eli tärkeimmistä vaatimuksista saatiin toteutettua sata prosenttia, mutta myöhemmässä vaiheessa esitetyistä vaatimuksista aivan kaikkia ei saatu toteutettua. Toteuttamatta jäivät asiakkaiden suodatus asiakasnumeron perusteella, käyttäjätunnusten hallinnointi ja varsinainen järjestelmän käyttöönotto.

Asiakastietojen suodatus asiakasnumeron perusteella osoittautui oletettua hankalammaksi toteuttaa projektin aikataulun puitteissa. Lisäksi asiakkaita voi suodattaa nimen perusteella ja asiakasnumerot voi järjestää taulukkonäkymässä pienimmästä suurimpaan tai tosin päin, ja näin ollen suodatus asiakasnumeron perusteella päätettiin jättää toteuttamatta.

Käyttäjätunnusten hallinnointi eli lisäys, poisto ja muokkaus järjestelmästä käsin päätettiin jättää toteuttamatta, koska sen toteutus olisi vienyt suhteellisen paljon aikaa projektin aikataulusta. Käyttäjien hallinnointi onnistuu manuaalisesti suoraan tietokantaan. Lisäksi järjestelmä on rakennettu siten, että mahdollisesti lisättävien käyttäjien käyttöoikeustaso on automaattisesti alempi taso.

Kuten jo edellä käyttöönottovaihekappaleessa kerrottiin, ei varsinaista järjestelmän käyttöönottoa ehditty toteuttaa opinnäytetyön puitteissa. Tämä johtui projektiryhmästä riippumattomasta syystä eli siitä, ettei toimeksiantaja ehtinyt varaamaan palvelintilaa projektin ja opinnäytetyön toteutuksen aikana.

Oma arvio

Projektiryhmän ja asiakkaan välinen yhteistyö sujui ongelmitta, kuten muukin projektityö. Asiakas toi hyvin ilmi oman näkökulmansa projektiin, mutta jätti hyvin tilaa myös projektiryhmän omille näkemyksille. Yhteydenpito asiakkaaseen kävi kätevästi sähköpostilla, eikä mielipiteitä tai vastauksia kysymyksiin tarvinnut odotella paria työpäivää pitempään. Varsinainen valmis järjestelmä ylitti projektiryhmän omatkin odotukset, sillä Adobe Flex oli projektiryhmälle entuudestaan tuntematon sovelluskehitysympäristö. Projekti pysyi hyvin ennalta laaditussa aikataulussa ja järjestelmä oli valmis helmikuun puolessa välissä, mikä jätti hyvin aikaa testaukselle, sillä toimeksiantajasta johtuen järjestelmän käyttöönotto venyi myöhempään ajankohtaan.

6 TUTKIMUSTULOSTEN ANALYSOINTI

Tässä luvussa selvitetään opinnäytetyöstä saadut tulokset sekä vastataan tutkimuskysymyksiin ja analysoidaan saadut vastaukset. Saadut vastaukset hyödyttävät toimeksiantajaa, projektiryhmää sekä kaikkia, jotka ovat kiinnostuneita rikkaista internet-sovelluksista ja niiden hyödyntämisestä. Tutkimuksen tarkoitus on avata lukijalle RIA-sovellusten maailmaa ja tuoda esille niiden tarjoamia mahdollisuuksia niin sähköisen liiketoiminnan kuin käytettävyydenkin saralla.

Tuloksina opinnäytetyöstä saatiin tutkimus RIA-sovelluksista ja web-sovellusten käytettävyydestä. RIA-sovellusten tutkimusosio sisältää eri RIA-teknologioiden esittelyn ja vertailun sekä niiden mukanaan tuomat hyödyt ja mahdolliset ongelmat sähköiseen liiketoimintaan. Web-sovellusten käytettävyysosiossa tutkitaan ihmisten toimintaa ja tarpeita web-sovellusten käyttäjinä. Lisäksi opinnäytetyön tulokseksi saatiin asiakashallintajärjestelmä-projektin tuottamat projektidokumentit. Toteutetun järjestelmän tuomista tuloksista toimeksiantajan liiketoiminnan edistämiseksi ei voida vielä puhua, sillä tulokset ovat havaittavissa vasta järjestelmän oltua toiminnassa jo pidemmän ajanjakson.

Tutkimuksella toivotaan saavutettavan tietoa siitä, mitä kaikkea on otettava huomioon lähdettäessä suunnittelemaan rikkaita internet-sovelluksia. Tutkimuksesta on apua, ei pelkästään toimeksiantajalle, vaan kaikille sähköisen liiketoiminnan ja web-pohjaisen asiakashallinnan hyötyjä epäileville.

1. Mitä annettavaa RIA-sovelluksilla on sähköiseen liiketoimintaan?

Yritys voi hyödyntää RIA-sovelluksia kahdelle eri tavalla. Niitä voidaan käyttää tarjoamaan parempi käyttökokemus asiakkaille suunnatuissa sovelluksissa. Hyvä käyttökokemus antaa asiakkaalle paremman kuvan yrityksestä ja näin ollen edistää myös myyntiä. Esimerkiksi verkkokaupoissa voidaan RIA-teknologioita hyödyntäen tarjota kuluttajalle mahdollisuus räätälöidä tuote itselleen vaivattomasti ja päivittää räätälöidyn tuotteen ulkoasu sekä tiedot ilman sivun uudelleenlataamista.

Lisäksi RIA-sovelluksia voidaan hyödyntää yrityksessä sisäisesti tarjoamalla työntekijöille heidän työtään tehostavia sovelluksia. Työntekijät hyötyvät muun muassa sovel-

luksista, jotka tuovat reaaliaikaista dataa yrityksen tietovarannoista näytölle. Reaaliaikaisesti päivittyvät tiedot eri tietokannoista samalla näytöllä voivat olla avuksi muun muassa päätöksenteossa. RIA-sovelluksia on myös nopea kehittää ja helppo jakaa eteenpäin.

2. Mitä tulee ottaa huomioon web-sovelluksen käytettävyyttä suunniteltaessa?

Web-sovelluksen käytettävyyden suunnittelu on hyvä alkaa pohtimalla, ketkä luotavaa sovellusta tulevat käyttämään. Eri ihmiset voivat käyttää sovellusta hyvinkin eri tavoin, riippuen esimerkiksi heidän kokemuksestaan tietotekniikasta, iästä, toiminnan rajoitteista tai vaikkapa koulutuksesta. Tästä syystä sovelluksen tulevien käyttäjien toimintaa tulee havainnoida, jotta tiedetään, mitä he sovellukselta haluavat tai miten he sen kanssa toimivat niin, että sitä voidaan edelleen suunnitella ja kehittää käyttäjien toimintaa tukevaksi. Havainnoinnin perusteella eri tarpeet omaavat käyttäjät on hyvä jakaa eri käyttäjäryhmiin, jotta sovelluksesta voidaan luoda käytettävyydeltään mahdollisimman tehokas, selkeä ja yksilöllinen eri käyttäjäryhmille. Käytännössä tämä mahdollistaa sen, ettei tietyn käyttäjäryhmän käyttöliittymässä esiinny toiselle käyttäjäryhmälle tarkoitettua tietoa, vaan käyttöliittymä voidaan yksilöidä juuri tietylle ryhmälle. On sanomattakin selvää, että sovelluksen käytettävyys paranee, kun käyttöliittymässä ei esiinny turhaa tai vain harvoin tarvittavaa tietoa.

Havainnointi ja vuorovaikutus tuotteen kanssa ovat oleellisia seikkoja, jotka on syytä ottaa huomioon sovelluksen käytettävyyttä suunniteltaessa. Sovelluksen päällimmäisenä tavoitteena on taata käyttäjälle tehokas toiminnan apuväline, ja siksi käyttäjän pitää pystyä havaitsemaan kaikki toiminnon suorittamisen kannalta oleelliset sovelluksen osat. Käyttäjän on myös pystyttävä mieltämään käyttöliittymän eri toiminnot joksikin, ennen kuin voi niitä käyttää. Toimintoja onkin syytä korostaa joillain käyttäjälle jo entuudestaan tutuilla kuvioilla tai symboleilla. Esimerkiksi poisto-toimintoon on hyvä liittää roskakorin kuva.

Hyvässä käyttöliittymässä ulkoasu tukee muuta sisältöä ja luo tuotteesta yhtenäisen kokonaisuuden. Ulkoasun merkitys myös käytettävyydessä on oleellinen, sillä selkeä ja helppolukuinen ulkoasu voi parantaa työskentelynopeutta reilusti. Ulkoasun suunnittelussa oleellista on ottaa huomioon käyttöliittymän toimivuus ja sisältö. Ulkoasun täytyy korostaa näitä seikkoja eikä piilottaa niitä epäoleellisten elementtien taakse.

Suunnittelijan on lähdettävä liikkeelle siitä, mitä elementtejä tiettyyn näkymään tarvitaan, mikä on niiden järjestys, tärkeys, pituus, muoto ja miten ne suunnitellaan sisällön mukaan järkevästi.

Käyttöliittymäsuunnittelun oleellisimmat osat, jotka suunnittelijan tulee ymmärtää, ovat: käyttäjä ei pysty havaitsemaan käyttöliittymän kaikkia yksityiskohtia, eikä suunnittelija näe sovelluksensa käyttöliittymää, kuten ensikertalainen sen näkee. Tästä syystä myös käytettävyyden suunnittelussa käyttöliittymän testauksella on tärkeä rooli. Käyttöliittymää on suositeltavaa testata heuristisella arvioinnilla useita kertoja jo sovelluksen suunnittelun ja kehityksen aikana. Sen tavoite on löytää käytettävyysongelmia käyttöliittymästä erillisen tarkastuslistan avulla.

