TARTU ÜLIKOOL
MATEMAATIKA-INFORMAATIKATEADUSKOND

Arvutiteaduse instituut
Informaatika eriala

Kristjan Robam

Turvaline laohaldussüsteem programmeerituna keeles PHP

Bakalaureusetöö (6 EAP)

Juhendaja: Helle Hein

Autor: …… "……" august 2012
Juhendaja: …… "……" august 2012

Lubada kaitsmisele
Professor: …… "……" august 2012

TARTU 2012

Sisukord

Sissejuhatus

Kuna tänapäeval on suhteliselt suur vajadus ladudes ja muudes arveldamise asutustes arvutipõhiste haldusvahendite järele, siis sobiks valiti käesoleva bakalaureusetöö teemaks hästi näiteks laohaldussüsteemi programmeerimine. Käesoleva bakalaureusetöö eesmärgiks on seega seega luua lihtne laohaldussüsteem, mida on võimalik kasutada erinevates ladudes. Bakalaureusetööna valmiv laohaldussüsteem ei eelda kasutamisel sügavaid teadmisi programmeerimisest ning peaks olema seega lõpptarbijale lihtsasti kasutatav. Valmiv süsteem programmeeritakse keeles PHP. Selles programmeerimiskeeles on laohaldussüsteeme programmeeritud vähe. Vaatamata lihtsusele on loodaval süsteemil piisav funktsionaalsus lihtsamate operatsioonide sooritamiseks. Süsteemi edasiarendamisel on võimalik funktsionaalsust täiendada.

Antud laohaldussüsteemi koostamisel on kasutatud lisaks programmeerimiskeelele PHP ka muid programmeerimisvahendeid, nagu näiteks JavaScript, HTML5 [3] ja CSS [4]. Javascripti puhul on kasutatud ka tema teeki nimega JQuery [5]. Keele Javascript tehnoloogiatest on kasutatud ka tänapäeval ühte sageli kasutatavat tehnoloogiat AJAX [6, 7].

Bakalaureusetöös valmiva laohaldussüsteemi puhul on suur rõhk pandud ka süsteemi turvalisusele, ilma milleta oleks süsteem liiga lihtsalt kahjustatav mittesoovitud kontingendi poolt. Süsteem kasutab turvalisuse tagamisel muuhulgas ka HTTPS protokolli [8].

Internetist järele uurides võib märgata, et taolisi vabavaralisi süsteeme leidub hulganisti. Võtame nendest vaatluse alla kolm: *SUNTECH Target*, *inFlow* ja *myWMS LOS (Release 1.3)*. Esimese puhul tuli peale installeerimist välja, et selle programmi tööle saamiseks tuleb mingil numbril helistada, muidu sellele programmile ligi pääseda pole võimalik. Seetõttu jäi antud programmi proovimine poolikuks, sest helistamine võib osutuda kalliks. Teine süsteem, mis internetist leiti, oli selline, et vabavaralises süsteemis võib maksimaalselt salvestada 100 toodet ja klienti. Selle programmi miinuseks oli ka vähene funktsionaalsus. Näiteks polnud võimalik seal salvestada toodetele kommentaare ja importijat. Valmival laohaldussüsteemil on see võimalus olemas. Kui kolmas süsteem oli tööle pandud, siis tuli esmalt ekraanile teade "*An internal error has occurred*" ja siis

teade “*Service not available. Are you connected to the server?*”. Ilmselt oli viga selles, et server *JBoss* polnud korralikult tööle läinud, sest minnes nõutud lehele “*http://localhost:8080/los-mobile*” tekkis ekraanile teade “*The requested resource (/los-mobile) is not available ”*.

Eelnevast võib teha nii mõnegi järelduse. Näiteks, et tasuta lõunaid pole tegelikult olemas ning et taolised vabavaralised süsteemid ei ole otse kasutatavad.

