Sähköinen asiakasrekisteri

CASE: Vaajakosken hierontapalvelu

Leevi Mursula

Opinnäytetyö
Marraskuu 2010

Tietojenkäsittely
Luonnontieteiden ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä(t) MURSULA, Leevi	Julkaisun laji Opinnäytetyö	Päivämäärä 15.11.2010
	Sivumäärä 55	Julkaisun kieli Suomi
	Luottamuksellisuus ( ) saakka	Verkkojulkaisulupa myönnetty (X)
Työn nimi SÄHKÖINEN ASIAKASREKISTERI Case: Vaajakosken hierontapalvelu		
Koulutusohjelma Tietojenkäsittelyn koulutusohjelma		
Työn ohjaaja(t) TUIKKA, Tommi		
Toimeksiantaja(t) Vaajakosken hierontapalvelu HELMOLA, Mia		

Tiivistelmä

Opinnäytetyön tarkoituksena oli suunnitella ja toteuttaa Vaajakosken hierontapalvelulle sähköinen asiakasrekisteri, joka korvaisi tähän asti käytössä olleet paperiset asiakastietolomakkeet ja helpottaisi tietojen hakemista sekä arkistointia. Sovelluksen tuli olla käyttöjärjestelmäriippumaton ja käytettävissä vain yhdellä työasemalla. Asiakasrekisteristä tehtiin www-selaimella käytettävä.

Tutkimusosioissa selvitettiin ensin lähdemateriaaliin perustuen, millainen on käytettävyydeltään mahdollisimman hyvä web-lomake ja miten HTML-lomakkeita tulisi rakentaa, jotta täyttäminen olisi sujuvaa. Tämän jälkeen selvitettiin miten kehitysprojekti tulee toteuttaa ja mitä tulee ottaa huomioon kehityksen eri vaiheissa (määrittely, suunnittelu, toteutus, testaus, asennus ja ylläpito). Lopuksi toteutettiin asiakkaan vaatimusten mukainen asiakasrekisteri ja asennettiin se toimeksiantajan ympäristöön.

Tutkimuksesta voitiin päätellä, että web-pohjaiset lomakkeet on syytä miettiä tarkoin ja käyttää hyvän lomakkeen periaatteita hyödyksi. Kehitystutkimuksen tuloksena syntyi valmis asiakasrekisteri, joka täytti asiakkaan asettamat pakolliset vaatimukset, sekä sovelluksen kehityksen aikana syntynyt dokumentaatio. Resurssien vähyyden takia dokumentoinnissa ei menty kovin syvälliselle tasolle. Tärkeimpiä dokumentteja tai niiden osia lisättiin työn liitteisiin.

Aiheessa on potentiaalia jatkotutkimukselle ja –kehitykselle. Erittäin suuri potentiaali ja tarve voisi olla esimerkiksi ajanvaraustoiminnolle. Myös asiakkaan laskutukseen ja niiden rekisteröintiin liittyvät asiat saattaisivat kiinnostaa toimeksiantajaa tai vastaavia tahoja. Myös vaihtoehtoista toteutustapaa , kuten HTML5, voisi tutkia tulevaisuutta silmällä pitäen.

Avainsanat (asiasanat)
Asiakasrekisteri, web-sovellus, HTML-lomake, käytettävyys

Muut tiedot

Author(s)
MURSULA, Leevi

Type of publication
Bachelor´s Thesis

Date
15112010

Pages
55

Language
FInnish

Confidential

( ) Until

Permission for web publication
(X)

Title
ELECTRONIC CUSTOMER REGISTER
Case: Vaajakosken hierontapalvelu

Degree Programme
Business Information Systems

Tutor(s)
TUIKKA, Tommi

Assigned by
Vaajakosken hierontapalvelu
HELMOLA, Mia

Abstract

The objective of this thesis was to design and create an electronic customer register for Vaajakosken hierontapalvelu which would replace current customer forms and make searching and archiving easier. The system had to be operating system independent and usable from one computer only. The customer register was made to operate on a web-browser.

At first usability of forms and ways to build good HTML forms were determined, based on the source material. Next, the phases of software development (definition, design, implementation, testing, installation and management) and what needs to be taken care of when developing systems were studied. Finally, the customer register was created using the requirements from the assigner.

The conclusion of the study was that web-based forms have to be designed well and principles of good form used as guidelines. The result of the study was a documentation that was created during the project and a functional electronic customer register which filled all mandatory requirements set by the customer. The documents were created in somewhat restricted way because of the lack of the resources. Important documentations or part of them were included in the appendixes.

There could be potential to research this subject further. Appointment booking or billing register could have potential and as well the needed possibilities. Creating the customer register with alternate technique like HTML5 could also be interesting in the future.

Keywords
Customer register, web-application, HTML Form, usability

Miscellaneous

SISÄLTÖ

KUVIOT

TAULUKOT

KÄSITTEISTÖ

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

JavaScript on ohjelmointikieli, jota voidaan kirjoittaa suoraan HTML-koodin sekaan. Se mahdollistaa interaktiivisten toimintojen lisäämisen www-sivuille.

PHP (Hypertext Preprocessor) on ohjelmointikieli, jota käytetään mm. dynaamisten web-sivujen luonnissa.

SQL (Structured Query Language) on kyselykieli, jolla tietokantaan voidaan tehdä erilaisia hakuja, muutoksia ja lisäyksiä.

1 JOHDANTO

Asiakastietojen kerääminen ja ylläpitäminen on liiketoiminnassa erittäin tärkeää. Yrityksen on kyettävä jollain tavalla keräämään tietoa asiakkaistaan. Erityisesti terveys- ja hoitoalalla asiakastietojen on oltava kunnossa ja mielellään ajan tasalla. Terveydenhuollossa asiakasrekisterin tai -kortiston ylläpitäminen ja hoitojen kirjaaminen on lakisääteinen tehtävä. Yksityisen terveydenhuollon ammattihenkilön on toimitettava lääninhallitukselle toimintakertomus vuoden aikana tehdyistä hoidoista. Sähköiset sovellukset ja palvelut ovat nykyään paljon tehokkaampia ja nopeampia tapoja asiakastietojen hallintaan kuin paperiarkistot.

Vaajakosken hierontapalvelussa asiakastiedot ovat tähän asti olleet paperisilla yhteystietolomakkeilla, joihin on kerätty tarvittavat tiedot. Myös käynnit on merkitty asiakaskohtaiselle paperiselle lomakkeelle. Lomakkeita on saattanut olla asiakasta kohden useita, mikä on entisestään hankaloittanut arkistointia ja tietojen hakua sekä muokkaamista. Tässä asiassa nähtiin tarve muutokselle toimintojen tehostamiseksi. Kaupallisia vaihtoehtoja on olemassa, mutta toimeksiantajan pienuuden takia kaupallisiin sovelluksiin ei vielä tässä vaiheessa ollut suurta kiinnostusta. Myös toimeksiantajan tietoteknisen osaamisen ja tietämyksen puute puolsivat varta vasten räätälöityä ja mahdollisimman yksinkertaistettua järjestelmää.

Tämän opinnäytetyön tarkoituksena on suunnitella ja toteuttaa toimeksiantajalle sähköinen versio nykyisestä asiakasrekisteristä, jotta asiakkaiden tietojen ylläpitäminen ja arkistointi olisi helpompaa ja nykyaikaisempaa. Samalla tarkastellaan, millaisia etuja sähköisellä järjestelmällä saadaan mm. liiketoiminnallisesta näkökulmasta. Asiakasrekisteri tehdään alusta alkaen itse, joten tässä opinnäytetyössä tullaan tutustumaan sovelluskehityksen eri vaiheisiin. Tämä on myös tekijälle tärkein ammatillinen peruste aiheen valinnalle.

Sovellus tullaan toteuttamaan PHP-ohjelmointikieltä käyttäen, jotta asiakasrekisteristä saataisiin käyttöjärjestelmäriippumaton, selaimella käytettävä rekisteri. Valittu

ohjelmointikieli on tekijälleen jo entuudestaan tuttua sillä tasolla, että projektin tekeminen yhden miehen resursseilla on aikataulun ja vaatimusten puitteissa mahdollista.

2 TUTKIMUSASETELMA

Toisessa luvussa esitellään tutkimuksen tavoitteet ja rajaukset, toimeksiantaja sekä kuvataan käytettävät tutkimusmenetelmät. Lisäksi esitellään opinnäytetyön tutkimuskysymykset ja aikataulu.

2.1 Tavoitteet ja rajaukset

2.1.1. Toimeksiantaja

Toimeksiantajana opinnäytetyössä on Tmi Koulutettu hieroja Mia Helmola, aputoiminimeltään Vaajakosken hierontapalvelu (Y-tunnus 1534072-9). Keski-Suomessa sijaitseva yritys tarjoaa hyvinvointipalveluja, mm. hierontaa ja fysioterapiaa. Yritys on perustettu vuonna 1999. Nykyisiin tiloihinsa Vaajakosken keskustassa yritys siirtyi vuonna 2008. Työntekijöitä on kolme, joista yksi toimii tiloissa yrittäjänä. Yrityksessä on käytössä yksi kannettava tietokone sekä tulostin.

2.1.2. Tausta ja tavoite

Tutkimuksen tarkoituksena on selvittää ja tarkastella, miten Vaajakosken hierontapalvelun toimintaa voisi tehostaa tietotekniikan avulla, erityisesti asiakkaiden ja käyntien rekisteröinnin suhteen sekä luoda tältä pohjalta asiakasrekisterisovellus. Aiemmin asiakkaiden tiedot on kerätty paperisille lomakkeille, joihin on merkattu myös asiakkaan käynnit. Asiakas on itse täyttänyt lomakkeen, jota on sitten työntekijän toimesta täydennetty tarvittavin tiedoin, kuten mahdollisten lääkärien kuvausten osalta. Tämä on toiminut potilastietokorttina.

Toimeksiantajasta on tuntunut siltä, että asiakastietojen arkistointi ja varsinkin tietojen muokkaaminen tarvittaessa on ollut työlästä. Kortille on merkattu esimerkiksi tulosyy, saadut hoidot, asiakkaan oma kuvaus sekä nykyiset oireet. On selvää, että mikäli asiakas käyttää Vaajakosken hierontapalvelun palveluja hyväkseen useasti, kortin ylläpitäminen voi käydä erittäin työlääksi. Yhdestä asiakkaasta voi siis olla usei-

ta eri lomakkeita. Myös lomakkeiden kirjoitusala on rajallinen, joten käsinkirjoitettua tekstiä ei saa kovin pieneen tilaan mahtumaan, mikäli raportoitavaa on paljon.

Mahdollisten lähetteiden kirjoittaminen lääkärille on kynällä työläämpää kuin koneella kirjoitettuna. Myös tähän asiaan haluttiin muutosta, ja toteutettavassa rekisterisovelluksessa tulee olla mahdollisuus myös tällaisen lomakkeen kirjoittamiseen ja tulostukseen.

Toimeksiantaja on pohtinut mahdollisuutta kaupallisen asiakasrekisterin hankintaan, mutta haluaa ensin tarkastella mahdollisuutta yksinkertaisen ja heidän tarpeisiinsa soveltuvan ja riittävät toiminnallisuudet sisältävän sovelluksen luomisesta alusta lähtien. Opinnäytetyönä tällainen on mahdollista toteuttaa suhteellisen kustannustehokkaasti.

Tämän opinnäytetyön tavoitteena on ensisijaisesti tehdä toimeksiantajalle hänen vaatimustensa mukainen, selkeä ja helppokäyttöinen asiakasrekisteri. Lisäksi tekijä toivoo saavansa lisää tietotaitoa ohjelmistokehityksen alueelta.

2.1.3. Tutkittavan alueen rajaus

Tutkimuksessa käsitellään sovelluksen määrittelyä, suunnittelua ja toteutusta. Näissä sovelletaan sovelluskehityksen periaatteita, mutta laajuus ja dokumentointi pidetään sellaisena, että työ on mahdollista saada ajan ja resurssien puitteissa valmiiksi. Erityisesti suunnittelun ja toteutuksen osalta valitaan sellaisia menetelmiä, jotka ovat tekijän kannalta hyödyllisiä. Rekisterin käytön tueksi kirjoitetaan selventävä käyttöopas. Teoriaosuudessa pyritään tarkastelemaan mm. lomakkeiden käytettävyyttä, sillä lopullinen sovellus tulee pääasiassa käyttämään lomakkeita tiedon syöttöön ja esittämiseen.

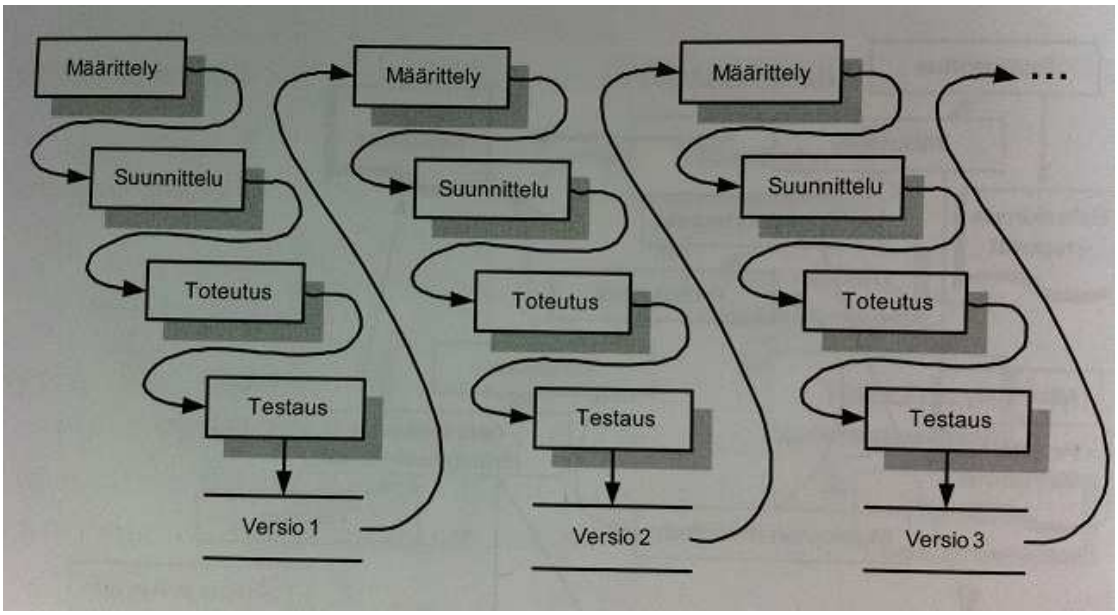
Asiakasrekisterissä ei tulla käyttämään erityisiä käyttäjätasoja, sillä työntekijöiden määrä ja se, että sovellus tulee toimimaan vain paikallisesti, eivät anna tähän aihetta.