RIA-sovellusten käytettävyys poikkeaa edellisen sukupolven sovelluksista ennen kaikkea vuorovaikutuksen osalta. RIA-sovellukset tarjoavat runsaamman valikoiman erilaisia elementtejä parantamaan sovelluksen käyttöliittymää. Tällaisia ovat esimerkiksi kattavammat ja helposti toteutettavat virheilmoitukset tai mahdollisuus pitää käyttäjä paremmin selvillä sovelluksen tilasta ja sen eri vaiheista. Myös RIA-sovellusten kaaviot ja grafiikka ovat vertaansa vailla tämän päivän web-sovellusten kehityksessä. Oleellista on kuitenkin muistaa, että liika on liikaa myös tässä asiassa ja kiinnittää erityishuomiota sovelluksen selkeyteen.

RIA-sovellukset ovat erittäin dynaamisia ja reagoivat huomattavasti nopeammin käyttäjän komentoihin kuin perinteiset HTML-pohjaiset sivustot. Tämä on mahdollista siksi, että sovellus kerää ainoastaan muuttuneen tiedon eikä koko sivua tarvitse ladata uudestaan, kuten ennen. Uudistus vähentää tietoliikenteen määrää käyttäjän ja palvelimen välillä, mikä tekee sovelluksesta nopeamman, tehokkaamman ja käyttäjäystävällisemmän.

Yhteenvetona voidaan todeta, että RIA-sovellusten käytettävyydessä, kuten yleisestikin web-sovellusten käytettävyydessä, on ensiksi otettava huomioon, ketkä sovellusta tulevat käyttämään. Hyvässä sovelluksessa otetaan huomioon seikkailunhaluiset käyttäjät, mutta myös tavalliset käyttäjät, jotka eivät ole niin herkkiä testaamaan sovelluksen uusia ominaisuuksia ja toimintoja.

3. Kuinka kehitysprojekti käytännössä toteutetaan?

Kehitysprojektin toimintamalliksi valittiin prototyyppimalli, sillä toimeksiantaja antoi varsin löyhät vaatimukset toteutettavan asiakashallintajärjestelmän suhteen. Prototyyppimallin avulla näitä vaatimuksia oli helppo tarkentaa ja täsmentää testattavan prototyypin valmistumisen jälkeen. Prototyyppimallin avulla voidaan myös kätevästi varmistaa, että käyttöliittymästä löytyvät kaikki tarvittavat toiminnot ja että käyttöliittymä vastaa asiakkaan toiveita. Lisäksi se on oiva tapa testata järjestelmän suorituskykyä, jo aikaisessa vaiheessa.

Mitä suunnittelussa tulee ottaa huomioon?

Suunnitteluvaiheessa on tavoitteena siirtää asiakkaan toiveet tekniselle kielelle, järjestelmän toteutuksen kuvaukseksi. Tärkeää on määrittää asiakkaan toiveet ja tarpeet jo alussa mahdollisimman tarkasti, jotta toteutettavasta järjestelmästä tulee asiakkaan toimintaa tukeva ja käytettävyydeltään tehokas. Projektin alkuvaiheessa laadittiin esitutkimus, jonka pohjalta projekti päätettiin toteuttaa. Esitutkimuksessa järjestelmää oli suunniteltu vasta varsin kevyesti ja se olikin enemmän yleiskatsaus kaikkeen siihen, mitä toteutettavalla järjestelmällä voidaan saavuttaa kuin varsinaiseen suunnitteluun liittyvä dokumentti. Siinä kuitenkin otetaan ensiaskeleet myös suunnittelun kannalta, joten esitutkimus kannattaa laatia perusteellisesti, jotta sen pohjalta olisi hyvä jatkaa projektia oikeaan suuntaan.

Vaatimusmäärittelyllä ja sen pohjalta luoduilla käyttötapauksilla on suuri rooli kartoitettaessa asiakasvaatimuksia ja kuvailtaessa niitä ohjelmistovaatimuksiksi. Vaatimusmäärittelyssä luodaan pohja koko kehitystyölle ja testaukselle, ja näin ollen määrittelyn on oltava tarkka, kattava ja yksityiskohtainen, sillä sen pohjalta luodaan järjestelmän lopulliset toiminnot. Vaatimusmäärittelyn luontiin kannattaa varata reilusti aikaa myös asiakkaalta, sillä usein kaikkia vaatimuksia ei saada määriteltyä yhdessä palaverissa.

Käyttötapaukset laaditaan vaatimusten pohjalta, ja niiden tarkoitus on kuvata järjestelmän toiminnallisuus, kun käyttäjä suorittaa jonkin tapahtumaketjun. Käyttötapaukset jaetaan useille käyttäjäryhmille, jotka määritetään sen perusteella, miten he järjestelmää käyttävät tai mitkä toiminnot heille ovat sallittuja. Käyttötapausten tulee olla asiakkaan ja tulevien käyttäjien ymmärrettävissä, niiden tulee kuvata asiakasvaatimuksia, olla testattavia ja kooltaan ja tarkkuudeltaan täsmällisiä. Liika on liikaa myös

käyttötapauksia laadittaessa, eikä kaikkia pienimpiä yksityiskohtia kannata ottaa mukaan vielä tässä vaiheessa. Käyttötapausten tärkein tarkoitus on toimia kommunikointivälineenä kehitettäessä asiakasvaatimuksia ohjelmistovaatimuksiksi.

Käyttöliittymäsuunnitelma perustuu vaatimusmäärittelyyn ja käyttötapauksiin, joiden pohjalta luodaan myös järjestelmän ulkoasu. Ulkoasun lisäksi käyttöliittymäsuunnitelmassa selvitetään, miten ja millä komponentilla järjestelmän eri toiminnot ovat toteutettavissa. Käyttöliittymäsuunnittelussa on tärkeää ottaa huomioon myös jo aiemmin käytettävyys tutkimuskysymykseen vastattaessa esitetyt hyvän käyttöliittymän ominaisuudet. Käyttöliittymäsuunnitelman avulla asiakas näkee, miten eri toiminnot ovat toteutettavissa ja miltä järjestelmä tulee näyttämään.

Ohjelmiston arkkitehtuurilla esitetään järjestelmän eri rakenneosia ja niiden välisiä suhteita. Arkkitehtuuri luo yhteisen pohjan kaikelle järjestelmän suunnittelu- ja toteutustyölle. Arkkitehtuurissa tärkein seikka on ohjelmiston toteutusfilosofia. Järjestelmänkehittäjä saa arkkitehtuurin toteutusfilosofiasta mallin siitä, miten järjestelmä toteutetaan. Usein hyvälle arkkitehtuurille on ominaista se, että jos jotain asiaa ei tiedä, se on pääteltävissä toteutusfilosofian perusteella.

Kuinka sovellus toteutetaan?

Toteutusvaiheessa järjestelmää alettiin toteuttaa ennalta suunniteltujen toimintojen ja ulkoasun mukaiseksi. Ensimmäiseksi on vuorossa oikean RIA-teknologian ja sen mukanaan tuoman kehitysympäristön valinta.

Tänä päivänä on olemassa monia RIA-teknologioita, joista oikean valitseminen juuri tietyn sovelluksen toteuttamiseksi voi olla vaikeaa. Joka teknologialla on omat vahvuutensa ja heikkoutensa kuin myös yksilölliset ominaisuutensa tietoturvassa, käytettävyydessä ja integroinnissa. Teknologian valinnassa oleellisessa osassa ovat toteutettavan järjestelmän vaatimukset ja tarpeet. Oikeaa RIA-teknologiaa valittaessa on syytä kiinnittää huomiota seuraaviin seikkoihin:

- **Kehittämisen ja käyttämisen helppous:** jotkut teknologiat ovat helpompia oppia kuin toiset ja niillä sovellusten kehittäminen on yksinkertaisempaa kuin toisilla.

- **Integroitavuus:** mitä parempi integroitavuus eli mitä helpommin teknologia on otettavissa käyttöön muiden järjestelmässä käytettyjen teknologioiden kanssa, sitä helpommin vältytään tulevaisuudessa odottamattomilta integrointi-ongelmilta.

- **Tietoturva:** hyvä ratkaisu on myös tietoturvallinen.

- **Selain- ja käyttöjärjestelmäriippumattomuus:** Suurin osa internetin käyttäjistä käyttää Internet Explorer -selainta, mutta myös esimerkiksi Firefox-selain on suosittu. On suotavaa, että sovellus toimii samalla tavalla selaimesta riippumatta.

- **Teknologian tulevaisuuden näkymät:** noudattaako ratkaisu standardeja ja onko se esimerkiksi avoimen lähdekoodin projekti? Tällä on tärkeä rooli siirrettävyyden kannalta tulevaisuuden infrastruktuureissa.

- **Koulutus kustannukset:** Uuden teknologian oppiminen voi olla hidasta ja kallista.

- **Liitännäisten riippuvuus:** Täytyykö käyttäjän asentaa uusia ohjelmia tai liitännäisiä tietokoneelleen, jos sovelluksia kehitetään tietyllä teknologialla?

- **Kehitystyökalut:** hyvät kehitystyökalut nopeuttavat kehittämisen elinkaarta ja helpottavat kehittäjien työtä.

- **Teknologian tausta:** Kuka on teknologian takana? Korjataanko teknologian virheitä ja tuleeko uusia versioita? On hyvä, jos teknologialla on aktiivinen kehitysyhteisö ja dokumentaatio.

Omaan projektiimme valitsimme RIA-teknologiaksi Adobe Flex 3:n, sillä se on helposti integroitavissa toimimaan jo entuudestaan tuttujen PHP- ja MySQL-kielien kanssa. Flex:ä kehitetään jatkuvasti ja uusi versio neljä on jo tulossa kovaa vauhtia markkinoille. Lisäksi Flex-järjestelmien kehitykseen käytetty Adobe Flex Builder 3 on opiskelijakäyttöön ilmainen ja varsin kattava kehitystyökalu.

