Barez Azarqaderi

Web-pohjainen laskutusjärjestelmä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

10.11.2017

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Barez Azarqaderi Web-pohjainen laskutusjärjestelmä 30 sivua 10.11.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistosuunnittelu
Ohjaaja(t)	Lehtori Vesa Ollikainen

Tämän insinöörityön tarkoitus oli suunnitella ja toteuttaa laskutusjärjestelmä web-pohjaisena. Tarkoituksena oli toteuttaa helppokäyttöinen järjestelmä, joka auttaa yrityksiä laskujen hallinnassa. Toteutuksessa käytettiin nykyaikaisia tekniikoita, kuten PHP:tä, MySQL:ää, HTML5:tä ja CSS3:aa sekä jQueryä ja JavaScriptiä.

Projektin ideana oli toteuttaa nettisivut palvelun käyttäjälle, jossa käyttäjä voisi luoda, muokata sekä poistaa laskuja. Käyttäjällä olisi myös mahdollisuus luoda vakioasiakkaita sekä hyvin myyntiin meneviä tuotteita, joita voisi lisätä helposti laskuun sen luomisessa.

Työn toteutus suoritettiin WampServerilla, joka on helppokäyttöinen palvelinympäristö, jonka avulla voidaan asentaa PHP-, MySQL- ja Apache-palvelimet Windows-ympäristössä.

Avainsanat	laskutus, PHP, MySQL, HTML, CSS, jQuery

Author(s) Title Number of Pages Date	Barez Azarqaderi Web-Based Invoice System 30 pages 10 November 2017
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Vesa Ollikainen, Principal Lecturer

The purpose of this thesis was to design and develop a web-based invoice system. It was meant to help users to easily manage invoices and control them over the internet without the need to install it on a personal computer which is usually done.

The purpose was to create a user friendly website using the latest technologies such as PHP, MySQL, HTML5 and CSS3 where the user can easily create a new invoice, browse invoices, edit and delete invoices. On this website, the user can also create and manage regular customers and regularly sold products, in such a way that they can be easily added into invoices when creating one.

The website was developed on WampServer, which installs all the required servers like PHP, Apache and MySQL. It is fully functioning web server that can be set up on one's computer.

Keywords	invoice, PHP, MySQL, HTML, CSS, jQuery

Sisällys

Lyhenteet

Apache	Palvelinohjelma. Avoimeen lähdekoodiin perustuva HTTP-palvelinohjelma, jolla siirretään tiedostoja selaimen ja WWW-sivujen välillä.
PHP	Hypertext Preprocessor. Ohjelmointikieli, jota käytetään dynaamisten web-sivujen luomiseen Web-palvelinympäristöissä laajan luokkakirjastonsa ansiosta.
PDO	PHP Data Objects. Mahdollistaa monien tietokantojen käyttämisen PHP:ssä.
MySQL	Relaatiotietokantaohjelmisto. Ohjelmisto, jolla hallinnoidaan tietokantoja.
Javascript	Komentosarjakieli. Käytetään pääasiassa Web-ympäristössä.
HTML	Hypertext Markup Language. Web-ympäristössä käytettävä merkintäkieli, jolla esitetään sivun rakenne ja sisältö.
CSS	Cascading Style Sheets. Tyyliohjaiden laji, jolla Web-ympäristössä sivuille voidaan tehdä erilaisia tyylejä, kuten tekstin värin vaihtoa.
Relaatio-tietokanta	Tietokanta, joka perustuu relaatiomalleihin.
Bootstrap	CSS- ja HTML-kehikko, joita käytetään Web-projektien toteutuksissä.
jQuery	jQuery on ilmainen javascript-kirjasto, joka on tarkoitettu kaikille selaimille. Se on hyvin suosittu, koska sitä on helppo ymmärtää.
PDF	Portable Document Format. Siirrettävä tiedostomuoto.

1 Johdanto

Yleisesti suuret yritykset tarvitset jonkinlaisen ohjelman, jonka avulla voidaan ylläpitää varastoa, laskuja ja kanta-asiakkaita. Näin ollen on kehitetty monenlaisia laskutusjärjestelmiä, mutta suurin osa niistä joudutaan asentamaan tietokoneella, jolloin niitä ei pysty käyttämään kukaan muu kuin tietokoneen käyttäjä itse. Tämä myös aiheuttaa lisäkuluja, kun sama ohjelma on asennettava moneen eri tietokoneeseen.

Hyvä ratkaisu tähän olisi käyttää web-pohjaisia ratkaisujärjestelmiä, jolloin monet käyttäjät voivat käyttää sitä yhtä järjestelmää ilman toisen ohjelman asennusta omaan tietokoneeseen, jolloin säästytään lisäkuluilta. Tämä ratkaisu ei tietenkään ole ilmainen, sillä palveluntarjoajalle on maksettava palvelun käytöstä ja ylläpidosta, mutta sitä palvelua voidaan päivittää tiheämmin vastaamaan käyttäjän tarpeita, eikä kulut ole yhtä suuria kuin tietokoneeseen asennettavat ohjelmat.

Joitakin jo olemassa olevia web-pohjaisia järjestelmiä on esim. Isolta Oy, joka tarjoaa asiakkailleen web-pohjaisen järjestelmän, jossa asiakas voi tehdä kaiken tarvitsemanssa laskutukset mistä tahansa paikasta, kunhan on internetyhteys. Tämä on kätevää, sillä ohjelmaa pystyy käyttämään myös muulla laitteella kuin henkilökohtaisella tietokoneella. Esimerkiksi asiakas voi käyttää tablettia tai kännykkää laskun tekemiseen.

Työssä olisi kuitenkin tarkoitus kehitellä pienimuotoinen laskutusjärjestelmä, joka olisi pienyrityksen käytössä. Tässä yritetään saada vähintään toimiva prototyyppi valmiiksi, jossa olisi mahdollisuus luoda, poistaa ja muokata laskuja sekä lähettää laskuja sähköpostitse ja tulostaa. Kehitystä olisi tarkoitus jatkaa opinnäytetyön jälkeen ja lisätä siihen erilaisia ominaisuuksia, kuten varasto ja sen hallinta.

Tarkoituksen on toteuttaa opinnäytetyö käyttämällä nykyaikaisia tekniikoita, kuten PHP- [3] ja MySQL-palvelimien [7] viimeisimpiä versioita ja HTML5- [6] ja CSS3-ohjelmointikieliä [13] esittämään sivustoa loppukäyttäjälle.

2 Tarpeen kuvaus ja toimintaympäristö

2.1 Tarpeen kuvaus

Laskutusjärjestelmää lähdetään kehittämään kiinnostuksena taloushallinnosta ja www-sivujen tekemisesta. Näitä kahta tässä yritetään saada toimimaan yhdessä. Kiinnostus kasvoi, kun tuli kokeiltua yhden palveluntarjoajan laskutusjärjestelmää pienessä autojen huoltoja tekevässä yrityksessä.

Siitä lähtien on ollut tarkoituksena kehittää kyseinen järjestelmää joko opinnäytetyönä tai harrastuksena.

2.2 Toimintaympäristö

Kehitettävä sovellus on tarkoitus käyttää pienyrityksessä, joka harjoittaa autojen korjaamista ja niiden huoltoja. Yritys toimii toiminimellä. Yrityksen nimiä tässä ei mainita, koska se ei ole mitenkään mukana tässä projektissa, vaan tämä projekti on lähdetty kehittämään kiinnostuksen perusteella.

Laskutusjärjestelmä perustuu yrityksen tarpeisiin tarjoten joitakin tärkeita osia. Koska yrityksellä ei ole kovin paljon asiakaskuntaa ja liikevaihto tulee olemaan hyvin pieni, on laskutusjäjestelmälle asetetut tavoitteet riittäviä yritykselle. Kun yritys kehittyy ja edistyy, on laskutusjäjestelmää jatkokehitettävä vastaamaan silloiseen tarpeeseen tai tarpeisiin.

Yrityksen tarpeena on laskujen hallinta, tuotteiden hallinta ja asiakaiden hallinta. Lasku-osiossa on pystyttävä suorittamaan perustoimenpiteet, kuten laskujen luominen, poistaminen ja muokkaaminen sekä merkitseminen maksetuksi tai keskeneräiseksi. Muissakin osioissa on tarpeena kyseiset perustoimenpiteet. Muita tarpeita ei ole esiintynyt tällä hetkellä.

3 Laskutusjärjestelmät

Laskutusjärjestelmiä on monenlaisia. On tietokoneelle asennettavia ohjelmia, pilvipalveluissä suoritettavia ja mobiilille tarkoitettuja sovelluksia sekä tietenkin selainpohjaisia.