1. Nõuete analüüs

Üheks oluliseks etapiks tarkvara loomisel on nõuete analüüs, mis peaks andma ülevaate, millised nõuded on esitatud antud tarkvarale ning analüüsima, kas need on ka täidetud. Nõuded jagunevad funktsionaalseteks ja mittefunktsionaalseteks. Käesolevas peatükis esitame ülevaate loodava süsteemi nõuetest.

1.1. Mittefunktsionaalsed nõuded

Antud süsteemile esitatavad mittefunktsionaalsed nõuded on järgmised:

1. Süsteemi operatsioonisüsteemiks on eeldatud Windows XP. Põhimõtteliselt võib süsteem töötada ka muudes operatsioonisüsteemides, kuid nendes süsteemi ülesseadmist ei kirjeldata. Tegelikult peaks olema teistes operatsioonisüsteemides süsteemi ülesseadmine samuti lihtne, sest teistes operatsioonisüsteemides peaks lihtsalt installeerima Apache serveri, PHP programmeerimiskeele ning andmebaasisüsteemi MySQL.

2. Arvuti, millel süsteem tööle pannakse, peaks olema selline, mis suudaks lubada töötada üheaegselt programmeerimiskeelega PHP, andmebaasiga MySQL ja Apache serveriga. Kindlasti peaks süsteem töötama vähemalt järgmiste parameetritega arvutil: protsessor: Pentium 4 1,7GHz, mälu 256MB, kõvaketas 20GB. Väiksemate parameetritega arvuti puhul süsteem võib töötada, kuid võib ka mitte töötada.

3. Süsteemi kasutamisel ei tohi jääda paroolid brauserile meelde. Kuna tänapäeval enamus brausereid suudab hoida paroole meeles, siis tuleks see ära keelata, et igaüks antud süsteemi siseneda ei saaks. Kui parool on alles, siis võidakse süsteemi sisse logida

seni, kuni keegi on tühjendanud brauseri mälu. Internet Exploreri korral on näiteks paroolide salvestamise keelamine "*Tools->Internet options->Content->AutoComplete Settings*" alt, kus tuleb võtta linnuke ära kasti "*User names and passwords*" eest. Firefoxis toimib see näiteks "*Tööriistad->Sätted->Turvalisus*" kasti "*Veebilehtede paroolid salvestatakse*" eest linnukese ära võtmise teel. Kui ollakse kindlad, et süsteem on piisavalt kaitstud, siis võib paroolide salvestamise alles jätta, kuid sellisel juhul tuleks pärast töö lõpetamist süsteemist välja logida, et vältida veateadetega ekraane, mis võivad ette tulla, kui jäädakse sisselogituks.

4. Laohaldussüsteemil peab olema kaasaegne disain. Antud süsteemi puhul on see nõue igati täidetud, sest on kasutatud kaasaegse disaini võtteid nagu näiteks CSS ja JQuery. Kindlasti oleks võimalik süsteemi kujunduselt palju võimsamaks teha, kuid antud süsteemi puhul on rohkem keskendutud funktsionaalsusele.

5. Süsteem peab olema lihtsasti kasutatav. Iga tegevuse puhul antud süsteemis on hästi näha, millega seal täpselt tegeletakse ning milline menüüpunkt on valitud.

1.2. Funktsionaalsed nõuded

1. Antud süsteemis on algselt arvestatud kahe kasutajatüübiga: tavakasutaja ja administraator. Tavakasutajaid on võimalik hiljem juurde lisada, kuid süsteemi administraatoreid ei saa olla rohkem kui üks. Kasutatava süsteemi parameetrid on määratud administraatori poolt. Administraator võib lisaks parameetrite määramisele(punkti *Sätted* alt) sooritada ka süsteemi kasutajate tegevusi.