Järjestelmän perusta luotiin Adobe Flex Buider 3:lla ja sen käyttämillä MXML- ja ActionScript 3 -ohjelmointikielillä. MXML-ohjelmointikielellä luotiin pääasiassa järjestelmän rakenne ja ulkoasu. ActionScript 3 -kielellä lisättiin järjestelmään toiminnallisuudet. MySQL-tietokantaa käyttävissä toiminnallisuuksissa käytettiin lisäksi PHP-ohjelmointikieltä, jolla kommunikoitiin yhdessä MySQL-tietokannan kanssa. Kommunikaatio tapahtui eri tarkoituksiin yksilöidyillä SQL-kyselyillä.

Kuinka sovelluksen toimintaa testataan?

Sovelluksen toimintaa testataan jatkuvasti projektin alusta loppuun saakka. Testaus myös suunnitellaan järjestelmätestidokumentissa, jossa käydään läpi tärkeimmät testitapaukset ja määritellään lähinnä, mitä testataan ja mitä ei. Sovellusta testataan koko elinkaaren ajan, jotta nähdään, että se toimii niin kuin pitää ja toteuttamista voidaan jatkaa. On myös kaksi suurempaa tai merkittävämpää testauksen vaihetta, ensimmäinen silloin, kun sovellus on saatu vastaamaan vaatimuksia ja se on sekä toteuttajien että toimeksiantajan mielestä valmis. Toinen perusteellinen testausvaihe on, kun sovellus otetaan käyttöön yrityksen palvelimella. Näin ollen voidaan olla varmoja, että sovellus toimii varsinaisessa käyttöympäristössään niin kuin pitää.

Testauksessa löytyy varmasti virheitä tai ongelmia ja niihin pyritään reagoimaan heti. Havaitut virheet pyritään korjaamaan välittömästi mahdollisuuksien mukaan. Testausta ei periaatteessa voi olla liikaa, mutta se tulee sovittaa järkevästi muun projektin yhteyteen. Joka tapauksessa projektissa testataan kaikki vaatimukset, jotta voidaan olla varmoja, että ne täyttyvät.

7 POHDINTA

Tutkimus on erittäin ajankohtainen, koska RIA-sovellukset on suunta, johon web-ohjelmistokehityksessä ollaan menossa. Se on ollut jo jonkin aikaa pinnalla, mutta suosio tulee vain kasvamaan tulevaisuudessa. RIA-sovellusten hyödyt ovat kiistatta merkittävät verrattuna ongelmiin. Näin ollen ei ole olemassa syytä, mikseivät rikkaat internet-sovellukset tulisi jatkossakin yleistymään sähköisessä liiketoiminnassa sekä kuluttajille suunnatuissa että yrityksen sisäisissä sovelluksissa.

Internetiä käytetään yhä enemmän ja enemmän mobiililaitteilla. RIA-sovellusten kehittäjien tulisi ottaa tämä huomioon ja toteuttaa sovellukset niin, että niitä voidaan käyttää myös esimerkiksi kännyköillä ja PDA-laitteilla. Aiheena RIA-sovellukset mobiililaitteissa on kuitenkin niin laaja, että sen lähempi tarkasteleminen jätettiin pois tästä tutkimuksesta, koska se olisi vienyt tutkimusta toisille raiteille. Tässä voisikin olla oivallinen jatkotutkimuksen aihe asiasta kiinnostuneelle opiskelijalle.

Tutkimuksen teoriaosuudesta tuli selkeä ja johdonmukainen. Tutkimuskysymykset ja tutkimusmenetelmä olivat onnistuneesti valittuja ja ne tukivat matkaa kohti päämäärää. Ensimmäinen kysymys johdatti RIA-sovellusten maailmaan ja niiden hyödyntämiseen liiketoiminnassa. RIA-teknologioiden vertailussa keskityttiin vain yleisimpiin teknologioihin ja tuleekin muistaa, että on olemassa lukuisia muita teknologioita rikkaiden internet-sovellusten luomiseen. Näiden lähempi tarkastelu rajattiin tutkimuksesta pois, koska sen myötä tutkimus olisi paisunut liikaa ja olennainen osa olisi jäänyt vähemmälle huomiolle. Toinen kysymys toi esiin käytettävyyden näkökulman web-sovelluksissa. Käytettävyyden merkitys on suuri, kun kehitetään hyvää sovellusta ja sen huomioon ottaminen on tärkeää. Kolmas kysymys ja siihen vastaaminen valmisti varsinaisen sovelluksen luomiseen sekä yleisesti ohjelmistokehityksen maailmaan.

Työn tekeminen kertasi hyvin muun muassa ohjelmistokehityksestä opittuja taitoja ja se opetti paljon uutta esimerkiksi rikkaista internet-sovelluksista sekä web-sovellusten käytettävyydestä. Lisäksi varsinkin Adobe Flex tuli teknologiana hyvin tutuksi ja uskomme siitä olevan hyötyä tulevaisuudessa. Projektiryhmälle opinnäytetyön tekeminen opetti myös muun muassa aikataulutusta, yhteistyötä, priorisointia, asiakkaan kanssa työskentelyä, ongelmien ratkaisua sekä tiedonhankintaa. Nämä kaikki ovat tärkeitä asioita myöhemmin työelämässä menestymisen kannalta. Pääsääntöisesti ma-

teriaalia tutkimuksessa käsitellyistä aiheista löytyi hyvin, vaikka esimerkiksi RIA-sovelluksia käsittelevät julkaisut olivat lähes poikkeuksetta englanninkielisiä ja jotkin vanhentuneita siinä mielessä, että niissä käsiteltävistä teknologioista oli ilmestynyt jo uusia versioita.

LÄHTEET

5 Benefits Of Rich Internet Applications For Manufacturing ROI 2009. Manufacturing.net -sivusto. Viitattu 8.3.2010.
Http://www.manufacturing.net/Articles-5-Benefits-Of-Rich-Internet-Applications-For-Manufacturing-ROI-091809.aspx.

Adoben uusi alusta internet-sovelluksille on nyt valmis. 2008. Adoben tiedote. Viitattu 16.2.2010.
Http://shockwave-install.com/fi/aboutadobe/pressroom/pr/feb2008/Adoben_uusi_alu.pdf.

Allaire, J. 2002. Macromedian tiedote. Viitattu 8.3.2010.
Http://www.adobe.com/devnet/flash/whitepapers/richclient.pdf.

Benefits of RIA's. n.d. Kalalau enterprises -sivusto. Viitattu 8.3.2010.
Http://www.kalalau.org/solutions_ria_benefits.html.

Benefits of Rich Internet Applications. n.d. Adoben sivusto. Viitattu 24.2.2010.
Http://www.adobe.com/resources/business/rich_internet_apps/benefits/.

Browser vs. desktop. n.d. Adoben tuote-esittelysivusto.Viitattu 16.2.2010.
Http://www.adobe.com/products/air/comparison/.

Cheng, R. 2008. How to Choose an RIA Solution. Viitattu 23.2.2010.
Http://java.sys-con.com/node/781133.

Flex FAQ. n.d. Adobelta Flexistä useimmin kysytyt kysymykset. Viitattu 5.2.2010.
Http://www.adobe.com/products/flex/faq/.

Flex overview. n.d. Adobe Flex yleiskatsaus. Viitattu 5.2.2010.
Http://www.adobe.com/products/flex/overview/.

Fordvehicles. n.d. Fordvehicles -sivusto. Viitattu 8.3.2010.
Http://www.fordvehicles.com/.

Garret, J. 2005. Ajax: A New Approach to Web Applications. Viitattu 3.2.2010.
Http://www.adaptivepath.com/ideas/essays/archives/000385.php.

Ghoda, A & Scanlon, J, 2009. Accelerated Silverlight 3. New York, USA: Apress.

Haikala, I. & Märijärvi, J. 2004. Ohjelmistotuotanto. 10. uud. p. Hämeenlinna: Talentum Media.

JavaFX FAQ. n.d. JavaFX yleiset kysymykset. Viitattu 5.2.2010.
Http://javafx.com/faq/.

Maurer, D. 2006. Usability for Rich Internet Applications. Viitattu 22.1.2010. Http://www.digital-web.com/articles/usability_for_rich_internet_applications.

Nielsen, J. 1993. Usability Engineering. San Diego, CA: Academic Press Inc.

Nielsen, J. 2000. WWW-suunnittelu. Alkuperäisteoksen nimi: Designing Web Usability. Jyväskylä: Edita.

Peltomäki, J & Nykänen, O. 2006. Web-selainohjelmointi. Porvoo: Docendo.

Riastats. n.d. Rich Internet Application Statistics. Viitattu 23.2.2010. Http://www.riastats.com.

Search optimization techniques for RIAs. n.d. Adoben sivusto kehittäjille. Viitattu 25.2.2010. Http://www.adobe.com/devnet/seo/articles/techniques_ria.html.

Sinkkonen, I., Kuoppala, H., Parkkinen, J. & Vastamäki, R. 2006. Käytettävyyden psykologia. 3. uud. p. Helsinki: Edita.

Tapper, J., Labriola, M., Boles, M. & Talbot, J. 2008. Adobe Flex 3: Training from the source. United States of America.

The Battle for the RIA Throne: Flex vs. Silverlight. 2008. Mike Kavisin artikkeli Flexin ja Silverlightin taistelusta. Viitattu 16.2.2010. Http://madgreek65.blogspot.com/2008/08/battle-for-ria-throne-flex-vs.html.

The business benefits of rich Internet application for enterprises. 2008. Adoben tiedote. Viitattu 25.2.2010. Http://www.ashorten.com/wp-content/uploads/2009/01/Adobe_RIA_Enterprise_Web0109.pdf.