3.1 Ilmaiset laskutusohjelmat

Internetissä on paljon ilmaisia laskutusohjelmia, joita voi käyttää vapaasti ilman erillistä sopimusta palveluntarjoajan kanssa. Nämä ohjelmat on tarkoitettu yrityksille, joiden tarve laskuttaa on hyvin vähäistä, tai niille, jotka tekevät freelance-työtä. Ohjelmat tarjoavat kuitenkin vain laskun luomisen ja tallentamisen, jossa on seuraavat pakolliset tiedot:

- laskun lähettäjän tiedot kuten y-tunnus, nimi ja osoite
- laskutettavan henkilön tai yrityksen tiedot kuten nimi, osoite ja mahdollisesti myös y-tunnus
- tuotteet tai palvelut eriteltynä hintoineen, joista henkilö tai yritystä laskutetaan
- lähettäjän tilinumero
- laskun loppusumma.

Ilmaiset laskutusohjelmat ovat hyviä, koska ne ovat ilmaisia, mutta on niissä myös negatiivisia kohteita. Oheisessa taulukossa on mainittu joitakin hyviä ja huonoja puolia (Taulukko 1).

Taulukko 1. Ilmaisten laskutusjärjestelmien hyvät ja huonot puolet.

Selite	**Vahvuus**	**Heikkous**
Ilmainen, eli voi luoda niin paljon laskuja kun haluaa	✓	
Laskut joko tulostetaan tai tallennetaan tietokoneeseen, koska niitä ei säilytetä missään palvelussa		✓
Tuki on hyvin rajoittunut tai ei ole olemassa olleenkaan		✓
Ei sitoutumista, joten ohjelman käytön voi lopettaa milloin haluaa	✓	
Koska tietoja ei tallenneta mihinkään palveluun, raporttia ei ole näissä ilmaisissa ohjelmissa tarjolla		✓

3.2 Kaupalliset laskutusjärjestelmät

Kaupallisia laskutusjärjestelmiä löytyy paljon, kun Googlen hakukoneeseen kirjoittaa laskutusjärjestelmä hakusanaksi. Suurin osa tuloksista ovat kaupallisia organisaatioita, jotka myyvät kyseisen tuotteensa. Nämä tuotteet kuitenkin sisältävät monia erilaisia ominaisuuksia, joita ei yleensä löydy olleenkaan ilmaisista laskutusjärjestelmistä. Esim. kaupallisissa järjestelmissä on erilaisia raportteja, joita käytetään kirjanpidon helpottamiseksi.

Joitakin suosittuja laskutusjärjestelmien tarjoajia ovat Visma, Isolta, NetVisor, Zervant jne. Visman tarjoamat järjestelmät ovat tietokoneelle asennettavia ohjelmia ja muiden yritysten tarjoamat ovat selainkäyttöisiä pilvessä pyöriviä ohjelmia. Osa mainituista yrityksista tarjoavat palvelunsa myös ilmaiseksi rajoituksin. Esimerkiksi Isolta tarjoaa palvelunsa ilmaiseksi, mutta rajoituksena on 7 laskua per vuosi. Zervant tarjoaa 30 päivän rajoittamattoman kokeilujakson. Tämän jälkeen ohjelma lakkaa toimimasta, jolloin on hankittava maksullinen palvelu.

Yleensä kaupallisissa järjestelmissä käytetään kuukausihintaa. Mainituilla palveluntarjoajilla on kuukausihinta, mutta kaikilla on eri kuukausihinta riippuen siitä, mitä järjestelmä tarjoaa käyttäjälleen. Esim. Visma tarjoaa Passeli+ Standard-ohjelmansa [12] kuukausihintaan 23.80 €, johon lisätään vielä arvonlisäveroa. Passeli + Standard sisältää kaikki tarvittavat osiot pienyrityksen tarpeisiin. Visman sopimus on yleensä määräaikainen, joka kestää 36 kuukautta. Muutkin palveluntarjoajat tarjoavat palvelunsa asiakkailleen määräaikaisina sopimuksina, jotka yleensä kestävät vähintään vuoden tai kahden.

Kaupallisilla laskutusjärjestelmillä on niin hyviä kuin huonoja puolia. Taulukossa 2 esitellään muutamia hyviä ja huonoja puolia.

Taulukko 2. Ohessa joitakin hyviä ja huonoja puoli kaupallisista laskutusjärjestelmistä.

Selite	Vahvuus	Heikkous
Määräaikaissopimukset, joita ei saa irtisanottua ennen sopimuksen määräaikaisuuden päättymistä		✓
Järjestelmien tarjoamat raportit, jotka helpottavat kirjanpitoa	✓	
Palveluntarjoajan tarjoama tuki järjestelmässä esiintyneille ongelmille	✓	
Maksullinen tuki muille kuin järjestelmässä esiintyneille ongelmille, kuten käyttöönotossa		✓

3.3 Miksi uusi järjestelmä?

Tässä kehitetään uusi järjestelmä, koska ilmaiset laskutusjärjestelmät eivät tarjoa asiakkaiden tai tuotteiden hallintoja ja eivätkä laskut säily järjestelmässä. Maksulliset järjestelmät taas syövät osan yrityksen tuloista kuukausittain. Tämä järjestelmä kehitetään, jotta yrityksellä olisi mahdollisuus myös hallita asiakkaitaan ja tuotteitaan.

4 Lasku

Laskuja on monenlaisia, kuten normaalilasku, hyvityslasku, koontilasku jne. Tässä käytetään normaalilaskua laskujen luomisessa laskutusjärjestelmässä.

Laskun sisältö tulee määrittää ohjeiden mukaisesti [11]. Suomen arvonlisäverolan pykälässä 209 b on mainittu laskun tietosisällön pakolliset merkinnät, joita ovat

- laskun antamispäivä
- laskunnumero tai -tunniste
- myyjän y-tunnus
- myyjän ja ostajan nimi ja osoite
- tavaroiden tai palvelujen määrä ja luonne
- tavaroiden toimituspäivä

- yksikköhinta ilman veroa
- hyvitykset ja alennukset
- veron peruste verokannoittain
- verokanta
- suoritettavan veron määrä.

Joissain laskuissa voi olla poikkeustapauksiakin. Esimerkiksi laki voi määrätä lisätietojen antamista tai toisaalta keventää merkintätapoja.

4.1 Paperilasku

Paperilasku on paperille tulostettu lasku, jota lähetetään postitse laskussa osoitetulle henkilölle eli laskutettavalle. Posti kuljettaa sen ovelle, jolloin laskutettu henkilö maksaa saadun paperilaskun sisällön avulla laskunsa. Paperilasku on perinteisin tapa lähettää laskuja laskutettaville.

Kun posti kuljettaa paperilaskun, kestää se muutaman päivän ennen toimitusta laskutetulle. Toimituksen jälkeen laskun voi maksaa joko itse tietokoneella antamalla laskussa esitetyt tiedot verkkopankissa tai vaihtoehtoisesti pankin konttorilla käyttäen sen asiakaspalvelua.

4.2 E-lasku

E-lasku on sähköinen lasku, jota lähetetään laskussa osoitetulle henkilölle suoraan hänen pankinsa tilinumerolle, jolloin henkilö voi maksaa sen suoraan pankissa, tietokoneella. Henkilön ei tarvitse tehdä mitään muuta kuin vahvistaa laskua, jolloin pankki ohjaa veloitettavan summan laskun lähettäneelle.

Laskua voidaan lähettää NetPosti-palvelulle, joka näkyy paperilaskun muotona. Laskut säilyvät palvelussa vähintään 5 vuotta. Palvelun voi käyttää rekisteröitymällä ja kirjautumalla sisään pankkitunnuksia käyttäen.

Seuraavalla luvulla käydään läpi joitakin hyviä ja huonoja puolia e-laskusta ja paperilaskusta.

4.3 E-lasku vs. paperilasku

Sekä e-laskulla että paperilaskulla on hyviä ja huonoja puolia. Oheisissa taulukoissa esitetään joitakin hyviä ja huonoja puolia ensin e-laskusta ja sen jälkeen paperilaskusta.

Taulukko 3. E-laskun hyvät ja huonot puolet.