Süsteemi kasutaja tegevusteks on:

- toodete lisamine, muutmine, kustutamine, otsimine;
- toote nimede lisamine, kustutamine;
- tootjate lisamine, muutmine, kustutamine;
- importijate lisamine, muutmine, kustutamine;
- töötajate lisamine, muutmine, kustutamine;
- klientide lisamine, muutmine, kustutamine;
- arvete lisamine, kustutamine, otsimine, arve maksmata staatuse muutmine (süsteemil on ka krediidi võtmise võimalus);

- lao sätete vaatamine (toodete hulk, millel on madal laoseis; minimaalselt tooteid arves; soodustuste protsent; käibemaksu protsent);
- lao planeerijasse kirjete sisestamine;
- klientidele soodustuste lisamine ja kustutamine;
- statistika jälgimine (suurema summa eest tooteid ostnud kliendid ja suurema läbimüügiga tooted);
- lao varade lisamine, muutmine, kustutamine;
- laoga seotud dokumentide lisamine ja kustutamine;
- parooli vahetus;
- süsteemi info vaatamine.

2. Kui laohaldussüsteemi ei kasutata kasutajaliidesest, siis on süsteemil nõue, et pärast tabelisse "*arvekaupseos*" (Joonis 1) kirjete lisamist kontrollitaks, kas kõigis arvetes on vähemalt minimaalne kogus tooteid. Selleks saab kasutada protseduuri `tooteidv2hem()`. Protseduur annab veergu *v2ljund* tulemi, kas mõnes arves on tooteid minimaalsest kogusest vähem või ei. Kui mõnel arvel on tooteid vähem kui minimaalne kogus, siis tuleks kindlasti sinna minimaalne kogus tooteid lisada. Kui aga tabelisse "*arvekaupseos*" kirjeid ei lisata, siis ei pea seda protseduuri kasutama.

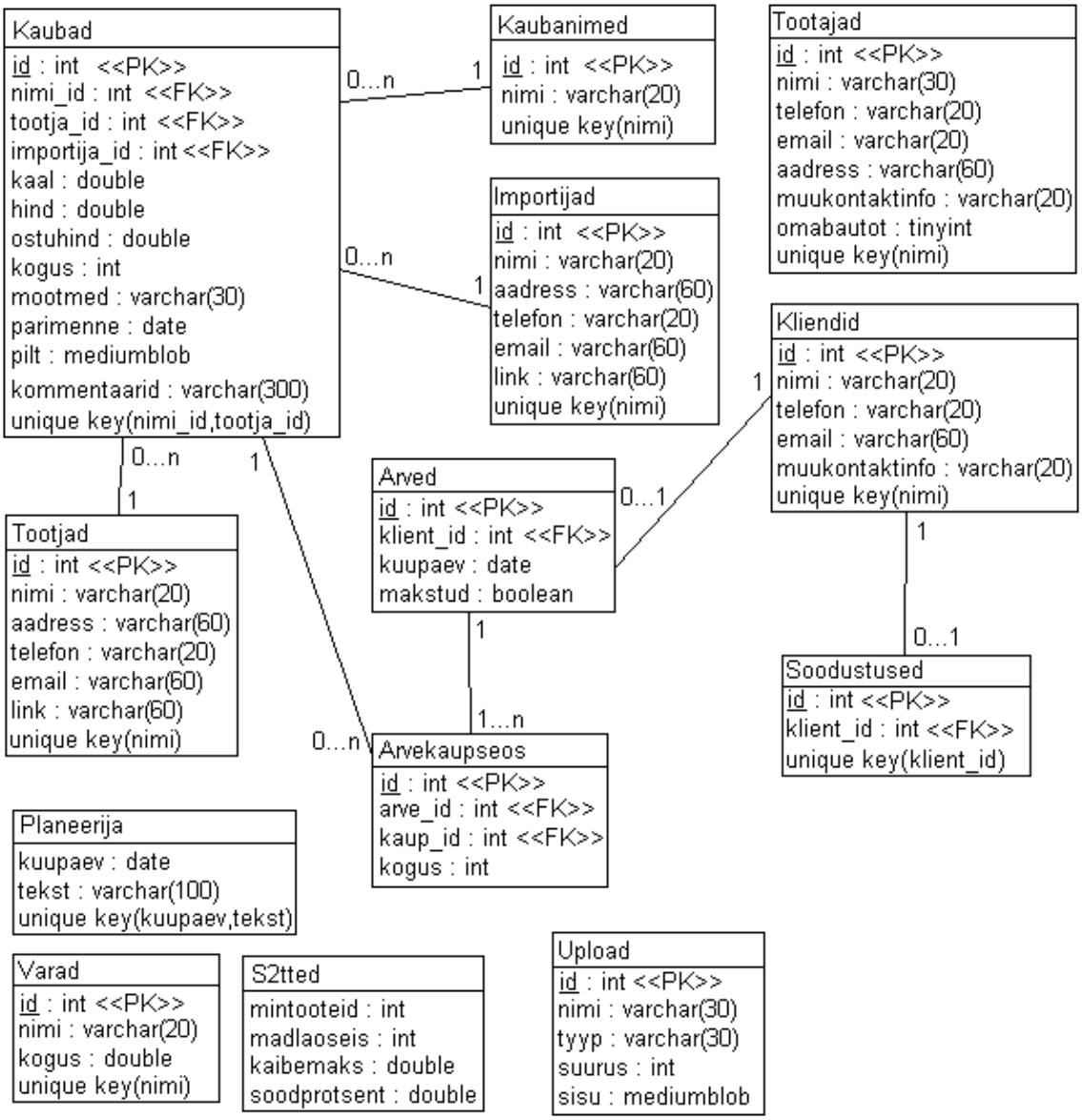
3. Antud laohaldussüsteemi puhul tuleks ka kontrollida pärast laos inventuuri tegemist protseduuri `tooteidv2hem()` abil, et arvetes poleks vähem tooteid, kui lubatud. Kui protseduur annab vastuseks jaatava vastuse, siis tuleks süsteemi logist (*C:\localhost.log*) järgi vaadata, milline kasutaja on lisanud arvesse vähem tooteid, kui lubatud ja seda kasutajat firma poolt karistada või hoiatada sellise tegevuse eest.