Van Elten, N. 2008. Good usability of Rich Internet Applications. Viitattu 25.1.2010. Http://www.jungleminds.com/publications/articles/good_usability_of_rich_internet_applications.

Verkkopalveluarkkitehtuuri. 2006. Tampereen Teknillisen Yliopiston sivusto. Viitattu 10.2.2010. Http://matwww.ee.tut.fi/hmopetus/hm-ohj/2006/pruju/hmohj06-041-051.pdf.

LIITTEET

Liite 1. Vaatimusmäärittely

Toiminnalliset vaatimukset:

1. Pääkäyttäjä voi kirjautua sisään järjestelmään, pakollinen
2. Pääkäyttäjä voi lisätä yksityisasiakkaan, pakollinen
3. Pääkäyttäjä voi poistaa yksityisasiakkaan, pakollinen
4. Pääkäyttäjä voi muokata yksityisasiakkaan tietoja, pakollinen
5. Pääkäyttäjä voi tulostaa valitun yksityisasiakkaan tiedot, pakollinen
6. Pääkäyttäjä voi hakea yksityisasiakkaita nimen perusteella, pakollinen
7. Pääkäyttäjä voi selata kaikkia yksityisasiakkaita listana, pakollinen
8. Pääkäyttäjä voi lisätä yritysasiakkaan, pakollinen
9. Pääkäyttäjä voi poistaa yritysasiakkaan, pakollinen
10. Pääkäyttäjä voi muokata yritysasiakkaan tietoja, pakollinen
11. Pääkäyttäjä voi tulostaa valitun yritysasiakkaan tiedot, pakollinen
12. Pääkäyttäjä voi hakea yritysasiakkaita nimen perusteella, pakollinen
13. Pääkäyttäjä voi selata kaikkia yritysasiakkaita listana, pakollinen
14. Pääkäyttäjä voi tallentaa yhteisiä muistiinpanoja, pakollinen
15. Pääkäyttäjä voi muokata yhteisiä muistiinpanoja, pakollinen
16. Pääkäyttäjä voi poistaa yhteisiä muistiinpanoja, pakollinen
17. Pääkäyttäjä voi tallentaa omia muistiinpanoja, pakollinen
18. Pääkäyttäjä voi muokata omia muistiinpanoja, pakollinen
19. Pääkäyttäjä voi poistaa omia muistiinpanoja, pakollinen
20. Pääkäyttäjä voi tulostaa omat ja yhteiset muistiinpanot, pakollinen
21. Pääkäyttäjä voi kirjautua ulos järjestelmästä, pakollinen
22. Pääkäyttäjä voi valita tulostettavat tietueet, valinnainen
23. Pääkäyttäjä voi lisätä peruskäyttäjätason käyttäjiä, valinnainen
24. Pääkäyttäjä voi poistaa peruskäyttäjätason käyttäjiä, valinnainen
25. Peruskäyttäjä voi kirjautua järjestelmään, pakollinen
26. Peruskäyttäjä voi tallentaa yhteisiä muistiinpanoja, pakollinen
27. Peruskäyttäjä voi muokata yhteisiä muistiinpanoja, pakollinen
28. Peruskäyttäjä voi tulostaa yhteiset muistiinpanot, pakollinen
29. Peruskäyttäjä voi lisätä yksityisasiakkaan, pakollinen
30. Peruskäyttäjä voi tulostaa valitun yksityisasiakkaan tietoja, pakollinen
31. Peruskäyttäjä voi selata kaikkia yksityisasiakkaita listana, pakollinen
32. Peruskäyttäjä voi hakea yksityisasiakkaita nimen perusteella, pakollinen
33. Peruskäyttäjä voi lisätä yritysasiakkaan, pakollinen
34. Peruskäyttäjä voi tulostaa valitun yritysasiakkaan tietoja, pakollinen
35. Peruskäyttäjä voi selata kaikkia yritysasiakkaita listana, pakollinen
36. Peruskäyttäjä voi hakea yritysasiakkaita nimen perusteella, pakollinen
37. Peruskäyttäjä voi kirjautua ulos, pakollinen

Ei-toiminnalliset vaatimukset:

1. Järjestelmän tulee toimia Flash –tuella varustetulla selaimella, pakollinen
2. Järjestelmän tulee olla käyttöjärjestelmäriippumaton, pakollinen
3. Järjestelmään kirjautuminen vaatii käyttäjätunnuksen ja salasanan, pakollinen
4. Käyttäjien salasanat tulee olla salattuna tietokannassa, pakollinen

Liite 2. Käyttötapaukset

Käyttötapaus:	**Pääkäyttäjä kirjautuu sisään järjestelmään.**
Suorittajat:	Järjestelmän pääkäyttäjät.
Esiehdot:	Järjestelmä on toiminnassa, pääkäyttäjä on saapunut kirjautumis-sivulle.
Kuvaus:	Pääkäyttäjä syöttää käyttäjätunnuksen ja salasanan niille varattuihin kenttiin. Pääkäyttäjä painaa Kirjaudu-painiketta.
Poikkeukset:	Käyttäjätunnus-kenttä on tyhjä tai väärin: Pääkäyttäjältä on jäänyt Käyttäjätunnus-kenttä tyhjäksi tai hän on syöttänyt väärän tunnuksen, kun hän painaa Kirjaudu-painiketta. Järjestelmä huomauttaa käyttäjätunnuksen tai salasanan olevan väärin. Salasana-kenttä on tyhjä tai väärin: Pääkäyttäjältä on jäänyt Salasana-kenttä tyhjäksi tai hän on syöttänyt väärän salasanan, kun hän painaa Kirjaudu-painiketta. Järjestelmä huomauttaa käyttäjätunnuksen tai salasanan olevan väärin.
Jälkiehdot:	Pääkäyttäjä on kirjautunut sisään järjestelmään. Pääkäyttäjä ohjataan automaattisesti muistiinpano-näkymään.

Käyttötapaus:	**Pääkäyttäjä lisää yksityisasiakkaan.**
Suorittajat:	Järjestelmän pääkäyttäjät.
Esiehdot:	Järjestelmä on toiminnassa, pääkäyttäjä on kirjautunut järjestelmään, pääkäyttäjä on siirtynyt yksityisasiakkaat – välilehdelle.
Kuvaus:	Pääkäyttäjä painaa Lisää-painiketta. Pääkäyttäjä syöttää avautuneeseen näkymään haluamansa tiedot. Asiakasnumero tulee automaattisesti järjestelmästä. Nimi – pakollinen Osoite Puhelinnumero Email Vastuuhenkilö Myynti Kuvaus Pääkäyttäjä painaa Tallenna-painiketta.
Poikkeukset:	Nimi-kenttä on tyhjä: Pääkäyttäjältä on jäänyt yksityisasiakkaan Nimi-kenttä tyhjäksi, kun hän painaa Tallenna-painiketta. Järjestelmä huomauttaa Nimi-kentän olevan pakollinen ennen kuin tallennus on mahdollista.
Jälkiehdot:	Tietokantaan on tallentunut uusi yksityisasiakas, jolla on yksilöllinen asiakasnumero ja järjestelmä ilmoittaa onnistuneesta asiakkaan lisäyksestä. Pääkäyttäjä ohjataan automaattisesti takaisin yksityisasiakkaidentaulukkonäkymään.

Käyttötapaus:	**Pääkäyttäjä poistaa yksityisasiakkaan.**
Suorittajat:	Järjestelmän pääkäyttäjät.
Esiehdot:	Järjestelmä on toiminnassa, pääkäyttäjä on kirjautunut järjestelmään, pääkäyttäjä on siirtynyt yksityisasiakkaat – välilehdelle.
Kuvaus:	Pääkäyttäjä valitsee poistettavan yksityisasiakkaan. Pääkäyttäjä painaa Poista-painiketta. Järjestelmä kysyy poiston vahvistuksen. Pääkäyttäjä painaa OK-painikketta.
Poikkeukset:	Pääkäyttäjä ei ole valinnut poistettavaa yksityisasiakasta. Järjestelmä ilmoittaa, ettei poistettavaa asiakasta ole valittuna. Pääkäyttäjä painaa Peruuta-painiketta poiston vahvistuksessa, pääkäyttäjä ohjataan automaattisesti takaisin taulukkonäkymään.
Jälkiehdot:	Yksityisasiakas on poistettu tietokannasta ja järjestelmää ilmoittaa onnistuneesta asiakkaan poistosta. Pääkäyttäjä ohjataan automaattisesti takaisin yksityisasiakkaidentaulukkonäkymään.

Käyttötapaus:	**Pääkäyttäjä muokkaa yksityisasiakasta.**
Suorittajat:	Järjestelmän pääkäyttäjät.
Esiehdot:	Järjestelmä on toiminnassa, pääkäyttäjä on kirjautunut järjestelmään, pääkäyttäjä on siirtynyt yksityisasiakkaat – välilehdelle.
Kuvaus:	Pääkäyttäjä valitsee tupla-klikkaamalla muokattavan yksityisasiakkaan. Pääkäyttäjä muokkaa avautuneessa näkymässä haluamansa tiedot. Asiakasnumeroa ei voi muokata. Pääkäyttäjä painaa Tallenna-painiketta.
Poikkeukset:	Nimi-kenttä on tyhjä: Pääkäyttäjältä on jäänyt yksityisasiakkaan Nimi-kenttä tyhjäksi, kun hän painaa Tallenna-painiketta. Järjestelmä huomauttaa Nimi-kentän olevan pakollinen ennen kuin tallennus on mahdollista.
Jälkiehdot:	Tietokannassa olevan yksityisasiakaan tietoja on muokattu ja järjestelmä ilmoittaa onnistuneesta muokkauksesta. Pääkäyttäjä ohjataan automaattisesti takaisin yksityisasiakkaidentaulukkonäkymään.