Selite	**Vahvuus**	**Heikkous**
E-laskusta ei synny hiilijalanjälkiä	✓	
Helppo ja nopea toimitus	✓	
Helppo maksaa vahvistamalla verkkopankissa, tai NetPostin palvelua käyttäen	✓	
Vaatii tietokoneen käyttöä, joita osalla suomalaisilla ei ole tai eivät osaa käyttää.		✓
Laskua ei näe ellei kirjaudu verkkopankkiin tietokonetta käyttämään.		✓

Taulukko 4. Paperilaskun hyvät ja huonot puolet

Selite	**Vahvuus**	**Heikkous**
Toimitus postitse, mikä voi joskus kadota tai mennä väärään asuntoon		✓
Paperilaskusta aiheutuu hiilijalanjälkiä, kun käyttää paperia		✓
Iäkkäille ihmisille helppo tapa saada laskua	✓	
Helppo pitää kirjaa laskuista, maksetuista ja maksamattomista	✓	
Paperilaskusta yleensä jonkinlainen kulutushinta laskussa		✓

Tässä työssä käsitellään paperilaskuja, koska on monia, jotka tarvitsevat vieläkin laskunsa paperina, kuten iäkkäät ihmiset. Tästä syystä asiakkaalle tulostetaan lasku,

jota annetaan hänelle joko paikan päällä tai postitetaan asiakkaan antamalle ja vahvistetulle osoitteelle.

5 Käytetyt tekniikat

Edellisissä luvuissa käsiteltiin laskutusjärjestelmiä ja vertailtiin erilaisia laskumuotoja. Siellä myös esiteltiin yritys ja sen asettamat vaatimukset. Seuraavaksi käydään läpi tekniikoita, joita hyödynnetään tässä ohjelmassa.

Koska laskutusjärjestelmä tulee olemaan dynaaminen, käytetään PHP-ohjelmointikieltä ohjelman kirjoittamiseen ja MySQL-relaatiotietokantaohjelmaa tietokannan ylläpitämiseen. Dynaaminen tarkoittaa sitä, että www-sivuihin voidaan lisätää tietoaa ilman koodin muokkaamista ja sitä, että sivut ladataan, kun selain sitä pyytää.

Toimintojen näyttäminen loppukäyttäjälle tapahtuu HTML-kielellä. Rinnalla käytetään CSS-kieltä, joka antaa HTML-koodille tyyliä, jotta loppukäyttäjän olisi helppoa lukea ja tehdä toimintoja.

5.1 HTML

Tietojen näyttämiseen käytetään HTML-merkintäkieltä, joka tulee sanoista HyperText Markup Language. HTML helpottaa tietojen esittämistä käyttäjälle. HTML-koodi koostuu tageista eli tunnisteista, joilla kerrotaan selaimelle, miten sisältöä näytetään. HTML-merkintäkieli on lähes kaikissa internetsivuissa käytetty.

Tunnisteet merkataan kulmasulkeissa muodossa

<tunniste>Sisältö</tunniste>

Esimerkiksi otsikkoa voidaan kirjoittaa h1-tunnistetta käyttäen seuraavasti:

<h1>Otsikko</h1>

HTML-tiedostojen pääte on yleensä *.html*, mutta käytetään myös *.php*-päätettäkin, kun tiedosto sisältää PHP-koodia.

5.2 CSS

CSS, eli Cascading Style Sheets on tyyliohjeiden laji, jolla internetsivuilla voidaan tehdä erilaisia tyylejä, kuten muuttaa tekstin kokoa, taustan väriä jne.

Tyyliohjeet voidaan lisätä HTML-tiedostoon <head></head>-tagien sisään. Tyyliohjeet joko kirjoitetaan suoraan HTML-tiedostoon tai erilliseen *.css*-päätteeseen tiedostoon, jotka linkitetään HTLM-tiedostoon.

CSS-tyylihje kirjoitetaan seuraavasti:

valitsin {attribuutti: arvo; }

Ennen valitsinta tulee joko piste (.) tai ristikkomerkki (#). Jos on piste, se tarkoittaa sitä, että sitä käytetään HTML-merkintäkielen *class*-atribuutin sisällä. Jos on ristikkomerkki, se taas tarkoittaa sitä, että se käytetään HTML-merkintäkielen *id*-atribuutin sisällä.

Esimerkiksi taustavärin voi vaihtaa kirjoittamalla

.taustavari {background-color: #ffffff;}

5.3 WampServer

Sovelluksen toimimisen kannalta tarvittiin web-palvelinympäristö, jossa olisi valmiiksi asennettuna PHP-, Apache- ja MySQL-palvelimet. WampServeria asentamalla saatiin täydellinen web-palvelinympäristö pystytettyä omaan tietokoneeseen, joka mahdollistaisi dynaamisten www-sivujen ajamista. WampServerin ollessa päällä päähakemistoon päästiin, kun selaimeen kirjoitettiin http://localhost/. Sieltä voitiin navigoida tietokantapalvelimeen sekä myös omaan hakemistoon. WampServerin versio, jota käytettiin tässä työssä, oli 3.0.4.

Xampp on toinen www-palvelinympäristöohjelma, joka toimii samantapaisesti kuin Wampserver ja jota olisi voinut vaihtoehtoisesti käyttää tämän projektin toteuttamiseen.

5.4 Apache HTTP server

Apache HTTP server on palvelin, jolla pystytään ylläpitämään avoimeen lähdekoodin perustuvaa HTTP-palvelinta. Apache HTTP server on julkaistu kaikille yleisille käyttöjärjestelmille, ja sen viimeisin julkaistu versio on 2.4.27. Tässä kuitenkin käytettiin versiota 2.4.18.

5.5 PHP-ohjelmointikieli

PHP tulee sanasta Hypertext Preprocessor. Se on ohjelmointikieli, jota käytetään pääasiassa dynaamisten www-sivujen luomiseen web-palvelinympäristössä. Se on hyvin suosittu ohjelmointikieli laajan luokkakirjastonsa ansiosta. Sitä käyttävät monet suuret julkaisujärjestelmät kuten WordPress, Magento ja phpBB. Yksi syy, jonka vuoksi sitä käytetään monesti, on sen kirjoittaminen HTML-merkintäkielessä, koska voidaan dynaaminen tieto näyttää sen sisällä (koodiesimerkki 1).

```
<!DOCTYPE html>
<html>
    <head>

        <title>PHP Esimerkki</title>
    </head>
    <body>
        <div>
            <?php

                // Tähän tulee php-koodi

                // Esimerkki
                echo "Hello World!";

            ?>
        </div>
    </body>
</html>
```

Koodiesimerkki 1. Raaka HTML-tiedosto, jossa käytetään PHP:ta.

Opinnäytetyössä käytetään PHP:n versiosta 5.1 tarjoama PDO-rajapintaa tietokantaan yhteydenottamiseen ja tietojen hakemiseen ja tallentamiseen tietokantaan [5]. PDO-lyhenne tulee sanoista *PHP Data Objects*, joka mahdollista erilaisten tietokantojen käyttämisen PHP-ohjelmien tietojen tallentamiseen.

```php
/**
*  Esimerkki luokasta PHP:ssa
*/
class LUOKKA{

    // tk = tietokanta
    private $tk;

    public function __construct($tk){
        $this->db = $tk;
    }

    // Tähän tulee php-koodi

    public function ekaFunktio(){

        // Tähän funktion toiminnot

    }

}
```

Koodiesimerkki 2. PHP-tiedosto, jossa esitellään luokan luomista.

Koodiesimerkissä 2 nähdään, miten luokkia luodaan. Sovelluksessa käytetään pääasiallisesti luokkia, joissa on funktioita suorittamaan erilaisia käskyjä tai toimintoja.

Työn olisi voinut toteuttaa myös muilla ohjelmointikielillä, kuten Pythonilla, .Net:llä jne. Tässä päädyttiin kuitenkin PHP-ohjelmointikieleen, koska siitä oli pientä kokemusta ennestään ja tarkoitus oli laajentaa osaamista.

PHP-ohjelmointikielestä on julkaistu monia versioita. Tässä opinnäytetyössä käytetään versiota 5.6.19.

5.6 MySQL-tietokantaohjelma

MySQL on suosittu relaatiotietokantaohjelma, jolla hallitaan tietokantoja. Se on käytetyin ohjelma tietojen säilyttämiseen Apache-pavelimessa. Tässä käytettiin PHP:n versiota 5.7.11.

5.7 JavaScript

JavaScript on on skriptikieli, jolla on monenlaisia kirjastoa. Kirjastoilla saadaan lisättyä dynaamisuutta sivustoihin, esimerkiksi datan syöttäminen ja sen tallentaminen voidaan suorittaa ilman sivujen päivittämistä tai lataamista uudelleen.

Projektissa JavaScriptiä on käytetty pääasiallisesti laskun luomissivulla, joka helpottaa asiakkaiden ja tuotteiden lisäämistä laskuun.