4. Pärast eelmise nõude täitmist tuleks peale lao inventuuri peatada MySQL server ja tühjendada süsteemi logifail selle kustutamise teel, et serveris ruumi kokku hoida. Hiljem, kui MySQL server uuesti tööle pannakse, tekib see fail uuesti. Algul tuleks siis klikkida *Task Manager*is "*mysqld.exe*" peale parema klahviga ning valida "*End Process*". Siis tuleks kustutada fail "*C:\localhost.log*". Seejärel võib MySQL serveri eelnevalt tehtud lingist uuesti käima panna.

Need oleks nüüd kõik funktsionaalsed nõuded, mis on süsteemile esitatud ning need on kõik ka täidetud.

2. Andmemudel

Joonisel 1. on näidatud süsteemi andmemudel esimesel normaalkujul.



Joonis 1. Laohaldussüsteemi andmemudel esimesel normaalkujul.

Andmebaasile on lisatud 10 trigerit. Esimene triger on lisatud tabelile "*s2tted*" sellepärast, et tabelisse ei oleks võimalik üle ühe kirje lisada ja sellepärast, et ei oleks võimalik lisada tabelisse mittesobilikke väärtusi. Teine triger ei luba uuendada tabelit "*s2tted*" mittesobilike väärtustega. Kolmas triger võimaldab lisada tabelisse "*tootajad*" ainult inimese täisnimesid ning nõuab, et kirjete lisamisel oleks mingi võimalikest kontaktinfodest olemas. Neljas triger lubab muuta tabelis "*tootajad*" nimesid ainult

selliselt, et need on inimese täisnimed ning nõuab, et kirjete muutmisel oleks mingi võimalikest kontaktinfodest olemas. Viies triger lubab juhul, kui toode on arves, vaid kustutada toote pilti ja muuta toote kogust ning see triger vaatab ka, et oleks võimalik uuendada tabeli "*kaubad*" väljasid nagu kogus, kaal, hind ja ostuhind ainult sobivalt, vastasel korral kuvab süsteem ette vea. Kuues triger lubab tabelisse "*kaubad*" sisestada ainult lubatud väärtusi. Seitsmes triger lubab tabelisse "*varad*" sisestada ainult sobivaid koguseid. Kaheksas triger lubab tabelis "*varad*" uuendada kogust ainult sobivaks. Üheksas triger lubab lisada tabelisse "*arvekaupseos*" ainult selliseid tooteid, mis juba ei sisaldu tabelis arvekaupseos ja lubab lisada ainult kogust mis on suurem nullist. Kümnes triger lubab uuendada tabelit "*arvekaupseos*" ainult selliselt, et kogus oleks suurem nullist. Kõik trigerid, nagu ka järgmised protseduurid, mis järgnevalt vaatluse alla võetakse, asuvad Lisas 3.