2.2 Tutkimusmenetelmät

Tämä opinnäytetyö on kehittämistutkimus. Työssä kehitetään projektiluontoisella hankkeella toimeksiantajan vaatimusten mukainen, selainpohjainen asiakasrekisteri. Tutkimuksessa kartoitetaan, millaisen sovelluksen toimeksiantaja tarvitsee. Tätä selvitetään mm. keskustelemalla toimeksiantajan kanssa ja tutustumalla nykyiseen paperiseen versioon asiakasrekisteristä.

Työssä ilmenevät kehittämisprojektille tyypilliset piirteet, joita ovat: ongelmalähtöisyys, osallistuvuus, suunnitelmallisuus, kertaluonteisuus, tavoitteellisuus ja rajalliset resurssit. Resursseja ovat tekijä, asiakkaan panos ja aikataulu. Työ on tarkoitus viedä läpi ilman erityisiä kuluja, joten budjetti on käytännössä 0 €. Kertaluonteisuus tulee ilmi siten, että kehitettävää järjestelmää ei voida suoraan käyttää muissa yhteyksissä ilman räätälöintiä. Pyrkimys on kuitenkin saada aikaan sellainen lopputulos, että mahdollinen räätälöinti toiselle asiakkaalle ei veisi liikaa aikaa ja resursseja.

Projekti toteutetaan käyttäen ohjelmistokehityksen periaatteita ja siihen liittyviä perusvaiheita: määrittely, suunnittelu, toteutus ja ylläpito, johon liittyvät lähinnä asennus ja koulutus sekä ohjeen kirjoittaminen. Projektin toteuttamisessa käytetään sovelletusti prototyyppimallia (kuvio 1), sillä se soveltuu tämän tyyppiseen projektiin, jossa tekijälle annetaan suhteellisen paljon vapauksia, eikä toimeksiantajalla ollut aivan selvää ajatusta lopputuloksesta.



KUVIO 1. Esimerkki prototyyppimallista (Haikala & Märijärvi 2006, 44)

2.3 Tutkimuskysymykset

Tutkimuskysymykset ilmaisevat tutkimusongelmia, jotka on esitetty kysymysten muodossa. Kysymykset on tarkoitus esittää mahdollisimman selkeinä. Työssä pyritään vastaamaan seuraaviin kysymyksiin:

1. Millaisen sovelluksen asiakas tarvitsee?

2. Miten kehitysprojekti toteutetaan?

3. Miten käytettävyys on otettava huomioon lomakkeiden suunnittelussa?

2.4 Aikataulu

Opinnäytetyö aloitettiin keväällä 2010. Tavoitteena oli saada työ valmiiksi syksyn 2010 aikana, ja tarkemmin ottaen valmista tuli olla siten, että opinnäytetyön ehtii palauttaa arvioitavaksi vuoden 2010 aikana. Kyseisestä aikataulusta ei ollut varaa myöhästyä, sillä opinto-oikeusaika alkoi olla vähissä. Toukokuussa 2010 idea opinnäytetyöstä alkoi olla selvillä, mutta pääasiassa ideat kerääntyivät kesän aikana. Tuol-

loin suoritettiin lähinnä suunnitelmia aikataulun suhteen, sekä kerättiin jonkin verran lähdemateriaalia. Tutkimussuunnitelma tehtiin heti elokuussa 2010, kun toimeksiantajalta oli saatu tarkempaa tietoa. Aloitusseminaari pidettiin tämän opinnäytetyön osalta 6.10.2010. Lopetusseminaari pidetään, kun työ on valmis (tavoite 15.11.2010).

TAULUKKO 1. Aikataulu

Aikataulu	Käytetty aika (h)	Pvm
Kokonaiskesto	397	1.5.-15.11.2010
Lähdeaineiston hankinta	35	1.5.-1.10.2010
Tietoperustan rakentaminen	60	1.7.-1.9.2010
1. Johdanto	8	18.10.2010
2. Tutkimusasetelma	8	19.10.2010
Tavoitteet ja rajaukset	3	
Tutkimusmenetelmät	2	
Tutkimuskysymykset	2	
Aikataulu	1	
3. Käytettävyyden huomioiminen web-pojaisissa lomakkeissa	10	20.10.2010
Lomakkeiden perusidea web-sivustolla	3	
Lomakkeiden rakenne	3	
Lomakkeiden käytettävyys	4	
4. Sovelluksen määrittely (sis. Palaverit toimeksiantajan kanssa)	30	1.8.-21.10.2010
Esitutkimus	14	
Vaatimusmäärittely	8	
Käyttötapaukset	8	
5. Sovelluksen suunnittelu	50	15.8.-23.10.2010
Käyttöliittymäsuunnittelu	20	
Arkkitehtuuri	5	
Tietokannan suunnittelu	25	
6. Sovelluksen toteutus	120	1.9.-25.10.2010
Toteutusympäristö	10	
Tietokannan luonti	30	
Kooditiedostot (sis. Koodaus)	60	
Testaus	20	
7. Sovelluksen käyttöönotto	45	8.-16.10.2010
Sovelluksen asennus toimeksiantajan ympäristöön	15	
Sovelluksen koulutus toimeksiantajalle	10	
Sovelluksen käyttöohje ja tietokannan varmuuskopiointi	20	
8. Tutkimustulosten analysointi	8	27.-28.10.2010
9. Pohdinta	8	5.-7.10.2010
Tiivistelmät, työn muokkaus ja viimeistely	15	8.-15.10.2010

3 KÄYTETTÄVYYDEN HUOMIOIMINEN WEB-POHJAISISSA LOMAKKEISSA

Tässä luvussa perehdytään tiedon syöttämiseen tarkoitettuihin lomakkeisiin ja niiden käytettävyyteen sekä hyvän lomakkeen periaatteisiin. Pääasiassa keskitytään asiakastietolomakkeisiin. Miten mahdollisimman selkeä lomake tulisi luoda, jotta lomakkeen käyttäjä voisi toimia tehokkaasti ja se olisi mahdollisimman esteetön? Lomakkeita on monesti tarpeen myös tulostaa tai ainakin esittää tietoa tulostettavassa muodossa. Tämäkin on otettava huomioon suunnittelussa. Luvussa esitettyjä periaatteita pyritään noudattamaan varsinaisen asiakasrekisterin kehityksen suunnittelu- ja toteutusvaiheessa.

3.1 Lomakkeiden perusidea web-sivustolla

Lomakkeet ovat web-sivuilla vuorovaikutteisuuden väline. Käyttäjä voi lomakkeiden kautta vaikuttaa jopa sivuston rakenteeseen. Lomake itsessään on osa käyttöliittymää, mutta tieto, jonka käyttäjä syöttää tekstikenttiin, valintalaatikoihin jne., on sivuston sisältöä. Yleisimmin lomake on esimerkiksi hakukenttä, johon voi syöttää tarkoitukseen sopivan hakusanan, minkä jälkeen lomakkeelle kirjoitettu tieto lähetetään selaimen käsiteltäväksi. Palautteena saadaan esimerkiksi hakutulos. Lomake koostuu siis yksinkertaisimmillaan normaalisti tekstikentästä tai valintaruudusta ja painikkeesta, jolla haluttu tapahtuma suoritetaan (form action). Lomakkeiden avulla voidaan myös muokata tietokannassa olevaa tietoa (tämä on tärkein lomakkeen olemassaolon peruste tässä kehitysprojektissa). Hyvin usein juuri tämä on sivustoilla olevien lomakkeiden tarkoitus. (Korpela & Linjama 2005, 268–269.)

Web-sivulla voi olla myös muita kuin HTML-lomakkeita. Sivustolle voidaan esimerkiksi liittää PDF-, Word- ja Excel-lomakkeita, jotka käyttäjä voi ladata koneelle ja täyttää tarvittavilla tiedoilla. Tällaisten lomakkeiden ongelma on kuitenkin se, että ne täytyy lähettää vastaanottajalle erikseen esimerkiksi sähköpostilla. Tämä tapa toimii hyvin mm. julkishallinnon palveluissa. Näin ollen asiakas voi itse tulostaa tarvitsemansa lomakkeet ja käsittely nopeutuu. (Korpela ym. 2005, 273.)

Ulkoasullisesti lomakkeissa ei ole valinnanvaraa. Mikäli ulkoasua halutaan muokata, on se yleensä tehtävä CSS-tyylitiedostojen kautta. Tulevaisuudessa HTML5 antaa kuitenkin lisää mahdollisuuksia ja ominaisuuksia myös lomakkeiden osalta, sillä se mahdollistaa monia uusia attribuutteja kehittäjien käyttöön. Tällä hetkellä HTML5 ei ole kuitenkaan erityisen hyvin tuettu ainakaan käytetyimpien selainten osalta. Tilanne tuntuu kuitenkin parantuvan, sillä myös Microsoft on siirtymässä HTML5:n kannalle. (Muglia 2010, TechCrunch.)

3.2 Lomakkeiden rakenne

Lomakkeen rakenne määrittää, minkä tyyppisiä kenttiä tarvitaan. HTML-lomakkeen syöttöön tarkoitettuja elementtejä on kolme: input, textarea ja select. Kenttien tyyppi määritellään type-osassa, joka kertoo, millaisen kentän selain käyttäjälle näyttää (esim. input type=”text” näkyy käyttäjälle yksirivisenä tekstialueena, johon voi syöttää rajatun määrän tekstiä). Tärkeimpiä ovat:

- Submit = luo painikkeen, jolla lomakkeen tiedot lähetetään selaimelle käsiteltäväksi.

- Hidden = välittää valmiiksi luotua dataa, eikä ole käyttäjän nähtävissä eli on ns. piilokenttä.

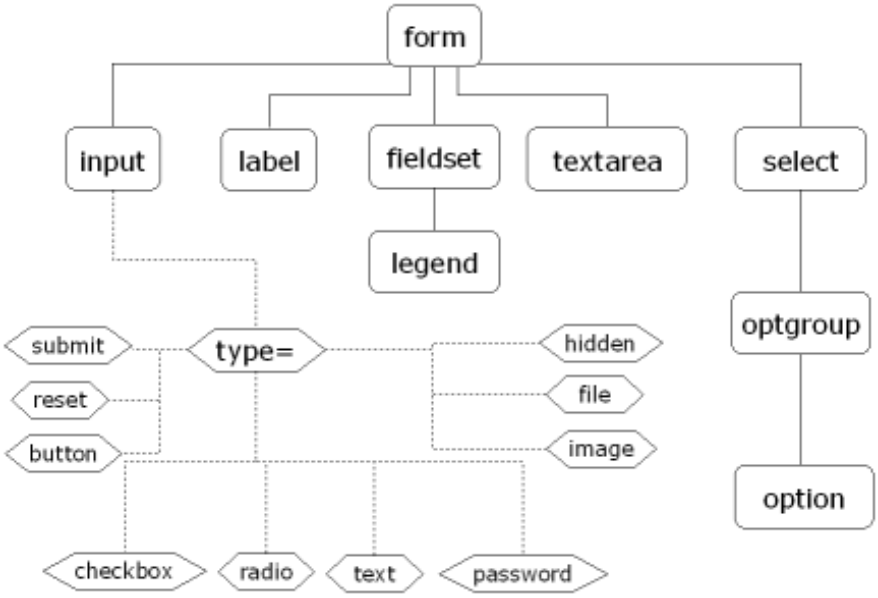
- Text = luo kentän, johon voi syöttää yhden rivin tekstiä. Vastaava muunnos tästä on password-kenttä, joka muuttaa kirjoitetun tekstin tilalle esim. mustia pisteitä, jotta muut eivät voi lukea ruudulta, mitä käyttäjä kenttään kirjoittaa.

- Textarea = tekstikenttä, jonka koon voi itse määrittää ja siihen voi syöttää tekstiä useamman kuin yhden rivin.

- Select = voidaan tehdä valikkokenttä, johon määritetään halutut vaihtoehdot, joista käyttäjä voi valita yhden tai joskus useamman vaihtoehdon.

- Radio, checkbox = käyttäjä voi valita haluamansa vaihtoehdot eri tyyliin kuin valikkokentässä, nappien avulla

- (Korpela ym. 2005, 273.)



KUVIO 2. Lomakkeen elementit (Ekonoja ym. 2010, WWW-lomakkeet.)

3.3 Lomakkeiden käytettävyys

Lomakkeet on syytä rakentaa tiettyjä käytettävyyden perusperiaatteita noudattaen. Seuraavassa esitellään hyviä muistisääntöjä, mitä kannattaa muistaa, kun web-sivustoa suunnitellaan. Seikat pätevät myös sähköisen lomakkeen suunnitteluun, joten on hyvä ottaa ideat esille ja tutkia, miten päästäisiin miellyttävään käyttökokemukseen. ISO 9241-11:n mukaan käytettävyys on aina kiinni myös kontekstista, eli käyttäjiä ja tarpeita on erilaisia. Tällöin onkin tärkeää, että käyttäjät ja mahdolliset käyttötilanteet ovat jollain tavalla tunnettuja. Ilman käyttäjiä ei ole käytettävyyttä, eikä sitä voi silloin mitata. (Parkkinen 2002, 31–32.)