JavaScriptistä käytetään myös lyhennettä *JS*, ja niiden tiedostojen pääteet ovat *.js.*

5.8 Bootstrap

Bootstrap on valmis CSS-, JS- ja HTML-kehikko, jota voi käyttää MIT-lisenssin alla. Siinä on valmiiksi joitakin CSS-, HTML- ja JS-koodeja, joita voi käyttää vapaasti. Se tukee myös reponsiivisuutta eli on mahdollista toteuttaa vain yksi toteutus, jota voidaan käyttää monella eri laitteilla, kuten mobiililaitteilla. Tämä myös helpottaa kehittämistä. Tässä totetutuksessa käytettiin versiota 3.3.7. Bootstrap käyttää HTML5- ja CSS3-kieliä kehikon toteutuksessa. Bootstrap on yleiseti tuettu kaikilla nykyisillä selaimilla.

5.9 Font-Awesome

Font-Awesome tarjoaa laajan ja helposti muokattavia kuvakkeita. Se perustuu avoimeen lähdekoodiin ja sitä voi käyttää vapaasti myös kaupallisissa projekteissa. Voidaan muokata CSS-kielellä kuvakkeiden väriä, kokoa ja varjoa jne. Helpoin tapa lisätä kuvakkeita on lisäämällä -tunniste, jolle antaa CSS-prefixin ja sen jälkeen kuvakkeen nimen (koodiesimerkki 3).

```
<div class="wrapper">
    <i class="fa fa-awesome"></i>
</div>
```

Koodiesimerkki 3. Font-Awesomen esittäminen HTML-tiedostossa.

Font-Awesome on kirjaisintyyppi, jonka sisällä on kuvakkeita. Kirjaisintyyppi eli fontti on hyvin kevyt kooltaan, ja sen sisältö on laaja sekä muokattavissa. Tämän takia työssä on käytetty Font-Awesomea.

Toinen tapa olisi ollut kuvakkeiden suunnitteleminen ja toteuttaminen MS Paint- tai Adobe Illustrator -ohjelmilla, mikä olisi vienyt huomattavasti enemmän aikaa ja vaivaa. Ja Kuvakkeiden koko tulisi olemaan huomattavasti suurempi kuin fontti, mikä veisi enemmän kovalevytilaa, ja ohjelman latausaika tuplaantuisi, kun se joutuisi lataamaan kaikki kuvakkeet.

Näin ollen parhaaksi on nähty käyttää erillistä kuvakekirjastoa Font-Awesome.

6 Suunnittelu

Koska sovellus on hyvin monipuolinen, tulee sillä olla myös monia osia. Päättelyn ja pohdinnan jälkeen on päätetty, että sovelluksen osat ovat seuraavat:

- TUOTE
- ASIAKAS
- LASKU
- ASETUKSET.

Näistä jokainen suorittaa omat toiminnot, jotka ovat merkittäviä sovellluksen toimimista ajatellen.

Lasku-osiossa on käyttäjän päästävä luomaan, muokkaamaan ja poistamaan niitä sekä päivittämään niiden tiloja, esim. jos lasku on maksettu, niin tila merkitään suoritetuksi, muuten keskeneräiseksi.

Asiakas-osiossa käyttäjä luo valmiiksi joitakin asiakkaita, joille lähetetään laskuja usein. Siinäkin käyttäjällä on luomis-, muokkaamis- ja poistamismahdollisuudet.

Tuote-osioon tallennetaan ne tuotteet, joita on hankittu myytäväksi ja ne tuotteet, jotka myydään usein. Esim. autohuollossa sama öljy tullaan käyttämään moneen eri autoon,

joten olisi järkevää tallentaa se ja lisätä sitä käyttönsä mukaan. Tuotteen tärkeitä tietoja, joita tallennetaan tietokantaan, ovat

- tuotteen tunniste
- tuotteen nimi
- tuotteen hinta.

Käyttäjällä on mahdollisuus luoda, muokata ja poistaa tuotteita.

Asetukset-osiossa käyttäjä täyttää tai päivittää yrityksensä tietoja, kuten

- yrityksen nimi
- y-tunnus
- tilinumero
- logo.

Nämä tiedot näytetään laskuissa, joita käyttäjä luo.

Sivuston rakenne koostuu ylätunnisteesta, sisällöstä ja alatunnisteesta (kuva 4).

Kuva 1. Sivuston rakenne piirrettynä.

Ylätunnisteessa on sivuston logo ja navigointipalkki. Navigointipalkilla navigoidaan sivuston eri osiin. Se sisältää seuraavat linkit:

- etusivu
- tuotteet
- asiakkaat
- laskut
- kirjaudu ulos.

Ylätunniste näytetään kaikissa sivuston osissa paitsi kirjautumissivulla. Ylätunniste ei ole dynaaminen, joten sen sisältö tulee olemaan aina sama.

Sisältö-osiossa näytetään kaikki olennaiset tiedot ja siinä tapahtuu myös kaikki käyttäjän suorittamat toiminnot, kuten asiakkaiden, laskujen ja tuotteiden hallinta sekä asetukseksien muutokset. Sisältö on dynaaminen, joka on aina erilainen eri sivuston osissa.

Alatunniste on samantapainen kuin ylätunniste, mutta tämän sisältö tulee olemaan erilainen, mutta kuitenkin staattinen. Siinä näytetään käyttäjälle sivuston tekijänoikeusteksti ja vuosi.

7 Toteutus

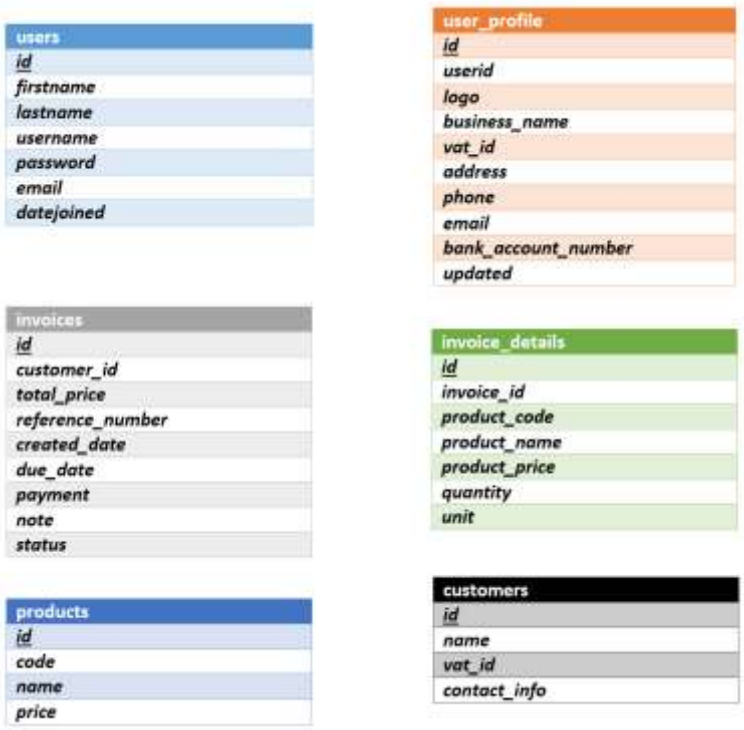
Suunnittelun jälkeen päästään varsinaiseen osaan eli toteutukseen. Tässä osiossa käydään läpi varsinaista toteutusta, eli käydään läpi sovelluksen eri kohtia, jotka vaativat esittelyä ja pilkkoamista kokonaiskuvan hahmottamiseksi.

Tässä ei käydä kovin yksityiskohtaisesti jokaista koodiriviä läpi vaan käydään läpi ne tärkeimmät kohdat sekä perusasiat.

Työ suoritetaan WampServer-palvelinympäristössä. Koska opinnäyettyö ei vaadi omaa domainia tai hosting-palvelua, niitä ei käsitellä tässä.

7.1 Tietokannan esittely

Tietokannassa on viisi taulua, jotka kukin tallentavat omat tietonsa. Niitä näytetään käyttäjän pyynnöstä (kuva 2). Käyttäjällä on kaksi taulua, johon tallennetaan käyttäjän tiedot. Tuotteilla on yksi taulu, johon tallennetaan tuotteen perustiedot. Asiakas-tauluun tallennetaan asiakkaan tiedot. Laskulla on kaksi taulu, jossa toisessa on laskun perustiedot ja toisessa laskutettavat tuotteet tai palvelut.



Kuva 2. Tietokantataulut ja niiden rakenteet

Users

Users-taulussa tallennetaan tietoja laskutusjärjestelmän käyttäjästä. Käyttäjä rekisteröityy täyttämällä lomakkeen, joka tallennetaan tietokantaan, user-tauluun. Rekisteröityessä käyttäjä antaa etunimen, sukunimen, sähköpostin ja käyttäjätunnuksen sekä salasanan.

User_profile

User_profile-taulussa tallennetaan muuta tietoa, kuten yrityksen nimen, y-tunnuksen, logon, sähköpostin ja tilinumeron. Nämä tiedot näytetään laskussa, kun sellainen on luotu.