Süsteemis on ka kolm protseduuri: `koiktooted(page int)`, `koikarved(page int)` ja `tooteidv2hem()`. Esimene protseduur väljastab kõik tooted koos väljadega *kaubanimi*, *tootja*, *importija*, *hind* ja *kogus* leheküljel "page". Lehekülgedel on kuni 50 kirjet. Teine protseduur väljastab kõik arved väljadega *id*, *klient*, *makstud* (ehk makstud või maksmata) ja *kuupaev* leheküljel "page". Igal leheküljel on siingi kuni 50 kirjet.

Kolmas protseduur väljastab, juhul kui tabelis "*arvekaupseos*" on mõnes arves vähem tooteid kui minimaalne kogus ette näeb, veateate. Süsteemi protseduure saab kutsuda välja MySQL käsurealt järgmiselt: `call procedure (parameeter1, parameeter2,…);` .

Süsteemi aruandeid on võimalik vaadata järgmiste linkide alt: "*Toodete nimekiri*", "*Töötajate nimekiri*", "*Klientide nimekiri*", "*Arvete nimekiri*", "*Soodustused*", "*Statistika*", "*Lao varad*", "*Dokumendid*" ja "*Süsteemi info*".

3. Kasutatavad tehnoloogiad

Antud süsteem kasutab tehnoloogiatest peale PHP ka tehnoloogiaid nimedega MySQL, HTML5, CSS, Javascript ja HTTPS protokolli. Antud süsteemi operatsioonisüsteemiks eeldatakse olevat Windows XP.

PHP puhul üheks olulisemaks mooduliks, mida kasutatakse, on sessioonid. Selle abil suudetakse süsteemi töö ajal salvestada olulisi muutujaid näiteks nagu kasutajanime ja parooli. Kui süsteem on töö lõpetanud, siis kustutatakse salvestatud muutujad. Kuigi kasutajanime on võimalik pärast igaühel kas brauseri mälus olevalt menüülehelt või parooli vahetamise lehelt järgi uurida, siis parooli järgi uurida pole võimalik.

Tähtis osa laohaldussüsteemi programmeerimisel keeles PHP [9] on objektorienteeritud lähenemisel. Sellest kirjutatakse ka lisas olevas peatükis "*PHP programmeerimiskeelest*". Näiteks kasutatakse selliseid objekte nagu `Kaup` ja `Tulemus`. Kuna esimene objekt on kasutuses mitmes failis, siis on see objekt pandud eraldi faili, et kõik seda objekti vajavad failid seda kasutada saaks. Objekti `Tulemus` kasutab aga ainult üks fail ja sellepärast eraldi faili selle objekti jaoks tehtud pole. Üks põhjus, miks objekorienteeritud programmeerimist kasutada, on kindlasti selles, et oleks võimalik objekte sorteerida. Selleks saab kasutada protseduuri `usort($j, "cmp")`, kus `$j` on mingi jada ja `cmp` on selles jadas olevate objektide järjestamise funktsioon, näiteks selline:

```
function cmp($a, $b) {
   if($a->labimuuk>$b->labimuuk) return 1;
   if($a->labimuuk<$b->labimuuk) return -1;
   return 0;
}
```

