WWW-SOVELLUSKEHITYS

Pirkka Huhtala

Opinnäytetyö
Toukokuu 2010

Mediatekniikka
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Tekijä(t)	Julkaisun laji	Päivämäärä
HUHTALA, Pirkka	Opinnäytetyö	3.5.2010
	Sivumäärä 53	Julkaisun kieli suomi
	Luottamuksellisuus ( ) saakka	Verkkojulkaisulupa myönnetty (X)

Työn nimi
WWW-SOVELLUSKEHITYS

Koulutusohjelma
Mediatekniikka

Työn ohjaaja(t)
NIEMI, Kari

Toimeksiantaja(t)
TOSSAVAINEN Seppo - Tietosuunta Oy

Tiivistelmä

Työn tarkoituksena oli tutkia www-sovelluskehitykseen liittyviä tekniikoita ja teknologioita, joiden avulla pystyttäisiin kehittämään järkevästi laaja web-käyttöliittymässä suoritettava sovellus Tietosuunta Oy:lle.

Työssä on selvitetty perinteisten www-teknologioiden lisäksi sovelluskehyksen käytön hyötyjä sovelluksen järkevän rakenteen, ylläpidettävyyden, skaalautuvuuden ja kehitystyön nopeuden näkökulmista. Verkosta löytyy tänä päivänä paljon valmiita avoimen lähdekoodin kirjastoja ja komponentteja, joita voidaan suoraan käyttää omassa sovelluksessa. Kun avointa lähdekoodia hyödynnetään omassa sovelluksessa, on tärkeää miettiä, miten se tulee vaikuttamaan lopputuotteen lisensointiin. Lisensointi vaikuttaa mm. sovelluksen levittämiseen, käyttöön ja muokkaamiseen. Työssä selvitettiin myös PHP:lla toteutettujen sovelluksien yleisimpiä tietoturvariskejä ja niiden ehkäisemistä. PHP on helppo ja nopea ohjelmointikieli omaksua ja kehittää ohjelmakoodia. Asialle löytyy myös kääntöpuoli, sillä nopeuden ansiosta on myös helppo laiminlyödä erilaisia tarkistuksia ja validointeja.

Selvitystyön tuloksien perusteella toteutettiin Tietosuunta Oy:lle englanninkielinen laskutusohjelma. Laajan web-sovelluksen kehitystyössä huomattiin käytännössä MVC-arkkitehtuurimallin käytön edut. Mallin avulla sovelluksen rakenne saatiin paremmin hallintaan, koska käyttöliittymä, liiketoimintalogiikka ja sovelluslogiikka pystyttiin erottamaan toisistaan. Lisäksi yritykselle tuli arvokasta tietoa ja kokemusta JQueryn käytöstä osana käyttöliittymän toteutusta.

Avainsanat (asiasanat)

WWW-palvelut, sovelluskehys, PHP, tietoturva

Muut tiedot

Author(s)
HUHTALA, Pirkka

Type of publication
Bachelor´s Thesis

Date
3.5.2010

Pages
53

Language
Finnish

Confidential

( ) Until

Permission for web publication
(X)

Title
WEB APPLICATION DEVELOPMENT

Degree Programme
Media Engineering

Tutor(s)
NIEMI, Kari

Assigned by
TOSSAVAINEN Seppo - Tietosuunta Oy

Abstract

The purpose of the Bachelor's thesis was to study techniques and technologies related to web application develop that could reasonably enable development of a large application executable in the web interface for Tietosuunta Oy.

The thesis examined the traditional web technologies in addition to the benefits of using an application framework in the perspectives of rational structure, maintainability, scalability, and the speed of development. Nowadays the web is full of working open source libraries and components, which can be directly used in one's own application. When using open source in a project it is important to consider how it will affect the final product's licensing. Licensing affects the application's distribution, use and modification. The thesis also examined the most common security risks and their prevention when developing PHP applications. PHP is a fast and easy programming language to acquire and develop program code. It also has its disadvantages, because the speed also makes it easy to neglect the different amendments and validations.

On the basis of results from the study, an invoicing application was developed for Tietosuunta Oy. The benefits of using an MVC-architecture model were noticed in the development of the large application. The model allowed better control in the application's structure because the interface, business logic and application logic could be separated from each other. In addition, the company gained valuable knowledge and experience of using JQuery in the implementation of the application interface.

Keywords
Web services, framework, PHP, web security

Miscellaneous

SISÄLTÖ

KUVIOT

TAULUKOT

1 TYÖN TAVOITTEET JA LÄHTÖKOHDAT

1.1 Toimeksiantaja

Tietosuunta Oy on vuonna 1989 perustettu yritys, jonka toimiala on suunnitella ja toteuttaa ohjelmistoja pk-yrityksille. Yhtiön toimipiste sijaitsee Ylistönmäen teknologiakeskuksessa Jyväskylässä. Tietosuunnan tuotteisiin kuuluvat taloushallinnon ohjelmat (laskutus, kirjanpito, palkanlaskenta, ostoreskontra, jäsenrekisteri, vesilaskutus, sopimuslaskutus) Windows-käyttöjärjestelmiin sekä pienyrityksen verkkokauppa tuote- ja sisällönhallintoineen sekä maksupainikkeineen. Taloushallinnon ohjelmat ovat saatavissa myös etäyhteytenä, jolloin asiakkaille on tarjottavissa myös käyttöjärjestelmäriippumaton vaihtoehto. Yrityksen henkilöstömäärä on tällä hetkellä viisi ja asiakkaita yrityksellä on yli 2000. Tietosuunta Oy:llä on Iso-Britanniaan rekisteröity yhtiö AC Balance Limited, joka toimii Internetissä markkinointi ja jakeluyhtiönä, ja toimittaa Tietosuunnan valmistamia taloushallinnon ohjelmia EU -alueelle englannin-, saksan-, ranskan-, italian-, espanjan-, ruotsin- ja suomenkielisinä versioina.

1.2 Tausta

Internet alkaa olla läsnä ihmisten jokapäiväisessä elämässä mm. työelämän kautta. Ihmiset ovat alkaneet luottaa yhä enemmän web-pohjaisiin sovelluksiin, niiden suorituskykyyn ja luotettavuuteen. Tästä syystä yritykset ovat alkaneet myös kiinnostua webin mahdollisuuksista. Vaatimustasojen noustessa alkavat sovellusten koot kasvaa, jolloin ne vaativat tarkkaa suunnittelu-, kehitys- ja ylläpitotyötä. Sovelluksien kehittäjiltä tullaan siis vaatimaan paljon. (Brandon 2008, 1-5.)

Tietosuunta Oy on yritys, joka haluaa pysyä aallon harjalla myös web-käyttöliittymässä toteutetuissa sovelluksissa. Yritys on kehittänyt vuodesta 1989 asti taloushallinnon ohjelmia Windows-käyttöjärjestelmille. Heiltä löytyi kiinnostusta web-teknologioiden eri käyttömahdollisuuksista ja sitä kautta tarvetta saada laajentaa taloushallinnon tuotteita selainpohjaisiksi verkkoon. Selainta hyödyntämällä sovelluksista saataisiin pääte- ja käyttöjärjestelmäriippumattomia. Sovelluksia varten ei tarvitsisi myöskään ladata erillisiä lisäohjelmia, vaan käyttöön riittäisi selainohjelma, käyt-

täjätunnus ja salasana. Näin asiakkaille tarjottaisiin helposti lähestyttävä ja sijaintiriippumaton vaihtoehto.

1.3 Tehtävät

Työn case-tapauksena toimi Tietosuunta Oy:lle toteutettava englanninkielinen laskutusohjelma. Tämä ohjelmamoduuli edustaa kaikkia Tietosuunnan Windows-ohjelmia, jotka toimivat samalla ohjelmakoodilla. Kun laskutusohjelma on toteutettu web-sovelluksena, niin muutkin Windows ohjelmamoduulit voidaan siirtää web-ohjelmiksi.

Työn tarkoitus oli tutkia eri web-teknologioita ja tekniikoita, joiden avulla pystyttäisiin järkevästi kehittämään laaja selainkäyttöliittymässä suoritettava sovellus. Teknologioita täytyi tutkia molempien asiakas- ja palvelinpään osalta. Eri vaihtoehtoja oli tarjolla useita ja niistä täytyi selvittää case-tapaukseen sopivimmat.

Tarkoituksena oli myös ottaa selvää web-sovelluksien tietoturvaan liittyvistä riskeistä ja niiden ehkäisemisestä. Laskutusohjelmassa tulisi liikkumaan suuria määriä dataa asiakkaan ja palvelimen välillä, jolloin syötteiden tarkistukset täytyisivät olla kunnossa. Yhteys täytyisi myös suojata, koska sovelluksessa tultaisiin käsittelemään osakseen hyvin arkaluontoista tietoa, jota ei mielellään haluttaisi näkyvän sovelluksen ulkopuolelle.

2 WWW-TOIMINTAYMPÄRISTÖ

2.1 WWW-palvelut

WWW-palveluita on vaikea jakaa mihinkään tiettyihin kategorioihin. Ne eroavat toisistaan niin piirteiltään, toiminnallisuuksiltaan kuin vaatimuksiltaan. Koot vaihtelevat pienistä verkkosivuista suuriin sovelluksiin, intranetteihin ja ekstranetteihin. (Brandon 2008, 1-5.) Taulukossa 1 on pyritty jaottelemaan WWW-palveluita niiden toiminnallisuuksien mukaan.

TAULUKKO 1. WWW-sovellusten jako toiminnallisuuksien mukaan (Brandon 2008, 4)

Toiminnallisuus / Kategoria	**Esimerkkejä**
Informatiivinen	Online-uutiset, tuotekuvastot, uutiskirjeet, manuaalit, raportit, e-kirjat
Interaktiivinen	Rekisteröintilomakkeet, online-pelit
Kaupallinen	Verkkokaupat, verkkopankit, matkojen online-varaukset
Työkulkupainotteinen	Online suunnittelu ja aikataulutus, varastonhallinta, tilan seuranta, toimitusketjun hallinta
Yhteistyöympäristöt	Vuorovaikutteiset suunnittelutyökalut
Online-yhteisöt, kauppapaikat	Foorumit, online-kauppapaikat, online-tavaratalot, online-äänestys

2.2 Staattinen vs. dynaaminen web

Webin yksi perimmäisistä ideoista on jakaa HTML-dokumentteja hyperlinkkien avulla. Mikäli sisältö on muuttumatonta tai harvoin päivitystä vaativaa, perinteinen HTML-dokumentti on toimiva ratkaisu. Staattisen web-dokumentin päivitys vaatii aina tiedoston itsensä muokkaamisen ja lähettämisen palvelimelle. Näin on siis tehtävä aina, jos halutaan jonkin muutos sisältöön. Tämä prosessi on yleensä hyvin aikaa vievä, koska dokumentit saattavat olla monimutkaisia ja tiedostot on aina haettava erikseen tietokoneen kiintolevyltä ja siirrettävä oikeaan paikkaan palvelimelle. Tähän lisätään vielä se, että loppukäyttäjän vuorovaikutusmahdollisuudet dokumenttia käsitellessä ovat minimaaliset. (Rantala 2005, 3.)

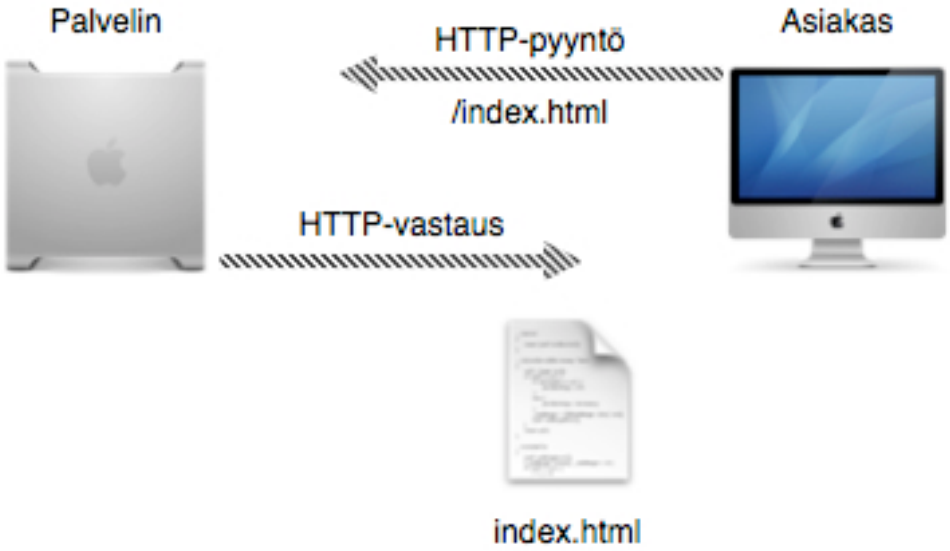
Jos käyttäjille halutaan antaa esimerkiksi kommentointioikeus sivulla julkaistaviin uutisiin, täysin staattinen HTML-dokumentti menettää merkityksensä. Tähän tarvitaan avuksi dynaamisuutta, joka tarkoittaa sivun muodostamisen automatisointia ja sitä kautta palvelinpuolen ohjelmointia. Dokumenttia ei ole kuitenkaan pakko muodostaa kokonaan palvelinpuolen tekniikoilla, koska dynaaminen osa voi olla myös vain palanen muuten staattisen dokumentin sisällä. Muuttuvan osan sisältö ja esitystapa riippuu lähtötilanteesta ja siitä miten automatisointi on hoidettu. (Rantala 2005, 4.)

Lisäämällä dynaamisuutta web-dokumentteihin sovelluksista saadaan rikkaampia ja vuorovaikutteisempia. Dynaaminen web-sivusto vaatii yleensä palvelimelta hieman enemmän. Esimerkiksi kehittäessä sovellusta PHP:lla palvelimelta vaaditaan PHP-tuki ja sen lisäksi yleensä myös jokin tietokantapalvelin.

2.3 Asiakas-palvelin –malli

Käyttäjä avaa selaimen ja kirjoittaa osoitekenttään suosikkisivustonsa URL:n ja painaa enteriä. Hetken kuluttua hänen eteensä avautuu jälleen tuttu etusivu ja hän alkaa selailla sivustoa ylälaidassa olevan navigaation avulla. Tämä on karkea esimerkki asiakas-palvelin-mallin toiminnasta.