Invoices

Invoices-taulussa tallennetaan tietoja laskun perustiedoista, kuten laskun luomispäivämäärä, eräpäivä, viitenumero, maksutapa, laskutettavan asiakkaan id ja laskun tila sekä lisätietoja-kenttä.

Invoice_details

Tässä taulussa tallennetaan myytävien tuotteiden tunnistetta, nimeä, hintaa ja määrää sekä yksikköä. Tässä tallennetaan myös invoice-taulun id:n.

Customers

Customers-taulussa tallennetaan pakollisen id:n lisäksi nimi, y-tunnus ja yhteystiedot. Yhteystiedot, kuten osoite, puhelinnumero ja sähköposti tallennetaan yhteen soluun, koska sitä on helpompi lisätä laskuun.

7.2 Yhteys tietokantaan

Tietokantaan otetaan yhteyttä käyttämällä PHP:n tarjoamaa PDO-rajapintaa. PDO-rajapinta on joustavampi ja tehokkaampi kuin vanhempi ”mysql”. PDO:lla voi myös yhdistää muihin tietokantoihin kuin MySQL-tietokantaan. Tässä työssä käytämme MySQL-tietokantaa. Seuraavassa joitakin tietokantoja, jotka tukevat PDO-rajapintaa [5].

- MySQL
- Microsoft SQL Server
- Oracle
- PostgreSQL
- SQLite.

```php
// Tarvittavat tiedot yhteydenottamiseen
$host = "localhost";
$db_name = "billme";
$username = "root";
$password = "";

// PDO-yhteys
try {

    $db = new PDO("mysql:host={$host};dbname={$db_name}", $username, $password);
    $db->exec("SET NAMES utf8");
}

// Virheet
catch(PDOException $exception){
    echo "Connection error: " . $exception->getMessage();
}
```

Koodiesimerkki 4. Tässä esitellään tietokantaan yhteydenottamista.

Koodiesimerkissä 4 nähdään, miten yhteydenottaminen toimii PDO-rajapinnalla. Aluksi on asetettu tarvittavat merkkijonot, kuten tietokannan nimi, käyttäjä, salasana ja host. Koska WampServerissä ei ole asetettu salasanaa MySQL-käyttäjälle, se on jätetty tyhjäksi.

Varsinainen yhteydenottaminen tapahtuu try-catch-lauseella. Tryn sisällä otetaan yhteyttä tietokantaan. Yhteyden epäonnistuessa näytetään virheet catchin sisällä. Myös muut tietokantakyselyt tapahtuvat *try-catch*-lauseella.

7.3 Tarvittavat luokat

Järjestelmässä käytetään pääluokkia, joissa suoritetaan kaikki oleelliset toiminnot, kuten tietojen haut, tallennukset ja muokkaukset sekä poistot. Tässä on luotu neljä pääluokkaa, joilla kaikilla on omat tietokantataulunsa yllä mainittuihin toimintojen suorittamiseen. Luokat ovat USER, PRODUCT, CUSTOMER ja BILL.

USER-luokka

Luodussa PHP:n USER-luokassa on funktioita, joilla lisätään, muokataan tai poistetaan tietoja tietokantatauluissa *users* ja *user_profile*. Siinä on myös funktioita, joilla kirjataan käyttäjää sisään, verrataan salasanojaa ja tarkistetaan käyttäjän olemassaoloa tietokannassa.

```php
/**
*
* Funktiolla kirjaudutaan sisään
*
*/
public function login($username, $pass){
    try
    {
        $stmt = $this->db->prepare("SELECT * FROM users WHERE username =:username LIMIT 1");
        $stmt->execute(array(':username'=>$username));
        $row = $stmt->fetch(PDO::FETCH_ASSOC);
        if($stmt->rowCount() > 0){
            if(password_verify($pass, $row['password'])){
                $_SESSION['user_session'] = $row['id'];
                return true;
            }else{
                return false;
            }
        }
    }
    catch(PDOException $e)
    {
        echo $e->getMessage();
    }
}
```

Koodiesimerkki 5. Tässä on koodiesimerkki sisäänkirjautumisesta.

Koodiesimerkissä 5 oleva login-funktio on tarkoitettu käyttäjän sisäänkirjautumista varten. Funktio saa kaksi parametriä, jotka ovat käyttäjätunnus ja salasana. Nämä tiedot käyttäjä antaa kirjautumislomaketta täyttäessä (kuva 8). Annetut käyttäjätunnus ja salasana tarkistetaan, jonka jälkeen ne laitetaan tämän funktion parametreiksi. Tämä funktio hakee ensi kyseistä käyttäjää tietokannasta. Käyttäjän löytyessä verrataan annettua salasanaa ja tietotakannassa tallennettua salasanaa toisiinsa. jolloin sen löytyessä tarkistetaan salasanojen täsmäävään toisiinsa.

Tarkistus ja uudelleenohjaus tapahtuu koodiesimerkin 6 mukaisesti. Kun käyttäjän antamat tiedot ovat okein, ohjataan käyttäjä etusivulle. Muuten ilmoitetaan käyttäjälle, että annetut käyttäjätunnus ja salasana eivät täsmää.

Rekisteröityminen

Käyttäjä pystyy rekisteröitymään käyttämällä kirjautumissivussa olevaa rekisteröintilomaketta. Lomaketta täytettyä kaikki annetut tiedot tarkistetaan, jonka jälkeen ne tallennetaan tietokantaan.

Lomakkeessa kysytään perustietojen lisäksi käyttäjätunnusta ja salasanaa. Näillä käyttäjä kirjautuu sovellukseen ja alkaa käyttää sitä. Käyttäjällä on annettu vapaat kädet

käyttäjätunnuksensa valitsemiseen, mutta sitä on kuitenkin rajoitettu kahdeksaan merkkiin.

Salasanan tallentaminen

Kun käyttäjä rekisteröintilomakkeessa antaa haluamansa salasanan salasana-kenttään, se tallennetaan tietokannassa kryptattuna eli salattuna. Näin ollen kukaan ei pääse käsiksi salasanoihin, jos tietomurto tapahtuu tietokannalle.

Salasana salataan käyttäen PHP:n tarjoamaa *password_verify()*-funktiota. Funktio on ollut mukana PHP:n versiosta 5.1, joka on kehitetty hyvin pitkälle, joten sen käyttäminen on hyvin suositeltavaa pienelle tietokannalle. Tämä riittää tällä hetkellä tähän projektiin, mutta laajentaessa salausta kannattaa vaihtaa laajalle tietokannalle sopivaksi salaukseksi, kuten MD5-salaus.

Sisäänkirjautuminen

Koska ohjelmaa käyttäessä on kirjauduttava sisään, on tehty funktio, joka tarkistaa aina sivun päivittyessä käyttäjän kirjautumista. Kun huomataan, että käyttäjä ei ole kirjautunut, hänet ohjataan kirjautumissivulle, jossa käyttäjän on annettava käyttäjätunnuksensa ja salasanansa kirjautuakseen sisään. Kun käyttäjä on kirjautunut sisään, hänet ohjataan etusivulle.

Kuva 3. Kirjautumissivu, johon käyttäjää ohjataan, kun käyttäjän halutessa käyttää sovellusta.

USER-luokan sisälle on luotu functio *logged_in()*, joka palauttaa arvon true, jos käyttäjän sessio on asetettu eli käyttäjä on kirjautunut sisään (koodiesimerkki 6).

```
/**
* Funktiolla tarkistetaan asetettua sessiota käyttäjälle
*
*/
public function logged_in(){
    if(isset($_SESSION['user_session'])){
        return true;
    }
}
```

Koodiesimerkki 6. Käyttäjän sisäänkirjautustarkistusfunktio.

Funktio tulee käytettyä sovelluksen ylätunnisteessa. Koska ylätunniste tulee olemaan kaikissa sivuissa, niin kerran kirjoitettu koodi riittää käyttäjän kirjautumistarkistusta varten. Koodiesimerkissä 7 nähdään funktion käyttöä ylätunnisteessa.

```
/**
* Jos käyttäjä ei ole kirjautunut,
* ohjaamme kirjautumissivulle
*/
if(!$user->logged_in()){
    redirect('entry.php');
}
```

Koodiesimerkki 7. Esitys koodiesimerkkissä 6 esiintyneestä funktion käytöstä.

Funktion palauttaessa arvoa epätosi ohjautuu käyttäjä kirjautumissivulle. Muuten käyttäjä on kirjautunut, ja hänelle annetaan oikeudet suorittaa toimintoja.