See võrdlusfunktsioon annab tulemuseks 1 ehk objekt `$a` on pärast objekti `$b`, kui objekti `$a` isendiväli `labimuuk` on suurem objekti `$b` isendiväljast `labimuuk`, -1 ehk objekt `$a` on enne objekti `$b`, kui objekti `$a` isendiväli `labimuuk` on väiksem objekti `$b` isendiväljast `labimuuk` ja 0 ehk objektid on võrdsed, kui objektide `$a` ja `$b` isendiväljad `labimuuk` on võrdsed. PHP protseduur `usort($j, "cmp")` järjestab antud juhul objektidest isendiväljaga `labimuuk` koosneva jada `$j` funktsiooni "`cmp`"

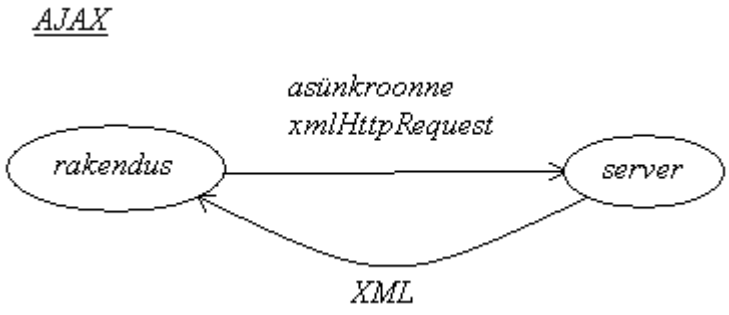
järgi kasvavalt ära.

Laohaldussüsteemis on kasutatud ka programmeerimiskeelt nimega Javascript, mis on PHP programmide peaaegu lahutamatu osa. Seega koosnevad PHP lehed väga tihti mõlemast, nii keelest Javascript, kui ka keelest PHP. Näiteks võiks tuua järgmise koodilõigu.

```
print 'var r=confirm("Selline klient on juba olemas.");
     if(r==true){
      window.location.href="muudak.php?id='.$kliendi_id.'";
}';
```

Selles öeldakse kasutajale, et “Selline klient on juba olemas.” ja jaatava vastuse puhul suunatakse kasutaja juba selle olemasoleva kliendi muutmise leheküljele “*muudak.php*” mingi olemasoleva kindla parameetriga.

Veel kasutab antud laohaldussüsteem tehnoloogiat nimega AJAX. AJAX on tehnoloogia kiirete ja dünaamiliste veebilehekülgede loomiseks. AJAX tähendab lahtiseletatult asünkroonset Javascript ja XML [10] koostoimimist nagu Joonisel 2. näidatud. Asünkroonne Javascript ja XML koos toimima panduna on selline, kus rakendus saadab serverisse ja võtab sealt informatsiooni lubades süsteemil paralleelselt edasi töötada. Vahetatav informatsioon on XML või mõnel teisel kujul. Sünkroonsel lähenemisel peab rakendus ära ootama tellitud informatsiooni ja alles siis saab kasutaja asuda muude tegevuste juurde. Sellepärast kasutatakse tehnoloogiat nimega AJAX suurte serverist tagasisaadavate andmemahtude puhul ning teist tehnoloogiat väikeste tagasisaadavate andmemahtude puhul.



Joonis 2. Tehnoloogia AJAX tööpõhimõte.

Antud süsteemi kood kasutab näiteks sellist AJAX tehnoloogia koodilõiku:

```
function xmlhttpPost() {
    var xmlHttpReq = false;
```

```
    var self = this;
if (window.XMLHttpRequest) {
        self.xmlHttpReq = new XMLHttpRequest();
    }
else if (window.ActiveXObject) {
        self.xmlHttpReq = new
ActiveXObject("Microsoft.XMLHTTP");
    }
    self.xmlHttpReq.open('POST', "paring.php", true);
    self.xmlHttpReq.setRequestHeader('Content-Type',
'application/x-www-form-urlencoded');
    self.xmlHttpReq.onreadystatechange = function() {
        if (self.xmlHttpReq.readyState == 4) {
            updatepage(self.xmlHttpReq.responseText);
        }
    }
    self.xmlHttpReq.send(getquerystring());
}
function getquerystring() {
    var form     = document.forms['f1'];
    var nimi = form.nimi.value;
    qstr = 'nimi=' + escape(nimi);
    return qstr;
}
function updatepage(str){
    document.getElementById("result").innerHTML = str;
}
```