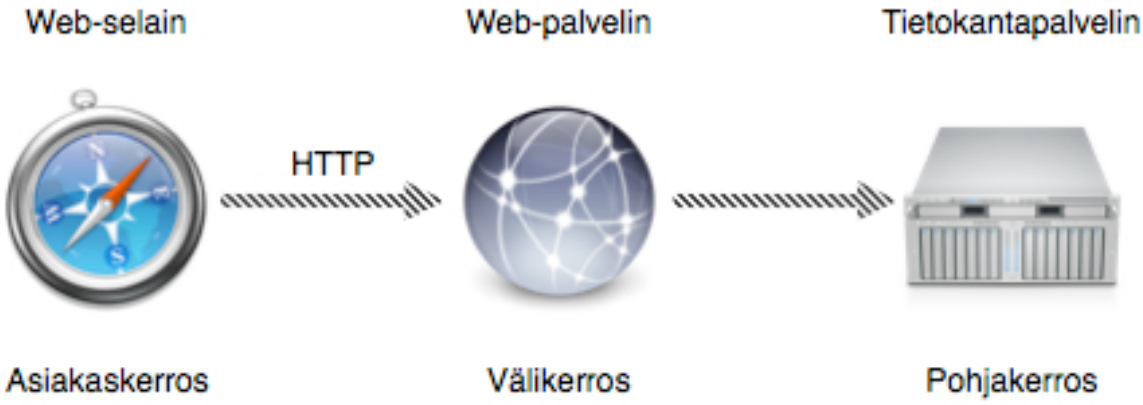
Asiakas-palvelin-malli koostuu kolmesta eri osasta: asiakasohjelmasta, palvelinohjelmasta ja yhteyskäytännöstä (protokolla). Käyttäjä lähettää asiakasohjelmalla, joka webissä on selainohjelma, palvelupyynnön (request) web-palvelimelle. Pyyntö lähetetään käyttäen HTTP-protokollaa. Samaa yhteyskäytäntöä hyödyntäen palvelin vastaa (response) asiakkaalle palauttamalla pyydetyn resurssin, joka voi olla esimerkiksi HTML-dokumentti. (Rantala 2005, 2.) Kuviossa 1 on havainnollistettu asiakas-palvelin-mallin toiminta.



KUVIO 1. Asiakas-palvelin-malli

2.4 Kolmikerrosmalli

Malli koostuu nimensä mukaisesti kolmesta kerroksesta: asiakas-, palvelin- ja pohjakerroksesta. Kolmikerrosmallia sovelletaan tilanteessa, jossa asiakas ei pysty suoraan toimimaan tietokantapalvelimen asiakkaana, jolloin on väliin toteutettava sovelluslogiikka. Välikerroksen tehtäviin kuuluu toimia asiakkaana tietokantapalvelimelle ja palvelimena asiakkaalle. Välikerros voidaan toteuttaa esimerkiksi Apache-palvelinta ja PHP-ympäristöä hyödyntäen. (Rantala 2005, 254.) Tietokantapalvelimeksi voidaan valita esimerkiksi MySQL, josta lisää luvussa 3.3.3. Kuviossa 2 on kuvattu kolmikerrosmalli.



KUVIO 2. Kolmikerrosmalli

2.5 MVC-arkkitehtuuri

Model-view-controller eli malli-näkymä-ohjain on arkkitehtuurimalli, jonka tarkoituksena on jakaa sovellus kolmeen toisistaan riippumattomaan osaan: näkymään, malliin ja ohjaimeen. MVC erottaa käyttöliittymän, liiketoimintalogiikan ja sovelluslogiikan toisistaan. Arkkitehtuurimalli tulee kyseeseen varsinkin laajemmissa sovelluksissa, mutta sitä suositellaan käytettävän aina mahdollisuuksien mukaan. Luokkien uudelleenkäytettävyys on yksi MVC:n vahvuuksista. Sovelluksen laajennettavuus ja virheiden helpompi löytäminen ja kitkentä ovat myös niitä piirteitä, jotka nostavat MVC:n

avainasemaan sopivaa arkkitehtuurimallia valittaessa. MVC:ssä myös pyritään siihen, ettei samoja ohjelmakoodeja toistettaisi kahteen kertaan. (Lecky-Thompson, Nowicki & Myer 2009, 306.)

MVC-mallin avulla pystytään tarvittaessa jakamaan sovelluksen kehitystyötä eri osa-alueiden osaajille. Käyttöliittymäsuunnittelijat voivat keskittyä ulkoasuun ja asiakaspuolen toiminnallisuuden suunnitteluun eikä heidän tarvitse tietää sovelluksen alla toimivasta logiikasta mitään. Samoin ohjelmoijat voivat keskittyä täysin luokkien, tietokantayhteyksien ja tietoturvan toteuttamiseen. (McArthur 2008, 201.) Seuraavaksi käydään läpi arkkitehtuurimallin kolme eri osaa.

Malli

Mallin (Model) tehtävä on hoitaa sovelluksen liiketoimintalogiikka (business logic). Se vastaa yhteydestä tietovarastoon, joka voi olla esimerkiksi tietokanta. Malli hakee, muokkaa ja varastoi tietoa sen mukaan, miten ohjain sitä käskee. Sovellukset sisältävät normaalisti useita malleja. Mallien avulla saadaan aikaan ohjelman toiminnot. (Lecky-Thompson, Nowicki & Myer 2009, 306-307.)

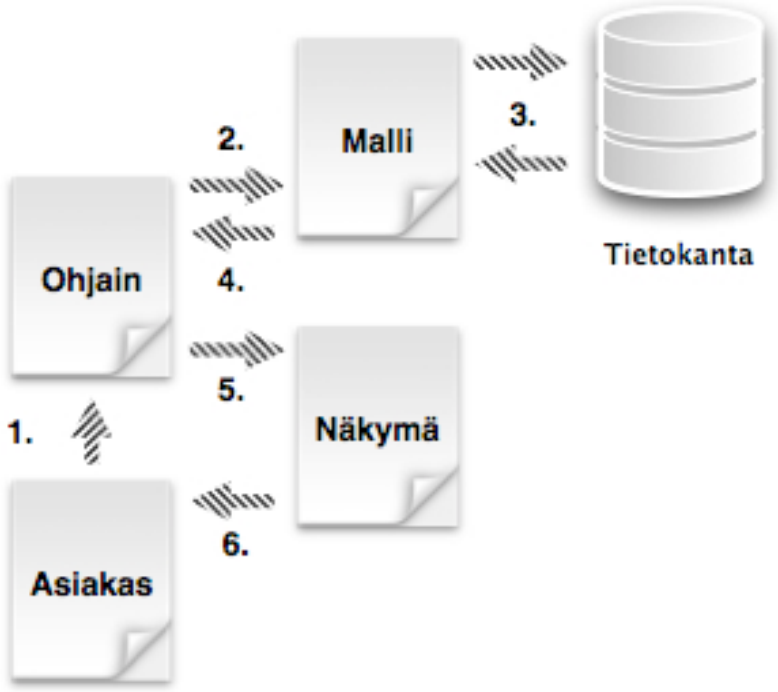
Näkymä

Näkymän (View) tehtävä on saattaa data luettavaan muotoon. Web-sovelluksessa näkymän muodostamisessa voidaan käyttää esimerkiksi HTML-merkkauskieltä tai XML:ää. (Mts. 307.)

Ohjain

Malli ja näkymä eivät ole tietoisia toisistaan. Ohjain (Controller) toimii niiden kahden välissä. Ohjain käsittelee käyttäjältä tulleet syötteet ja hoitaa sovelluksen toimintavuon. (Mts. 307.)

Kuviosta 3 nähdään vielä MVC-arkkitehtuurin karkea kulkukaavio. (1) Ohjain vastaanottaa asiakkaalta tulleen syötteen. (2-3) Ohjain valitsee syötteen perusteella mallin, joka hakee tietoa tietokannasta. (4-5) Ohjain siirtää mallilta saadun tiedon näkymälle. (6) Näkymä luo saadusta datasta esityksen asiakkaalle.



KUVIO 3. MVC-arkkitehtuurin kulkukaavio

2.6 Sovelluskehys

Kun puhutaan sovelluskehyksestä (framework), puhutaan usein ohjelmakirjastosta, jossa on valmiiksi toteutettu sovelluksille ominaiset perusominaisuudet (O'Brien 2007). Perinteiseen luokkakirjastoon verraten sovelluskehys sisältää vuorovaikutuksen ohjauksen, ajonaikaisia riippuvuuksia luokkien välillä sekä mallin olioiden välisistä vuorovaikutuksista (Juslin 1999). Kehykset nopeuttavat kehitystyötä, koska ohjelmoija voi keskittyä itse sovelluksen tekemiseen (O'Brien 2007). Sovelluskehys ei itsessään vielä ole valmis sovellus, vaan sen päälle luodaan varsinainen toteutus. Taulukkoon 2 on listattu muutamia tärkeitä ja hyödyllisiä asioita, joita koetaan sovelluskehyksen käyttöönotosta saatavan.

TAULUKKO 2. Listaus sovelluskehyksistä saatavista hyödyistä (Top 10 Reasons Why You Should Use a PHP Framework 2009)

- Parempi tiedostojen ja ohjelmakoodien organisointi, koska sovelluskehys sisältää selkeän hakemistorakenteen, jota tulee noudattaa.
- Sisältävät paljon valmiiksi tehtyjä kirjastoja ja komponentteja perusominaisuuksien toteuttamiseen.
- Useimmat noudattavat MVC-arkkitehtuurimallia.
- Automaattiset datan filtteröinnit ja tarkistukset tietoturvariskien ehkäisemiseksi.
- Nopeampi sovelluskehitys, koska koodin kirjoittaminen vähenee.
- Sovelluksen suorituskyky parantuu, koska käyttöön ladataan vain tarvittavat luokat ja ohjelmakoodit.
- Aktiivinen käyttäjäyhteisö, jolloin apua on saatavissa nopeasti erilaisilta foorumeilta ja sähköpostilistoilta.
- Soveltuu hyvin projekteihin, joissa työskennellään tiimeissä. Sovelluksen eri osat voidaan jakaa eri osa-alueiden osaajille, kuten käyttöliittymäsuunnittelijoille ja tietokantasuunnittelijoille.

PHP sovelluskehyksiä löytyy markkinoilta nykyään useita. Osa on paremmin ylläpidettyjä ja tuettuja kuin toiset. Kehyksiä löytyy myös laajuudeltaan eri kokoisia. Suurinta osaa yhdistää kuitenkin se, että ne ovat saatavilla avoimen lähdekoodin lisenssillä. (Lecky-Thompson, Nowicki & Myer 2009, 355-35.) Erityisominaisuudet ovat avainasemassa tarpeisiin sopivaa kehystä valittaessa (Gilmore 2008, 605). Seuraavaksi käydään läpi pääpiirteittäin neljä suosittua vaihtoehtoa sovelluksen alustaksi.

CakePHP

CakePHP on ammentanut paljon vaikutteita Ruby on Rails -sovelluskehyksestä. CakePHP:n kehityksen aloitti Michal Tarynowicz vuonna 2005. Se on yksi suosituimmista PHP:lla toteutetuista sovelluskehyksistä. CakePHP haluaa tuoda yksinkertaisuutta ja skaalautuvuutta sovelluksiin. (Mts. 605.) CakePHP:n vahvuuksiin kuuluvat tiedonkäsittelymetodit. Sovelluskehyksestä löytyvät yksinkertaiset haut tietokantaan, eikä varsinaisia kokonaisia SQL-lauseita tarvitse kirjoittaa. (Golding 2008, 290-291.)

CodeIgniter

CodeIgniter on erityisesti web-sivustojen toteutuksia varten kehitetty sovelluskehys. Se on kevyt, mutta tarjoaa paljon valmiiksi luotuja ohjelmakomponentteja. (CodeIgniter Introduction 2007.) Sovelluskehyksen vahvuuksiin lukeutuvat helppo asennus, nopea kehitystyön aloittaminen ja kattava online-dokumentointi (Welcome to CodeIgniter 2010).

Symfony

Symfony on Fabien Potencierin luoma ja ranskalaisen web-ohjelmistoyrityksen Sensio sponsoroima sovelluskehys. Se on rakennettu kolmen muun avoimen lähdekoodin ohjelman päälle. Creole hoitaa tietokantarajapinnan, Mojavi MVC-arkkitehtuurin ja Propel ORM-kerroksen. (Gilmore 2008, 606.) Object-Relational Mapping (ORM) tarkoittaa tekniikka, jonka avulla pystytään käsittelemään tietokantoja samalla tavalla kuin käsiteltäisiin luokkia ja olioita. Yleensä joudutaan erikseen määrittelemään metodit luokkien ominaisuuksien arvojen hakemiseksi ja käsittelemiseksi. (Propel 2009.) Symfonyn vahvuuksiin kuuluvat automaattiset lomakkeiden tarkistukset, sivunumerointi, ostoskorin hallinta ja Ajax-tuki (Gilmore 2008, 606). Symfonysta löytyvät myös työkalut testaukseen, vianetsintään ja dokumentointiin (Symfony Introduction, 2007). Askeet ja Yahoo! Bookmarks webpalvelut ovat kehitetty käyttäen Symfonya (O'Brien 2007).

Zend Framework

Neljästä sovelluskehyksestä ehkä kaikista tunnetuin Zend Framework on PHP-tuotteita ja palveluita tarjoavan Zend Technologiesin luomus. Sama yritys on ollut PHP:n moottorin takana kolmosversiosta lähtien. Kehys on näistä neljästä sovelluskehyksestä laajin. Zend Frameworkille löytyy verkosta todella suuri käyttäjäyhteisö ja läpikotainen online-dokumentaatio. Nämä on ovat mm. yksiä avaintekijöitä nopealle kehitykselle, koska vastaukset ongelmatilanteisiin on löydettävissä usein todella nopeasti. Itse sovelluskehys on joustava. Se ei nojaa mihinkään tiettyyn paradigmaan. Kuitenkin sovelluskehyksissä tuttu MVC-arkkitehtuurimalli on täysin toteutettavissa. (Golding 2008, 293.) Zend Framework tarjoaa kattavan kirjaston web-sovelluksille tyypillisiä ohjelmakomponentteja. Sovelluskehys sisältää myös web-sovelluksen rakenteen kannalta ei niin välttämättömiä moduuleita kuten PDF:n ja RSS-syötteiden

luonnin ja rajapintoja ulkopuolisiin verkkopalveluihin kuten Amazon, Flickr ja Yahoo. (Gilmore 2008, 607.)

Taulukkoon 3 on koottu ja verrattu edellä esiteltyjen neljän sovelluskehyksen pääpiirteitä. Taulukosta huomataan, että sovelluskehyksillä on keskenään eroavaisuuksia, jotka saattavat vaikuttaa ratkaisevasti sopivaa sovelluskehystä valittaessa. Jos halutaan, että sovellus voidaan asentaa palvelimille, joissa on käytössä vanhempi PHP:n versio, joudutaan pudottaa Symfony ja Zend Framework pois vaihtoehdoista. CodeIgniter on kehyksistä ainut, joka sisältää valmiin mallinejärjestelmän. Toisaalta se on myös ainut, joka ei sisällä valmiina tukea ORM:lle ja Ajax:lle. CodeIgniteristä puuttuu myös valmis käyttäjäkirjautuminen sekä muita lisämoduuleita. Tämä taas voi kertoa siitä, että CodeIgniter on verratuista neljästä sovelluskehyksestä kaikista kevyin.