3.3.1. Nielsenin käytettävyyden teesit

Käytettävyys on hyvin tärkeä, ellei tärkein osa verkkosivuston suunnittelussa ja toteutuksessa. Sama asia koskee myös sivustolla olevia lomakkeita. Jakob Nielsenin teorian mukaan käytettävyys jakaantuu viiteen tekijään:

- Opittavuus, eli miten hyvin käyttäjä osaa toimia ensimmäisellä kerralla.
- Tehokkuus, eli voiko sivuston tai sovelluksen niksit oppinut käyttää sitä tehokkaammin ja saada enemmän hyötyä.
- Muistettavuus, eli onko käyttö helppoa, kun sen on oppinut.
- Virheettömyys, eli voiko käyttäjä tehdä paljon virheitä, ja jos voi, niin miten niistä ilmoitetaan.
- Miellyttävyys

Samat seikat pätevät myös muuhunkin kuin pelkästään web-sivustoihin tai ohjelmiin, kuten esimerkiksi laitteisiin ja niiden käytettävyyteen. (Parkkinen 2002, 28.)

3.3.2. Lomakkeen rakentamisen peruspilareita

Lomakkeiden tekoon annetaan monissa lähteissä ohjeita. Yksi tällainen lähde on Tietoyhteiskunnan kehittämiskeskus ry:n esteettömyysopas (Korpela 2003). Opas listaa kahdeksan kohtaa, jotka tulisi ottaa huomioon, kun lomakkeita rakennetaan websivulle. Oli kyseessä sitten millainen lomake tahansa, esitellyt kohdat olisi hyvä pitää mielessä.

1. Kirjoita lomake-elementti (form element) niin suppeaksi, että sen sisällä on vain itse lomake täyttöohjeineen.

2. Kirjoita ennen lomakkeen syöttökenttiä kaikki ne tiedot, jotka käyttäjän on syytä tietää, ennen kuin rupeaa täyttämään lomaketta.

3. Kirjoita lomakkeen kentät HTML-dokumenttiin sellaiseen järjestykseen, joka vastaa lomakkeen normaalia täyttämisjärjestystä.

4. Anna ennen kutakin kenttää selitys siitä, mikä tieto käyttäjää pyydetään antamaan ja missä muodossa, esimerkiksi näin: Anna nimesi (sukunimi, etunimi).

5. Vältä tilanteita, joissa käyttäjä joutuu valitsemaan kovin monesta valmiista vaihtoehdosta. Käyttäjälle voi olla helpompi käyttää tekstinsyöttökenttää.

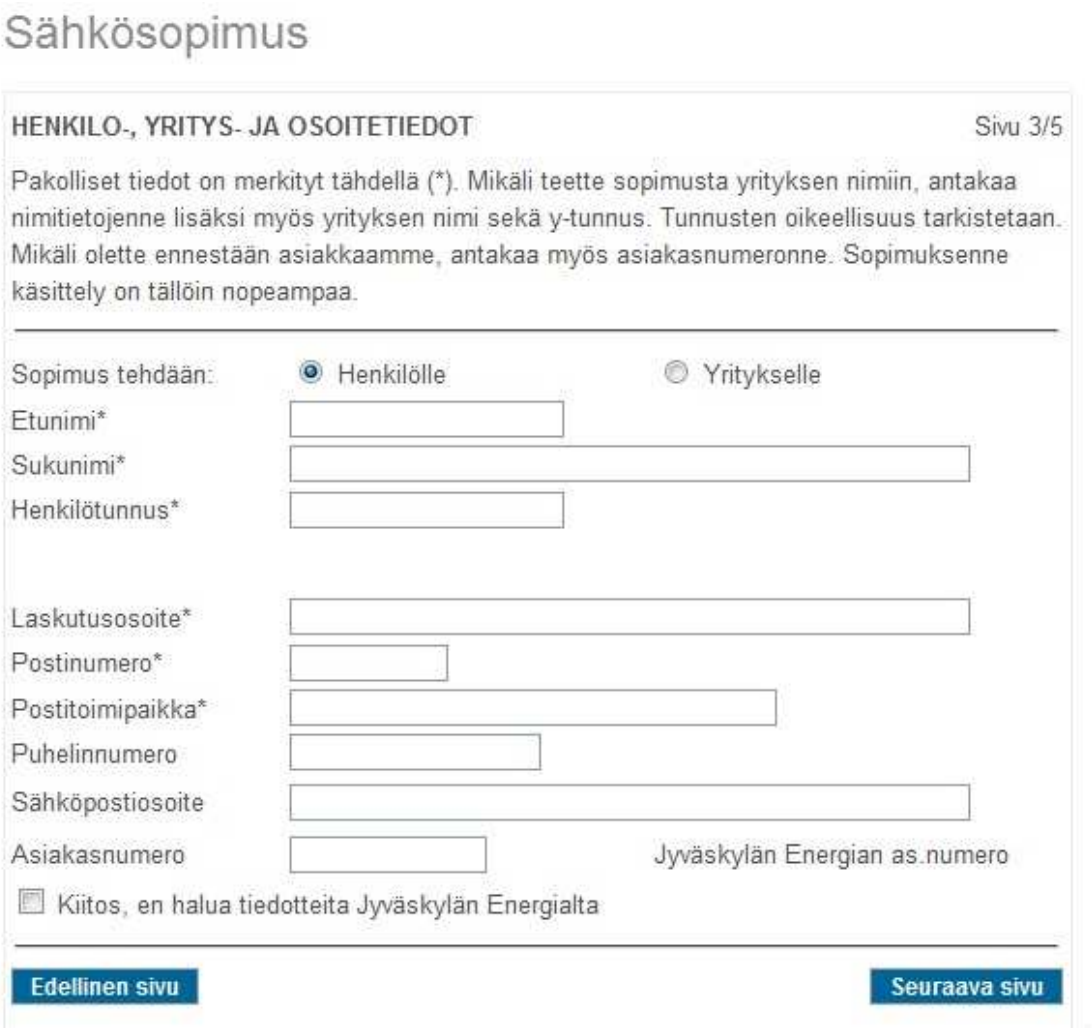
6. Älä sisällytä lomakkeeseen <input type=”reset”> -kenttää tai sijoita se ensimmäiseksi.

7. Sisällytä lomakkeeseen aina <input type=”submit”> -kenttä tai vastaava (viimeiseksi kentäksi tietysti).

8. Vältä käyttämästä <input type=”submit”> -kentän asemesta ”kuvapainiketta” <input type=”image”>. Jos kuitenkin käytät kuvapainiketta, kirjoita sen kuvalle vaihtoehtoinen teksti kolmeen kertaan: alt-määritteen, name-määritteen ja value-määritteen arvoksi.

(Korpela 2003, 31.)

Esimerkkinä yksinkertaisesta ja hyvästä lomakkeesta käy Jyväskylän Energian sivuilla oleva sähkösopimuksen tekoon tarkoitettu lomake (kuvio 3). Lomakkeelta selviävät hyvin pakolliset kentät, ohjeet täyttöön on annettu lomakkeen alussa, täyttäminen etenee loogisesti ja käyttäjälle vielä ilmoitetaan, missä vaiheessa sopimusta ollaan menossa.



KUVIO 3. Sähkösopimuksen tekoon tarkoitettu lomake (Sähkösopimus. Jyväskylän Energia 2010)

Lomakkeen suunnittelussa tulisi miettiä, miten käyttäjä täyttää lomaketta. Käyttäjän huomio pitää ohjata tärkeisiin kohtiin. Lomakkeella on aina oltava otsikko, pakolliset kentät on ilmoitettava ja turhaa tietoa ei saa antaa eikä kysyä. Toteutuksessa on syytä huomioida se, että yleensä hyvän lomakkeen täyttämisen täytyisi onnistua myös pelkkää näppäimistöä käyttäen. Kun lomakkeen lähetyspainiketta on painettu, tulee käyttäjän saada aina ilmoitus tapahtuman onnistumisesta tai mikäli syötteissä on puutteita, tulee siitä ilmoittaa käyttäjälle. (Parkkinen 2002, 101–104.)

Kenttiin kannattaa tehdä esimerkiksi JavaScript -tarkistukset, mikäli tietojen on oltava tietyssä muodossa. Hyvin usein sähköpostikentässä on tarkistus, jotta käyttäjä

syöttäisi teknisesti toimivan osoitteen. Osoitteen olemassaolosta ei voi kuitenkaan tässä vaiheessa olla varma. Tarkistuksissa kannattaa kuitenkin käyttää harkintaa, sillä liian pikkutarkat säännökset saattavat hidastaa käyttäjän toimia tai jopa karkottaa tämän pois sivustolta. (Korpela ym. 2005, 294.)

Paperisen lomakkeen pohjalta luotu sähköinen versio (esim. asiakastietojen keräämisen tarkoitettu lomake) ei saisi kuitenkaan olla välttämättä suoraan yksi yhteen -kopio. Luonnissa on otettava huomioon, että HTML-lomakkeelle kirjoitettu tieto menee aina ohjelman käsiteltäväksi. (Korpela ym. 2005, 268.)

4 SOVELLUKSEN MÄÄRITTELY

Luvussa neljä käydään läpi Vaajakosken hierontapalvelulle tehtävän asiakasrekisterin ensimmäinen vaihe eli määrittely. Toimeksiantajan kanssa käytyjen keskustelujen sekä saadun materiaalin perusteella asiakasrekisterisovellukselle laadittiin esitutkimuksen mukaisesti vaatimusmäärittely, käyttötapaukset ja käyttöliittymäsuunnittelu. Kaikki vaatimukset ja ideat hyväksytettiin asiakkaalla ennen varsinaisen suunnittelun aloittamista.

4.1 Esitutkimus

Varsinaisen tutkimuksen ensimmäisenä vaiheena oli keskustella toimeksiantajan kanssa vallitsevasta tilanteesta ja tutustua ympäristöön, johon sähköistä asiakasrekisteriä tarvittiin. Heti alkuun oli selvää, että toteutuksen tulisi olla mahdollisimman yksinkertainen. Toimeksiantajalta saatiin omat kappaleet asiakastietokortteja (liite 1), joille Vaajakosken hierontapalvelun asiakas täyttää omat tietonsa ensimmäisen käynnin yhteydessä. Muun muassa arkistoinnin ja tietojen muokkaamisen helpottamiseksi toimeksiantaja haluaa kirjata tiedot paperilomakkeelta sähköiseen muotoon. Paperisilta lomakkeilta oli suhteellisen helppo merkitä kaikki ne kohdat ja kentät, jotka toimeksiantajan mielestä olivat olennaisia ja tarpeellisia. Eräänä toimeksiantajan työtä helpottavana asiana tuli keskustelun myötä ilmi se, että paperisen lomakkeen etsiminen ottaa oman aikansa, vaikka ne olisivatkin kansiossa aakkosittain järjestettynä. Tietokoneella toimivaan järjestelmään voidaan tehdä käyttäjähaku, joka nopeuttaa tietojen tarkastamista.

Toimeksiantajalla on käytössään kannettava tietokone ja Internet-yhteys. Toiveena oli, että sähköinen asiakasrekisteri ja sen tietokantaan syötetyt tiedot olisivat käytössä vain tällä yhdellä tietokoneella. Asiakkaiden yksityisyydensuojan takia ulkopuolista palveluntarjoajaa ei haluttu. Tässä tapauksessa tarvetta rekisterin käyttöön muulta päätteeltä tai Internet-yhteyden yli ei ollut tarvetta. Käyttäjiä asiakasrekisterillä tulisi olemaan kahdesta kolmeen, eli käytännössä koko Vaajakosken hierontapalvelun henkilökunta. Erillisille tunnuksille ei kuitenkaan toimeksiantajan puolelta nähty tar-

vetta. Tästä huolimatta asiakasrekisteriä päätettiin lähteä esitutkimuksen perusteella kehittämään selainpohjaiseksi.

Käynnit kirjataan asiakkaan käyttämän lomakkeen kääntöpuolelle. Tässä hankaluutena on toimeksiantajan mielestä lomakkeen rivien rajallinen määrä, jolloin täyttöä joudutaan jatkamaan uudelle lomakkeelle. Tällöin asiakkaalla voi olla useita lomakkeita. Myös käynnin toimenpiteen kuvauskenttä on paperilla kooltaan rajallinen, mutta sähköisessä rekisterissä tähän voidaan vaikuttaa. Käyntien tulisi olla myös helposti lisättävissä asiakaskohtaisesti.

Esitutkimuksen tarkoituksena on mm. tarkastella eri toteutusvaihtoehtoja, varsinkin kun kyseessä on alusta asti tehtävä sovellus. Java-toteutus kirjattiin alkuvaiheessa esitutkimusdokumenttiin vaihtoehtoisena menetelmänä. Ensisijainen vaihtoehto (PHP ja MySQL) tuli kuitenkin valituksi. Tähän valintaan vaikuttivat niin tekijän jo olemassa oleva kokemus PHP-ohjelmointikielestä, kuin käytettävissä olevat resurssitkin.

Määrittelyvaiheessa tilanteen kartoittamista vaikeuttivat hieman toimeksiantajan tietoteknisten taitojen puute, sekä selkeän vision puuttuminen. Näin ollen tekijä sai projektissa melko vapaat kädet. Määrittelyn osalta tapaamisia piti järjestää useampi, jotta kokonaiskuva tarpeista saatiin selkeämmälle tasolle.

4.2 Vaatimusmäärittely

Esitutkimuksessa esille tulleiden seikkojen perusteella listattiin perusvaatimukset, joita sovelluksen tulisi täyttää. Vaatimusten perusteella voidaan todeta, tuliko järjestelmästä sellainen kuin haluttiin. Vaatimukset jaetaan toiminnallisiin ja ei-toiminnallisiin vaatimuksiin. Toiminnallisilla vaatimuksilla kuvataan ohjelmiston toteutettavat ominaisuudet, otetaan mahdollisesti kantaa käyttöliittymään liittyviin kysymyksiin ja kommunikointiin muiden järjestelmien kanssa. Ei-toiminnallisilla vaatimuksilla otetaan kantaa esimerkiksi vasteaikoihin, suoritustehoon ja käytettävyyteen. Vaatimusten tulee olla mahdollisimman selkeitä, eikä niihin saa jättää liiaksi

varaa tulkinnalle. Varsinkin ei-toiminnallisten vaatimusten tulee olla mitattavissa olevia. Näin ollen esimerkiksi selainten osalta piti vaatimukseen merkata selkeästi, mitä selaimia varmasti tuetaan ja mikä versio tai mistä versiosta eteenpäin.