Algul kutsutakse peale sündmust “`onkeyup`” toote nime kastis välja keele Javascript funktsioon `xmlhttpPost`. Funktsioon `xmlhttpPost` saadab failile *paring.php* vormist “`f1`” välja toote nime väärtuse ja saab tagasi faililt *paring.php* vastuse HTML (XML alamkeel) näol. Konkreetsel juhul saab funktsioon tagasi HTML toote nime vastete dropdown valik. See dropdown valik pannakse HTML “`result`” nimega `div` (see on keeles HTML üks faili osa) märgisesse, kus kasutajal on võimalik toote nime vasteid valida. Antud süsteemis kasutatakse tehnoloogiat AJAX veel lisaks päris mitmes kohas.

Antud laohaldussüsteem kasutab veel lisaks ka keele Javascript teeki nimega JQuery. Seda teeki kasutatakse menüüelementide aktiivseks muutmiseks. Täpsemalt kasutatakse antud süsteemis järgmist üpris olulist JQuery koodilõiku:

```
var make_button_active = function()
{
  var siblings =($(this).siblings());
  if(!$(this).hasClass('active')) {
    var siblings =($(this).siblings());
    siblings.each(function ()
      {
         if($(this).hasClass('active')) {
            $(this).removeClass('active');
                  $(this).addClass('link2');
         }
      }
    );
   $(this).removeClass('link2');
   $(this).addClass('active');
   top.frames['sisu'].location.href=$(this).find('a').attr(
'href');
  }
}
$(document).ready(
  function()
  {
      $(".link2").click(make_button_active);
      $(".active").click(make_button_active);
  }
)
```

Selle koodiga pannakse menüü elementidele stiiliklassidega `link2` ja `active` peale funktsioon, mis vahetab nende menüüelementide klasse selliselt, et kui valitakse uus menüüelement, siis võetakse vanal menüüelemendil stiiliklass `active` pealt ära ja pannakse peale stiiliklass `link2` ja uuelt võetakse stiiliklass `link2` pealt ära ja pannakse peale klass `active` ning kui jäetakse sama menüüelement aktiivseks, siis stiiliklasside vahetamist ei toimu. Lisaks, kui vajutatakse uut menüüelementi, siis

avatakse menüüelemendile vastav link süsteemi raamis “*sisu*”.

Nagu igas tänapäeva veebirakenduses, kasutatakse ka selles keelt nimega CSS. Järgneva CSS failiga(*stiil.css*) määratakse HTML lehe tagataust ja fondi värv, vormide äärekaugus, stiiliklassid nimedega `.kiri`, `.menyylink` ja `.menyy`, mis pannakse hiljem peale märgistele `div`.

Vastav stiilifail *stiil.css* näeb välja järgmine:

```
body {background-color:#F5DA81; color: black;}
body#teine {background-color:#87CEFA; color: black;}
form{margin:0;}
.kiri{background:#01DFA5; color:#0000A0; width: 132px;}
.menyylink{text-decoration: none;}
.menyy{color:#0000A0; background:#01DFA5; width: 132px;
text-align: left;}
```

Siiski kasutatakse HTML osade stiilide määramiseks väga tihti ka tavapärast lähenemist nagu näiteks järgmise rea puhul:

```
style="height: 10px; font-size: 0"
```

Selles muudetakse mingi elemendi kõrguseks 10 pikslit ja fondi suuruseks 0.

Tehnoloogia HTML5 puhul kasutab süsteem päris paljudes failides meta märgise atribuuti `charset`. Seda sellepärast, et süsteem tunnustaks ka eestikeelseid tähti. See näeb välja järgmine: “`<meta charset="Windows-1257">`”. Erinevalt keelest HTML4, kus kasutati eestikeelse tähestiku valimiseks koodi: ”`<meta http-equiv="Content-Type" content="text/html; charset=Windows-1257">`”on see ülestähendus palju lühem ja lihtsam.