Lomakkeiden tarkistus ja virheiden ilmoittaminen

Käyttäjän antamaa lomakkeen sisältöä tarkistetaan aina ennen niiden hyväksymistä. Kun käyttäjä täyttää väärin tai jättää täyttämättä lomakkeen kenttiä, ilmoitetaan käyttäjälle virheistä. Tarkistetaan jokainen kenttä erikseen, jotta käyttäjälle olisi helppoa korjata niitä sen mukaan kun virheilmoitus esiintyy. Tarkistukset suoritetaan PHP-koodilla.

Jos kenttä on tyhjä tai se ei täytä asetettuja määräyksiä, asetetaan luotuun *error*-nimiseen taulukkoon virheilmoitus, jossa kerrotaan käyttäjälle, missä ongelma esiintyy ja mitä on tehtävä ongelman korjaamiseksi (koodiesimerkki 8).

```
if(isset($_POST['username']) && !empty($_POST['username'])){
    $username = clearData($_POST['username']);
}else{
    $error[] = 'Anna käyttäjätunnus!';
```

Koodiesimerkki 8. Koodissa esitellään esimerkki lomakkeissa annettujen tietojen tarkistuksesta ja mahdollisten virheiden ilmoittamista.

Koodiesimerkissä 8 tarkistetaan, onko *username*-niminen kenttä asetettu, eikä se ole tyhjä. Kun molemmat arvot pitävät paikkaansa, puhdistetaan annetun kentän arvo *clearData*-nimisella funktiolla (koodiesimerkki 9).

```
/**
* Funktiolla puhdistetaan käyttäjän antamaa dataa
*
*/
function clearData($data){
    $data = trim($data);
    $data = stripcslashes($data);
    $data = htmlentities($data);

    return $data;
}
```

Koodiesimerkki 9. Kyseessä on funktio, jonka avulla puhdistetaan käyttäjän antamaa dataa.

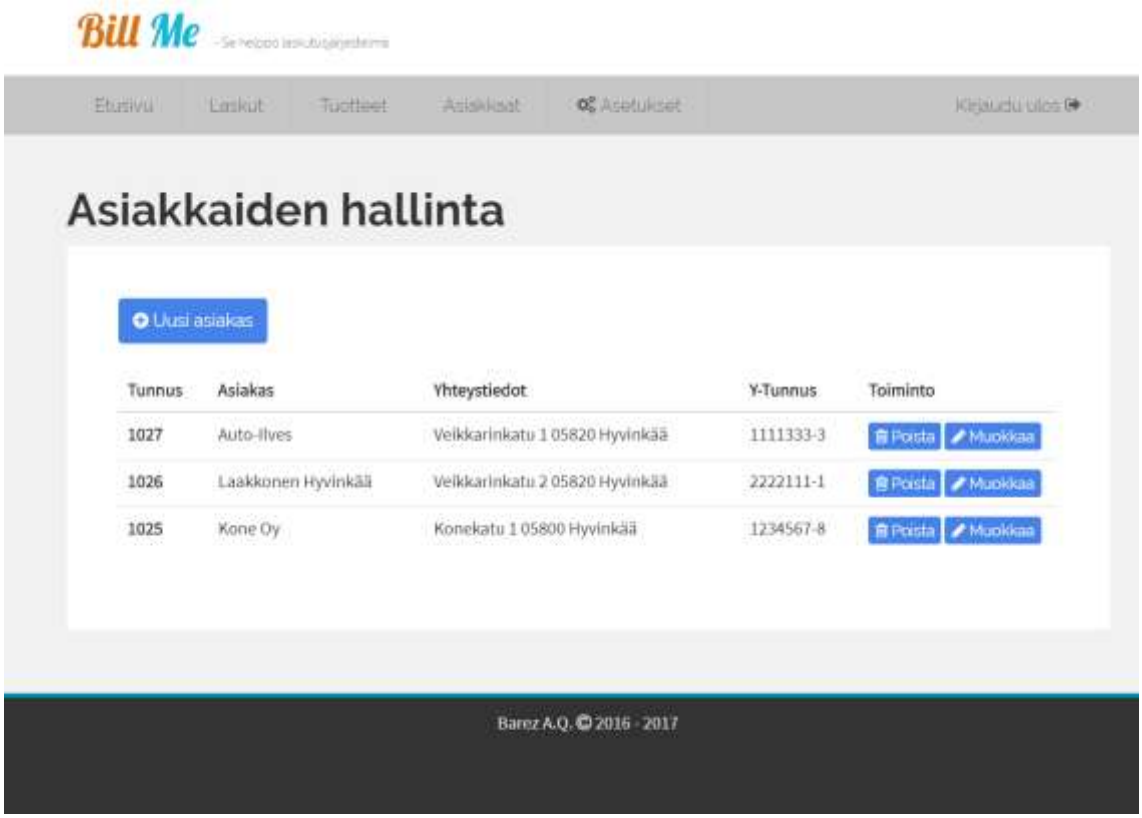
Koodiesimerkissä 10 käydään läpi *error*-taulukkoa, kun sitä on asetettu. Ensin *if*-lauseella käydään läpi *error*-taulukkoa siitä, onko sillä vähintään yksi arvo vai onko se kokonaan tyhjä. Kun *if*-lause on käyty läpi ja huomattu, että *error*-taulukko sisältää arvon tai arvoja, toteutuu *if*-lause, jolloin käydään läpi myös sen sisällä olevat toiminnot. Koska *error* on taulukko, sen sisällön tulostaminen tapahtuu joko *for*-lauseella tai tässä käyttettyä *foreach*-lausella.

```
if(isset($error)){
    echo '<div class="alert alert-danger">';
    foreach($error as $error){
        echo $error.'<br>';
    }
    echo '</div>';
}
```

Koodiesimerkki 10. Kyseinen koodipätkä on esimerkki siitä, miten lomakkeessa annettuja tietoja voisi tarkistaa ja tarpeen vaatiessa ilmoittaa virheistä.

CUSTOMER-luokka

Kyseisellä luokalla käyttäjä hallitsee asiakkaita.



Kuva 4. Perusnäkymä asiakkaiden hallintasivulta.

Kuvassa 4 nähdään peusnäkymä asiakkaiden hallintasivulta, jossa näytetään tietokannassa olevat asiakkaat ja niiden tiedot. Painamalla *Uusi asiakas*-nappia pääsee kuvassa 5 näkyvään lomakkeeseen, jossa luodaan uutta asiakasta.

Kuva 5. Lomake uuden asiakkaan lisäämiseen tietokantaan.

```
public function deleteCustomer($id){
    if(!empty($id) && is_numeric($id)){
        $sql = "DELETE FROM customers WHERE id = '".$id."' LIMIT 1";
    }

    try{
        $stmt = $this->db->prepare($sql);
        if($stmt->execute()){
            return $stmt;
        }
    }catch(PDOException $e){
        echo "Virhe!".$e->getMessage();
    }
}
```

Koodiesimerkki 11. Tämä koodipätkä on luokassa CUSTOMER käytetty funktio, jolla on tarkoitus poistaa asiakkaita tietokannasta sen tietyllä id:llä.

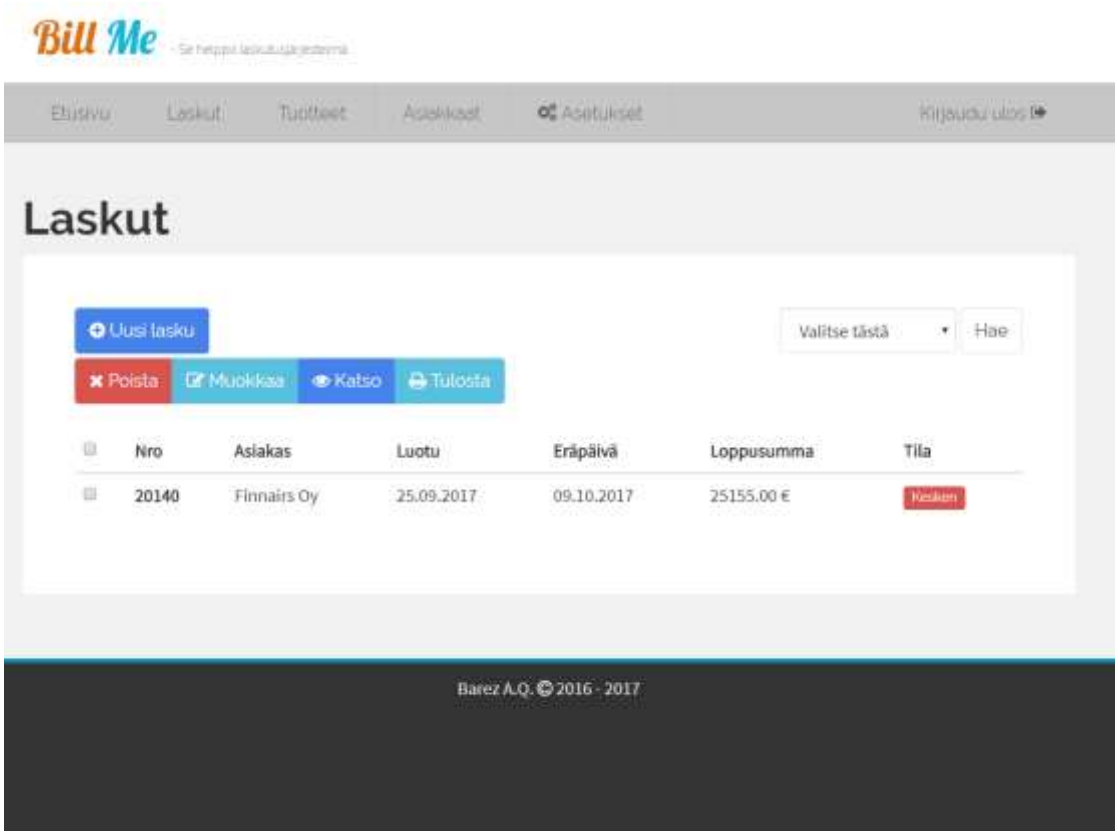
Funktiolla *deleteCustomer* poistetaan asiakasta id:llä. Samantapaista funktiota käytetään myös muissa luokissa.