Vaatimuksiin merkittiin listausvaiheessa, onko vaatimus pakollinen vai valinnainen. Tähän päädyttiin määrittelyvaiheessa siksi, että toimeksiantajan kanssa ideoita heiteltiin avoimesti ja esille tuli myös joitakin ideoita, joita ei aikataulun puitteissa olisi ollut mahdollista tehdä. Samoin opinnäytetyön rajauksen vuoksi oli pakko tiputtaa tiettyjä, hyviä ajatuksia pois. Myös tekijän ohjelmointiosaaminen vaikutti siihen, että osa vaatimuksista oli lähes pakko laittaa merkinnällä valinnainen. Valinnaiset vaatimukset olivat kuitenkin sellaisia, että ne eivät toimeksiantajan prioriteetissa olleet korkealla. Pakollisina päävaatimuksina voidaan työn osalta pitää asiakastiedon lisäystä ja muokkausta, käynnin lisäystä asiakkaalle sekä tulostettavaa palautelomaketta.

4.3 Käyttötapaukset

Toiminnalliset vaatimukset purettiin käyttötapauksiksi, joilla pyritään kuvaamaan järjestelmän toiminnallisuus käyttäjän tekemien toimien kautta. Käyttötapaukset kirjattiin käyttötapauskaavioilla. Näissä kaavioissa kerrotaan yleensä tekstimuotoisesti, käyttäjän näkökulmasta, mitä kunkin toiminnallisuuden suorittaminen vaatii. Lisäksi kerrotaan, mitkä ovat esiehdot toiminnallisuudelle, ketkä ovat suorittajia, kuvaus käyttötilanteesta, mahdolliset poikkeukset ja lopputulos. Lisäksi voidaan vielä kertoa muut vaatimukset, joita käyttötapaukselta vaaditaan. Esimerkki tällaisesta use-case -käyttötapauksesta esitetään kuviossa 4 (yksi tämän opinnäytetyön käyttötapauksista). Käyttötapaukset toimivat hyvänä pohjana käyttöliittymän suunnittelussa. (Haikala & Märijärvi 2006, 159.)

Käyttötapaus: Käyttäjä voi lisätä asiakkaan

Suorittaja:	Käyttäjä
Esiehdot:	Järjestelmän on käynnissä, käyttäjä kirjautunut, lisää asiakas painettu
Kuvaus:	Käyttäjä syöttää asiakkaan tiedot niille varattuihin kenttiin [poikkeus: virheelliset tiedot] ja painaa tallenna-painiketta [poikkeus: käyttäjä on jo olemassa].
Poikkeukset:	Käyttäjä on jo olemassa: käyttäjälle ilmoitetaan jo olemassaolevasta tunnuksesta. Virheelliset tiedot: käyttäjä on syöttänyt vääriä merkkejä esim. sähköpostiosoite -kenttään.
Kuvitus:	-
Jälkiehdot:	Käyttäjä on lisännyt asiakkaan järjestelmään ja saa siitä ilmoituksen.

KUVIO 4. Esimerkki käyttötapauksen esitystavasta

5 SOVELLUKSEN SUUNNITTELU

Viidennessä luvussa suunnitellaan sovellukselle vaatimusmäärittelyn perusteella käyttöliittymä, arkkitehtuuri ja tietokanta. Myös vaatimusmäärittelyä tarkennetaan tarpeen mukaan. Varsinkin tietokannan valintaan ja suunnitteluun tulee kiinnittää erityistä huomiota, sillä asiakasrekisteri on tietokannassa, jota selaimelle rakennettavalla sivustolla hallitaan vaatimusten mukaisesti. Tietokannasta tehdään ER-kaavio, jonka pohjalta suunnitellaan tietokannan taulut.

5.1 Käyttöliittymäsuunnittelu

Käyttöliittymäsuunnittelussa ei ole tarkoitus ainoastaan suunnitella ohjelman tai sivuston ulkoasua. Visuaalisen ilmeen suunnittelu on toki tärkeintä, mutta suunnittelussa tulee ottaa huomioon myös muut kuin ulkonäölliset seikat. Painikkeet ja toiminnot tulisi suunnitella siten, että käyttäjä hyötyy sovelluksesta mahdollisimman paljon ja käyttö on loogista. Tässä auttaa käyttöliittymäoperaatioiden listaaminen. Määrittelyvaiheessa kirjoitetut käyttötapaukset voidaan purkaa operaatioihin. Lista kannattaa jakaa mahdollisuuksien mukaan ryhmiin toiminnan mukaan.

Dialogikaaviolla saadaan esitettyä ns. ”pohjapiirustus”, johon on merkitty kaikki näkymät ja se, mitä operaatioita näkymä sisältää. Tämän perusteella on hyvä siirtyä luomaan visuaalista ulkoasua. Dialogikaavion avulla voidaan tehdä suunnitelmat näkymien ulkoasusta ja merkitä kunkin näkymän komponentit. Komponentit ovat siis sivustolle tulevat toiminnot, esimerkiksi ”Lisää käyttäjä” -nappi. Tämän työn käyttöliittymäsuunnitelman näkymät on esitelty liitteessä 4.

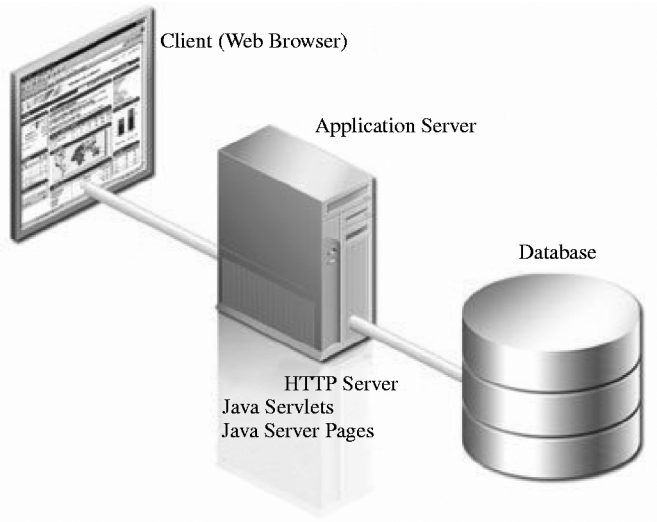
Käyttöliittymän visuaaliseen puoleen voidaan suunnitteluvaiheessa vaikuttaa mm. käyttöliittymädemolla, jossa tehdään ulkoasultaan valmiin näköinen versio, mutta josta toiminnallisuudet puuttuvat. Tämän projektin osalta luotiin käyttöliittymädemo, jota esiteltiin toimeksiantajalle suhteellisen aikaisessa vaiheessa. Ulkoasullisesti työssä käytettiin Vaajakosken hierontapalvelun Internet-sivuston värimaailmaa ja

kuvitusta. Rekisteristä pyrittiin saamaan mahdollisimman yhteneväinen sivuston kanssa.

5.2 Arkkitehtuuri

Arkkitehtuuri antaa yhteisen pohjan kaikelle järjestelmään liittyvälle suunnittelu- ja toteutustyölle. Arkkitehtuurin filosofia antaa kehittäjälle mallin siitä, miten uusi ominaisuus toteutetaan järjestelmään. Samaten se antaa ylläpitäjälle tietoa siitä, minkä tapaista ratkaisua on etsittävä ja mihin muutokset on kohdistettava. Hyvälle arkkitehtuurille on ominaista, että jos jotain asiaa ei tiedä, se on arvattavissa toteutusfilosofian perusteella. (Haikala & Märijärvi 2006, 317.)

Tässä työssä otettiin käyttöön kolmikerrosmalli (kuvio 5), sillä web on ympäristönä sellainen, että tietokantojen asiakas-palvelin-mallia ei voi käyttää. Selaimet eivät pääsääntöisesti osaa toimia suoraan tietokantapalvelimen asiakkaana. Kolmikerrosmallissa selain toimii asiakaskerroksena. Välikerros toimii asiakaskerroksen ja pohjakerroksen välissä, joka on tässä tapauksessa web-palvelimella oleva PHP-tulkki. Pohjakerroksella tarkoitetaan tietokantapalvelinta. Palvelimet voivat toimia samalla palvelinkoneella tai erillisillä palvelimilla. Tässä työssä yksi kone hoitaa kaikki vaiheet.



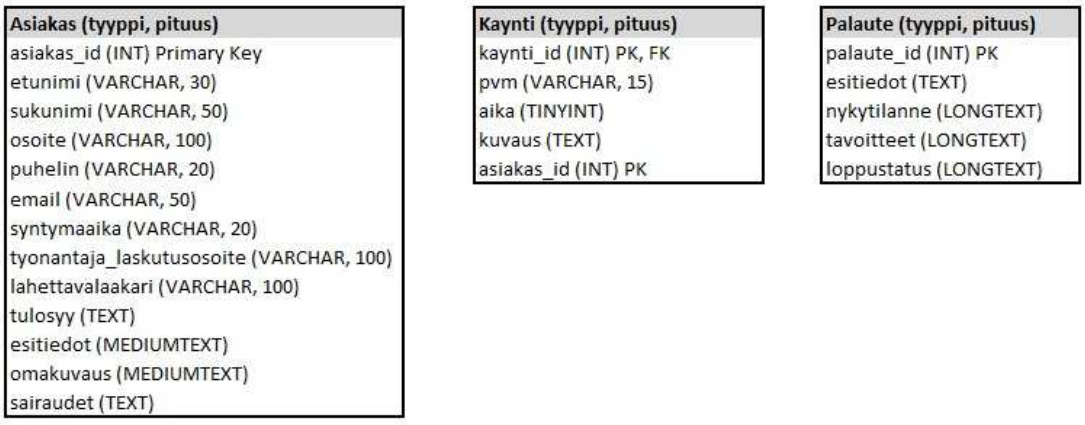
KUVIO 5. Esimerkki kolmikerrosmallista (Emerald 2010)

5.3 Tietokannan suunnittelu

Tietokannan valintaan tämän työn osalta ei ollut juurikaan kilpailua, sillä MySQL-tietokanta oli selvästi paras vaihtoehto. Se on ilmainen ja WAMP-serveriohjelmistossa se on valmiiksi asennettuna. WAMP-ohjelmistoa käytettiin tässä työssä tietokannan hallintaan ja tarkastuksiin, sillä sen sisältämä PHPMyAdmin -työkalu on suhteellisen helppokäyttöinen ja sisältää graafisen käyttöliittymän. MySQL-tietokanta on hyvin yleisesti käytetty ratkaisu PHP-ohjelmointikielen kanssa. Kyseinen ratkaisu on myös tekijälle entuudestaan tuttu.

Tulevalle tietokannalle suunniteltiin rakennetta ER-mallin kautta. Kaavion avulla on helppo tarkastella, millaisia tietokantatauluja tarvitaan ja mikä niiden sisältö olisi. Myös taulujen yhteydet on helpompi miettiä graafisella mallilla.

Työssä päätettiin käyttää mahdollisimman yksinkertaista tietokantarakennetta, sillä tarvetta kovin monimutkaisiin toteutuksiin ei tullut ilmi, kuten suunnitelman ER-mallista voidaan todeta (liite 3). Mallin mukaisesti tietokannan taulurakenne esitetään kuviossa 6.



KUVIO 6. Tietokantarakenne, taulut

Taululle Kaynti tehtiin viiteavain asiakas_id, jonka perusteella kunkin asiakkaan käynnit voidaan listata ja osoittaa luotaessa juuri tietylle asiakkaalle. Palauteosion ei tar-

vinnut olla sidoksissa asiakkaisiin, joten se jätettiin omaksi taulukseen ilman yhteyksiä. Tässä yhteydessä palautelomake ei tarkoita asiakkaan antamaa palautetta, vaan fysioterapeutin kirjoittamaa lähetettä esimerkiksi asiakasta hoitavalle lääkärille. Tälle ominaisuudelle suunniteltiin taulu, koska vaatimuksena oli, että palaute pitäisi pystyä tallentamaan hetkellisesti ja jatkaa kirjoittamista myöhemmin. Myös tulevaisuutta ajatellen tämä oli syytä huomioida, sillä jatkossa taulua on helppo muokata, mikäli palautteista halutaan asiakaskohtaisia ja tallentaa kunkin asiakkaan palautelomake.

Useisiin taulun sarakkeisiin ei haluttu kovin tarkkoja määrityksiä, vaan toivomus oli, että lomakkeen kentät olisivat suhteellisen vapaamuotoisia. Tällä haluttiin myös vähentää erilaisten kenttien tarvetta, eli mm. osoitteen tapauksessa ei haluttu erikseen postinumeroa ja postitoimipaikkaa, vaan koko osoitetieto haluttiin yhteen kenttään. Kaynti -taulun päivämääräkentän kanssa tilanne oli sama. Ehdotus oli, että kenttään tulisi timestamp-määrite, jolla päivämäärä tulisi lisäyksen yhteydessä automaattisesti. Tätä ei kuitenkaan haluttu toimeksiantajan puolesta toteuttaa, sillä merkkaukset eivät välttämättä tule reaaliajassa. Monet attribuutit suunniteltiin TEXT- tai MEDIUMTEXT -tyyppisiksi, koska toimeksiantaja ei halunnut rajoittaa vapaamuotoisiin kenttiin kirjoitettavien merkkien määrää. Tietoinen rajoittaminen mm. hakutulosten hidastumisen pelossa kävi mielessä, mutta toimeksiantajan suunnalta arveltiin, että rekisterin koko ei välttämättä kasvaisi hirveän suureksi ja toisaalta tekstiä pitäisi mahdollisesti saada mahtumaan runsaasti.

Esimerkiksi MEDIUMTEXT mahdollistaa 16 772 215 merkkiä ja vie tallennustilaa kirjoitetun merkkimäärän + 3 tavua. TEXT tietotyyppi tukee 65 535 merkkiä ja vie tilaa merkkien määrä + 2 tavua. (Gilmore 2005, 576--577.)

6 SOVELLUKSEN TOTEUTUS

Tässä luvussa kerrotaan, miten määrittely- ja suunnitteluvaiheen mukainen asiakasrekisteri toteutetaan. Toteutuksessa käytetty ympäristö esitellään mm. käytettyjen ohjelmistojen kautta. Osiossa otetaan esille koodauksen vaiheita, tietokannan luonti ja testausta.