TAULUKKO 3. Neljän tyypillisimmän sovelluskehysvaihtoehdon vertailu (PHP Frameworks 2007)

Sovelluskehys	CakePHP	CodeIgniter	Symfony	Zend
PHP4-tuki	**x**	**x**		
PHP5-tuki	**x**	**x**	**x**	**x**
MVC-arkkitehtuurimalli	**x**	**x**	**x**	**x**
Useita tietokantarajapintoja	**x**	**x**	**x**	**x**
Mallinejärjestelmä		**x**		
ORM	**x**		**x**	**x**
Validointi / filtteröinti	**x**	**x**	**x**	**x**
Ajax-tuki	**x**		**x**	**x**
Käyttäjäkirjautuminen	**x**		**x**	**x**
Moduulit (RSS, PDF ym. hyödyllisiä komponentteja)	**x**		**x**	**x**

2.7 Avoin lähdekoodi ja lisensointi

Ohjelmistokehityksessä kohdataan väistämättä jossain vaiheessa lisensointikysymykset. Lisenssi määrää mm., kenellä on oikeudet ohjelmaan, sen julkaisuun, levittämiseen ja kehittämiseen. Sovelluksia voidaan julkaista suljetun, osittain avoimen tai kokonaan avoimen lähdekoodin ohjelmistoina riippuen valitusta lisenssistä. Suljetun ja avoimen ero on se, että avoimen lähdekoodin (open source) sovelluksia voidaan vapaasti muokata ja jakaa, kun taas suljettu sovellus (proprietary) on tehty tiettyä tarkoitusta varten, yksityiskäyttöön, eikä lähdekoodiin pääse ulkopuoliset käsiksi. Tässä luvussa selvitetään, mitä tarkoitetaan avoimella lähdekoodilla, ja käydään karkeasti läpi neljä yleisintä sovelluksissa käytettävää lisensointityyppiä.

Avoin lähdekoodi

Avoimen lähdekoodin perimmäinen idea on poistaa yksinoikeudet tehdystä työstä. Siinä missä suljetun koodin sovellus rajaa ohjelman henkilökohtaiseen käyttöön tiettyä tarkoitusta varten muuntelemattomana, avoimen lähdekoodin idea on toimia päinvastoin ja poistaa nämä rajoitukset. Avoimen lähdekoodin sovelluksia voidaan vapaasti jakaa. Ohjelmien mukana jaettavaa lähdekoodia saa muokata omien tarpeidensa mukaan ja luoda haluttaessa omia uusia sovelluksia. Johdannaissovelluksiin pätevät samat lisenssiehdot, mitä alkuperäisen sovelluksen lisenssissä on määritelty. Jos alkuperäissovellusta saa jakaa ja muokata oman halunsa mukaan, on johdettu sovellus vastaavassa asemassa. Tällä tavoin saadaan esimerkiksi estettyä lähdekoodin sulkeutuminen. (Mts. 4-6.)

Avoimen lähdekoodin vaalimisesta koetaan saatavan suurta hyötyä perinteiseen suljettuun koodiin verrattuna. Innovatiivisuus on yksi merkittävistä tekijöistä. Mitä enemmän saadaan henkilöitä mukaan projektiin viemään sovellusta eteenpäin, sitä enemmän projekti saa arvoa. Sovelluksen vakaus kohenee, kun ohjelmoijia saadaan enemmän testaamaan. Yksittäinen pätevä ohjelmoija voi kitkeä havaitsemansa rajoitteet ja virheet nopeasti, koska hänellä on suora pääsy lähdekoodiin. Ohjelman kehitys nopeutuu ja ongelmat vähenevät, koska ohjelmoijan ei tarvitse olla riippuvainen organisatorisista prioriteeteista ja järjestelyistä. Sovelluksille saadaan myös jatkuvuutta. Kun suljetun sovelluksen kehitystyö päättyy, tulee ohjelma elinkaarensa määränpäähän. Oh-

jelmistotalo ei enää tue sovellusta ja päivitykset loppuvat. Avoimen lähdekoodin sovelluksen kehitystyö voi myös yhtä lailla loppua, mutta kuka tahansa voi sitä halutessaan jatkaa ja muokata sen halutessaan täysin eri käyttötarkoitusta varten. (Mts. 6-7.)

MIT

MIT-lisenssi antaa täydet oikeudet sovelluksen muokkaamiselle ja jakamiselle. Sovellus tarjotaan sellaisenaan kuin se on. Toimivuus ja käytöstä koituvat seuraukset ovat käyttäjän vastuulla. Kehittäjä voi halutessaan määrätä sovelluksen jatkokehittäjiä toimimaan samoin ehdoin, mutta tähän ei pakoteta. (Laurent 2004, 14-15.)

BSD

BSD-lisenssin ainut ero MIT-lisenssiin on se, ettei alkuperäissovelluksen kehittäjän nimeä saa käyttää johdannaissovelluksien eteenpäin viemisessä ja mainostamisessa ilman kehittäjän omaa lupaa. Tällä tavoin suojataan sovelluksen alkuperäiskehittäjän maine, jos on esimerkiksi kyse heikosti ohjelmoidusta johdannaissovelluksesta. (Mts. 15-16.)

GPL

GPL:n (GNU General Public Licence) ideana on pitää avoimen lähdekoodin sovellukset vapaana, niin että niitä voidaan vapaasti jakaa ja muokata ilman erillistä sovelluksen alkuperäistekijän lupaa. Tämä tarkoittaa myös sitä, että johdannaissovelluksen on noudatettava samoja määritteitä. GPL:n alainen sovellus on oltava vapaa patenteista. Patentoitu koodi on lisensoitava erikseen. (Mts. 36-37.)

LGPL

LGPL (GNU Lesser General Public Licence) sisältää paljon samoja määritteitä kuin GPL, mutta se antaa kehittäjälle mahdollisuuden julkaista johdannaissovellukset ja kirjastot valitsemallaan muulla lisenssillä. Tämä tarkoittaa sitä, että LGPL-lisensoitu sovellus voidaan lisensoida haluttaessa esimerkiksi GPL:n alaisuuteen. (Mts. 51-52.)

MPL

MPL:llä (The Mozilla Public Licence) lisensoidun sovelluksen lähdekoodista osa voidaan määrittää haluttaessa suljetuksi. Tämä tarkoittaa sitä, että lähdekoodiin voidaan

yhdistää jonkin muun lisenssin alaista koodia, mikä esimerkiksi GPL:ssä on tiukasti kiellettyä. (Mts. 63.)

3 WWW-TOTEUTUSTEKNIIKAT

3.1 Kommunikointiosa

3.1.1 URI

Lyhenne URI tulee sanoista Uniform Resource Identifier, ja se tarkoittaa yleistä resurssin tunnistetta. URI koostuu kahdesta alakäsitteestä: osoitteesta ja nimestä. Tarkoituksena on ollut, että webissä oleviin resursseihin voitaisiin viitata URI:n molemmilla osilla. (Korpela & Linjama 2005, 414.)

URL

Uniform Resource Locator eli URL on URI:n osoiterakenne. Se voi olla joko suhteellinen tai absoluuttinen. Suhteellisella osoitteella tarkoitetaan lyhenteellistä viittausta absoluuttiseen osoitteeseen tietyin säännöin. (Mts. 407.) Jos osoitteen edestä puuttuu protokolla, on kyseessä suhteellinen URL. Taulukossa 4 on esitelty URL:n rakenne.

TAULUKKO 4. URL:n rakenne

Protokolla	**Palvelin**	**Polku**	**Kysely**	**Fragmentti**
http://	www.noise.fi	/uutiset/index.html	?rid=11401	#11401

URL:n protokollaosasta nähdään, mitä yhteyskäytäntöä käytetään resurssin hakemiseen. Se voi olla esimerkiksi HTTP, HTTPS tai FTP . Palvelinosan domain eli Internet-nimi on se paikka, josta haettava resurssi on saatavilla. (Mts. 407-409.)

Polku on merkkijono, jolla yksilöidään resurssi. Palvelin tulkitsee polun haluamallaan tavalla, jolloin polku on riippumaton hakemistoista ja tiedostoista. On kuitenkin hyvin yleistä, että sillä viitataan juuri kansioihin ja niissä oleviin tiedostoihin. (Mts. 409-410.)

Kyselyosa alkaa kysymysmerkillä, jonka jälkeen tulee &-merkillä eroteltuja nimi-arvo-pareja. Näitä voidaan lukea esimerkiksi palvelimella olevalla PHP sovelluksella. Kuviossa 4 on esimerkki PHP-sovelluksesta, joka vastaanottaa URL:n kyselyosasta nimi-arvo-parit.

```php
<?php

    // URL -> http://www.esimerkki.fi/index.php?etunimi=Matti&sukunimi=Meikalainen

    //Index.php
    echo $_GET["etunimi"]." ".$_GET["sunuimi"];;

    // Tulostaa Matti Meikalainen

?>
```

KUVIO 4. Esimerkki PHP-sovelluksesta, joka vastaanottaa nimi-arvo-parit

Kyselyosa päättyy ristikkomerkkiin (#), jonka jälkeen seuraa fragmenttiosa. Fragmentilla viitataan tiettyyn osaan haetun resurssin sisällä. (Mts. 410.)

URN

Uniform Resource Name eli URN on URI:n nimirakenne. URN:n avulla verkossa oleva resurssi nimetään pysyvästi. Nimeä ei voi siirtää jollekin toiselle. URN on myös yksikäsitteinen, joten se voi viitata vain yhteen ainoaan resurssiin. Yhteen resurssiin voi kuitenkin liittyä monia eri URNeja (Varjonen 2000.) URN:t ovat palvelin- ja protokollariippumattomia (Korpela & Linjama 2005, 414).

URLia käytettäessä käyttäjän täytyy tietää resurssin tarkka sijainti, tiedostonimi ja tiedostopääte. URNia käyttäessä ei tarvitse tietää kuin resurssiin viittaava nimi. (What is URN 2004.) Kuviossa 5 on esitelty esimerkkitapaus URN:n käytöstä.

```
URN:ISBN:951425533X
```

KUVIO 5. URN:n merkintätapa (Varjonen 2000)

3.1.2 HTTP

HTTP on lyhenne sanoista Hypertext Transfer Protocol, ja se tarkoittaa standardoitua tiedonsiirrossa käytettävää yhteyskäytäntöä asiakkaan ja palvelimen välillä (Rantala 2005, 99). HTTP-tiedonsiirto alkaa, kun asiakas lähettää pyynnön TCP:n kautta (yleensä portti 80) palvelimelle, joka taas lähettää asiakkaalle takaisin tilakoodin ja vastauksen (Davis & Phillips 2007, 2). Protokollan avulla voidaan siirtää monentyyppistä dataa kuten HTML-dokumentteja, kuvia ja ääntä.

3.1.3 JSON

JavaScript Object Notation eli JSON on asiakkaan ja palvelimen välillä käytettävä tiedonsiirtomuoto (JQuery Cookbook 2009, 405). JSON on riippumaton käytettävästä ohjelmointikielestä. Sen käsittelyyn käytetään kuitenkin C-kielestä tuttuja tapoja. (JSON n. d.) JSONia on kevyt ja helppo käsitellä. JSON muodostetaan koodaamalla haluttu olio merkkijonomuotoon. Tämän jälkeen se voidaan lähettää palvelimelle sellaisenaan. Vastaavasti vastauksena saatu JSON parseroidaan takaisin oliomuotoon. (JQuery Cookbook 2009, 405.) Kuviossa 6 on esitelty, kuinka luodaan JSON-merkkijono. Luomiseen on käytetty apuna www.json.org sivustolta saatavaa JSONin käsittelyyn tarkoitettua JavaScript-kirjastoa.

```
1 <script type="text/javascript">
2     // Koodaamaton (Oliomuoto)
3     var object = {etunimi: 'Matti', sukunimi: 'Meikalainen'};
4     // Koodattu (Merkkijonomuoto)
5     var string = '{"etunimi":"Matti","sukunimi": "Meikalainen"}';
6 </script>
```

KUVIO 6. Esimerkki JSONin käytöstä erillisestä JavaScript-kirjastosta saatavien funktioiden avulla

3.2 Asiakaspuoli

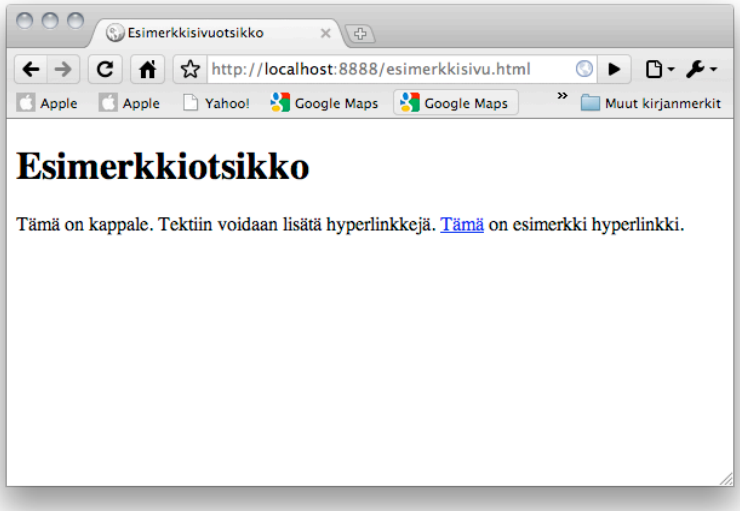
3.2.1 (X)HTML

Hypertext MarkUp Language eli HTML on web-dokumenttien kuvauksessa käytettävä merkintäkieli. Sen määritelmä perustuu SGML:ään. HTML sisältää tageja eli merkkejä, joita käytetään dokumentin elementtien muodostamiseen. Tageilla voidaan erotella, mikä osa hypertekstiä on otsikko, kappale, linkki, lista jne. (What is HTML 4, 2002.) Kuviossa 7 on esimerkki HTML-merkkauksesta HTML 4.01 Transitional standardin mukaan. Kuviossa 8 on nähtävissä, miltä edellä mainittu merkkaus näyttää selaimelle tulostettuna.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Esimerkkisivuotsikko</title>
</head>
<body>
        <h1>Esimerkkiotsikko</h1>

            <p>
                Tämä on kappale. Tektiin voidaan lisätä hyperlinkkejä.
                <a href="http://www.esimerkkisivu.com" target="_blank">Tämä</a>
                on esimerkki hyperlinkki.
            </p>
</body>
</html>
```

KUVIO 7. Esimerkki HTML-merkkauksesta käyttäen HTML 4.01 Transitional –dokumenttityyppiä



KUVIO 8. Esimerkkisivu.html tulostettuna Googlen Chrome-verkkoselaimelle

XHTML (The Extensible HyperText Markup Language) on XML:ään pohjautuva merkintäkieli ja HTML:n laajennus (What is XHTML 2002).

3.2.2 CSS

CSS (Cascading Style Sheets) on ohje, jolla määritetään web-dokumenttien tyylit. CSS:n avulla saadaan erotettua tyyli merkkauskielestä, jolloin tagien sisäiset tyyliin vaikuttavat omat attribuutit saadaan poistettua. (Bos 2010.) Tällä tavalla pystytään paremmin hallitsemaan dokumentin ulkoasu. HTML-dokumentissa tyylit voidaan määritellä joko suoraan style-attribuutilla (ks. kuvio 9) elementin sisällä, dokumentin header-osassa (ks. kuvio 10) tai erillisessä CSS-tiedostossa (ks. kuviot 11 ja 12). CSS:llä pystytään määrittää tyylit myös eri päätelaitteille. (Three ways to insert CSS 2010.)