TUOTE-luokka

Tuote-luokassa päästään tallentamaan uusia tuotteita tietokantaan. Näin niitä on helppoa lisätä laskuun. Tuotteiden hallintasivu tulee näyttämään pitkälti samanlaiselta kuin asiakkaiden hallintasivu, mutta tässä tallennetaan eri tietoja.

LASKU-luokka

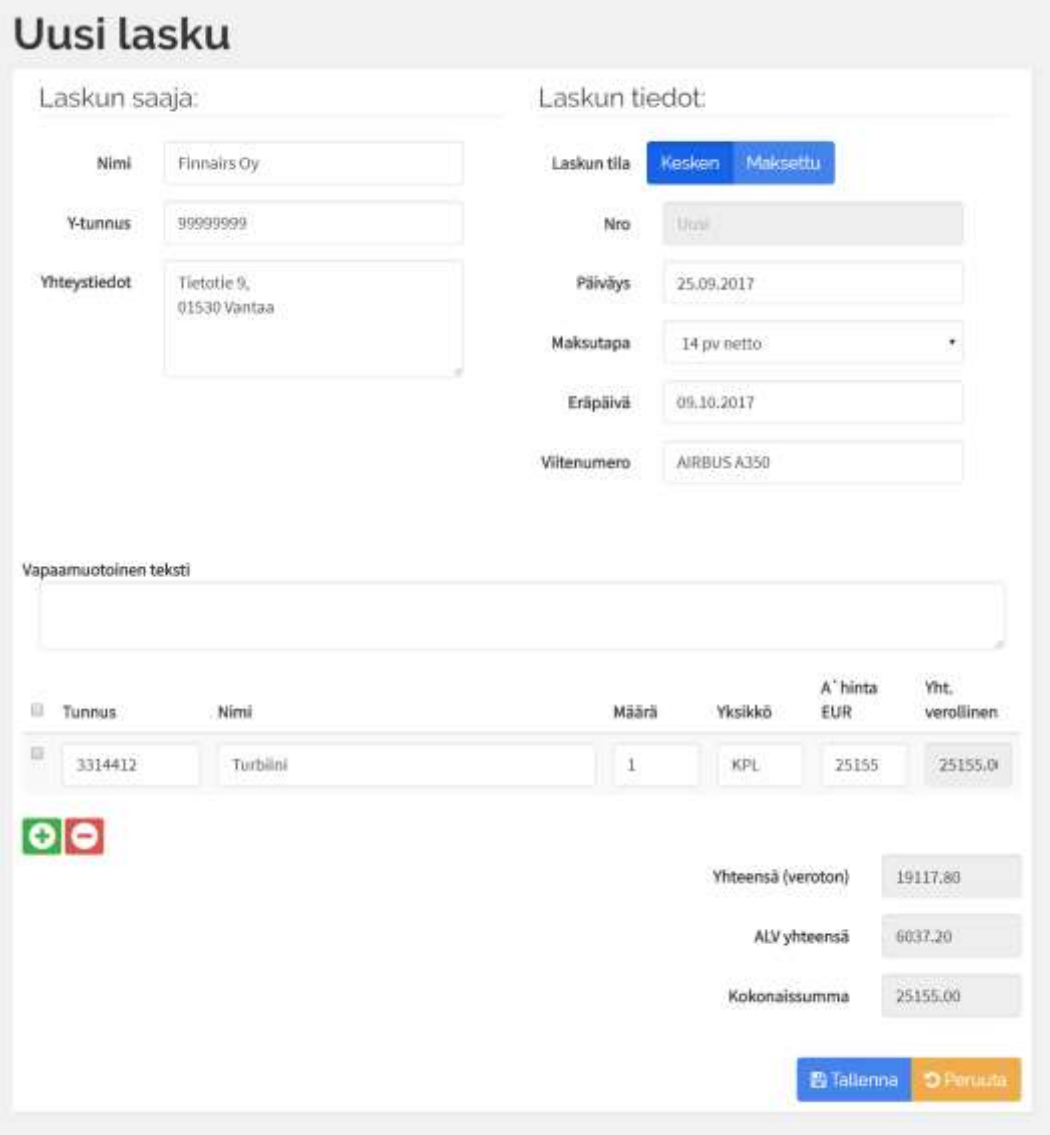
Lasku-luokassa on kirjoitettu kaikki siihen liittyvät toiminnot, kuten lisäykset, poistot ja muokkaukset.



Kuva 6. Perusnäkymä laskujen hallintasivulta.

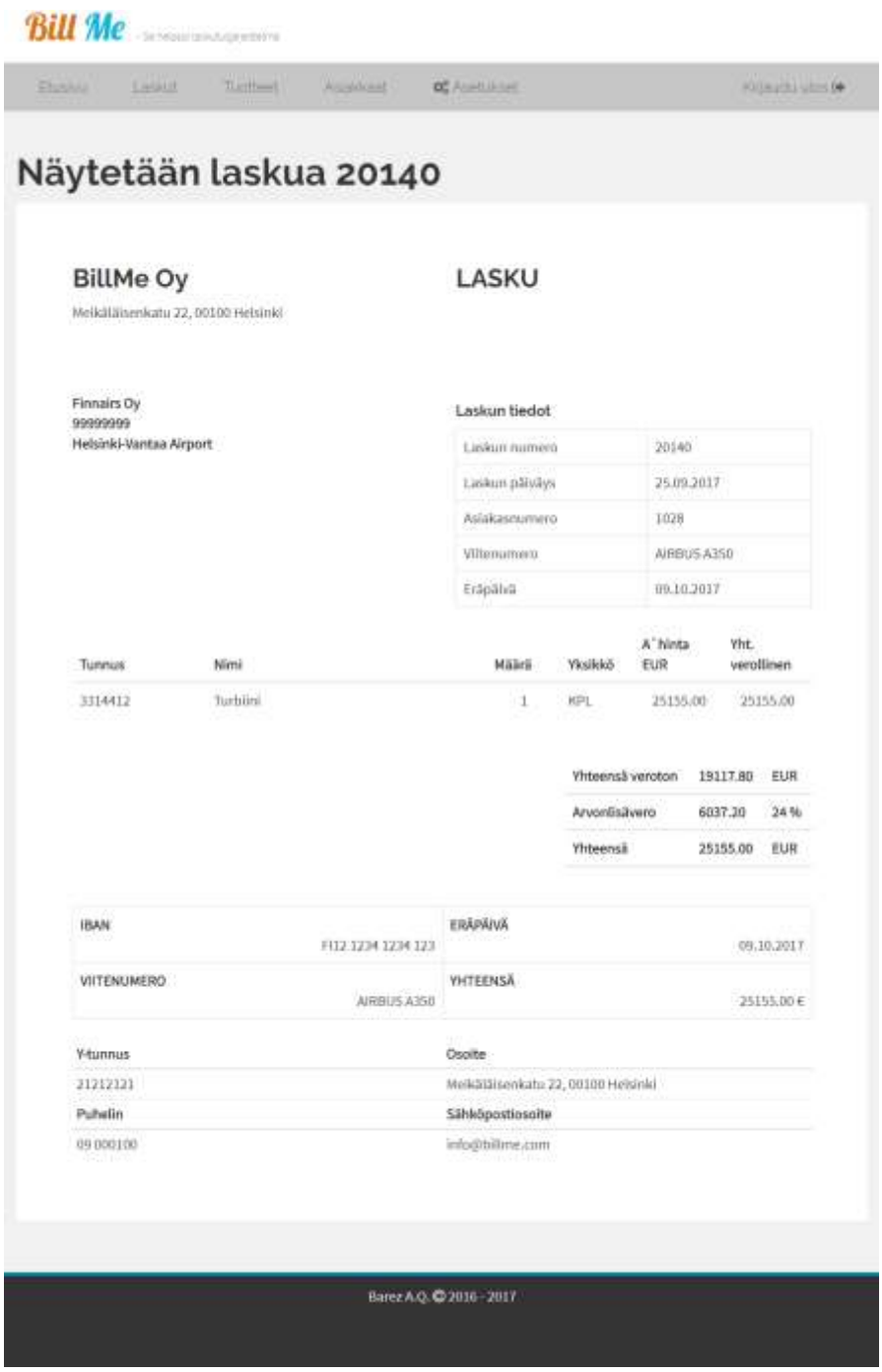
Perusnäkymässä nähdään laskujen tietoja. Siinä on myös erilaisia toimintonappeja, joilla voidaan poistaa, muokata tai katsoa laskua. Laskut voidaan myös rajata siten, että näytetään vain suoritetut laskut tai vain kesken olevat. Perusnäkymässä laskut näytetään viimeisimmästä vanhimpaan.