6.1 Toteutusympäristö ja apuohjelmat

Seuraavassa esitellään, millaisella alustalla tämän työn sähköinen asiakasrekisteri toteutettiin ja millaisia työkaluja käytettiin toteutuksen apuna.

6.1.1. Laitteisto

Opinnäytetyön toteutus tehtiin pääasiallisesti tekijän pöytäkoneella, jossa käyttöjärjestelmänä on Windows 7 Ultimate (64 bit). Palvelimena toimi WAMP-ohjelmisto, josta käytettiin versiota 2.0. WAMP sisältää APACHE 2.2.11, PHP 5.3.0 ja MySQL 5.1.36 versiot valmiiksi asennettuina. Tämä ratkaisu otettiin käyttöön, koska asiakasrekisterille ei ollut eikä ollut tulossa omaa, erillistä palvelinta. WampServerin hyviin puoliin lukeutuu helppous, sillä kuten aiemmin mainittiin, tarvittavat asiat ovat valmiiksi asennettuina ja eri toimintoja voi ottaa käyttöön tai pois päältä ilman asetustiedoston muokkausta. Myös eri versioiden välillä toimiminen onnistuu, mikäli tälle olisi työn aikana tullut tarvetta. Tietokannan käyttö ja asetusten muokkaaminen on myös helppoa graafisen käyttöliittymän avulla.

Koska toimeksiantajan tietotaito oli varsin vähäistä, pyrittiin toteutuksen etenemistä raportoimaan muilla keinoilla, esim. kuvankaappauksilla, jotka antoivat tarvittavan informaation. Itse sovelluksen toimintaa näytettiin asiakkaalle muutaman kerran tekijän kannettavalla tietokoneella, jota käytettiin tarvittaessa myös testailuun. Kannettavan tietokoneen käyttöjärjestelmänä on Windows 7 Home Premium (64 bit) ja muutoin samat ohjelmistot kuin pöytäkoneessa.

6.1.2. Dropbox

Koko työn ajan käytössä oli Dropbox ”pilvipalvelu”, joka mahdollistaa tiedostojen jakamisen, varmuuskopioinnin ja historiatiedot. Dropbox tarjoaa 2 gigatavua ilmaista levytilaa yrityksen palvelimilla. Ohjelma oli asennettuna kaikille työssä käytetyille tietokoneille, mukaan lukien toimeksiantajan kannettava. Dropbox luo kansion tietokoneelle, joka toimii kuten mikä tahansa kansio resurssien hallinnassa. Erona on se, että merkattu kansio on yhteydessä palvelimeen, johon tiedostot synkronoidaan, kun kansioon siirretään uusia tiedostoja tai olemassa olevia muokataan. Muutokset sisällössä näkyvät kaikissa päätteissä reaaliajassa. Tästä huolimatta tiedostot ovat myös fyysisesti kullakin koneella, joten kansion sisältöä voi käyttää myös yhteydettömässä tilassa. Muutokset synkronoidaan aina, kun Internet-yhteys on saatavilla. Kansioita ja tiedostoja voidaan laittaa jakoon eri tunnusten välillä. Näin toimittiin toimeksiantajan kanssa, sillä välimatka haittasi säännöllistä tapaamista ja sähköpostin välityksellä tiedostojen siirto vie turhan paljon aikaa ja tilaa.

Dropbox -palvelun avulla toimeksiantaja näki reaaliajassa jaetun kansion sisällön ja pystyi tarkastamaan tilannetta sen kautta. Esimerkiksi käyttöliittymän suunnittelussa kansioon tallennettiin kuvakaappaukset, joita toimeksiantaja kommentoi halutessaan. Myös toteutusvaiheessa kooditiedostot tallennettiin päivän päätteeksi jaettuun kansioon. Vaikka kaikki koneet olisivat hajonneet samaan aikaan, olisi tiedostot saatu helposti haettua Internet-käyttöliittymän kautta takaisin. Dropbox oli siis osa versionhallintaa ja paikka varmuuskopioille.

6.1.3. Notepad++

Koodaustyökaluna tässä työssä käytettiin Notepad++ -tekstieditoria, joka on tarkoitettu erityisesti lähdekoodin kirjoittamiseen ja selaamiseen. Editorissa voidaan valita, mitä ohjelmointikieltä kirjoitetaan, jolloin eri tasot ja esim. funktioiden aloitukset ja lopetukset on helppo havaita. Tämän tarkempaa kehitysympäristöä koodausvaiheessa ei tarvittu. Ohjelma on myös ilmainen eikä vaadi suuresti tehoja. Ohjelmasta käytettiin versiota 5.7.

6.2 Tietokannan luonti

Ensimmäinen vaihe toteutuksessa oli tietokannan perustaminen suunnitelmien pohjalta. Suunnitelmavaiheessa tarpeeksi hyvin mietitty taulurakenne helpotti tässä operaatiossa ja nopeutti lisäyslauseiden kirjoittamista. Vaikka PHPMyAdminin graafinen käyttöliittymä tarjoaa tietokannan rakentamisen myös tätä kautta, suoraan tekstieditorilla kirjoitetut SQL-lauseet tuntuivat luontevammalta keinolta. Tietokannan luontiin käytetyt SQL-lauseet on esitetty liitteessä 5.

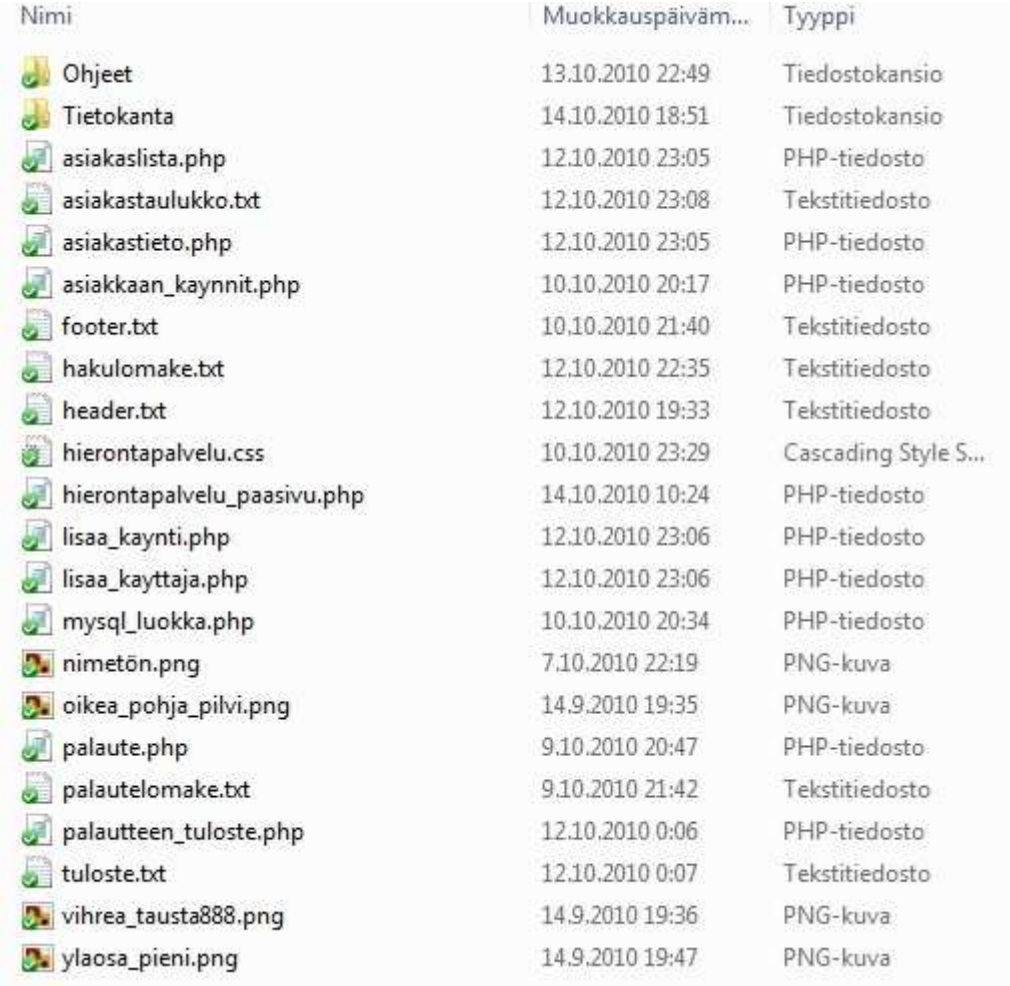
Viite-eheys asiakkaan ja käyntien välillä otettiin huomioon siten, että mikäli asiakas jostain syystä poistetaan, kyseisen asiakkaan ID-numeron sisältävät käynnit poistetaan (ON DELETE CASCADE). Samoin, mikäli käyttäjän tietoja muokataan, tai jostain syystä asiakas_id vaihtuu, muutos huomioidaan kyseisen asiakkaan käyntien riveissä (ON UPDATE CASCADE).

Tietokantaan lisättiin myös testidataa testikäyttäjän muodossa (myös käynti testikäyttäjälle) ja lisäksi palautelomakkeelle annettiin oletusarvot.

6.3 Kooditiedostot

Koska työssä käytetään WampServeriä, kooditiedostot tulee lisätä wamp-ohjelman www-kansioon. Ensimmäisenä vaiheena luotiin mm. tietokannan hallintaan tarkoitettu MySQL-tietokantaluokka. Pohjana käytettiin W. Jason Gilmoren PHP & MySQL 5 -kirjan ohjetta tietokantaluokan rakentamiseen. Tietokantaluokan teko helpottaa tietokannan kanssa toimimista, sillä muutoksia ei välttämättä tarvitse tehdä kaikkiin skripteihin. Erityisesti yhteyden luominen tietokantapalvelimeen helpottuu hieman. (Gilmore 2005, 647 - 650.)

Tiedostojen nimeämisessä pyrittiin selkeyteen, eli jo tiedoston nimestä voisi päätellä, mitä ominaisuutta ja tehtävää kukin tiedosto edustaa. Kooditiedostojen listaus on esitetty kuviossa 7.



Nimi	Muokkauspäivämäärä	Tyyppi
Ohjeet	13.10.2010 22:49	Tiedostokansio
Tietokanta	14.10.2010 18:51	Tiedostokansio
asiakaslista.php	12.10.2010 23:05	PHP-tiedosto
asiakastaulukko.txt	12.10.2010 23:08	Tekstitiedosto
asiakastieto.php	12.10.2010 23:05	PHP-tiedosto
asiakkaan_kaynnit.php	10.10.2010 20:17	PHP-tiedosto
footer.txt	10.10.2010 21:40	Tekstitiedosto
hakulomake.txt	12.10.2010 22:35	Tekstitiedosto
header.txt	12.10.2010 19:33	Tekstitiedosto
hierontapalvelu.css	10.10.2010 23:29	Cascading Style S...
hierontapalvelu_paasivu.php	14.10.2010 10:24	PHP-tiedosto
lisaa_kaynti.php	12.10.2010 23:06	PHP-tiedosto
lisaa_kayttaja.php	12.10.2010 23:06	PHP-tiedosto
mysql_luokka.php	10.10.2010 20:34	PHP-tiedosto
nimetön.png	7.10.2010 22:19	PNG-kuva
oikea_pohja_pilvi.png	14.9.2010 19:35	PNG-kuva
palaute.php	9.10.2010 20:47	PHP-tiedosto
palautelomake.txt	9.10.2010 21:42	Tekstitiedosto
palautteen_tuloste.php	12.10.2010 0:06	PHP-tiedosto
tuloste.txt	12.10.2010 0:07	Tekstitiedosto
vihrea_tausta888.png	14.9.2010 19:36	PNG-kuva
ylaosa_pieni.png	14.9.2010 19:47	PNG-kuva

KUVIO 7. Kooditiedostot

Koodi pyrittiin kommentoimaan mahdollisimman kattavasti, jotta mahdollinen jatkokehitys onnistuisi hyvin. Kommentointi helpottaa erityisesti silloin, kun useampi henkilö työskentelee samojen koodien parissa.

6.4 Testaus

Testaus on ohjelmistokehityksessä erittäin merkittävä seikka. Sillä pyritään osoittamaan, että ohjelmassa on virheitä, mutta virheettömyyttä testauksella ei pystytä osoittamaan edes yksinkertaisissa tapauksissa. Testaukseen liittyvät vaiheet ovat testauksen suunnittelu (jossa on tarkoitus tehdä esim. testitapaukset), testiympäristön luonti, testin suorittaminen ja tulosten tarkastelu. Yleensä näihin kuluu suuri osa resursseista. Juuri tästä syystä testaus on se vaihe, josta tingitään, mikäli aika tms. on

vähissä. Testauksen määrä ei kuitenkaan välttämättä ole sama kuin testauksen tehokkuus. Testauksen suunnittelulla on siis suuri rooli ohjelmistokehityksessä. (Haikala & Märijärvi 2006, 283 - 287.)

Tämän työn osalta testitapaukset valittiin mustalaatikkotestaukseen mukaisesti. Siinä testitapaukset valitaan testattavan järjestelmän toiminnallisten vaatimusten perusteella tutustumatta ohjelman toteutukseen. (Haikala & Märijärvi 2006, 283 - 287.)

Järjestelmätestauksella pyritään tarkastelemaan koko järjestelmää ja tuloksia verrataan määrittelydokumentaatioon ja asiakasdokumentaatioon. Järjestelmätestauksella testataan myös se, ovatko ei-toiminnalliset vaatimukset tulleet täytetyiksi. Testauksesta pitäisi olla vastuussa mahdollisimman riippumattoman tahon. Kuitenkin tämän työn osalta tekijä oli päävastuussa myös testauksesta. Työn aikana tehtiin järjestelmätestisuunnitelma. Siinä määritettiin, mitä ja miten testataan. Järjestelmää päätettiin testata seuraavasti: tekijän toimesta koko koodausvaiheen ajan, vaatimusten testaaminen, käytettävyyden testaus (aikaresurssin puitteissa) ja testaus käyttöönoton yhteydessä.

7 SOVELLUKSEN KÄYTTÖÖNOTTO

Luvussa seitsemän kerrotaan tämän työn sovelluksen käyttöönotosta. Tarkoitus on selvittää sovelluksen asennus sekä käyttökoulutuksen järjestäminen ja käyttöoppaan teko. Myös varmuuskopiointi otetaan huomioon.