```
1 <p style="color:sienna;margin-left:20px">This is a paragraph.</p>
```

KUVIO 9. Tyylimäärittely kappale-elementin attribuuttina (CSS How to 2010)

```
<head>
<style type="text/css">
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
</style>
</head>
```

KUVIO 10. Tyylimäärittely web-dokumentin header-osassa (CSS How to 2010)

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css" />
</head>
```

KUVIO 11. CSS-tyylitiedoston linkitys web-dokumentin header-osassa (CSS How to 2010)

```
hr {color:sienna;}
p {margin-left:20px;}
body {background-image:url("images/back40.gif");}
```

KUVIO 12. Tyylimäärittely erillisessä CSS-tiedostossa (CSS How to 2010)

3.2.3 JavaScript

JavaScript on mm. HTML-dokumenteissa käytettävä oliopohjainen skriptikieli, jota hyödyntämällä voidaan luoda dynaamisuutta ja vuorovaikutteisuutta web-sivulle. Kieli perustuu ECMAScript-standardiin. JavaScript on tulkattava ohjelmointikieli, mikä tarkoittaa sitä, ettei ohjelmaa tarvitse kääntää ensin, vaan sitä luetaan ja suoritetaan ajon aikana rivi riviltä. Tulkattavuuden ansiosta ohjelmia on helppo ja nopea kehittää ja testata. Muutokset astuvat heti voimaan, kun dokumentti ladataan selaimessa uudelleen. (Peltomäki & Nykänen 2006, 90-97.) Kuviossa 13 on kuvattu JavaScript-esimerkkisovellus.

```
 8
 9      <script type="text/javascript">
10
11          function tulosta_teksti() {
12              document.write("Hei maailma!");
13          }
14
15      </script>
16
17  </head>
18  <body>
19
20      <p>
21          <script type="text/javascript">
22
23              // Dokumentin ladattua, tulostaa "Hei maailma!"
24              window.onload = tulosta_teksti();
25
26          </script>
27      </p>
28
29  </body>
```

KUVIO 13. JavaScript-esimerkkisovellus

JavaScript-ohjelma voi olla joko muutaman rivin skripti tai kokonainen sovellus. Skriptin avulla voidaan laajentaa jo olemassa olevaa käyttöliittymää käsittelemällä ja automatisoimalla sen eri toimintoja. JavaScript ei rajoitu vain selaimeen, vaan sitä voidaan käyttää myös muissa isäntäohjelmissa. Ohjelmilta vaaditaan vain JavaScript-tulkki. (Mts. 90-97.)

3.2.4 JQuery

JQuery on avoimen lähdekoodin JavaScript-kirjasto. Sen tarkoituksena on helpottaa HTML:n dokumenttioliomallin (DOM) käsittelyä (Linldey 2009, 1-2.) Slogani kuuluukin: ”Kirjoita vähemmän, saa enemmän aikaan” (JQuery 2010). JQueryn vahvuuksiin kuuluvat kätevät valitsimet CSS:n tyylisesti ja metodien ketjuttaminen (Linldey 2009, 1-2). Kuviosta 14 nähdään, miten koodin kirjoittaminen JQuerylla ja puhtaalla JavaScriptilla eroavat toisistaan.

```
 9      <script type="text/javascript">
10
11          // JavaScript
12          var value = document.getElementById("UsernameInput").value;
13
14          //JQuery
15          var value = $("UsernameInput").val();
16
17      </script>
18
19  </head>
```

KUVIO 14. Kuvasta nähdään kuinka JQueryn syntaksi lyhentää tarvittavan JavaScript-koodin kirjoittamisen määrää

3.2.5 Ajax

Perinteisen web-sovelluksen toiminta voidaan kuvata pyynnön lähettämisellä palvelimelle, joka vastaa pyydetyllä HTML-dokumentilla. Tämä tarkoittaa sitä, että koko sivu ladataan aina uudestaan, kun käyttäjä esimerkiksi klikkaa sivuston valikosta valitsemaansa linkkiä edetäkseen sivustolla. Asynchronous JavaScript and XML eli Ajax antaa mahdollisuuden siihen, että dataa voidaan lähettää ja vastaanottaa ilman, että sivua tarvitsee rakentaa kokonaan uudestaan. Tekniikan avulla voidaan muuttaa pelkästään tiettyä osaa sivusta. Näin palvelusta saadaan nopeampi ja käyttäjäystävällisempi. (Peltomäki & Nykänen 2006, 296.)

Ajaxin toiminta perustuu XMLHttpRequest-olioon, joka antaa mahdollisuudet asynkronisien pyyntöjen lähettämiseen palvelimelle. XMLHttpRequest ei ole W3C:n standardi. Siitä huolimatta, että XMLHttpRequest olio luodaan hieman eri tavalla eri selaimilla (ks. kuvio 15), sen käyttö on kaikin puolin samanlaista ja siitä saatavan suuren hyödyn myötä myös hyvin suositeltavaa. (Mts. 299-300.)

```
function GetXmlHttpObject()
{
if (window.XMLHttpRequest)
  {
  // code for IE7+, Firefox, Chrome, Opera, Safari
  return new XMLHttpRequest();
  }
if (window.ActiveXObject)
  {
  // code for IE6, IE5
  return new ActiveXObject("Microsoft.XMLHTTP");
  }
return null;
}
```

KUVIO 15. JavaScriptilla voidaan toteuttaa XMLHttpRequest-olion luontia varten oma funktio. Funktion avulla voidaan tarkistaa, mitä verkkoselainta asiakas käyttää ja näin luoda oikeanlainen XMLHttpRequest-olio Ajaxin käyttöä varten (Ajax Suggest 2010)

3.3 Palvelinpuoli

3.3.1 PHP

PHP kehitys aloitettiin vuonna 1994 Rasmus Lerdorfin toimesta. Kieli tunnettiin tällöin nimellä Personal Home Page Tools. (Rantala 2005, s. 9.) Vuonna 1997 julkaistiin PHP:n toinen versio PHP 2.0, jolloin nimi muuttui Personal Home Page / Form Interpreter:ksi (PHP / FI). PHP 3.0 julkaistiin vuonna 1998, jolloin kielen suosio alkoi kasvamaan nopeaa vauhtia. Tällöin PHP:n nimi sai nykyisen muotonsa: PHP: Hypertext Preprocessor. (Gilmore 2008, 2.) Tällä hetkellä PHP:stä on julkaistu viides versio PHP 5.

PHP on palvelinpuolen skriptikieli ja sitä käytetään web-sovelluksien ohjelmointiin. Kielen syntaksi muistuttaa paljon C-kieltä (Rantala 2005, 9.) PHP on helppo omaksua ja sen avulla pystytään luomaan nopeasti tehokkaita sovelluksia (Suehring, Converse & Park 2009, 5). Kuten kuviosta 16 nähdään, yksinkertaisimmillaan PHP-skripti voi koostua yhdestä rivistä, kun esimerkiksi C-kielessä (ks. kuvio 17) täytyy sisällyttää pakollisia kirjastoja (Gilmore 2008, 7).

```php
<?php echo "Hello World!"; ?>
```

KUVIO 16. Esimerkki yksinkertaisesta PHP-skriptistä

```c
#include <stdio.h>

int main(void)
{
  printf("Hello world!\n");
  return 0;
}
```

KUVIO 17. Hello world -sovellus C-kielellä kirjoitettuna

PHP on heikosti tyypitetty kieli, mikä tarkoittaa sitä ettei muuttujia tarvitse välttämättä erikseen luoda, tyypitellä tai tuhota, vaan PHP tekee nämä toimenpiteet ajon aikana automaattisesti (Mts. 7). PHP-koodia voidaan upottaa suoraan HTML-dokumenttiin, jolloin saadaan luotua dynaamisia web-sivuja. Käyttäjän onkin vaikea suoraan sanoa, mikä osa HTML-dokumentista on toteutettu PHP:lla, kun se on tulostettu näytölle. PHP sisältää myös olio-ohjelmointiominaisuudet, joidenka ansioista sovelluksista ja niiden lähdekoodeista saadaan paremmin skaalattavia, järjestettyjä ja uudelleen käytettäviä. (Rantala 2005, s. 64.)

PHP on avoimen lähdekoodin ohjelmisto, jolloin se on saatavissa täysin ilmaiseksi. Kieli tarvitsee toimiakseen web-palvelinohjelman. Tähän käytetään yleensä Linux-alustalle asennettua Apachea. (Rantala 2005, 9-10.) PHP:lle löytyy suuri ja aktiivinen käyttäjäyhteisö. Verkosta on saatavilla runsaasti valmista ohjelmakoodia ja dokumentaatiota. Käyttäjät ovat myös valmiita antamaan neuvoja ja auttamaan eri ongelmatilanteissa. (Suehring, Converse & Park 2009, 9.)

3.3.2 Smarty

Smarty on mallinejärjestelmä (template engine), jonka tarkoituksena on erottaa PHP-sovelluslogiikka ja HTML-esitys toisistaan (ks. kuvio 18). Smarty kääntää mallineet (ks. kuvio 19) ja luo niistä PHP-skriptit ennalta määrättyyn kansioon. Näitä skriptejä käytetään jatkossa esityksien (ks. kuvio 20) toteuttamiseen. Näin säästetään resursseja,

koska Smartyn ei tarvitse aina kääntää mallinetta uudestaan, vaan ainoastaan silloin, kun siitä on uudempi versio löydettävissä. (What is Smarty 2008.)

Mallineeseen voidaan haluttaessa sisällyttää logiikkaa kuten muuttujia ja ohjausoperaattoreita. Ei ole kuitenkaan suositeltavaa, että mallineisiin sisällytetään liiketoimintalogiikkaa. Se on kuitenkin täysin mahdollista, koska mallineeseen voidaan kirjoittaa haluttaessa myös puhdasta PHP-koodia. Tässä rikotaan kuitenkin mallinjärjestelmän perimmäistä ideaa. Smarty ei osaa erottaa esitys- ja liiketoimintalogiikkaa toisistaan. PHP-jäsentäjän hoitaessa esityksien rakentamisen, Smarty on nopea ja tehokas. Järjestelmään voidaan luoda myös omia funktioita ja muuttujien käsittelijöitä, mikä tekee siitä myös laajennettavan. (Mts.)

index.php

```
include('Smarty.class.php');

// create object
$smarty = new Smarty;

// assign some content. This would typically come from
// a database or other source, but we'll use static
// values for the purpose of this example.
$smarty->assign('name', 'george smith');
$smarty->assign('address', '45th & Harris');

// display it
$smarty->display('index.tpl');
```

KUVIO 18. PHP-skriptissä määritetään muuttujat mallinetta varten (Crash Course 2008)

index.tpl

```
<html>
<head>
<title>User Info</title>
</head>
<body>

User Information:<p>

Name: {$name}<br>
Address: {$address}<br>

</body>
</html>
```

KUVIO 19. Malline sisältää muuttujien esityksen (Crash Course 2008)

```
<html>
<head>
<title>User Info</title>
</head>
<body>

User Information:<p>

Name: george smith<br>
Address: 45th & Harris<br>

</body>
</html>
```

KUVIO 20. Tulostus ajon jälkeen (Crash Course 2008)

3.3.3 MySQL

MySQL on avoimen lähdekoodin SQL-tietokannan hallintajärjestelmä. Sen kehitti ruotsalainen TcX DataKonsult AB. Ensimmäinen versio julkaistiin vuonna 1996. Tätä nykyä MySQL:ää kehittää Oracle Corporation ja siitä on julkaistu 5.5 versio. (What is MySQL, 2010.)

MySQL on suosittu sen helposta käytettävyydestä, nopeudesta, vakaudesta ja lisensointitavasta (Suehring, Converse & Park 2009, 4). PHP-ympäristössä MySQL on

yleisin valinta tietokantajärjestelmäksi. Yleisesti puhutaankin LAMP-ympäristöstä (Linux, Apache, MySQL, PHP). (Rantala 2005, 255.)

4 PHP JA TIETOTURVA

4.1 PHP:n konfigurointi

Kun ottaa muutaman asian huomioon PHP:n asetuksissa, saa varmistettua hyvän pohjan web-sovelluksen tietoturvalle. Uudemmat PHP:n versiot sisältävät oletuksena paremmin määritellyt asetukset kuin aikaisemmat. Voi kuitenkin olla niin, että palvelimelle on päivitetty uusin PHP versio, mutta php.ini-tiedosto on jäänyt koskematto-maksi. (Dickinson 2005, 2.)

Ensimmäisenä on hyvä tarkistaa virheraportointi. Kun sovellusta kehitetään testipalvelimella, asetus on hyvä olla tietysti asetettuna. Tuotantopalvelimelta on kuitenkin hyvä tarkistaa, että asetus on poissa päältä. Jos virheraportointi on asetettuna, voi hakkeri mahdollisesti käyttää hyödyksi saamiaan virheviestejä. (Mts. 2.)

Yksi vakavimmista tietoturvariskien aiheuttajista on ollut register_globals-direktiivi. Aikaisemmin se oli oletuksena asetettuna php.inissä, mutta siitä aiheutuneiden useiden ongelmien syystä on se on nykyään pois päältä. (Mts. 2.) Tämän direktiivin ollessa asetettuna loppukäyttäjät pystyivät syöttämään muuttujia suoraan sovellukseen. Nämä muuttujat toimivat sovelluksessa globaaleina.

Php.inistä löytyy safe_mode-direktiivi. Se kannattaa asettaa voimaan. Direktiivin avulla rajataan se, että ainoastaan PHP-skriptien omistavalla käyttäjällä on oikeus lukea paikallisia tiedostoja. (Mts. 2.)

Disable_functions-direktiivillä saadaan estettyä ei haluttujen funktioiden suoritus. Näin pystytään estämään vahingollisten PHP-skriptien ajo. (Mts. 2.)

Yleisesti kannattaa pitää PHP:n versio ajan tasalla, koska ohjelmistojen kehittyessä yleensä myös tietoturvaa parannetaan (Hamid 2010).

4.2 Syötteiden tarkistus

Web-sovellukset sisältävät useasti lomakkeita, joissa on syötteitä. Syötteiden kautta käyttäjät pääsevät syöttämään dataa, joka hyvin usein tallennetaan tietokantaan. Näissä tilanteissa käyttäjä saattaa vahingossa syöttää sellaista dataa, josta voi koitua vahingollisia seurauksia. Toisaalta on taas käyttäjiä, joilla on päämääränä löytää aukot, joiden kautta voi saada aikaan mahdollisimman paljon tuhoa. Syötteiden tarkistus onkin yksi tärkeimmistä asioista web-sovelluksen tietoturvaa rakentaessa. (Synder & Southwell 2005, 229.)

PHP:ssä muuttujien heikko tyypittely on oikotie ongelmille. Yleinen tahallinen tai tahaton hyökkäys tuleekin siitä, kun käyttäjä syöttää väärän tyyppistä tai väärän pituista dataa. Käyttäjän syöttämä tieto saattaa myös sisältää erikoismerkkejä tai binäärikoodia. Syötteiden tarkistamisen laiminlyönti saattaa johtaa esimerkiksi kaikkien tietokantataulussa olevien rivien tuhoutumiseen. (Mts. 229.) Syötteet tulee siis aina siivilöidä, ja päästää lävitse ainoastaan hyväksyttyä dataa (Dickinson 2005, 1).

Syötetty data saattaa olla merkkijono tai numero. Tyyppi on tarkistettava sen mukaan, minkälaista dataa halutaan syötteestä tulevan. Tyypin lisäksi pituus kannattaa myös tarkistaa. Liikaa data saattaa kaataa sovelluksen ja pahimmassa tapauksessa koko palvelimen. Kenttään johon pyydetään esimerkiksi sähköpostia, on tärkeää, että annetun datan formaatti on oikea. (Synder & Southwell 2005, 231-239.)