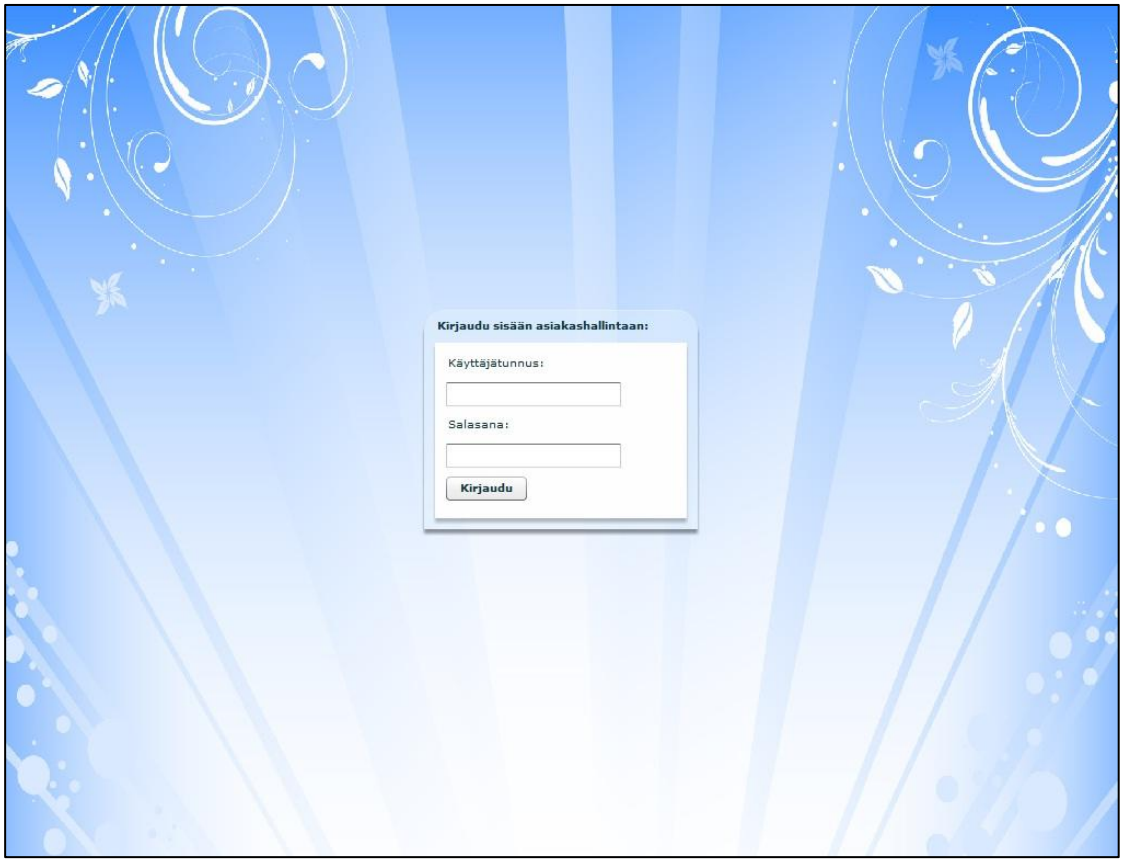
Käyttötapaus:	**Pääkäyttäjä lisää yritysasiakkaan.**
Suorittajat:	Järjestelmän pääkäyttäjät.
Esiehdot:	Järjestelmä on toiminnassa, pääkäyttäjä on kirjautunut järjestelmään, pääkäyttäjä on siirtynyt yritysasiakkaat – välilehdelle.
Kuvaus:	Pääkäyttäjä painaa Lisää-painiketta. Pääkäyttäjä syöttää avautuneeseen näkymään haluamansa tiedot. Asiakasnumero tulee automaattisesti järjestelmästä. Yritys – pakollinen Y-tunnus Yhteyshenkilö Osoite Laskutusosoite Puhelinnumero Email WWW-sivu Toimiala Vastuuhenkilö Myynti Kuvaus Pääkäyttäjä painaa Tallenna-painiketta.
Poikkeukset:	Yritys-kenttä on tyhjä: Pääkäyttäjältä on jäänyt Yritys-kenttä tyhjäksi, kun hän painaa Tallenna-painiketta. Järjestelmä huomauttaa Yritys-kentän olevan pakollinen ennen kuin tallennus on mahdollista.
Jälkiehdot:	Tietokantaan on tallentunut uusi yritysasiakas, jolla on yksilöllinen asiakasnumero ja järjestelmä ilmoittaa onnistuneesta asiakkaan lisäyksestä. Pääkäyttäjä ohjataan automaattisesti takaisin yritysasiakkaidentaulukkonäkymään.

Käyttötapaus:	**Pääkäyttäjä poistaa yritysasiakkaan.**
Suorittajat:	Järjestelmän pääkäyttäjät.
Esiehdot:	Järjestelmä on toiminnassa, pääkäyttäjä on kirjautunut järjestelmään, pääkäyttäjä on siirtynyt yritysasiakkaat – välilehdelle.
Kuvaus:	Pääkäyttäjä valitsee poistettavan yritysasiakkaan. Pääkäyttäjä painaa Poista-painiketta. Järjestelmä kysyy poiston vahvistuksen. Pääkäyttäjä painaa OK-painikketta.
Poikkeukset:	Pääkäyttäjä ei ole valinnut poistettavaa yritysasiakasta. Järjestelmä ilmoittaa, ettei poistettavaa asiakasta ole valittuna. Pääkäyttäjä painaa Peruuta-painiketta poiston vahvistuksessa, pääkäyttäjä ohjataan automaattisesti takaisin taulukkonäkymään.
Jälkiehdot:	Yritysasiakas on poistettu tietokannasta ja järjestelmää ilmoittaa onnistuneesta asiakkaan poistosta. Pääkäyttäjä ohjataan automaattisesti takaisin yritysasiakkaidentaulukkonäkymään.

Käyttötapaus:	**Pääkäyttäjä muokkaa yritysasiakasta.**
Suorittajat:	Järjestelmän pääkäyttäjät.
Esiehdot:	Järjestelmä on toiminnassa, pääkäyttäjä on kirjautunut järjestelmään, pääkäyttäjä on siirtynyt yritysasiakkaat – välilehdelle.
Kuvaus:	Pääkäyttäjä valitsee tupla-klikkaamalla muokattavan yritysasiakkaan. Pääkäyttäjä muokkaa avautuneessa näkymässä haluamansa tiedot. Asiakasnumeroa ei voi muokata. Pääkäyttäjä painaa Tallenna-painiketta.
Poikkeukset:	Yritys-kenttä on tyhjä: Pääkäyttäjältä on jäänyt Yritys-kenttä tyhjäksi, kun hän painaa Tallenna-painiketta. Järjestelmä huomauttaa Yritys-kentän olevan pakollinen ennen kuin tallennus on mahdollista.
Jälkiehdot:	Tietokannassa olevan yritysasiakaan tietoja on muokattu ja järjestelmä ilmoittaa onnistuneesta muokkauksesta. Pääkäyttäjä ohjataan automaattisesti takaisin yritysasiakkaidentaulukkonäkymään.

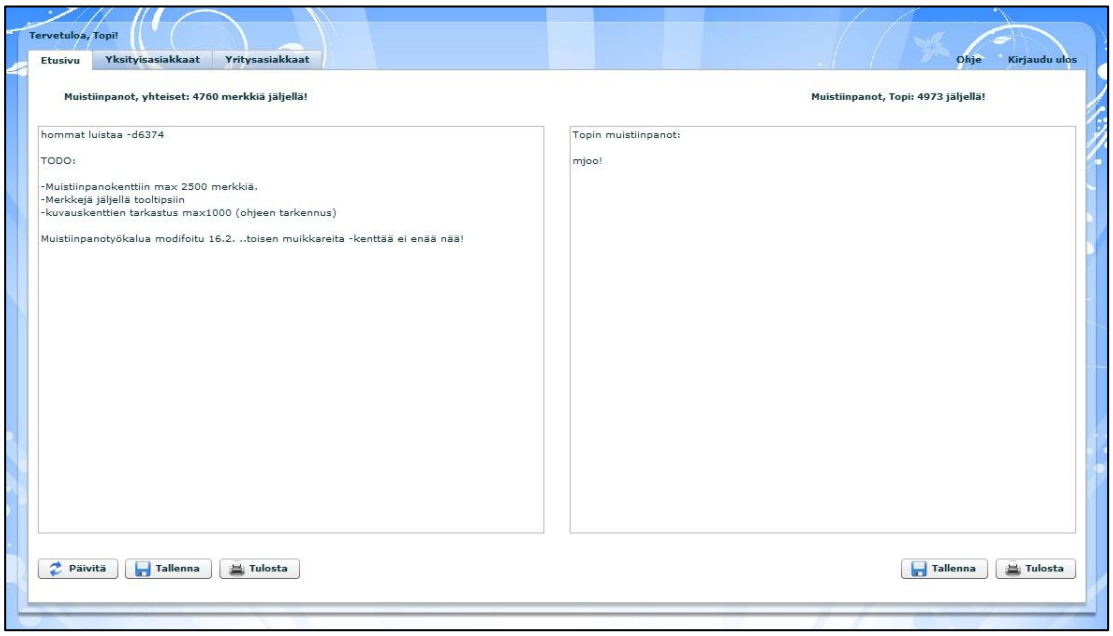
Liite 3. Käyttöliittymäsuunnitelma

Kirjautuminen:



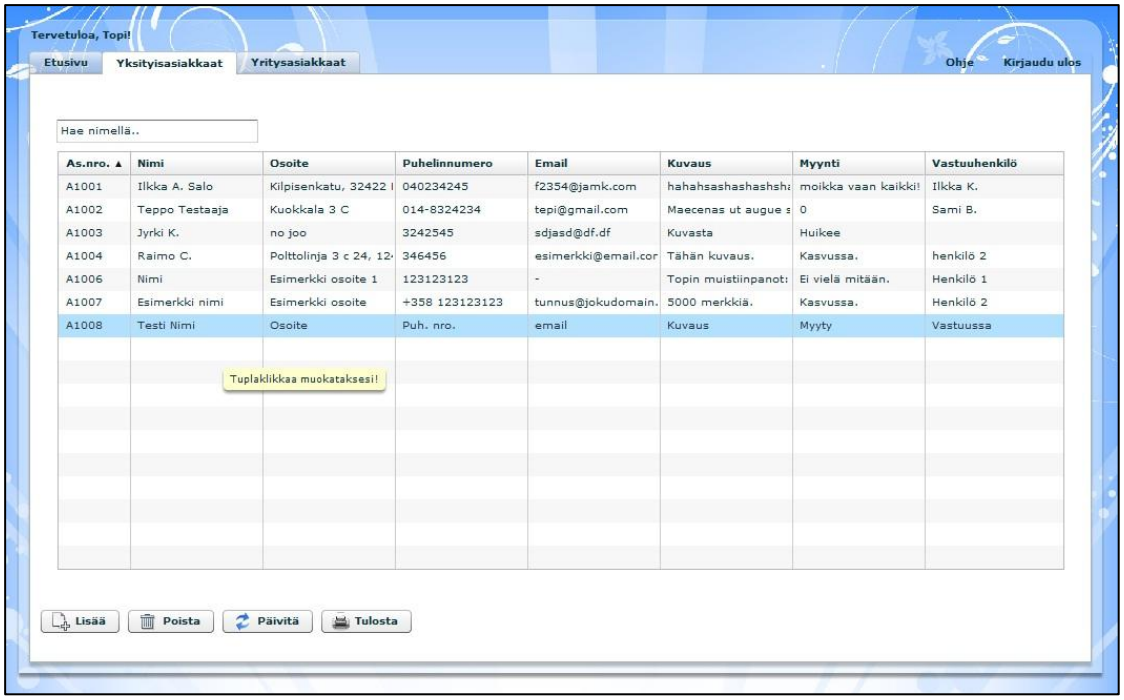
Komponentti	Kuvaus
Käyttäjätunnus-kenttä	Käyttäjä voi syöttää maksimissaan 50-merkkiä sisältävän käyttäjätunnuksen.
Salasana-kenttä	Käyttäjä voi syöttää maksimissaan 50-merkkiä sisältävän salasanan. Kenttään syötetyt merkit esitetään suojattuina.
Kirjaudu-painike	Lähettää käyttäjän syöttämät tiedot php-tiedostoon, joka tarkistaa ne MySQL-tietokannasta.

Muistiinpanotyökalu:



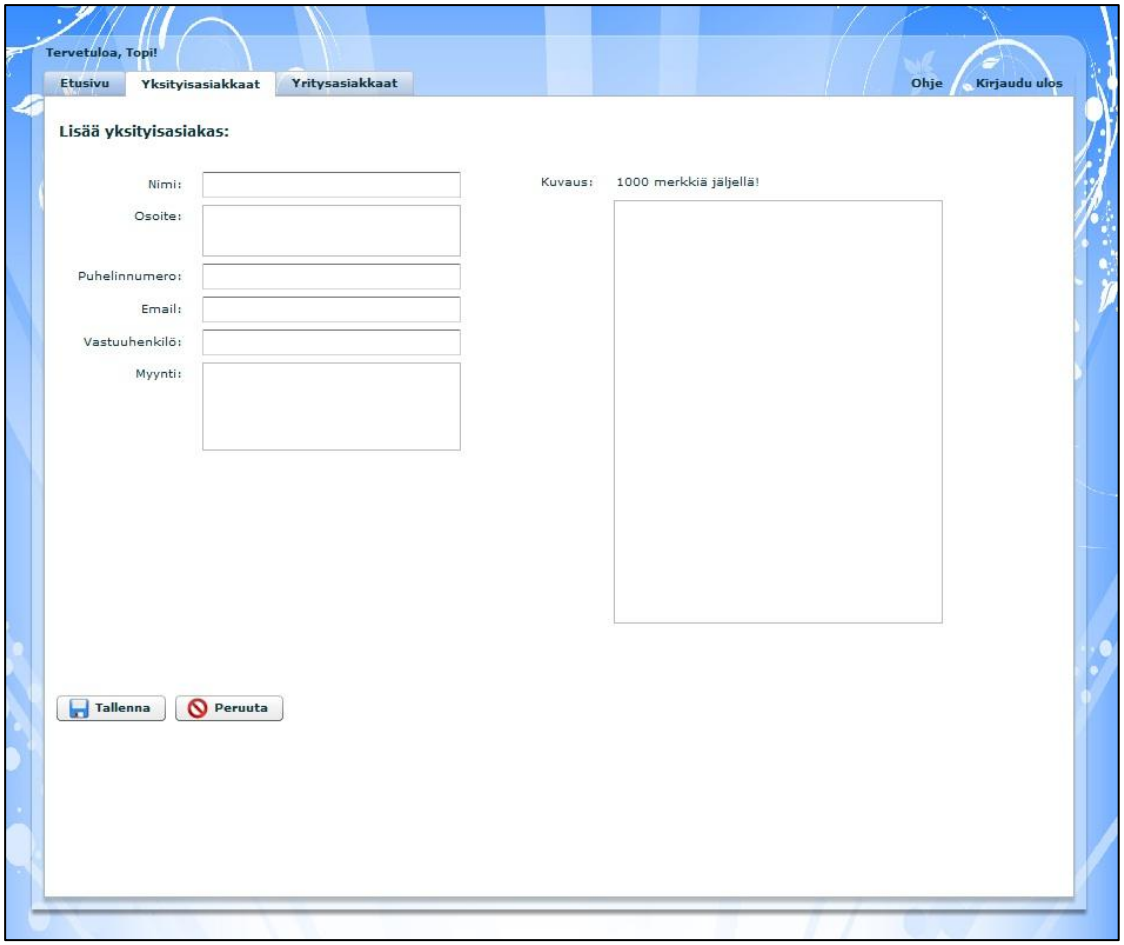
Komponentti	Kuvaus
Etusivu-välilehti	Avaa järjestelmän etusivun eli muistiinpanosivun.
Yksityisasiakkaat-välilehti	Avaa yksityisasiakkaiden taulukkonäkymän.
Yritysasiakkaat-välilehti	Avaa yritysasiakkaiden taulukkonäkymän.
Ohje-linkki	Avaa järjestelmänkäyttöohjeet popup-ikkunaan.
Kirjaudu ulos-linkki	Siirtää käyttäjän ulos järjestelmästä ja kirjautumissivulle.
Yhteiset muistiinpanot-kenttä	Käyttäjä voi syöttää maksimissaan 5000-merkkiä pitkän muistiinpanon.
Pääkäyttäjän muistiinpanot-kenttä	Kirjautunut pääkäyttäjä voi syöttää maksimissaan 5000-merkkiä pitkän henkilökohtaisen muistiinpanon. Kenttä ei näy muille käyttäjille.
Päivitä-painike	Päivittää yhteiset muistiinpanot.
Yhteisten muistiinpanojen Tallenna-painike	Tallentaa yhteisiin muistiinpanoihin tehdyt muutokset.
Yhteisten muistiinpanojen Tulosta-painike	Avaa yhteisten muistiinpanojen tulostusnäkymän.
Pääkäyttäjän muistiinpanojen Tallenna-painike	Tallentaa kirjautuneen pääkäyttäjän muistiinpanoihin tehdyt muutokset.
Pääkäyttäjän muistiinpanojen Tulosta-painike	Avaa kirjautuneen pääkäyttäjän muistiinpanojen tulostusnäkymän.

Yksityisasiakaat taulukkonäkymä:



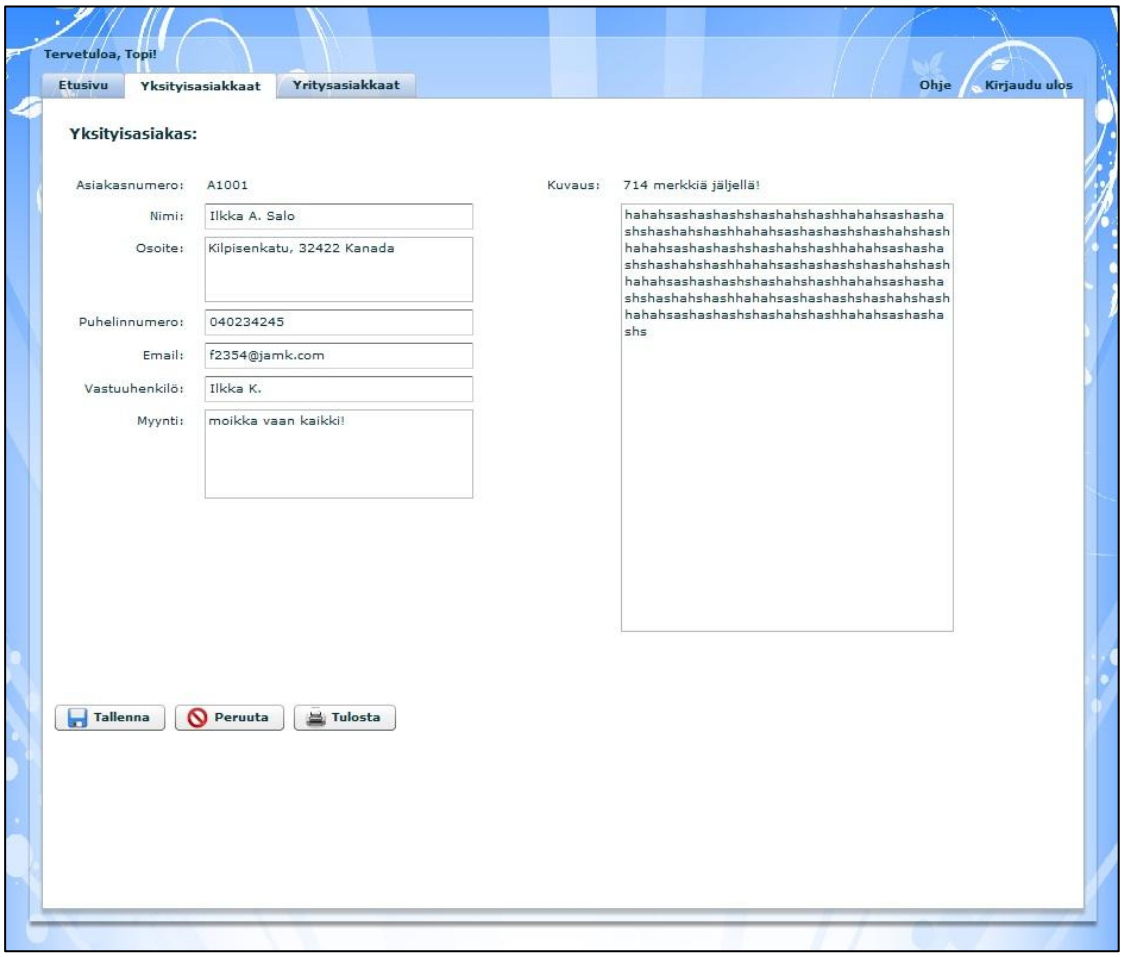
Komponentti	Kuvaus
Etusivu-välilehti	Avaa järjestelmän etusivun eli muistiinpanosivun.
Yksityisasiakkaat-välilehti	Avaa yksityisasiakkaiden taulukkonäkymän.
Yritysasiakkaat-välilehti	Avaa yritysasiakkaiden taulukkonäkymän.
Ohje-linkki	Avaa järjestelmänkäyttöohjeet popup-ikkunaan.
Kirjaudu ulos-linkki	Siirtää käyttäjän ulos järjestelmästä ja kirjautumissivulle.
Hae nimellä-kenttä	Käyttäjä voi syöttää merkkejä, jolloin taulukkonäkymä automaattisesti suodattaa esiin ne nimet joissa kyseinen merkki tai merkit esiintyvät.
Taulukkonäkymä	Käyttäjä voi drag&drop-tyylisesti järjestellä taulukonsarakkeita ja niiden leveyksiä. Käyttäjä voi klikkaamalla sarakkeen otsikkoa järjestään sarakkeen tiedot pienimmästä suurimpaan tai aakkosjärjestykseen. Tuplaklikkaamalla haluamaansa riviä käyttäjälle aukeaa kyseisen rivin asiakastietojen muokkausnäkymä.
Lisää -painike	Avaa käyttäjälle yksityisasiakkaan lisäys -näkymän.
Poista -painike	Poistaa poistonvahvistuksen jälkeen valitun asiakastiedon.
Päivitä-painike	Päivittää taulukkonäkymän.
Tulosta-painike	Avaa tulostus näkymän.

Yksityisasiakkaan lisäys:



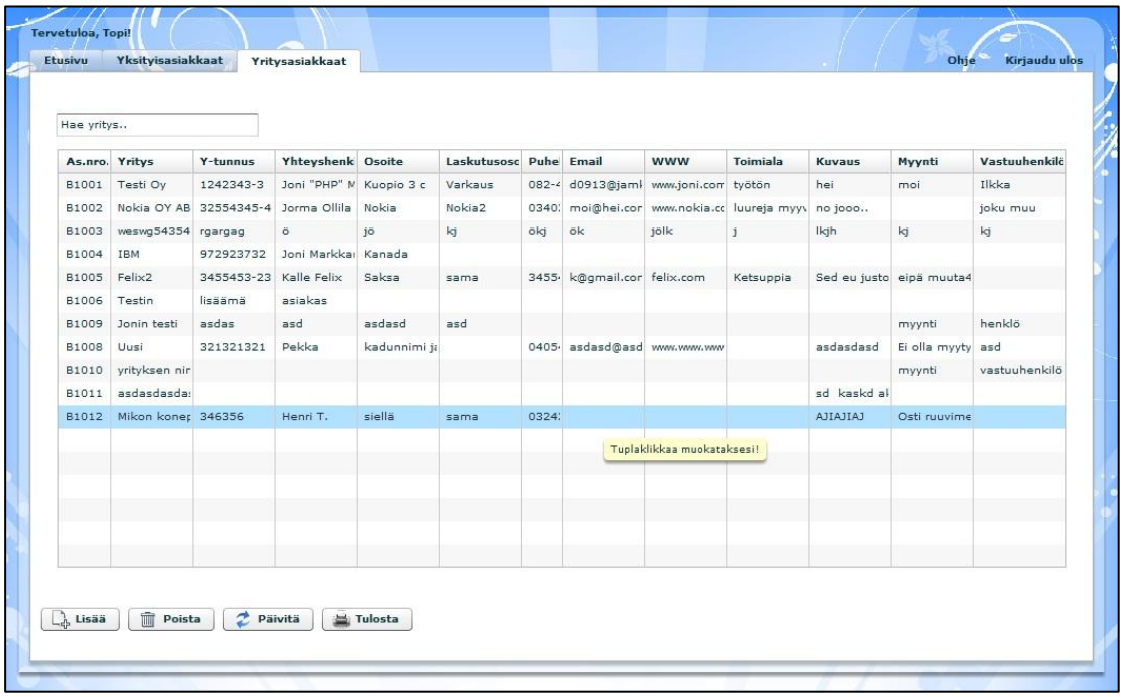
Komponentti	Kuvaus
Etusivu-välilehti	Avaa järjestelmän etusivun eli muistiinpanosivun.
Yksityisasiakkaat-välilehti	Avaa yksityisasiakkaiden taulukkonäkymän.
Yritysasiakkaat-välilehti	Avaa yritysasiakkaiden taulukkonäkymän.
Ohje-linkki	Avaa järjestelmänkäyttöohjeet popup-ikkunaan.
Kirjaudu ulos-linkki	Siirtää käyttäjän ulos järjestelmästä ja kirjautumissivulle.
Lisää yksityisasiakas-lomake	Käyttäjä voi syöttää haluamansa asiakastiedot. Pakollisena kenttänä Nimi–kenttä. Lisäksi Kuvaus–kentässä merkkilaskuri.
Tallenna-painike	Painettaessa tarkistaa lomakkeen tiedot, tallentaa asiakkaan ja vie käyttäjän takaisin yksityisasiakkaiden taulukkonäkymään.
Peruuta-painike	Tyhjentää mahdollisesti täytetyt kentät ja vie käyttäjän takaisin yksityisasiakkaiden taulukkonäkymään.

Yksityisasiakkaan muokkaus:



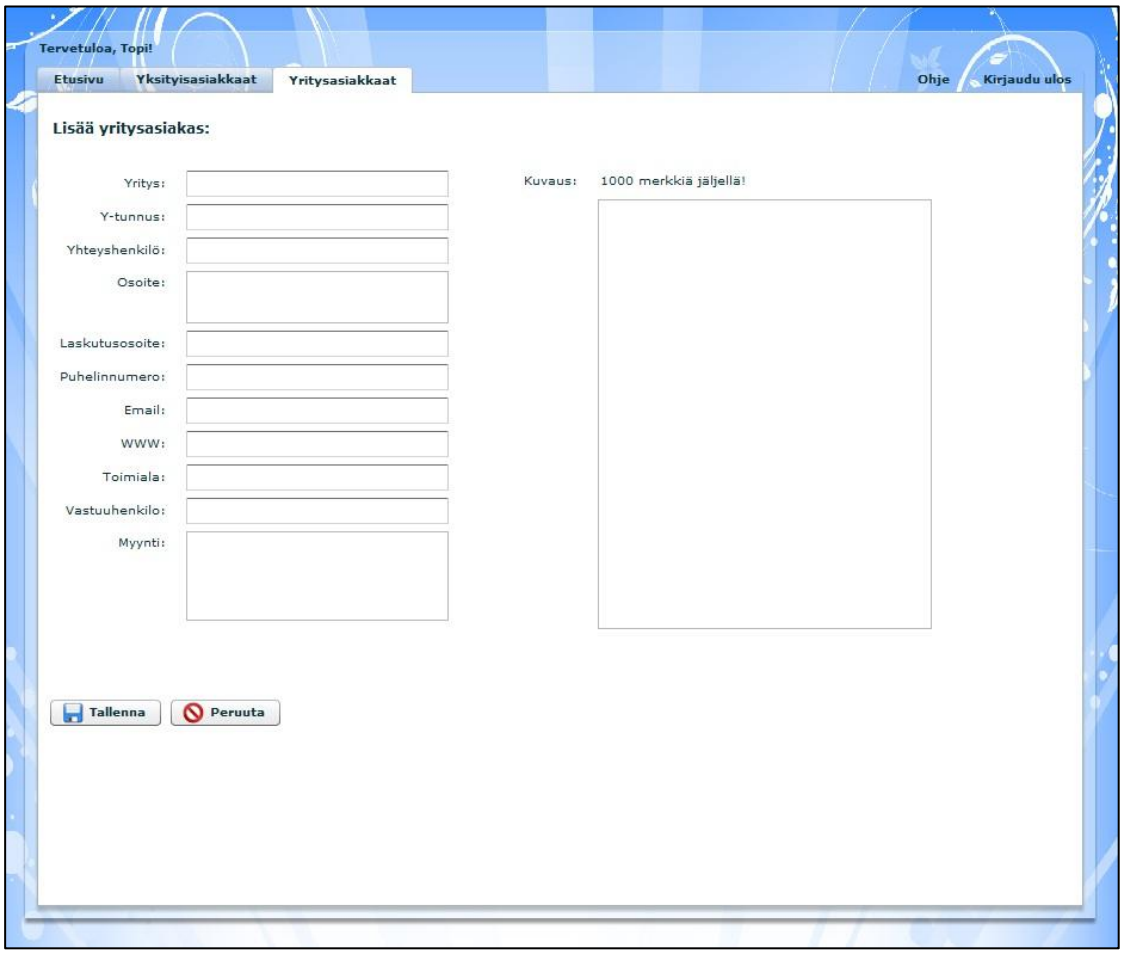
Komponentti	Kuvaus
Etusivu-välilehti	Vie käyttäjän järjestelmän etusivulle eli muistiinpanosivulle.
Yksityisasiakkaat-välilehti	Vie käyttäjän yksityisasiakkaiden taulukkonäkymään.
Yritysasiakkaat-välilehti	Vie käyttäjän yritysasiakkaiden taulukkonäkymään.
Ohje-linkki	Avaa järjestelmänkäyttöohjeet popup-ikkunaan.
Kirjaudu ulos-linkki	Siirtää käyttäjän ulos järjestelmästä ja kirjautumissivulle.
Yksityisasiakas-lomake	Käyttäjä voi muokata haluamansa asiakastiedot. Pakollisena kenttänä Nimi-kenttä. Lisäksi Kuvaus-kentässä merkkilaskuri.
Tallenna-painike	Painettaessa tarkistaa lomakkeen tiedot, tallentaa asiakkaan ja vie käyttäjän takaisin yksityisasiakkaiden taulukkonäkymään.
Peruuta-painike	Tyhjentää mahdollisesti täytetyt kentät ja vie käyttäjän takaisin yksityisasiakkaiden taulukkonäkymään.
Tulosta-painike	Avaa tulostus näkymän.