7.1 Sovelluksen asennus toimeksiantajan ympäristöön

Sovelluksen valmistuttua se täytyy asentaa toimeksiantajalle. Tämän työn tapauksessa tekijä kävi henkilökohtaisesti hoitamassa asennuksen toimeksiantajan ympäristöön. Asennus tapahtui toimeksiantajan käytössä olevalle kannettavalle tietokoneelle, jossa käyttöjärjestelmänä on Windows 7 Home Premium. Toimitiloissa on käytössä langaton Internet-yhteys. Tietokone on suojattu käyttäjän salasanalla ja pääsy tilaan, jossa konetta käytetään, on estetty muilta kuin henkilökunnalta. Koska sähköinen asiakasrekisteri toimii paikallisesti tällä kyseisellä koneella, erillistä salasanasuojausta itse rekisteriin ei nähty tässä vaiheessa tarpeelliseksi.

Tietokoneelle asennettiin ensimmäisenä uusin versio WAMP-ohjelmistosta, jonka kautta asiakasrekisteri toimii. Wampserver asetettiin aukeamaan automaattisesti offline-tilassa, jotta palvelin ei ainakaan vahingossa avaisi tietä ulkopuoliselle tietoliikenteelle. Tietokannan asennus tapahtui tekstitiedostosta, johon tarvittavan tietokannan luontilauseet on tallennettu. PHPMyAdmin-käyttöliittymän kautta tietokantojen luonti ja testidatan lisääminen on helppoa. Tietokannalle luotiin käyttäjätunnus ja salasana, jotta tietokantayhteyden muodostaminen PHP-koodista onnistuu. Graafinen käyttöliittymä tarjoaa helpon tavan ajaa SQL -muotoiset luonti- ja lisäyslauseet suoraan kopioimalla ja liittämällä tekstin tiedostosta.

Tietokannan luonnin jälkeen sähköisen asiakasrekisterin kansio kopioitiin WAMPin www-kansioon. Asiakasrekisterin pääsivusta luotiin työpöydälle oma kuvake, jonka kautta asiakasrekisteriin on mahdollista päästä suoraan. Kuvakkeelle luotiin oma ikonikuva Vaajakosken hierontapalvelun logolla, jotta tunnistaminen työpöydältä

olisi helpompaa, eikä se sekoittuisi muiden selainkuvakkeiden kanssa. Muita asennustoimenpiteitä ei asiakasrekisterin puolesta tarvinnut tehdä.

7.2 Sovelluksen koulutus toimeksiantajalle

Toimeksiantajan kanssa oli sovittu heti alusta alkaen, että toteutettava järjestelmä tulee myös opastaa ainakin kahdelle Vaajakosken hierontapalvelun työntekijälle. Koulutus toimi samalla myös järjestelmätestauksen osana testisuunnitelman mukaisesti. Mikäli puutteita tulisi tässä vaiheessa ilmi, voitaisiin niihin puuttua. Käytännössä tällaiseen ratkaisuun päädyttiin, koska toimeksiantajalla ei ollut mahdollisuutta testata järjestelmän käyttöä ilman asennusta. Välimatka tekijän ja toimeksiantajan välillä piti tapaamiset vähissä, joten tekijällä oli vapaus asentaa järjestelmä siinä vaiheessa, kun katsoi sen olevan käyttökunnossa. Tässä käytettiin mittarina pakollisten vaatimusten täyttymistä testien jälkeen.

Asiakasrekisterin käyttäjiä opastettiin ns. kädestä pitäen, eli tekijä kävi läpi kaikki toiminnot. Tämän jälkeen molemmat tulevista käyttäjistä kokeilivat toiminnot läpi testitapausten mukaisesti. Tällä saatiin varmistettua sovelluksen toiminta toimeksiantajan ympäristössä. Tapaamisen lopuksi todettiin, että vaatimukset täyttyivät, joten valmis tuote oli toimeksiantajan käytettävissä.

7.3 Sovelluksen käyttöohje ja tietokannan varmuuskopiointi

Sähköiselle asiakasrekisterille tuli luoda ohjeistus, sillä toimeksiantajan mielestä pelkkä koulutus ei välttämättä olisi riittävä. Käyttöohje päätettiin luoda erillisenä tiedostona, johon koottiin kaikki asiakasrekisterin toiminnot. Ohjeesta tehtiin kronologisesti etenevä alkaen koko sovelluksen asennuksen vaiheista ja päättyen varmuuskopiointi-osioon. Asennus pyrittiin selittämään mahdollisimman yksinkertaisesti ja kuvankaappauksia käyttäen, jotta toimeksiantaja voisi itse suorittaa asiakasrekisterin asennukset. Ohjeessa käytettiin kuvankaappauksia mahdollisimman paljon, jotta käyttäjän olisi helpompi ymmärtää, mitä kukin vaihe vastaa ruudulla.

Varmuuskopiointi on tietokantojen kanssa toimittaessa isossa roolissa, sillä varsinkin tämän työn toimeksiantajan tietokantaan syöttämät tiedot voivat olla hyvin merkittäviä yrityksen kannalta. Varmuuskopiointiin otettiin työn osalta kantaa käyttöohjeessa, johon tekijä selosti, miten phpmyadminista saadaan otettua tietokannan varmuuskopio suoraan erillisenä tiedostona. Tieto kirjataan haettuun tiedostoon SQL-muotoisissa lauseissa, joten tietokannan palauttaminen on hyvin helppoa.

Toimeksiantaja päätti itse hoitaa varmuuskopioinnin ulkoiselle USB-muistille vähintään kerran kuukaudessa tai välittömästi, mikäli asiakastietoja tarvitsee syöttää isommassa erässä. Erillinen tallennuspaikka on tämän tapauksen osalta tärkeä, sillä sovellus ja tietokanta sijaitsevat samalla tietokoneella. Näin ollen kannettavan tietokoneen rikkoutuminen vie kaiken mukanaan. Esimerkiksi USB-muistitikulla tai ulkoisella kiintolevyllä tiedon täydellinen menetys ei ole niin suuri riski.

8 TUTKIMUSTULOSTEN ANALYSOINTI

Luvun tarkoituksena on selvittää tästä opinnäytetyöstä saadut tulokset. Tutkimuskysymyksiin on tarkoitus saada vastaukset ja analysoida ne. Vastausten olisi tarkoitus hyödyttää ensisijaisesti toimeksiantajaa ja tekijää, sillä aikaiseksi saatu järjestelmä on räätälöity juuri tätä projektia silmällä pitäen ja toimeksiantajayrityksen toiminnan tehostamiseksi, mutta myös vastaavanlaisessa tilanteessa olevat ja muut kiinnostuneet tahot voivat hyötyä saaduista tuloksista.

Opinnäytetyön tuloksena syntyi tutkimus, jossa tarkastellaan sähköistä asiakasrekisteriä erityisesti HTML-lomakkeiden käytettävyyden näkökulmasta. Osiossa selvitetään muun muassa hyvän käytettävyyden periaatteet ja se, mitä tulee ottaa huomioon, kun paperista asiakaslomaketta lähdetään siirtämään sähköiseksi. Yhdeksi tutkimustulokseksi saatiin myös tietojärjestelmän kehitysprojektin tuottamat dokumentit siinä laajuudessa, kuin projektille katsottiin tarpeelliseksi. Tällä pyritään myös lisäämään tietoa projektin kehityksen vaiheista ja siinä huomioon otettavista seikoista. Sähköisen asiakasrekisterin todellisesta hyödystä pitkällä aikavälillä ei voida tehdä tässä vaiheessa johtopäätöksiä, sillä se vaatisi pidemmän aikajakson tarkastelua. Projektissa käyttäjälähtöisyys oli tärkeässä asemassa, ja tämä saatiin toimeksiantajan vapaamuotoisen palautteen perusteella onnistumaan.

1. Millaisen sovelluksen asiakas tarvitsee?

Sovellukset kehitetään asiakkaan tarpeisiin, joten kaikki lähtee vaatimusten keräämisestä. Vaatimuksia on hyvä kerätä reilusti, kuten tämän työn kohdalla havaittiin. Näiden kerääminen voi olla hyvinkin hankalaa tai helppoa sen mukaan, millainen toimeksiantaja on. Toimeksiantajalla ei ollut kovin paljon omia ideoita ja näkemyksiä siitä, millainen sovelluksen tulisi olla. Tästä johtuen vaatimuksia jouduttiin miettimään melko paljon ja vaatimusten määrä olisi voinut olla korkeampi, jotta suunnittelu ei olisi ollut niin suurpiirteistä. Vaatimuksia tehtäessä niiden selkeyteen tulee kiinnittää huomiota, ja se onnistui tämän työn kohdalla hyvin. Pakollisten ja valinnaisten vaatimusten merkintä osoittautui hyväksi ratkaisuksi. Aikataulupaineiden pienentä-

miseksi tämän asian huomioiminen on erittäin tärkeää. Valinnaisuus jättää tämän tyyppisissä pienissä ja resursseiltaan vaatimattomissa projekteissa pelivaraa. Vaatimukset tulee erotella toiminnallisiin ja ei-toiminnallisiin. Ei-toiminnalliset vaatimukset määrittelevät, miten esimerkiksi tekniset seikat tulevat vaikuttamaan suunnittelussa ja toteutuksessa. Tämän projektin kohdalla ei-toiminnallisten vaatimusten pohjalta voitiin todeta, että sovellus vastaa parhaiten asiakkaan toiveita, jos sovellus ei ole käytettävissä kuin yhdellä koneella. Selainpohjaisuuden valinta mahdollistaa kuitenkin tulevaisuudessa sen, että sovellus voidaan siirtää myös palvelimelle ja käytettäväksi mistä tahansa. Tämä tosin vaatii tietoturvapuolen suunnittelua uudelleen.

Esitutkimuksen tekeminen mahdollisimman kattavasti on erittäin tärkeää. Hyvistä ja selkeistä vaatimuksista on helppo tehdä tarvittavat käyttötapaukset. Käyttötapauksista on helppo lähteä suunnittelemaan esim. käyttöliittymää. Käyttötapauksista voidaan tarkastella tilannetta käyttäjän näkökulmasta, sillä niiden tarkoitus on kuvata järjestelmän toiminnallisuus. Siten sovellukselle saadaan tiettyä selkeyttä, ja suunnittelussa päästään heti huomioimaan käyttäjälähtöisyys. Esitutkimuksen perusteella voitiin todeta, että asiakasrekisterin tulee olla mahdollisimman yksinkertainen käyttää.

Käyttöliittymäsuunnittelun perusteella voidaan selvittää mm. sovelluksen ulkoasulliset seikat. Projektin osalta ulkoasu kehitettiin toimeksiantajan Internet-sivujen mukaiseksi. Sähköisestä asiakasrekisteristä tehty käyttöliittymädemo antoi erittäin hyvän kuvan siitä, mitä asiakas loppujen lopuksi halusi. Sen avulla toimeksiantaja kykeni antamaan tarkempia ohjeita, millaisia kenttiä jne. lopulliseen asiakasrekisteriin tarvitaan.

Kaikkiaan voidaan todeta, että aktiivinen kanssakäyminen asiakkaan kanssa on kehityksen alkuvaiheessa elintärkeää. Kaikki esille tulleet asiat on syytä dokumentoida hyvin ja kattavasti, jos toimeksiantajalla ei ole selkeitä vaatimuksia kehityksen alkuvaiheessa. Esitutkimus, vaatimusmäärittelydokumentti ja käyttöliittymäsuunnittelu kertovat hyvin, millaisen sovelluksen toimeksiantaja tarvitsee ja millä tekniikalla toteutusta kannattaa lähteä rakentamaan.

2. Miten kehitysprojekti toteutetaan?

Kehitysprojektin toteutukseen on olemassa useita toimintamalleja (esim. vesiputousmalli, spiraali ja prototyyppimalli). Tähän projektiin valittiin prototyyppimalli siitä syystä, että toimeksiantajalla ei ollut antaa tarkkoja vaatimuksia. Prototyyppimallin avulla voitiin luoda demo, minkä jälkeen vaatimuksia tarkennettiin ja pohdittiin uudestaan. Ensimmäinen versio oli toimeksiantajan mukaan jo sen verran hyvä, että tätä päätettiin lähteä muokkaamaan lähinnä asiakastietolomakkeen osalta. Käyttöliittymä ja ulkoasu olivat toimeksiantajan mielestä heti miellyttäviä. Voidaan sanoa, että tässä asiassa myös onnella oli osuutensa. Myös yrityksen Internet-sivujen mukainen toteutus auttoi varsinkin ulkoasullisesti. Tämä seikka kannattaa ottaa huomioon, mikäli vastaavanlaista asiakasrekisteriä tai muuta vastaavaa lähdetään toteuttamaan.

Toteutusvaiheessa määrittelyn kautta suunnitelmiksi muodostuneet dokumentit ovat ratkaisevassa osassa. Suunnitelmat nopeuttavat projektin toteuttamista, sillä niissä tarpeet ja lähtökohdat on asetettu. Näin ollen itse toteuttamisvaihe saadaan sujumaan nopeammin. Hyvin helposti voi käydä niin, että toteutusvaihe ja suunnitteluvaihe menevät päällekkäin. Toteutuksesta voi tulla aikaa vievää yritys ja erehdys -tyylistä toimintaa. Tämän projektin osalta näin oli päästä käymään, sillä toteutus haluttiin aloittaa mahdollisimman aikaisessa vaiheessa, ja tekijä on käytännössä työn ainoa resurssi. Suunnittelulle oli varattu ehkä vähän liian vähän aikaa. Voidaankin sanoa, että aikataulutuksella on iso merkitys kehitysprojektissa. Aikataulun asettaminen on muutenkin tärkeää, sillä liian tiukka aikataulu saattaa viedä tilaa esimerkiksi testaukselta.