4.3 SQL-injektio

SQL-injektiosta tai SQL-hyökkäyksestä puhutaan silloin, kun käyttäjä pääsee syötteen kautta vaikuttamaan tietokantaan ei halutulla tavalla. Tämä johtuu yleensä aina huonosti tarkistetuista syötteistä. Heikosti puhdistettujen syötteiden ansiosta hakkeri pystyy esimerkiksi hakemaan kaikki taulun tiedot syöttämällä tekstikenttään, joka on suoraan yhteydessä SELECT-lauseeseen, niinkin lyhyen skriptin kuin 1=1. Tällä tavoin hakkeri voi saada mahdollisesti käsiinsä koko taulun rakenteen ja pahimmassa tapauksessa luottamuksellista tietoa. Äkillisiä tuhoisia seurauksia saadaan aikaan, jos sama skripti pystytään syöttää UPDATE- tai DELETE-lauseeseen. Riippuen MySQL-palvelimen asetuksista ja sovelluksessa käytettävästä tietokantarajapinnasta, kenttien

kautta on mahdollista syöttää myös kokonaisia kyselyitä. PHP:n mysql- ja mysqli-laajennuksissa nämä ovat kuitenkin oletuksena estetty. (Mts. 249-253.)

Tällaisia hyökkäyksiä vastaan löytyy onneksi keinot puolustautua. Ensimmäisenä olisi hyvä rajata kyselyissä olevat muuttujat heittomerkein. Seuraavaksi täytyisi tarkistaa annetun arvon tyyppi. PHP:sta löytyy valmiita funktioita tätä toimenpidettä varten. Ehkä tärkein asia, mitä pitäisi muistaa syötteiden tarkistuksessa, on eliminoida kaikki haitalliset merkit lisäämällä kenoviiva niiden eteen. PHP:sta löytyy mysql_real_escape_string funktio, jolla tämä saadaan hoidettua automaattisesti (Mts. 255-256.)

4.4 Istunnon kaappaus

Istunnoilla pystytään pitämään esimerkiksi käyttäjä kirjautuneena johonkin järjestelmään. Normaalisti tätä varten tallennetaan tietoja istuntotaulukkoon, joka PHP:ssa on $_SESSION. Istunnon tunnistetta kuljetetaan HTTP:ssa edes takaisin asiakkaan ja palvelimen välillä, ja on täysin mahdollista, että joku pääsee väliin ja pystyy sieppaamaan sen. Näin hakkeri voi saada käyttäjän tiedot haltuunsa ja pääsee kirjautumaan järjestelmään kuin olisi itse kyseinen käyttäjä. (Mts. 319.)

Istuntojen kaappaus perustuu siis siihen, että hakkeri saa selville istunnon tunnisteen eli sessionid:n. Hakkereilta löytyy keinot tunnisteen kaappaukselle. Tietovirtoja pystytään salakuuntelemaan. Vaikka ne ovat yleensä suuria, niitä voidaan paloitella valituilla sanoilla, jotka voisivat olla esimerkiksi *login* tai *phpsessid*, pienempiin osiin. Sovellukset saattavat kuljettaa sessionid:ta URL:ssa, mikä voi johtaa siihen, että käyttäjä voi itse vahingossa paljastaa oman tunnisteensa. Tämä voi tapahtua esimerkiksi niin, että käyttäjä on kirjautuneena kuvapalveluun ja postittaa kuvalinkin keskustelupalstalle. Se joka menee osoitteeseen, saa haltuunsa käyttäjän istunnon, koska tunniste oli URL:ssa. Hakkeri voi myös pystyttää välityspalvelimen ja yrittää saada käyttäjän huijattua käyttämään sitä, jolloin kaikki tapahtumat välittyvät sen kautta. Tämä tapahtuu usein niin, että käyttäjä saa sähköpostin, joka sisältää linkin. Käyttäjä luulee menevänsä linkin kautta esimerkiksi www.cdon.com:n sivuille, mutta todellisuudessa siirtyikin hakkerin välityspalvelimelle. (Mts. 319-320.)

Istunnon kaappausta vastaan pystytään myös suojautumaan. Tunniste kannattaa sijoittaa evästeisiin ja välttää URL:ssa kuljettamista. Tunnisteelle kannattaa myös asettaa aikaraja, jonka jälkeen se vanhentuu ja ei ole enää käytettävissä. Käyttäjän tilan muuttuessa kannattaa tunniste luoda uudelleen. Tämä tarkoittaa esimerkiksi sitä, että jos käyttäjä kirjautuu palvelusta ulos ja hetken päästä takaisin sisään. Tehokkain keino suojautua kaappauksilta on suojata tietoliikenne SSL-protokollalla. Tätä yhteyskäytäntöä hyödyntämällä saadaan liikkuva data asiakkaan ja palvelimen välillä salatuksi. (Mts. 322-324.)

4.5 Cross site scripting

Cross site scriptingista tai yleisesti XSS:stä puhutaan silloin, kun web-sivustolle saadaan syötettyä haitallista koodia. Tällaisen haittaohjelman ansiosta käyttäjä voi tietämättään lähettää dataa haitalliselle sivustolle, joka kerää esimerkiksi käyttäjien luottokorttitietoja. XSS:ään käytetään usein JavaScriptiä. Puhutaankin, että ainut tapa kokonaan suojautua XSS:ltä on asettaa JavaScript pois päältä ja estää kuvien näyttö selaimessa. Nykyään sivustot vaativat useasti kuitenkin JavaScriptin tukea. (Mts. 263.)

XSS-hyökkäykset voidaan jakaa kahteen kategoriaan: ulkoisiin ja paikallisiin. Ulkoisista hyökkäyksistä puhutaan silloin, kun sähköpostiviestissä tai jollain sivustolla klikataan linkkiä, avataan kuva tai lähetetään lomake, joka sisältää haittaohjelman. Nämä haittaohjelmat yrittävät yleensä kaapata käyttäjän istunnon. Paikalliset hyökkäykset tapahtuvat nimensä mukaan lokaalisti. Hakkeri tai robotti syöttää haitallista koodia esimerkiksi kommenttina foorumille tai yleisesti kenttiin, jotka toimivat käyttäjien syötteinä. Kun tietämätön käyttäjä lataa tämän sivun, tapahtuu jotain odottamatonta ja vahingollista. (Mts. 265-266.)

XSS-hyökkäystekniikoita on erilaisia. Näihin lukeutuvat mm. HTML- ja CSS-hyökkäykset, JavaScript-haittaohjelmat, väärennetyt linkit ja kuvien lähteet, ja huijauslomakkeet. (Mts. 267-271.)

XSS hyökkäyksiin pystytään vastamaan aivan kuten muihinkin tietoturvariskeihin. Peruslähtökohtana on aina tarkistaa käyttäjän antama syöte. HTML-merkkaukset saadaan poistettua koodaamalla kaikki erikoismerkit niitä vastaaviksi entiteeteiksi. Käyt-

täjän syöttämät linkit on syytä tarkistaa etteivät ne sisällä mitään skriptaukseen liittyvää merkkausta. Sovelluksen herkimmät osat voidaan mahdollisesti myös eriyttää ja luoda niille kokonaan oma käyttöliittymä. Käyttäjän toimintaa voidaan yrittää myös ennakoida ja luoda logiikkaa sen mukaan. (Mts. 272-278.)

5 CASE: LASKUTUSOHJELMA TIETOSUUNTA OY:LLE

5.1 Yleiskuvaus

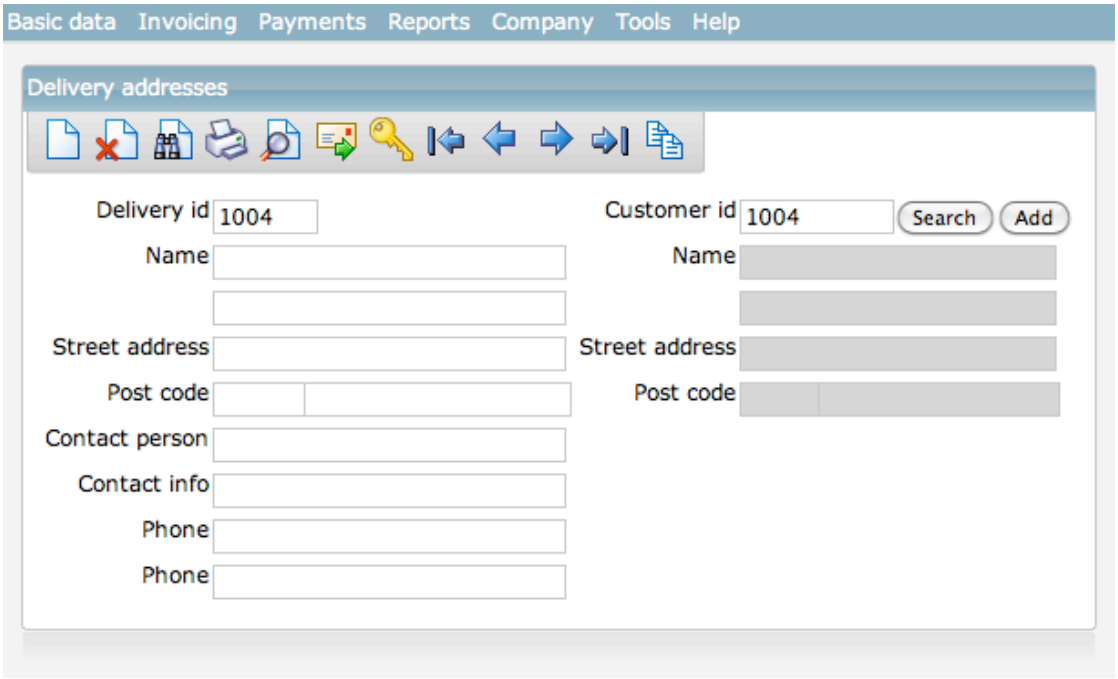
Tietosuunta Oy:n laskutusohjelma on suunniteltu hoitamaan yrityksen laskutusasiat helposti ja vaivattomasti. Tuote soveltuu yrityksien omatoimisiin laskutuksiin tai tilitoimistoille. Ominaisuuksiin kuuluu laskujen luonti, tulostus ja lähetys, asiakas- ja tuoterekisteri, raportit ja myyntireskontra. Web-käyttöliittymän ansiosta laskutukset eivät ole enää tiettyihin tietokonepäätteisiin rajoittuneita vaan töitä voidaan tehdä niin työ- kuin kotikoneelta, jolloin samat tiedot ja lähtötilanteet säilyvät. Web-sovellusta on kehitetty vuodesta 2009 lähtien ja opinnäytetyön myötä ensimmäinen kaupallinen versio alkoi olemaan valmiina.

Sovellusta tullaan tarjoamaan palveluna, mikä tarkoittaa sitä, että sovellus asennetaan Tietosuunnan valitsemalle palvelimelle ja käyttöä varten vaaditaan oma käyttäjätunnus ja salasana. Sovellukseen on tarkoitus kehittää rekisteröitymisosa ja käyttäjätunnuksien hallintaosa. Niiden toteutus jäi kuitenkin tämän opinnäytetyön ulkopuolelle. Käyttäjätunnuksien luonti voidaan aluksi hoitaa manuaalisesti.

Sovelluksen tarjoaminen palveluna tuo muutamia huomioita koskien lisenssiehtoja. Sovellusta suoritetaan Tietosuunnan palvelimella ja siellä olevilla ohjelmilla, jolloin voidaan puhua, että ohjelma on yrityksen sisäisessä omassa käytössä. Asiakkaat käyttävät pelkästään käyttöliittymää ja varsinainen prosessointi ja ohjelmakoodi ovat käyttäjältä ulottumattomissa. Asiakas saa tällöin pelkästään palvelun vastauksen. Varsinaista ohjelmaa ei siis jaeta. Lisenssiehdot tullaan tarkistamaan vielä kertaalleen palvelinohjelmien (Apache, MySQL, PHP) ja avoimen lähdekoodin kirjastojen (JQuery, Smarty) ja komponenttien osalta ennen, kun ohjelma lanseerataan käyttöön.

5.2 Käyttöliittymä

Sovellusta suoritetaan WWW-selaimessa. Ohjelma on pyritty kehittämään niin, että se toimii kaikilla suosituimmilla selaimilla. Käyttöliittymä koostuu kahdesta alueesta: valikosta ja toiminnallisesta alueesta (ks. kuvio 21). Valikon avulla navigoidaan sovelluksen eri osien välillä. Näitä osia ovat erilaiset lomakkeet ja raportit. Toiminnallisen alueen tehtävä on olla alusta lomakkeiden ja raporttien käsittelylle. Tällä alueella tapahtuu siis varsinainen työskentely.



KUVIO 21. Yläosassa nähdään sovelluksen valikko, jonka alapuolella sijaitsee työskentelyalue

5.2.1 Lomake

Lomakkeiden avulla käyttäjä pystyy käsittelemään tietokannassa olevia tietoja. Lomake voi sisältää työkalupalkin, joka koostuu erilaisista toiminnoista riippuen käsittelyssä olevasta lomakkeesta. Näitä toimintoja ovat mm. tietueiden lisäys, poisto, selaus,

kopiointi ja lähetys. Lomakkeessa voi myös olla syötekohtaisia ominaisuuksia, kuten modaalisia ikkunoita, hakuja, laskentaa ja viittauksia muihin lomakkeisiin.

5.2.2 Raportti

Raporttien tehtävä on koostaa tietokannassa olevat tiedot havainnollistaviksi esityksiksi. Raportilla voidaan esimerkiksi hakea kaikki laskut. Raportointiin liittyy modaalinen ikkuna raporttien rajausta varten. Raportit voidaan avata joko navigaation tai lomakkeiden työkalupalkin kautta.

5.3 Käyttäjät

Sovelluksen käyttäjä voi olla yrityksen alaisuudessa työskentelevä työntekijä: esimerkiksi kirjanpitäjä. Tällöin käyttäjällä on oikeudet pelkästään kyseisen yrityksen laskutustietoihin. Sovelluksen käyttäjä voi olla myös tilitoimistossa työskentelevä henkilö, jolloin käyttäjän hallinnassa voi olla useita yrityksiä. Tilitoimiston hoitamat yritykset voidaan jakaa työntekijöiden kesken niin, että käyttäjillä on oikeudet hallita vain heille määrättyjen yritysten asioita.