Uuden laskun voi luoda painamalla *Uusi lasku*-nappia, josta pääsee kuvassa 7 näkyvään lomakkeeseen. Kun lomakkeen tiedot täytetty ja painettu *tallenna*-nappia yritetään tallentaa laskua, jolloin sen onnistuessa käyttäjä ohajataan laskujen hallintasivulle kertomaan, että tallennus onnistui.



Kuva 7. Uuden laskun luominen.

Lomakeessa kysytään saajan tietoja, laskun perustietoja ja laskutettavan tuotteen tai palvelun tietoja. Laskun perusitetoja ovat päiväys, eräpäivä, viitenumero maksutapa ja laskun tila. Käyttäjälle näytetään myös loppusumma.



Kuva 8. Laskun näyttäminen käyttäjälle

Käyttäjän halutessa vilkaista laskua hän voi valitse kyseisen laskun hallintasivulta ja painaa *katso*-nappia, jolloin näytetään valittu lasku (kuva 8).

8 Ratkaisun arviointi

Ongelma lähdettiin ratkaisemaan siten, että se vastaisi vähintään toiminimellä toimivan pienyrityksen tarvetta laskujen hallinnassa. Ongelmaa ratkaistiin erilaisilla kokeiluilla, jossa tarvittiin koodaustaitoja ja teknistä ymmärtämistä sekä taloushallinnon tietämystä.

Yrityksen ensimmäisenä vaatimuksena oli laskujen hallinta, joka sisältäisi laskujen luomisen, poistamisen ja muokkaamisen. Toteutus tapahtui siten, että ensin koodattiin laskujen luomista. Sen jälkeen toteutettiin luotujen laskujen listaus, jonka jälkeen poistaminen oli seuraavana listassa. Kun laskujen poistaminen toteutui, niin toteutettiin laskun muokkaamisosa. Näiden vaadittujen toimintojen toteutusten jälkeen listättiin muutama pieni perustoiminto, jotka helpottaisivat laskujen selaamista maksettujen ja maksamattomien laskujen välillä. Laskuissa toivottiin myös laskujen tulostamista PDF-muotoon, joka jäi tässä vaiheessa pois toteutuksesta.

Seuraavana vaatimuksena oli asiakkaiden hallinta. Asiakkaiden hallinnassa vaadittiin samat toiminnot, joita vaadittiin myös laskujen hallinnasta. Eli poisto, luominen ja muokkaaminen olivat tärkeitä toimintoja. Asiakkaiden hallinta-osio toteutettiin kaikki vaatimuksia täyttäen, joten prosenttimääräisesti toteutus oli onnistunut.

Seuraavan ja viimeisenä vaatimuslistalla oli tuotteiden hallinta, joka helpottaisi laskujen luomista, kun tuotteet voitiin lisätä laskuihin kirjoittamalla tuotteen tuotekoodin lasku-osioon, jolloin tuotteen tiedot lisättiin automaattisesti laskuun. Tuotteiden hallinta toteutettiin pitkälti samalla tavalla kuin asiakkaiden hallinta ja siinäkin saatiin toteutettua asiakkaan vaatimat osat.

Kaiken kaikkiaan toteutus on onnistunut hyvin pitkälle, jolloin voidaan aloittaa jo beta-testaukset.

Tietoturvallisesti ohjelmaa voisi vielä hiota vastaamaan nykyisiä standardeja, vaikka osa niistä on jo käytetty tässä projektissa. Tämä on suositeltavaa vähintään silloin, kun halutaan jatkokehittää laajentaen ohjelmaa. Tämä koskee myös tietosuojaa.

Tässä tietosuojaan ei kiinnitetä kovin paljon huomiota, koska tallennettavat tiedot ovat hyvin pieniä ja vähäisiä toimintaympäristössä.

Ohjelmaa on voitu ratkaista myös muilla ohjelmointikielillä ja myös muilla tavoin. Tässä opinnäytetyössä näytetään yksi esimerkki siitä, miten ohjelmaa tulisi rakentaa.

9 Yhteenveto

Tässä opinnäytetyössä pyrittiin tutustumaan erilaisiin ohjelmointikieliin ja niiden avulla kehittämään laskutusjärjestelmää. Tavoitteena oli kehittää laskutusjärjestelmää pienelle yritykselle, joka helpottaisi laskujen hallintaa.

Sovellusta kehitettäessä saatiin prototyyppi valmiiksi. Kuitenkin päästiin kehittämään sen verran pitkälle, että käyttäjä pääsee luomaan ja hallitsemaan laskuja. Käyttäjä pääsee myös tallentamaan uusia tuotteita ja asiakkaita tietokantaan sekä hallitsemaan niitä.

Tavoitteestä jäi laskun tulostaminen PDF-muotoon. Tämä on kuitenkin kehiteltävissä, kun sovellusta jatkokehitellään tulevaisuudessa. Jatkokehityksessä voidaan lisätä myös muitakin toimintoja, kuten PDF-laskun lähettämistä suoraan sähköpostiosoitteeseen.

Tässä käytiin läpi monesti teknistä puolta antaen erilaisia esimerkkejä ja joitakin oikeita koodia, joita oli käytetty projektissa. Tässä kuitenkin on tuotu esiin enemmän PHP-puolta, koska se oli monimutkaisin kaikista muista kielistä, joita on käytetty tässä projektissa.

Lähteet

1 Lasku, verkkodokumentti., päivitetty 11.1.2017, luettu 12.1.2017: <https://fi.wikipedia.org/wiki/Lasku>.

2 Apache HTTP server, verkkodokumentti, luettu 15.11.2016: <https://httpd.apache.org/>.

3 PHP, verkkodokumentti, päivitetty 15.8.2016, luettu 10.9.2016: <https://fi.wikipedia.org/wiki/PHP>.

4 PHP, verkkodokumentti, luettu 10.9.2016: <https://secure.php.net/>.

5 PHP Data Objects, verkkodokumentti, luettu 15.9.2016: <https://secure.php.net/manual/en/book.pdo.php>.

6 HTML5 Tutorial, verkkodokumentti, luettu 10.8.2016: <https://www.w3schools.com/html/default.asp>.

7 SQL Tutorial, verkkodokumentti, luettu 13.9.2016: <https://www.w3schools.com/sql/default.asp>.

8 Bootstrap, verkkodokumentti, luettu 11.8.2016: <https://getbootstrap.com/docs/3.3/>.

9 Bootstrap Tutorial, verkkodokumentti, luettu 1.9.2016: <https://www.w3schools.com/bootstrap/default.asp>.

10 jQuery Tutorial, verkkodokumentti, luettu 12.2.2017: <https://www.w3schools.com/jquery/default.asp>.

11 Laskutusvaatimukset arvonlisäverotuksessa, verkkodokumentti, päivitetty 7.7.2014, luettu 18.10.16: <https://www.vero.fi/fi-FI/Syventavat_veroohjeet/Arvonlisaverotus/Laskua_koskevat_vaatimukset/Laskutusvaatimukset_arvonlisaverotuksess(33169)>.

12 Passeli+Standard, verkkodokumentti, luettu 19.10.2017: <https://www.visma.fi/passeli/standard/>.

13 Cascading Style Sheets, verkkodokumentti, luettu 15.9.2016: <https://fi.wikipedia.org/wiki/Cascading_Style_Sheets>.