Tämän työn tyylisen sähköisen asiakasrekisterin pohjana toimii tietokanta. Tietokannan suunnittelu on pidettävä tämänkaltaisissa projekteissa pääosassa, sillä se muodostaa asiakasrekisterin pohjan. Tietokanta kannattaa suunnitella hyvin jo varhaisessa vaiheessa, sillä muutosten tekeminen voi olla työlästä. Suunnittelu kannattaa aloittaa esimerkiksi ER-mallista. Tämän projektin osalta ER-mallin luonti oli melko helppoa, sillä suurin osa tarvittavista attribuuteista saatiin suoraan paperilomakkeelta. Tietokannan suunnittelussa vastaan tuli muutamia ongelmia, joiden ratkaisu kesti

yllättävän kauan. Esimerkiksi yhteyden tekeminen asiakas- ja käynti taulujen välille tuotti hivenen ongelmaa. Ensin ajatuksena oli luoda asiakas- ja käynti -taulu, joiden tiedot yhdistetään välitaululla, johon merkitään asiakas_id ja käynti_id. Asiaa tarkasteltiin kuitenkin uudelleen ja todettiin, että yhdellä asiakkaalla voi olla monta käyntiä, mutta yhdellä käynnillä voi olla vain yksi asiakas. Käynti tauluun merkittiin soluksi asiakas_id, joka myös merkattiin viiteavaimeksi. Tämä kertoo käynnit -taulusta sen, että käynti yhdistetään käyttäjään viiteavaimen perusteella (asiakas- taulun asiakas_id = käynti- taulun asiakas_id).

Kehitystyökalut ja -ympäristö valikoidaan valitun tekniikan ja tarpeiden mukaisesti. Web-sivuston toteutuksessa helpottavina työkaluina voidaan sanoa tämän työn pohjalta olevan ainakin WampServer -palvelinympäristön, jossa tietokantaa ja itse sivustoa voidaan testata ilman erillisen palvelimen tai palvelintilan hankintaa. Ohjelmointia helpottaa, mikäli käytettävä työkalu osaa esimerkiksi tunnistaa eri koodauskielet ja merkitä rivit tietyillä väreillä ja merkinnöillä. Tämä helpottaa huomattavasti virheiden havaitsemista, ja tämän työn yhteydessä tulleen kokemuksen pohjalta myös vähentää turhien virheiden määrää.

Kehitysprojektissa testauksen merkitys on suuri. Työtä tulee testata niin toteutuksen aikana kuin sen jälkeenkin. Testaamisesta on syytä luoda testisuunnitelma, jossa pohditaan tarvittavat testimenetelmät ja oletetut tulokset. Tämän työn kohdalla vaatimusten toteutuminen oli testauksen perustana. Vaatimukset tulivat täytetyiksi toimeksiantajan kanssa tehdyn toiminnallisuustestin perusteella, vaikka testaus olikin suunniteltua suppeampi. Aikataulullisesti testaus on yleensä se vaihe, josta tingitään.

Asennus- ja ylläpitovaiheessa on otettava huomioon koulutuksen ja ohjeistuksen tarve. Tämän työn asiakasrekisterin toiminta opastettiin toimeksiantajalle asennuksen yhteydessä. Ohjeiden kirjoittamisessa tuli ottaa huomioon käyttäjien tietoteknisen tietämyksen taso. Tässä tapauksessa tietämystä oli melko vähän, joten ohjeistuksesta pyrittiin saamaan mahdollisimman yksinkertainen, mutta samalla kattava.

3. Miten käytettävyys on otettava huomioon lomakkeiden suunnittelussa?

Lomakkeiden teossa on ensisijaisesti otettava huomioon käyttäjät ja heidän tarpeensa. Paperisen lomakkeen sähköinen versio ei saisi olla yhden suhde yhteen, vaan lomake tulisi suunnitella siten, että toiminta ottaa huomion tietotekniikan mahdollisuudet ja heikkoudet. Tässä kannattaa tietenkin käyttää harkintaa (pdf jne.).

Tämän työn perusteella seuraavat seikat tulee huomioida lomakkeiden suunnittelussa:

- Lomakkeella tulee olla vain tarpeellinen tieto täyttöohjeineen.
- Pakolliset kentät tulee ilmoittaa ja/tai merkitä.
- Lomakkeen tulee täyttöjärjestyksen suhteen looginen.
- Kentissä olisi hyvä olla selitteet, missä muodossa tieto tulee lisätä.
- Mikäli lomakkeella on vaihtoehtokenttiä, eri vaihtoehtoja ei saa olla liikaa.
- Vapaamuotoinen tekstikenttä voi olla parempi kuin vaihtoehtokentät.
- Reset-painiketta tulisi välttää.
- Lähetys-painike tulee olla viimeisenä.
- Lähetyspainikkeen tulisi olla painike, ei kuva.
- Lomakkeen täyttö tulisi onnistua pelkkää näppäimistöä käyttäen.

Tässä työssä paperista asiakastietolomaketta muokattiin sähköiseen versioon jonkin verran. Osa kentistä jätettiin pois ja muutamia lisättiin, sillä esimerkiksi kenttä sähköpostiosoitteelle katsottiin tarpeelliseksi. Muutenkin lomakkeilla käytettiin hyvin

paljon vapaata tekstikenttää, sillä liian tarkkojen rajausten todettiin haittaavan käyttäjien työtä. Tarkistus tuli vain sähköpostikentälle, mikä voidaan laskea tekijän mielestä pieneksi puutteeksi. Myös tekstikenttien merkkimäärien näyttö olisi ollut syytä tehdä toteutusvaiheessa (vapaan tekstin kenttiin). Jälkikäteen ajateltuna tästä olisi saattanut olla hyötyä käyttäjän näkökulmasta.

9 POHDINTA

Opinnäytetyö onnistui kokonaisuutena kohtalaisen hyvin, vaikka aikataululliset seikat olivat haastavia koko toteutuksen ajan. Aihetta olisi voinut kokonaisuutena käsitellä vieläkin syvällisemmin ja laajentaa itse asiakasrekisterin ominaisuuksia. Tämä olisi kuitenkin vaatinut enemmän resursseja tai pidemmän kehitysajan. Toteutusvaihe venyi kokonaisuutena pidemmäksi kuin oli alun perin suunniteltu. Tähän vaikutti erityisesti noin vuoden tauko opiskelusta ennen varsinaisen opinnäytetyön aloitusta. Näin ollen monet asiat piti selvittää ja palauttaa mieleen uudelleen, erityisesti ohjelmoinnin ja tietokannan suunnittelun sekä toteutuksen osalta. Näistä seikoista huolimatta opinnäytetyön toteutus sujui lähes alussa suunnitellun aikataulun mukaisesti. Myös arvioitu sivumäärä liitteineen toteutui lähes täydellisesti.

Lähdemateriaalia aiheesta oli tarjolla melko paljon, varsinkin tietojärjestelmien kehityksestä. Tässä asiassa oleellisen löytäminen meinasi tuottaa hankaluuksia. Lomakkeiden suunnittelun osalta tietoa oli vaikea lähteä syventämään tämän tarkemmin, sillä suurin osa ajasta olisi mennyt lähteiden hankintaan.

Tulosten analysoinnin osuuteen olisi voinut käyttää enemmänkin aikaa, mutta valmis asiakasrekisteri toimi tekijän mielestä tarpeeksi merkittävänä tuloksena. Lopputuloksen tarkastelu oli vaikeaa, koska mittavampaa palautetta asiakasrekisterin toimivuudesta tuotantokäytössä ja lopullisista vaikutuksista yrityksen toimintojen tehostumiseen ei ollut mahdollista saada näin lyhyellä aikataululla. Toimeksiantajaan tullaan kuitenkin olemaan yhteydessä myöhemmin mahdollisten muutosten ja puutteiden korjaamiseksi, mikäli jotain suurta epäkohtaa ilmenee.

Sähköisen asiakasrekisterin toteutusta voisi pohtia myös vaihtoehtoisilla ja uusilla tekniikoilla. Tulevaisuudessa esimerkiksi HTML5 lienee yksi merkittävistä toteutusvaihtoehdoista. Tekijän kiinnostus tätä vaihtoehtoa kohtaan nousi aivan kirjoittamisen loppuvaiheessa, joten muutosta ei voinut edes harkita.

Tämän työn tuloksena kehitetylle asiakasrekisterille on olemassa potentiaalisia jatkokehitysmahdollisuuksia. Eräänä jatkokehityksen kohteena voisi olla asiakasrekisterin yhteyteen toteutettava ajanvarausjärjestelmä, jossa olisi työntekijäkohtainen kalenteri. Kalenteriin voitaisiin varata aikoja ajanvaraajan toimesta. Tulevaisuudessa vastaavalle Internet-palvelulle voisi olla hyvinkin tarvetta. Tällöin ideaa voisi laajentaa siten, että myös asiakkaat voisivat varata aikoja suoraan sivulta omilla tunnuksillaan. Toimeksiantaja esitti myös ajatuksen asiakkaan maksujen hoitamista rekisterin yhteyteen liitetyn lisäominaisuuden avulla. Tällöin toimeksiantaja voisi rekisterisovelluksen avulla laskea mm. kelakorvausten määriä. Aiheesta voisi saada tehtyä vaikka opinnäytetyön, mikäli kiinnostusta löytyy.

LÄHTEET

Ekonoja, A., Lahtonen, T. & Mäntylä, J. 2010. WWW-lomakkeet – Luento 7. Viitattu 25.10.2010. http://appro.mit.jyu.fi/www/luennot/luento7/.

Getting started. n.d. Notepad++ -opassivusto. Viitattu 25.10.2010. http://sourceforge.net/apps/mediawiki/notepad-plus/index.php?title=Getting_Started.

Gilmore, W. 2005. PHP & MySQL – Tehokas hallinta. Jyväskylä: Gummerus.

Haikala, I. & Märijärvi, J. 2006. Ohjelmistotuotanto. 11. uud. p. Jyväskylä: Gummerus.

Hovi, A., Huotari, J. & Lahdenmäki T. 2005. Porvoo: WS Bookwell.

HTML Tutorial. n.d. http://www.w3schools.com/html/default.asp.

Korpela, K. 2003. CSS-tyylit. Porvoo: WS Bookwell.

Korpela, K. 2003. WWW-sivut jokaiselle sopiviksi – Esteettömien verkkosivujen tekemisen opas kaikille tekijöille ja teettäjille. Helsinki: TIEKE Tietoyhteiskunnan kehittämiskeskus ry.

Korpela, K. & Linjama, T. 2005. Web-suunnittelu. Porvoo: WS Bookwell.

Muglia, B. 2010. Microsoft Has Seen The Light. And It's Not Silverlight. Viitattu 9.11.2010. http://techcrunch.com/2010/10/30/rip-silverlight-on-the-web/

MySQL Avaimet. n.d. Viitattu 22.10.2010. http://www.ratol.fi/opensource/mysql/avaimet.htm.

Parkkinen, J. 2002. hyvään verkkopalveluun! käytettävyysopas verkkoviestijöille. Tampere: Tammer-Paino Oy.

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

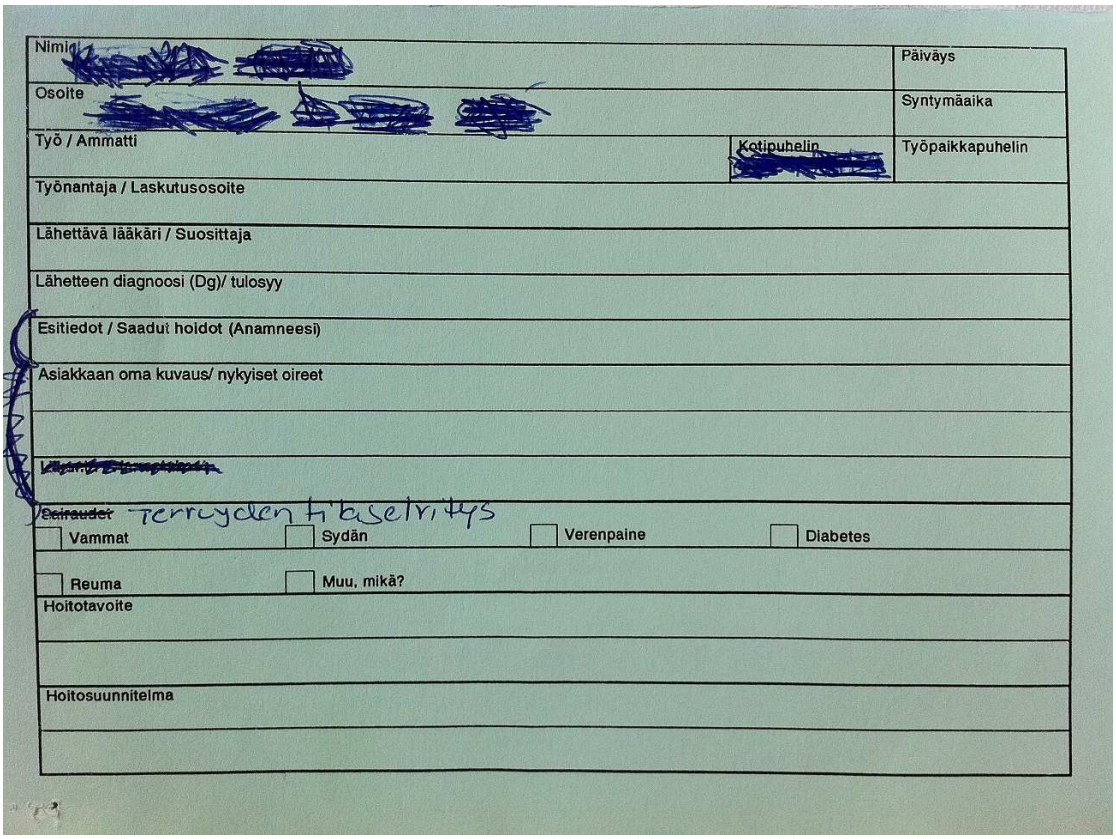
Presentation. n.d. WampServerin esittelysivu. Viitattu 26.10.2010.
http://www.wampserver.com/en/presentation.php.

Three-tier-model. n.d. Viitattu 10.10.2010.
http://www.emeraldinsight.com/content_images/fig/3330080202004.png

What is Dropbox? n.d. Dropbox –esittelysivusto. Viitattu 26.10.2010.
https://www.dropbox.com/tour.