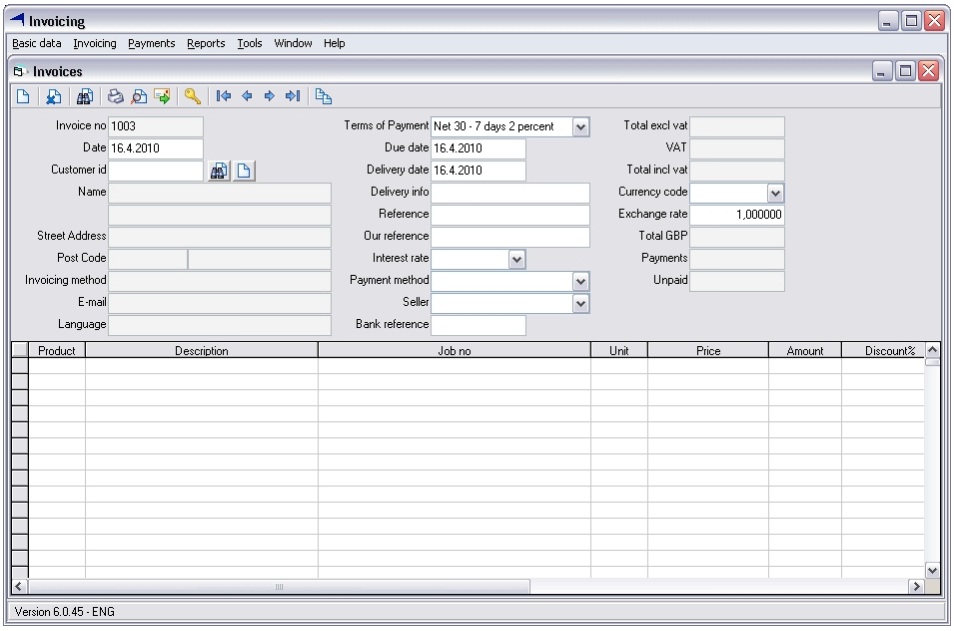
6 TOTEUTUS

6.1 Yleistä

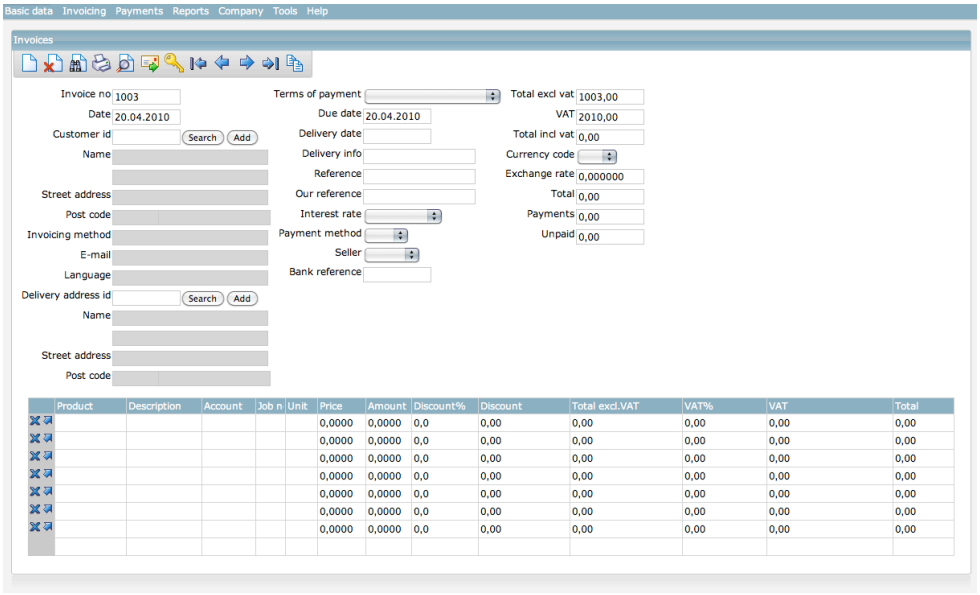
Sovellusta lähdettiin suunnittelemaan ja toteuttamaan niin, että se vastaisi mahdollisimman paljon vastaavaa Visual Basicilla toteutettua Windows-ohjelmaa (ks. kuvio 22 ja 23). Toiminnot ja ulkoasu täytyisivät olla lähellä alkuperäistä. Nykyisen Windows-ohjelmiston keskeinen sisältö on kokonaisten ohjelmistomoduulien toteutus samalla työvälineellä, toteutuksen joustavuus, käyttöliittymän huipputason toteutus, käytettävyys ja käytön helppous. Ohjelmisto on määrittelyihin perustuva, mukautuva ja laajennettavissa. Ohjelmisto toimii sekä ohjelmamoduulien kehitys- ja hallintavälineenä että käyttäjän ohjelmana. Web-sovellus toteutetaan niin, että Windows ohjelman määrittelyt ja logiikka siirretään uuteen ohjelmaan ja kaikki nykyiset Windows sovellukset voidaan siirtää web-sovelluksiksi mahdollisimman automaattisesti.

Sovelluksen kehittämisen kannalta säästettiin aikaa, koska ei tarvinnut suunnitella varsinaisia ominaisuuksia ja toimintoja, vaan keskityttiin siihen, miten alusta voitaisiin

toteuttaa käyttäen eri web-teknologioita. Tällöin myös asiakkaan, jolla on aikaisempaa kokemusta Windows-ohjelmasta, ei tarvitse opetella sovelluksen käyttöä uudestaan vaan voi aloittaa heti ohjelman tehokkaan käytön.



KUVIO 22. Laskun luonti -näkymä Windows-sovelluksessa



KUVIO 23. Laskun luonti selainkäyttöliittymässä

6.2 Ohjelmointi- ja kehitysympäristö

Ohjelmointityökaluna käytettiin aluksi Eclipse PDT:tä ja myöhemmin Aptana Studiota. Molemmat sovellukset ovat ilmaisia ja ladattavissa verkosta. Kumpikin sovellus pohjautuu Eclipseen, joka alkujaan on kehitetty Java-ohjelmoinnin työkaluksi. Sovellukset osoittautuivat hyviksi vaihtoehdoiksi sovelluskehitystä ajatellen, koska niillä pystyttiin hallitsemaan sujuvasti suurta projektikokonaisuutta hyvien käyttöliittymiensä ansiosta.

Syy Aptana Studioon siirtymiseen oli se, että sen koettiin olevan vielä enemmän nimenomaan WWW-sovelluskehitykseen keskittynyt sovellus. Aptana Studio sisältää suoraan mm. CSS-, XML- ja JavaScript-määrittelyt, joita Eclipse PDT:ssä ei ollut. Toki Eclipse PDT:hen voidaan ladata nämä määritykset jälkikäteenkin ohjelman sisäisen plugin-palvelun kautta. Aptanasta löytyi myös muutamia käyttöliittymään liittyviä parannuksia kuten väripipetti, automaattinen kommentointi ja HTML-tagien pikaluonnit.

Sovellusta testattiin ja suoritettiin lokaalisti WAMP-ympäristössä. Testipalvelin laitettiin pystyyn asentamalla Windows-käyttöjärjestelmään EasyPHP-ohjelmistokokonaisuus. EasyPHP sisältää Apachen, PHP-tulkin ja MySQL-tietokantapalvelimen. Näin saatiin kaikki paikallisen web-palvelimen pystyttämiseen tarvittavat ohjelmat samassa paketissa.

6.3 Asiakastekniikat

Asiakaspuolentekniikoiksi valittiin XHTML, CSS ja JQuery. XHTML valittiin siksi, koska sillä saadaan luotua hyvämuotoisia dokumentteja. Varsinaiset XHTML-dokumentit luodaan kuitenkin dynaamisesti palvelintekniikoilla. CSS:n avulla komponenttien muotoilu eriytettiin varsinaisesta XHTML-merkkauksesta. Näin sovelluksen ulkoasu saatiin paremmin hallintaan. JQuerylla hoidettiin käyttöliittymän dynaamiset osat ja Ajax. Kirjasto ei ollut kuitenkaan alusta asti käytössä, vaan aluksi ohjelmointiin puhtaasti JavaScriptilla. JQueryn ansiosta DOM saatiin paremmin hallintaan. Havaittiin myös, että kirjaston käyttöönotto nopeutti kehitystyötä huomattavasti.

6.4 Palvelintekniikat

Sovelluksen palvelinpuolen toteutustekniikoiksi valittiin PHP, MySQL ja Smarty. PHP valittiin siksi, koska se on saatavilla ilmaiseksi ja suosittu web-kehittäjien keskuudessa. Sen avulla oli myös nopeaa ja joustavaa kehittää ohjelmakoodia. PHP:n suuren suosion ansiosta verkosta oli löydettävissä paljon materiaalia, jolloin ratkaisuja eri ongelmatilanteisiin voitiin etsiä myös sitä kautta. Tietokantojen toteutukseen käytettiin MySQL-tietokantapalvelinta. MySQL valittiin sen suosion, vakauden ja suorituskyvyn vuoksi. PHP ja MySQL toimivat myös saumattomasti yhdessä, koska PHP sisältää valmiit funktiot tietokannan käsittelyä varten. Sovelluksen PHP-osaan otettiin lisäksi mukaan Smarty-kirjasto. Näin saatiin valmis mallinejärjestelmä, jonka ansiosta web-dokumenttien dynaaminen luonti nopeutui.

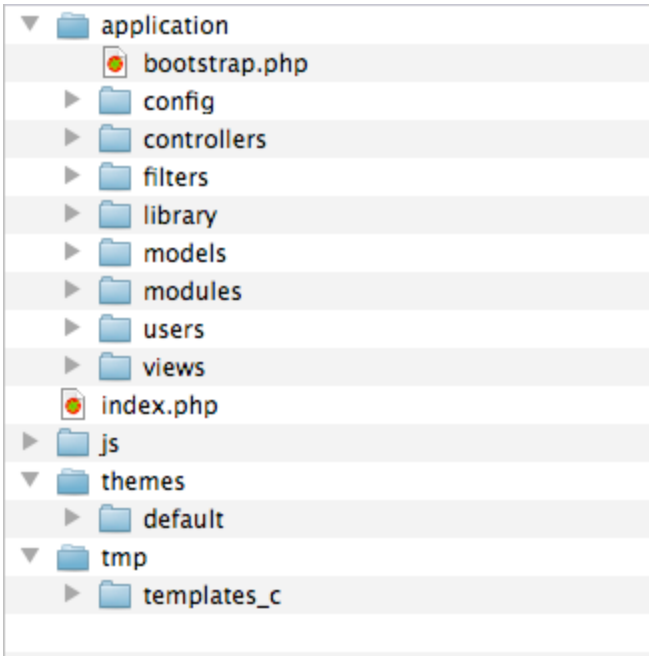
6.5 Arkkitehtuuri

Nykyinen Windows-ohjelmisto toimii määrittelyjen ohjaamana. Web-sovellus rakentuu näiden samojen määrittelyjen perusteella. Määrittelyt ovat sijoitettu erillisiin tiedostoihin. (ks. kuvio 24). Perusmalli on abstrakti kuvaus tietojen käsittelystä ja varastoinnista. Näiden tiedostojen pohjalta ohjelma luo lomakkeet tietojenkäsittelyä ja raportit tietojen koostavaa havainnollistamista varten näytölle ajonaikana automaattisesti. Uusia ominaisuuksia saadaan tarvittaessa luotua muuttamalla määrittelyitä. Sovelluksen ydin PHP-osa sisältää tarvittavat funktiot näiden tiedostojen käsittelyä varten.

```
 3 [Form]
 4 Caption=Delivery addresses
 5 RecordSource=SELECT * FROM delivery_addresses
 6 Relation RecordSource 1=customers
 7 RelationForm=Customers
 8 UpdatableKeyField=1
 9
10 [Report]
11 Report=Customers in alphabetical order
12 Select=
13
14 [Check Delete]
15
16 [Main Table Columns]
17 delivery_address_id ,Delivery id, 8 ,I  ,1
18 name            ,Name       ,28/60  ,K  ,1
19 name_2          ,       ,28/60  ,K  ,1
```

KUVIO 24. Katkelma lomakkeen määrittelytiedostosta

PHP-osan rakenne noudattaa MVC-arkkitehtuurimallia. MVC:n avulla saadaan sovelluksen logiikka paremmin hallintaan. Ohjelmakoodista saadaan myös paremmin ylläpidettävää ja uudelleen käytettävää. Myös käyttöliittymä pystytään erottamaan omaksi osakseen. Sovellus noudattaa tiettyä kansiorakennetta, jonka ansiosta saadaan pidettyä ohjelmakoodit hyvin organisoituina (ks. kuvio 25).



KUVIO 25. Sovelluksen kansiorakenne

Application-kansiossa sijaitsee kaikki sovelluksen palvelinpuolen toimintojen kannalta tärkeät tiedostot. Laskutusohjelman kansio, joka sisältää määrittelytiedostot, sijaitsee modules-kansion alapuolella. Uusi sovellus saadaan aina käyttöön lisäämällä vain uusi kansio eli moduuli modules-kansion alle. Määrittelytiedostojen ollessa vahvassa asemassa ohjelmien toimintojen kannalta, PHP-osa toimii vain alustana ohjelmien toteutuksille. MVC-arkkitehtuurin ansiosta ytimeen voidaan ohjelmoida helposti ohjelmakohtaisia ratkaisuja. Uudelleen käytettävät luokat ovat eriytetty omaan kansioon omaksi kirjastokseen (library). Näihin luokkiin kuuluvat mm. tietokantarajapinta, istunnot, suunnittelumallit. Malleihin (models) on sijoitettu määrittelytiedostojen käsittelyluokat. Varsinainen sovelluksen käyttöliittymä luodaan näkymien (views) avulla. Näkymät ovat toteutettu Smarty:n mallinejärjestelmän

Näkymät ovat toteutettu Smarty:n mallinejärjestelmän avulla. Ohjaimien (controllers) avulla luodaan sovelluksen toimintavuo. Käyttäjä ja käyttäjäyritys kohtaiset määritykset sijaitsevat users-kansiossa. Datan validointiin ja filtteröintiin liittyvät osat ovat omina luokkina filters-kansiossa. Yleiset määritykset kuten esimerkiksi tietokantayhteyteen vaadittavat asetukset sijaitsevat config-kansiossa. Application-kansio on suojattu .htaccess-tiedostolla. Näin estetään ulkopuolisten pääsy kansioon.

Application-kansion kanssa samalla tasolla sijaitsevat muut kansiot sisältävät asiakaspään määritykset. JS-kansiossa sijaitsee kaikki JavaScript-tiedostot. Themes-kansiossa ovat sovelluksen ulkoasuun liittyvät CSS-tiedostot ja kuvat. Tmp-kansio sisältää Smarty-mallineista käännetyt php-skriptit.

Sovellus käynnistetään index.php-tiedostosta. Tiedostossa luodaan FrontContoller-olio, joka vastaanottaa kaikki käyttäjältä saadut syötteet. Index.php:ssa luodaan myös datan tarkistus olio, jonka avulla putsataan kaikki syötteistä saatu data. Tiedostoon on myös sisällytetty tärkeä bootstrap.php, joka lataa automaattisesti kaikki tarvittavat luokat ja asetukset käyttöön.

Luokat on pyritty luomaan niin, että ne ovat mahdollisimman paljon toisistaan riippumattomia. Tällä tavoin niitä voidaan hyödyntää myös muissa projekteissa. Raportointijärjestelmä on mm. yksi tämmöinen täysin eriytetty komponentti.

Sovelluksen alustaksi ei otettu käyttöön valmista sovelluskehystä, koska konseptia oli jo tehty ennen kehitystyön alkua. Arkkitehtuurimalli mahdollistaa kuitenkin sovelluskehyksen käyttöönoton tulevaisuudessa, jos asiasta niin päätetään.