Yritysasiakkaat taulukkonäkymä:



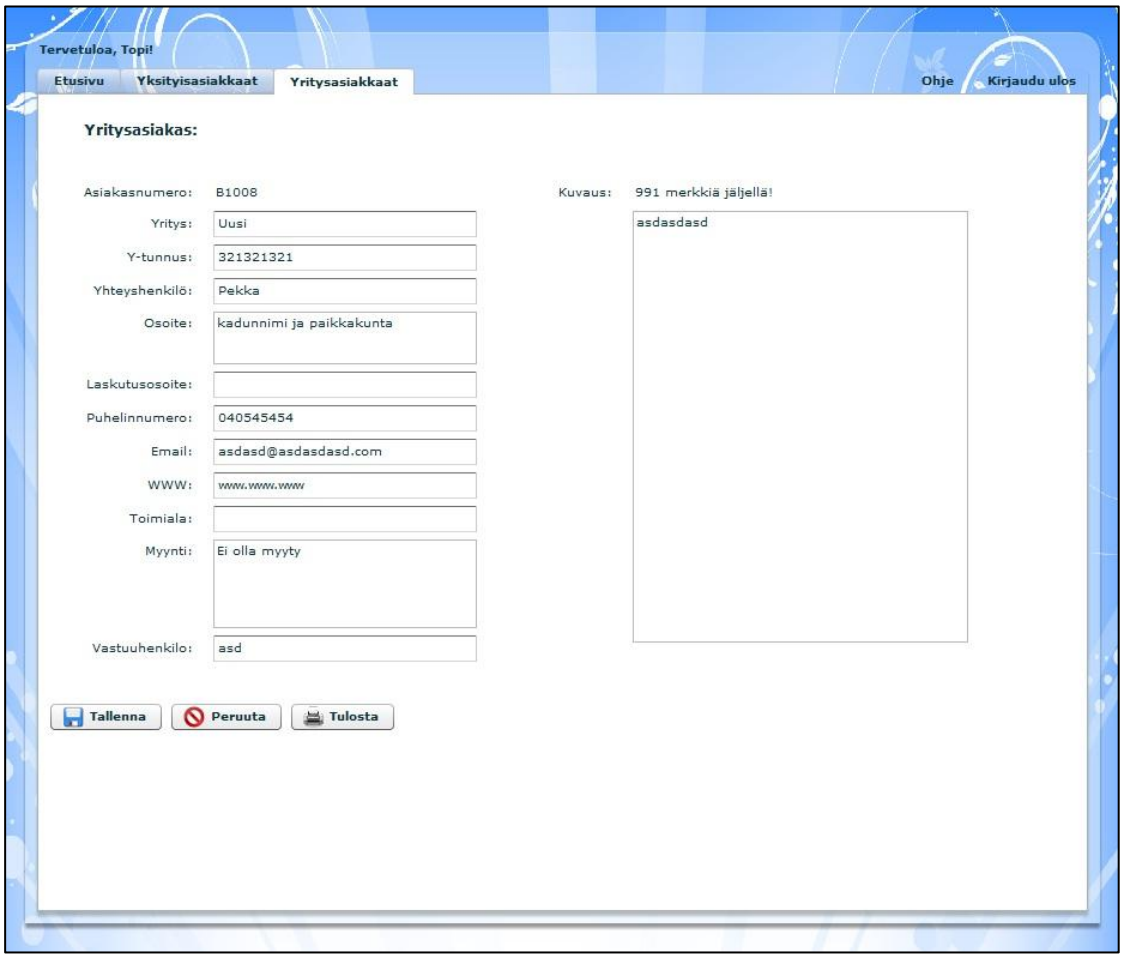
Komponentti	Kuvaus
Etusivu-välilehti	Vie käyttäjän järjestelmän etusivulle eli muistiinpanosivulle.
Yksityisasiakkaat-välilehti	Vie käyttäjän yksityisasiakkaiden taulukkonäkymään.
Yritysasiakkaat-välilehti	Vie käyttäjän yritysasiakkaiden taulukkonäkymään.
Ohje-linkki	Avaa järjestelmänkäyttöohjeet popup-ikkunaan.
Kirjaudu ulos-linkki	Siirtää käyttäjän ulos järjestelmästä ja kirjautumissivulle.
Hae yritys-kenttä	Käyttäjä voi syöttää merkkejä, jolloin taulukkonäkymä automaattisesti suodattaa esiin ne yritykset joissa kyseinen merkki tai merkit esiintyvät.
Taulukkonäkymä	Käyttäjä voi drag&drop-tyylisesti järjestellä taulukonsarakkeita ja niiden leveyksiä. Käyttäjä voi klikkaamalla sarakkeen otsikkoa järjestään sarakkeen tiedot pienimmästä suurimpaan tai aakkosjärjestykseen. Tuplaklikkaamalla haluamaansa riviä käyttäjälle aukeaa kyseisen rivin asiakastietojen muokkausnäkymä.
Lisää-painike	Avaa käyttäjälle yritysasiakkaan lisäys-näkymän.
Poista-painike	Poistaa poistonvahvistuksen jälkeen valitun asiakastiedon.
Päivitä-painike	Päivittää taulukkonäkymän.
Tulosta-painike	Avaa tulostus näkymän.

Yritysasiakkaan lisäys:



Komponentti	Kuvaus
Etusivu-välilehti	Vie käyttäjän järjestelmän etusivulle eli muistiinpanosivulle.
Yksityisasiakkaat-välilehti	Vie käyttäjän yksityisasiakkaiden taulukkonäkymään.
Yritysasiakkaat-välilehti	Vie käyttäjän yritysasiakkaiden taulukkonäkymään.
Ohje-linkki	Avaa järjestelmänkäyttöohjeet popup-ikkunaan.
Kirjaudu ulos-linkki	Siirtää käyttäjän ulos järjestelmästä ja kirjautumissivulle.
Lisää yritysasiakas-lomake	Käyttäjä voi syöttää haluamansa asiakastiedot. Pakollisena kenttänä Yritys-kenttä. Lisäksi Kuvaus-kentässä merkkilaskuri.
Tallenna-painike	Painettaessa tarkistaa lomakkeen tiedot, tallentaa asiakkaan ja vie käyttäjän takaisin yritysasiakkaiden taulukkonäkymään.
Peruuta-painike	Tyhjentää mahdollisesti täytetyt kentät ja vie käyttäjän takaisin yritysasiakkaiden taulukkonäkymään.

Yritysasiakkaan muokkaus:



Komponentti	Kuvaus
Etusivu-välilehti	Vie käyttäjän järjestelmän etusivulle eli muistiinpanosivulle.
Yksityisasiakkaat-välilehti	Vie käyttäjän yksityisasiakkaiden taulukkonäkymään.
Yritysasiakkaat-välilehti	Vie käyttäjän yritysasiakkaiden taulukkonäkymään.
Ohje-linkki	Avaa järjestelmänkäyttöohjeet popup-ikkunaan.
Kirjaudu ulos-linkki	Siirtää käyttäjän ulos järjestelmästä ja kirjautumissivulle.
Yritysasiakas-lomake	Käyttäjä voi muokata haluamansa asiakastiedot. Pakollisena kenttänä Yritys-kenttä. Lisäksi Kuvaus-kentässä merkkilaskuri.
Tallenna-painike	Painettaessa tarkistaa lomakkeen tiedot, tallentaa asiakkaan ja vie käyttäjän takaisin yritysasiakkaiden taulukkonäkymään.
Peruuta-painike	Tyhjentää mahdollisesti täytetyt kentät ja vie käyttäjän takaisin yritysasiakkaiden taulukkonäkymään.
Tulosta-painike	Avaa tulostus näkymän.