LIITTEET

LIITE 1. Paperinen asiakastietolomake



Nimi [redacted] Päiväys

Osoite [redacted] Syntymäaika

Työ / Ammatti Kotipuhelin [redacted] Työpaikkapuhelin

Työnantaja / Laskutusosoite

Lähettävä lääkäri / Suosittaja

Lähetteen diagnoosi (Dg)/ tulosyy

Esitiedot / Saadut hoidot (Anamneesi)

Asiakkaan oma kuvaus/ nykyiset oireet

[redacted]

~~Sairaudet~~ Terveyden hoitaselvitys

☐ Vammat ☐ Sydän ☐ Verenpaine ☐ Diabetes

☐ Reuma ☐ Muu, mikä?

Hoitotavoite

Hoitosuunnitelma

LIITE 2. Vaatimukset

Seuraavassa on listattuna järjestelmän toiminnalliset ja ei-toiminnalliset vaatimukset.

Toiminnalliset vaatimukset

1. Käyttäjä voi lisätä asiakastietoja [pakollinen]

2. Käyttäjä voi muokata asiakastietoja [pakollinen]

3. Käyttäjä voi poistaa asiakastietoja [valinnainen]

4. Käyttäjä voi lisätä hinnan käynnille [valinnainen]

5. Käynnin hinnoille voi laskea suoraan kelakorvauksen määrän [valinnainen]

6. Käyttäjä voi lisätä asiakkaalle palautteen [pakollinen]

7. Asiakkaalle voi lisätä sairaudet [pakollinen]

8. Käyttäjä voi lisätä asiakkalle tiedot käynnistä [pakollinen]

9. Käyttäjä voi tulostaa asiakastiedot [pakollinen]

10. Käyttäjä voi tulostaa palautteen [pakollinen]

11. Käyttäjä voi hakea käyttäjiä sukunimen mukaan [pakollinen]

12. Käyttäjä voi kirjautua sovellukseen [valinnainen]

Ei-toiminnalliset vaatimukset

1. Järjestelmän tulee toimia Mozilla Firefox 3, Internet Explorer 8 tai Google Chrome selaimilla.

2. Palvelimen tulee tukea PHP 4.x ja MySQL x.x

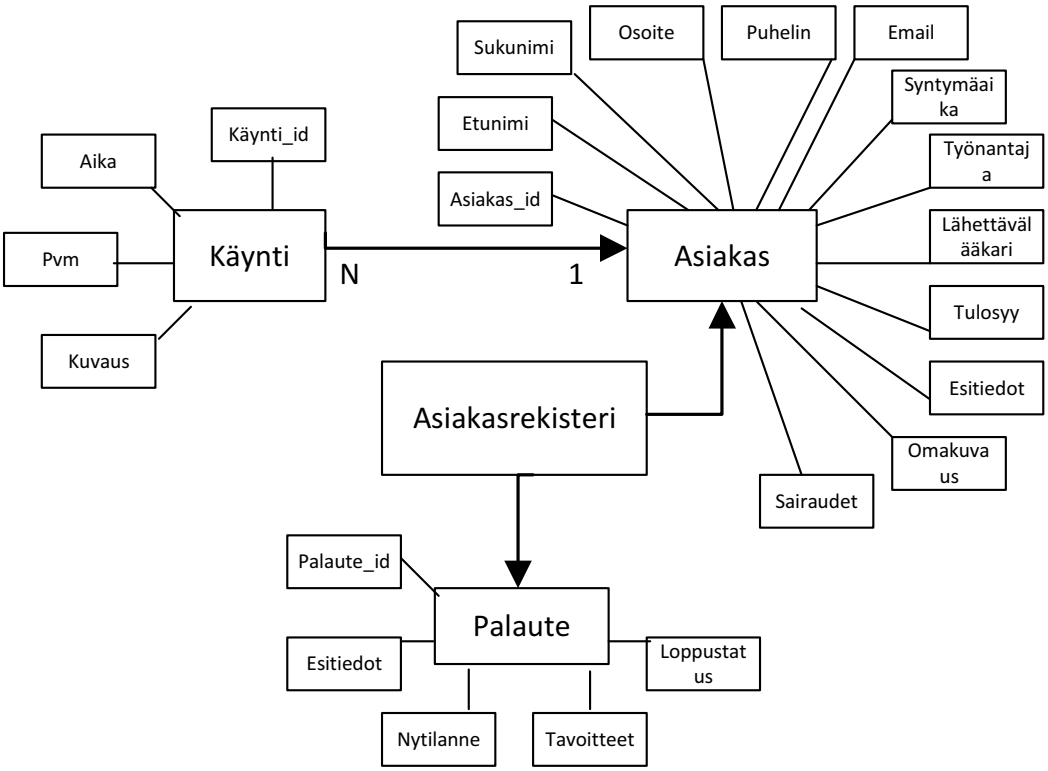
3. Tiedot tulee tallentaa MySQL-tietokantaan

4. Koneella on oltava WAMP ohjelmisto asennettuna

5. Sovelluksen tulee toimia paikallisesti

6. Sovelluksen tulee olla ulkoisesti Vaajakosken Hierontapalvelun värimaailman mukainen.

LIITE 3. Tietokannan ER-malli



LIITE 4. Käyttöliittymäsuunnitelma

Etusivunäkymä/haku

Komponentti	Kuvaus
Haku-linkki	Avaa käyttäjien hakunäkymän
Asiakkaat-linkki	Avaa asiakaslistauksen, jossa asiakkaat listattuna
Uusi asiakas-linkki	Avaa lomakkeet uuden asiakkaan lisäystä varten
Kirjoita palaute-linkki	Avaa lomakkeen palautteen kirjoittamista varten
Haku-kenttä	Käyttäjä voi syöttää hakukenttään vapaamuotoisen hakusanan (etsii sukunimistä)
Hae-painike	Lähettää käyttäjän hakukenttään syöttämän tiedon kyselynä MySQL-tietokantaan ja palauttaa tuloksen

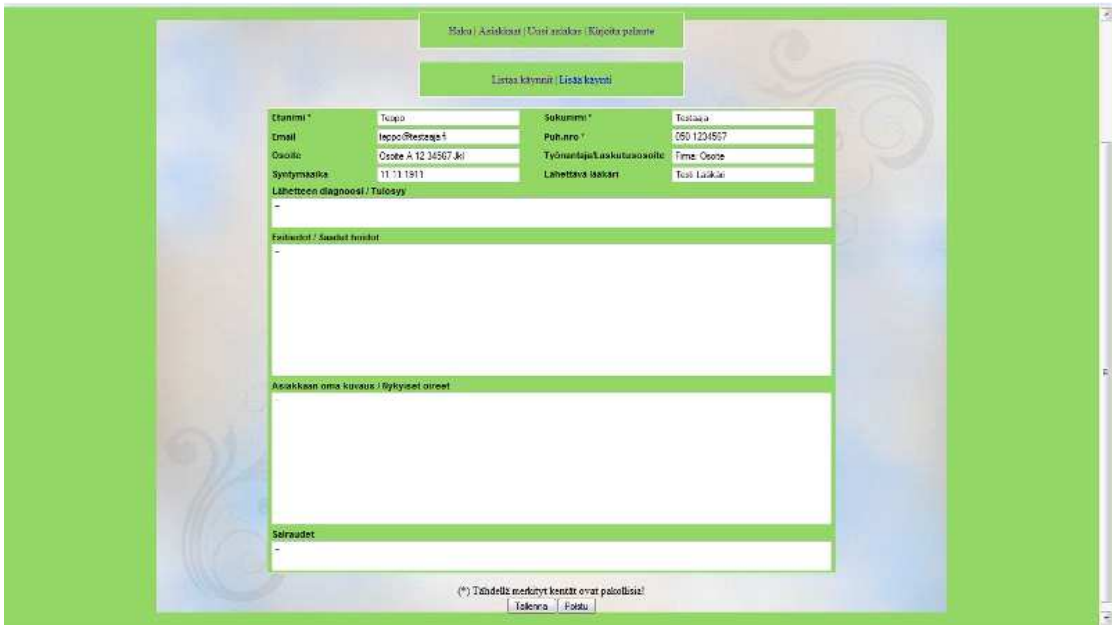
Asiakaslistaus

Komponentti	Kuvaus
Haku-linkki	Avaa käyttäjien hakunäkymän
Asiakkaat-linkki	Avaa asiakaslistauksen, jossa asiakkaat listattuna
Uusi asiakas-linkki	Avaa lomakkeet uuden asiakkaan lisäystä varten
Kirjoita palaute-linkki	Avaa lomakkeen palautteen kirjoittamista varten
Asiakkaan nimi-linkki	Käyttäjä voi asiakkaan nimeä klikkaamalla avata asiakkaan tiedot tarkastelua ja muokkausta varten

Uuden asiakkaan lisäys

Komponentti	Kuvaus
Haku-linkki	Avaa käyttäjien hakunäkymän
Asiakkaat-linkki	Avaa asiakaslistauksen, jossa asiakkaat listattuna
Uusi asiakas-linkki	Avaa lomakkeet uuden asiakkaan lisäystä varten
Kirjoita palaute-linkki	Avaa lomakkeen palautteen kirjoittamista varten
Asiakastiedot-lomake	Käyttäjä voi syöttää asiakastiedot niille varattuihin kenttiin, pakolliset merkitty tähdellä
Tallenna-painike	Tallentaa syötetyt tiedot tietokantaan ja vie käyttäjän takaisin asiakaslistaan
Poistu-painike	Palauttaa käyttäjän takaisin asiakaslistaan

Asiakkaan tiedot / muokkaus



Komponentti	Kuvaus
Haku-linkki	Avaa käyttäjien hakunäkymän
Asiakkaat-linkki	Avaa asiakaslistauksen, jossa asiakkaat listattuna
Uusi asiakas-linkki	Avaa lomakkeet uuden asiakkaan lisäystä varten
Kirjoita palaute-linkki	Avaa lomakkeen palautteen kirjoittamista varten
Listaa käynnit-linkki	Avaa käyttäjän käyntitiedot
Lisää käynti-linkki	Avaa käynnin lisäyslomakeen
Asiakastiedot-lomake	Käyttäjä voi syöttää asiakastiedot niille varattuihin kenttiin, pakolliset merkitty tähdellä
Tallenna-painike	Tallentaa muokatut tiedot tietokantaan
Poistu-painike	Palauttaa käyttäjän takaisin asiakaslistaan

Asiakkaan käyntien listaus

Komponentti	Kuvaus
Haku-linkki	Avaa käyttäjien hakunäkymän
Asiakkaat-linkki	Avaa asiakaslistauksen, jossa asiakkaat listattuna
Uusi asiakas-linkki	Avaa lomakkeet uuden asiakkaan lisäystä varten
Kirjoita palaute-linkki	Avaa lomakkeen palautteen kirjoittamista varten
Käynnit-taulukko	Näyttää asiakkaaseen liitetyt käynnit

Käynnin lisäys

Komponentti	Kuvaus
Haku-linkki	Avaa käyttäjien hakunäkymän
Asiakkaat-linkki	Avaa asiakaslistauksen, jossa asiakkaat listattuna
Uusi asiakas-linkki	Avaa lomakkeet uuden asiakkaan lisäystä varten
Kirjoita palaute-linkki	Avaa lomakkeen palautteen kirjoittamista varten
Lisää käynti-lomake	Lomakkeelle voi lisätä tiedot käynnistä
Tallenna-painike	Tallentaa käynnin tietokantaan asiakkaan id:n mukaisesti

Palautteen kirjoitus

Komponentti	Kuvaus
Haku-linkki	Avaa käyttäjien hakunäkymän
Asiakkaat-linkki	Avaa asiakaslistauksen, jossa asiakkaat listattuna
Uusi asiakas-linkki	Avaa lomakkeet uuden asiakkaan lisäystä varten
Kirjoita palaute-linkki	Avaa lomakkeen palautteen kirjoittamista varten
Palaute-lomake	Lomakkeelle voi kirjoittaa palautteen tiedot niille varattuihin kenttiin (huom. Kenttiä enemmän, kuin tässä näytetään)
Tallenna-painike	Tallentaa käynnin tietokantaan asiakkaan id:n mukaisesti
Tulosta-painike	Avaa lomakkeelle kirjoitetut tiedot tulostusnäkymään, josta lomakkeen voi tulostaa paperille

LIITE 5. SQL luontilauseet

```
-- asiakas-taulu
CREATE TABLE `Asiakas` (
`asiakas_id` INT AUTO_INCREMENT NOT NULL ,
`etunimi` VARCHAR( 30 ) NOT NULL ,
`sukunimi` VARCHAR( 50 ) NOT NULL ,
`osoite` VARCHAR( 100 ) ,
`puhelin` VARCHAR( 20 ) NOT NULL ,
`email` VARCHAR( 50 ) ,
`syntymaaika` VARCHAR( 20 ) ,
`tyonantaja_laskutusosoite` VARCHAR( 100 ) ,
`lahettavalaakari` VARCHAR ( 100 ) ,
`tulosyy` TEXT ,
`esitiedot` MEDIUMTEXT ,
`omakuvaus` MEDIUMTEXT ,
`sairaudet` TEXT ,
CONSTRAINT Asiakas_PK
                PRIMARY KEY (asiakas_id)
) ENGINE = innodb DEFAULT CHARSET=latin1;

-- käynti-taulu
CREATE TABLE `Kaynti` (
`kaynti_id` INT AUTO_INCREMENT NOT NULL ,
`pvm` VARCHAR ( 15 ) ,
`aika` TINYINT ,
`kuvaus` TEXT ,
`asiakas_id` INT NOT NULL ,
CONSTRAINT Kaynti_PK
                PRIMARY KEY (kaynti_id, asiakas_id) ,
CONSTRAINT Kaynti_FK_1
                FOREIGN KEY (asiakas_id)
                REFERENCES Asiakas (asiakas_id)
                                ON UPDATE CASCADE
                                ON DELETE CASCADE
) ENGINE = innodb DEFAULT CHARSET=latin1;


-- palaute-taulu
CREATE TABLE `Palaute` (
`palaute_id` INT NOT NULL AUTO_INCREMENT ,
`esitiedot` TEXT NOT NULL,
`nykytilanne` LONGTEXT ,
`tavoitteet` LONGTEXT ,
`loppustatus` LONGTEXT ,
CONSTRAINT Palaute_PK
                PRIMARY KEY (palaute_id)
) ENGINE = innodb DEFAULT CHARSET=latin1;
```