6.6 Toiminnot

Sovelluksessa on pyritty hakemaan mahdollisimman automatisoituja toimintoja. Päämäärä oli luoda käyttöliittymästä mahdollisimman käyttäjäystävällinen. Työskentely tulisi olla jouhevaa. Näiden tuloksien aikaansaamiseksi sovellukselta vaadittiin Ajax:n käyttöönottoa. Suurin osa ohjelman asiakaspuolen toiminnoista on ratkaistu kyseistä tekniikkaa hyödyntäen. Navigointi eri lomakkeiden ja raporttien välillä on toteutettu perinteisellä synkronisella tavalla.

6.6.1 Navigaatio

Navigaatio luodaan lomakkeen tapaan määrittelytiedoston perusteella (ks. kuvio 26). Tiedosto parseroidaan mallin avulla ja esitetään näkymässä listana. Lista on muotoiltu CSS:llä pudotusvalikoksi. Valikon linkit ovat määrittelytiedostossa omina osioina. Niitä voidaan pitää eräänlaisina skripteinä. Ne sisältävät tietoa esimerkiksi siitä, mitä toimintoa ja lomaketta kutsutaan.

```
1 [Invoicing]
2 Basic data
3 Company
4     Function=Edit
5     FormLayout=Company.fld
6 Exchange rates
7     Function=Edit
8     FormLayout=Currencies.fld
9 -
10 Customers
11     Function=Edit
12     FormLayout=Customers.fld
13 Delivery addresses
14     Function=Edit
15     FormLayout=Delivery_addresses.fld
16 -
17 Products
18     Function=Edit
19     FormLayout=Products.fld
20 -
21 Close
22     END
23 *
```

KUVIO 26. Katkelma valikon määrittelytiedostosta ja näkymä selaimella

Sovelluksen URL:t ovat muotoa index.php?page={ohjain}?action={metodi}. Kun valikosta valitaan jokin lomake tai raportti, URL:n loppuun lisätään yksi parametri lisää. URL on tämän jälkeen muotoa index.php?page={ohjain}&action={metodi}&s={skripti}. FrontController-luokka käsittelee URL:n parametrit. Parametrien perusteella valitaan ohjain ja suoritetaan haluttu toiminto.

6.6.2 Tietojenkäsittely

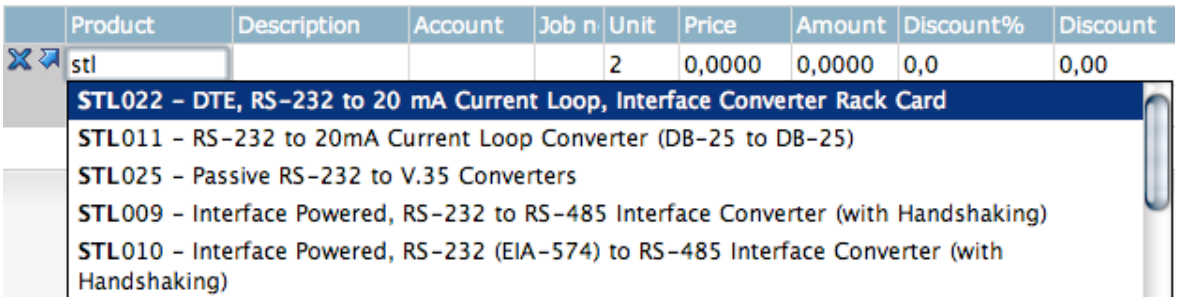
Lomakkeiden avulla päästään vaikuttamaan tietokannassa oleviin tietoihin. Tällöin voidaan puhua kolmikerrosmallista. Lomakkeet sisältävät syötteitä, joidenka kautta muokataan ja tallennetaan tietoa. Syötteet ovat pääasiassa tekstikenttiä tai valintalistoja. Kentän arvon muuttuessa laukeaa onchange-tapahtuma, joka kutsuu syötteelle määrättyä JavaScript-funktiota. Funktio lähettää kentän arvon, nimen ja tyypin palvelimelle, joka tallentaa arvon tietokantaan. Palvelin palauttaa arvon kentän tyypin mukaan muotoiltuna. Tiedot tallennetaan siis aina syöte kerrallaan riippuen siitä, onko kentän arvo muuttunut. Tällä tavoin saadaan tietojen tallennukset automaattisiksi, eikä erillistä tallenna-painiketta tarvita.

Laskennoissa vaaditaan selainta lähettämään useampia syötetietueita. Laskentoihin liittyvien kenttien arvojen muuttuessa kutsutaan tiettyä JavaScript funktiota, joka lähettää muuttuneen kentän sekä muiden laskennallisten kenttien tiedot palvelimelle. Laskennallisten kenttien lähetyksessä käytetään JSONia (ks. kuvio 27). Palvelin hoitaa varsinaiset laskennat ja tallentaa tulokset tietokantaan. Palvelin palauttaa muuttuneen kentän sekä tuloksien arvot kenttien tyyppien mukaan muotoiltuna. Tulokset esitetään niille määrätyissä kentissä.

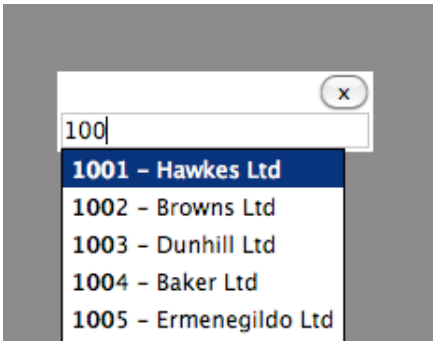
```
fields_values  {"price_excl_vat":"22","vat_percent":"22,00"}
```

KUVIO 27. Laskennalliset kentät ja niiden arvot JSON-muodossa

Tekstikenttä voi toimia myös tietojen hakemiseen muista lomaketietueista. Haku on joko kentässä itsessään (ks. kuvio 28) tai modaalisena ikkunana (ks. kuvio 29). Syötteen arvon muuttuessa kutsutaan JavaScript funktiota, joka lähettää hakusanat palvelimelle. Palvelin vastaa palauttamalla hakutulokset. Tuloksista valitaan sopivin vaihtoehto, minkä jälkeen kutsutaan toista JavaScrip-funktiota, joka lähettää hakutuloksen palvelimelle. Palvelin tallentaa arvon ja palauttaa lomaketietueen kenttien tyyppien mukaan muotoiltuna. Kuvassa 28 nähdään, kuinka laskuun voidaan hakea rivi tuotetaulusta.

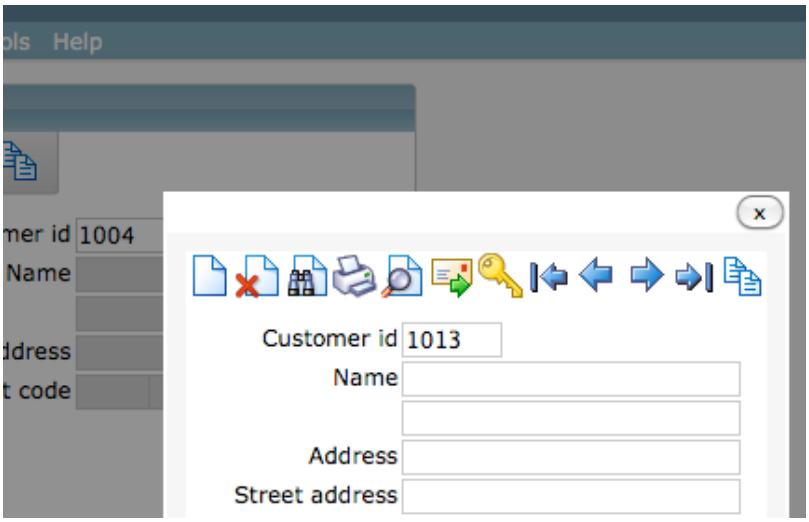


KUVIO 28. Syöte voi toimia hakukenttänä



KUVIO 29. Hakukenttä modaalisessa ikkunassa

Lomakkeissa voidaan viitata myös muihin lomakkeisiin. Näitä voidaan kutsua ulkoisiksi lomakkeiksi. Ulkoiset lomakkeet avataan erillisten painikkeiden kautta. Kun painiketta painetaan, avautuu ulkoinen lomake modaaliseen ikkunaan (ks. kuvio 30). Painike sisältää samalla jonkin erityistoiminnon, kuten esimerkiksi automaattisen uuden asiakastietueen luomisen. Ikkunassa olevaa lomaketta voidaan käsitellä täysin samalla tavalla kuin se olisi avattu normaalisti navigaation kautta.



KUVIO 30. Asiakastietojen käsittelyä erillisessä ikkunassa

Työkalupalkki sisältää myös dynaamisia osia. Kun käyttäjä painaa työpalkista kuvaketta, kutsutaan JavaSript-funktiota, joka lähettää tietyt toiminnalle ominaiset parametrit palvelimelle. Palvelin vastaa esimerkiksi palauttamalla uuden tietueen.

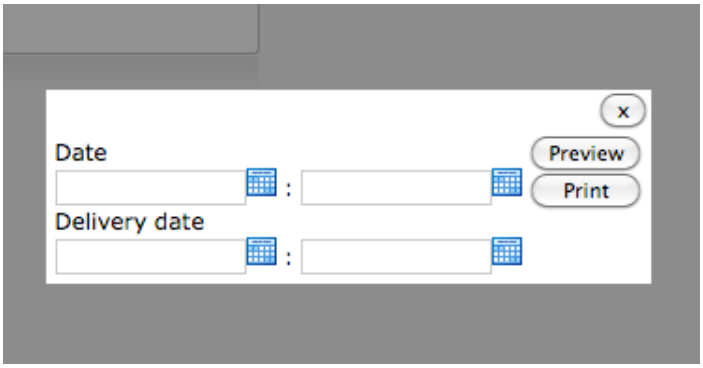
6.6.3 Raportointi

Raportit muodostetaan samalla tavalla kuin lomakkeet ja valikko eli ennalta luotujen määrittelytiedostojen parseroinnilla (ks. kuvio 31). Määrittelytiedostot sisältävät raporttien rakentamisen kannalta tärkeitä määrityksiä. Käsittelylle on luotu oma mallinsa. Malli käsittelee tiedoston ja hakee määrittelyjen perusteella tiedot tietokannasta.

```
1  [report layout]
2  NAME=Customers
3  TYPE=table
4  [sql query]
5  SELECT=customer_id,name,street_address,post_code
6  FROM=customers
7  WHERE
8  ORDER
9  [column headers]
10 LIST=Customer,Name,Street address, Post code
11 [sum]
12 COLUMN
```

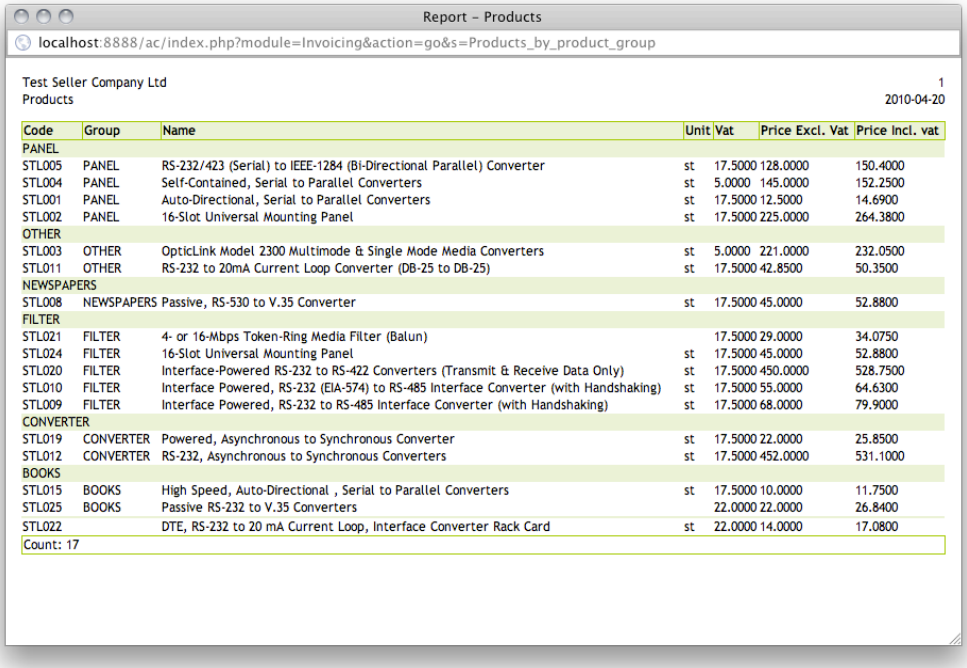
KUVIO 31. Katkelma raportin määrittelytiedostosta

Raportti voidaan esittää joko (esikatselu) HTML:nä tai (tulostus) PDF:nä. Raporttiin haettavia tietoja voidaan rajata erillisen modaalisen ikkunan kautta (ks. kuvio 32). Ikkunassa olevien kenttien avulla voidaan rajata raporttia tuomaan esimerkiksi yrityksen laskut tietyltä aikaväliltä. Kentät määräytyvät valikon skriptin mukaan. Raportteja on helppo luoda lisää, koska tarvitaan luoda vain uusi määrittelytiedosto ja lisätä uusi skripti valikon määrittelytiedostoon.



KUVIO 32. Raportin rajaus päivämäärän ja toimituspäivämäärän mukaan

Raportointia ei saatu opinnäytetyön aikana täysin valmiiksi. Aluksi oli tarkoitus luoda Crystal Reportsin raporttimäärittelyjen parseroija. Crystal Reportsilla luodut raporttimäärittelyt muutettiin XML-muotoon ja parseroitiin PHP:lla. Määrittelyt sisälsivät kuitenkin niin paljon eri asioita, joita tuli ottaa huomioon, joten tekniikka osoittautui liian haastavaksi. PHP:lla kokeiltiin myös ladata muistiin Crystal Reportsin dll-tiedosto ja sitä kautta saada Windows-ohjelman raportit suoraan käyttöön. Lopulta päätettiin kuitenkin luoda täysin oma raportointijärjestelmä. Pohja saatiin hyvälle mallille ja sitä on helppo kehittää jatkossa. Kuviossa 33 on nähtävissä esimerkki raportista.



KUVIO 33. Esimerkki tuotelistaraportista tuoteryhmän mukaan ryhmiteltynä

6.6.4 Kirjautuminen

Päästäkseen käyttämään sovellusta, käyttäjää vaaditaan kirjautumaan palveluun. Tähän tarvitaan käyttäjätunnus ja salasana. Kun asiakas tilaa palvelun, hänelle lähetetään sähköpostitse tarvittavat tiedot kirjautumista varten.

Kun käyttäjä saapuu sovelluksen domain-osoitteeseen, hänen eteensä avautuu kirjautumissivu. Sivu sisältää kentät käyttäjätunnuksen ja salasanan syöttämistä varten. Onnistunut kirjautuminen johtaa istunnon aloitukseen ja sovelluksen käyttöönottoon. Tällöin käyttäjällä on täydet oikeudet sovelluksen toimintoihin. Käyttäjä voi missä vaiheessa tahansa kirjautua ulos sovelluksesta, jolloin istunto lakkautetaan. Käyttäjä kirjataan ulos myös silloin, kun sovellus on ollut käyttämättä tietyn ajan yli.

6.7 Tietokanta

Sovelluksen tietovarastona toimii MySQL-tietokanta. Sovellus ei rajoitu pelkästään yhteen tietokantaan, vaan niitä on useita. Tietokannat hajautetaan ohjelma-

käyttäjäyritys-asiakasyritys kohtaisesti. Tietokantasuunnittelua ei tarvinnut toteuttaa, koska voitiin käyttää suoraan Windows-ohjelmalle luotua MySQL-tietokantamallia.

Yleisiä asetuksia ja käyttäjiä varten luotiin oma tietokanta. Tätä tietokantaa on helppo tulevaisuudessa laajentaa, koska se ei ole riippuvainen mistään tietystä ohjelmasta. Varsinaisista sovellustietokannoista tullaan huolehtimaan erinäisin PHP:llä luotujen automaattisten huolto- ja päivitysskriptien avulla.

6.8 Versionhallinta

Versionhallinnassa ei käytetty erillistä versionhallintajärjestelmää kuten Subversion. Sovelluksesta otettiin tasaisin väliajoin kopioita, jotka nimettiin sovelluksen nimen ja kopiointipäivämäärän mukaan. Aina kun edessä oli suurempi muutos, otettiin sovelluksesta kopio. Kopioidut kansiot siirrettiin niille ennalta määrättyyn paikkaan. Hakemistosta oli tarvittaessa helppo hakea ohjelmakoodin edellinen versio, mikäli ajauduttiin vakaviin ongelmiin koodia kehittäessä.

6.9 Testaus

Sovellusta testattiin koko ajan samalla, kun sitä kehitettiin eteenpäin. Sovelluksesta pyrittiin heti alusta asti karsimaan pahimmat virheet eri komponenteissa, moduuleissa sekä rajapinnoissa. Kehityksen loppuvaiheessa tullaan tekemään vielä erillinen alpha- ja betatestaus pienellä käyttäjäryhmällä, joka tulee luultavammin koostumaan Tietosuunnan nykyisistä asiakkaista. Tämän jälkeen tehdään vielä hyväksymistestaus, jonka jälkeen sovellus on valmis julkaistavaksi.

6.10 Tietoturva

Php:n ini-tiedostoon pystyttiin tekemään testipalvelimella tarvittavat muutokset tietoturvan parantamiseksi. Voi olla kuitenkin, että varsinaisella tuotantopalvelimelle tämä ei ole kuitenkaan mahdollista. Asia voidaan ratkaista asettamalla asetukset ajon aikana. Se ei kuitenkaan ole yhtä tehokas tapa, kuin asetukisen määrittäminen php.ini:ssä.

Sovelluksessa siirrettään asiakkaan ja palvelimen välillä luotettavaa dataa, mikä johtaa yhteyden suojaamiseen SSL-protokollaa käyttäen. On myös tärkeää ettei syötteiden kautta päästä antamaan vahingollista dataa, joka voisi pahimmassa tapauksessa aiheut-

taa mittavia vahinkoja tietokantaan. Syötteiden tarkistusta ja puhdistusta varten on luotu oma kerros. Syötteet puhdistetaan aina ennen kuin mitään dataa päästetään tietokantaan.

6.11 Lopputuotteen sijoitus

Sovelluksen valmistuttua se siirretään tuotantopalvelimelle, jossa se otetaan varsinaisesti käyttöön ja aloitetaan tarjoamaan palveluna asiakkaille. Opinnäytetyön aikana ei vielä ollut täysin varmaa sijoitetaanko se Tietosuunnan omalle fyysiselle, vaiko ulkopuolisen palvelutarjoajan palvelimelle. Omassa palvelimessa on se hyöty, että PHP ja MySQL ovat täysin omassa hallinnassa. Ulkopuolisella palvelutarjoajalla saattaa olla joitain rajoituksia.

7 YHTEENVETO

Opinnäytetyössä käytiin lävitse web-sovelluskehityksessä käytettäviä tekniikoita ja teknologioita. On kuitenkin syytä tietää etteivät ne rajoitu pelkästään mainittuihin, vaan tarjolla on myös useita muita. Ohjelmoijan on valittava suuresta teknologioiden kirjosta sopivimmat. Sovelluksia pystytään toteuttamaan monella eri tavalla. Yksi tärkeimmistä valintaan vaikuttavista tekijöistä on varmasti tehokkuus. Esimerkiksi JQuerylla saadaan lyhennettyä JavaScriptin kirjoittamisen määrää tai sovelluskehyksellä päästään keskittymään heti itse sovelluksen toteuttamiseen. Syy miksi työssä käytiin lävitse juuri kyseiset teknologiat oli se, että kaikkia yhdistää avoimen lähdekoodin lisenssit ja suuri suosio web-kehittäjien keskuudessa. Nämä olivat myös ne syyt, miksi suurin osa niistä valittiin laskutusohjelman toteutustekniikoiksi. Näin säästettiin kustannuksissa ja apua oli tarvittaessa saatavissa runsain määrin myös verkosta.

Webistä alkaa nykyään löytymään yhä enemmän työpöytätyyppisiä sovelluksia. Ihmiset alkavat tottumaan siihen ja samalla vaatimaan web-sovelluksilta enemmän. Näin myös web-kehittäjiltä tullaan vaatimaan entistä enemmän. Laajempaa sovellusta kehittäessä nousee esiin kysymykset, miten ohjelmakoodi ja ylipäätänsä koko sovellus saadaan pidettyä hallinnassa. Tämä vaatii pakosti ohjelmakoodin pilkkomista järkeviin osiin. Laskutusohjelmaa kehittäessä huomattiin, että jotain on tehtävä, jotta ohjelma saadaan paremmin ylläpidettäväksi ja uusien komponenttien lisääminen helpommaksi. Tämä vaati käyttöliittymän ja sovelluslogiikan erottamista toisistaan. So-

velluksessa aloitettiin toteuttaa MVC-arkkitehtuurimallia. Aluksi oli hieman hankalaa löytää ja kehittää mallit määrittelytiedostoista. Ne saatiin kuitenkin lopulta eri mutkien kautta toteutetuiksi. Toimintavuon ja käyttöliittymän kehittäminen olikin sitten helpompaa.

PHP:n ollessa tilaton muutamien Windows-ohjelman ominaisuuksien toteuttamisen ymmärtäminen oli haastavaa. Windows-ohjelmassa muuttujat pysyvät järjestelmän muistissa, kun taas PHP:lla joudutaan aina rakentamaan kaikki uudestaan. Ajaxin avulla tähän ongelmaan saatiin apua. Staattisten web-sivustojen muuttuessa yhä enemmän dynaamisiksi alkaa myös Ajaxin käyttöä näkymään yhä enemmän. Työpöytätyyppisiä web-sovelluksia kehittäessä Ajax tuntuu olevan ehdoton.

Web-sovelluksia kehittäessä on tärkeää muistaa tietoturvariskit, koska sovellukset ovat käytännössä kaikkien saatavissa ja näin murrettavissa. Oikeastaan minkään tyyppistä pienenpääkään sovellusta, joka sisältää syötteitä, ei tulisi vähätellä. Syötteet ovat aina niitä, joiden kautta käyttäjä pääsee periaatteessa vaikuttamaan sovelluksen sisälle. Tietoturva-asioita selvittäessä huomattiin, miten jo muutaman asian huomioimisella saadaan sovelluksen perustietoturva paremmaksi. Laskutusohjelmassa liikkuu osakseen hyvinkin luottamuksellista dataa, joten sovellukselta vaadittiin myös järeämpiä tekniikoita, kuten SSL:n käyttöönottoa.

Sovellusta kehittäessä olisi syytä miettiä myös lisensointiasioita. Tämä ratkeaa hyvin paljon sillä, miten lopputuotetta tullaan tarjoamaan. Avoimen lähdekoodin kirjastoja ja komponentteja käytettäessä täytyy tarkistaa, minkä lisenssin alla ne ovat julkaistu. Jos oma valmis sovellus julkaistaan pakettina, voi olla, että sen lähdekoodi on myös jaettava, jolloin käyttäjät saavat vapaasti muokata ja kehittää ohjelmaa omien halujensa mukaan. Laskutusohjelmaa tullaan tarjoamaan palveluna. Sovellusta suoritetaan Tietosuunnan omalla palvelimella, jolloin voidaan puhua, että se on yrityksen omassa käytössä. Lisensointiasiat eivät ole mitään yksinkertaisia juttuja. Niitä pitää oikeasti miettiä, jos sovelluksia aiotaan julkaista kaupallisessa mielessä. Tarvittaessa niihin joutuu hakea jopa juridista apua.

WWW-sovelluskehityksessä on paljon asioita, joita tulisi ottaa huomioon. Välillä tuntuu, että on mahdotonta hallita kaikkea kiitettävästi. Toisaalta voi myös ajatella, onko se kovin järkevääkään, koska sovelluksen kehitys voidaan jakaa eri osa-alueiden osaajille. Näin voi siis miettiä, haluaako paneutua esimerkiksi pelkästään käyttöliittymien suunnitteluun.

Opinnäytetyön ansiosta saatiin yritykselle rikasta tietoa koskien eri web-teknologioita ja tekniikoita. Laskutusohjelma saatiin kehitettyä siihen vaiheeseen, että se voidaan kohta valjastaa käyttöön. Raportointiosa jäi vielä hieman kesken, joten se vaatii jatkokehitystä opinnäytetyön jälkeen. Sovellukseen jäi myös muutamia muita osia, jotka tullaan toteuttamaan myöhemmin.

Opinnäytetyön aihe tuli sinällään helposti, koska se oli oikeastaan jatkumoa Tietosuunta Oy:ssa tehdylle työharjoittelulle. Aihe oli aluksi vain vaikea lyödä lukkoon, koska pääsin itse vaikuttamaan siihen. WWW-sovelluskehitys oli aiheena mielenkiintoinen. Tietoperustaa kirjoittaessa oma ammatillinen osaaminen syventyi. Mitä enemmän asioita oppi, sitä enemmän halusi oppia lisää. Tietoa oli saatavissa erilaisista kirjallisista teoksista ja suoraan verkosta. Joistain asioista oli tietoa entuudestaan, joten välillä oli vaikeaa olla kirjoittamatta omia mielipiteitä. Tämä myös osakseen hidastutti prosessin kulkua, koska myös perusasioihin oli pystyttävä viittaamaan. Kaiken kaikkiaan prosessi eteni kuitenkin hyvällä vauhdilla.

Laskutusohjelman kehittäminen oli myös antoisaa. Ohjelmistotuotanto oli itselle entuudestaan hieman vierasta. Aikaisempi kokemus koostui lähinnä web-sivustojen suunnittelusta ja toteutuksesta. Näin sain siis hyvää kokemusta myös siltä puolelta. Henkilökohtaisesti olen tyytyväinen lopputulokseen. Sovelluksen kehittäminen olisi voinut tosin olla hieman nopeampaa. Välillä tuntui, että meni liikaa aikaa samojen ongelmien kanssa painimiseen. Nämä ongelmat olisi mahdollisesti pystytty kitkemään hieman paremmalla suunnittelulla.

LÄHTEET

Ajax Suggest. 2010. Viitattu 7.4.2010. http://www.w3schools.com/ajax/ajax_example_suggest.asp

Brandon, D. 2008. Software Engineering for Modern Web Applications: Methodologies and Techonologies. Hershey (PA): Information Science Reference.

Bos, B. 2010. What are style sheets?. Viitattu 20.2.2010. http://www.w3.org/Style/

CodeIgniter Introduction. 2007. Viitattu 19.3.2010. http://www.phpframeworks.com/php-frameworks/index.php?id=9

Crash Course. 2008. Viitattu 3.3.2010. http://www.smarty.net/crashcourse.php

CSS How to. 2010. Viitattu 7.4.2010. http://www.w3schools.com/css/css_howto.asp

Davis, M. & Phillips, J. 2007. Learning PHP & MySQL. 2 p. Sebastol (CA): O'Reilly.

Dickinson, P. 2005. Top 7 PHP Security Blunders. Viitattu 10.3.2010. http://articles.sitepoint.com/article/php-security-blunders/2

Gilmore, W. 2008. Beginning PHP and MySQL: From Novice To Professional. 3 p. Berkeley (CA): Apress.

Golding, D. 2008. Beginning CakePHP: From Novice to Professional. Berkeley (CA): Apress.

Hamid, A. 2010. PHP MySQL Web Development Security Tips – 14 tips you should know when developing with PHP and MySQL. Viitattu. 12.3.2010. http://codingrecipes.com/php-mysql-web-development-security-tips-14-tips-you-should-know-when-developing-with-php-and-mysql?t=939

JQuery. 2010. Viitattu 30.3.2010. http://jquery.com/

JSON. n. d. Viitattu 9.4.2010. http://www.json.org/

Juslin, J. 1999. Oliopohjaiset kehykset. Viitattu 19.3.2010. http://zds.iki.fi/zds/tekstit/kehykset.shtml

Laurent, A. 2004. Understanding Open Source And Software Licencing. Sebastol (CA): O'Reilly.

Lecky-Thompson, E., Nowicki, S. & Myer, T. 2009. Professional PHP6. Indianapolis (IN): Wiley.

Linldey, C. 2009. JQuery Cookbook: Solutions & Examples for JQuery Developers. Sebastol (CA): O'Reilly.

Korpela, J. & Linjama, T. 2005. Web-suunnittelu. 2 p. Jyväskylä: Docendo.

McArthur, K. 2008. Pro PHP: Patterns, Frameworks, Testing and More. Berkeley (CA): Apress.

O'Brien, D. 2007. PHP frameworks, Part 1: Getting started with three popular frameworks. Viitattu 19.3.2010. http://www.ibm.com/developerworks/opensource/library/os-php-fwk1/

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

PHP Frameworks. 2007. Viitattu 19.3.2010. http://www.phpframeworks.com/

Propel. 2009. Viitattu 7.4.2010. http://propel.phpdb.org/trac/

Rantala, A. 2005. Web-ohjelmointi. Jyväskylä: Docendo.

Suehring, S. Converse, T & Park, J. 2009. PHP6 and MySQL Bible. Indianapolis (IN): Wiley.

Symfony Introduction. 2007. Viitattu 19.3.2010. http://www.phpframeworks.com/php-frameworks/index.php?id=3

Three ways to insert CSS. 2010. Viitattu 22.2.2010. http://www.w3schools.com/css/css_howto.asp

Top 10 Reasons Why You Should Use a PHP Framework. 2009. Viitattu: 7.4.2010. http://www.phpandstuff.com/articles/top-10-reasons-why-you-should-use-a-php-framework

Varjonen, Ville. 2000. URI, URN ja URL Dokumenttien nimeäminen ja paikantaminen. Oulun yliopiston kirjasto. Viitattu 16.2.2010. http://herkules.oulu.fi/vili/viittaus/uri.html

Welcome to CodeIgniter. 2010. Viitattu 19.3.2010. http://codeigniter.com/

What is HTML 4. 2002. Viitattu 20.2.2010. http://www.w3.org/TR/xhtml1/#html4

What is MySQL. 2010. Viitattu 3.3.2010. http://dev.mysql.com/doc/refman/5.5/en/what-is-mysql.html

What is Smarty. 2008. Viitattu 3.3.2010. http://www.smarty.net/manual/en/what.is.smarty.php

What is URN. 2004. Viitattu 16.2.2010. http://searchsoa.techtarget.com/sDefinition/0,,sid26_gci214164,00.html)

What is XHTML. 2002. Viitattu 20.2.2010. http://www.w3.org/TR/xhtml1/#xhtml