Veel on tehnoloogiatest kasutusel HTTPS protokoll ehk inglise keeles *Hypertext Transfer Protocol Secure*, mida kasutakse igal pool serverisse saadetavate ja sealt saadavate andmete krüpteerimiseks. Võrreldes HTTP protokolliga on see protokoll turvaline, sest HTTP ei kasuta mingit krüpteerimist andmete vahetuses. Selle installeerimisest on juttu Lisa teises punktis.

4. Süsteemi turvalisus

Nagu juba tehnoloogiate all öeldud, siis antud süsteem kasutab HTTPS protokolli [11] ehk turvalist HTTP protokolli. Ilma selleta oleks andmete vahetamine kasutaja ja serveri vahel üsnagi ebaturvaline, sest kurjategija võib pealt kuulata saadetavaid andmeid ning sealt vajaliku informatsiooni välja lugeda, et süsteemi sisse murda. Seega on väga vajalik, et Lisa teise punkti (*Vajamineva tarvara installeerimise juhend*) alt oleks installeeritud veebiserverile ka moodul OpenSSL, mis lisab serverile HTTPS protokolli toe.

Antud süsteemi kasutamine on kaitstud kasutajanime ja parooliga. Kui kasutaja sisse logib, siis pannakse programmeerimiskeeles PHP kasutaja info jada tüüpi muutujasse `$_SESSION`. Täpsemalt pannakse sinna elementi `$_SESSION['kasutaja']` kasutajanimi ja elementi `$_SESSION['parool']` parool. Kui kasutaja on süsteemist välja loginud "*Logi välja*" nupu abil, siis kustutatakse muutujast `$_SESSION` need andmed ära. Juhul, kui laohaldussüsteemi sessioon jääb avatuks, siis on võimalik ka teistel kasutajatel süsteemile ligi pääseda. Selleks tuleb alati pärast laohaldussüsteemis töö lõpetamist sealt välja logida.

Parimal juhul on võimalik teistel kasutajatel näha kasutajanime, kellega süsteemi viimati sisse logiti, sest süsteem jätab selle menüü ja parooli vahetuse lehel meelde, kuid selle äraarvamine ei anna suurt eelist süsteemi kurjategijal sisselogimiseks, sest võimalike proovitavate paroolide hulk on ülisuur.

Süsteemile on võimalik ligi pääseda ainult kasutajal, kellel on selleks antud kasutajanimi ja parool ning MySQL "*root*" kasutajal. Üldiselt on siis süsteem mõeldud tavakasutajatele ja administraatorile. Administraator võib lisaks tavakasutaja õigustele süsteemis muuta ka tabelit *"s2tted*", kus on võimalik muuta süsteemi sätteid.

Kuigi "*root*" kasutajal on kõige kõrgemad õigused andmebaasikeeles MySQL, vaadeldakse teda laohaldussüsteemi veebilehel kui tavakasutajat. Tegelikult on aga "*root* kasutaja õigused andmebaasis kõige suuremad ja need õigused võiks jätta vaid süsteemi ülesseadjale, sest nende õigustega võib süsteemis palju kurja korda saata.

Nagu nõuete analüüsis on kirjutatud, et võõrad kasutajad ligi ei pääseks, on ka oluline, et brauseris oleks ära keelatud kasutajanimede ja paroolide salvestamine.

Vastasel juhul on võimalik kas kasutajanimi ja parool järgi uurida või lihtsamal juhul neid niikaua kasutada, kuni keegi brauseri mälu ära tühjendab.

Üks oluline omadus, mis antud laohaldussüsteemil on, et parooli ei salvestata aadressiribal. Vastasel korral oleks võimalik süsteemile peale seda, kui keegi seal töö on ära lõpetatud, lihtne ligi pääseda, vaadates brauseri ajaloost külastatud linke ja uurides sealt välja süsteemi kasutaja ja parooli. Parooli hoitakse ainult muutujas `$_SESSION`, mis on veebikasutajale nähtamatu objekt.

Need oleks aspektid, millega antud süsteemi turvalisust puudutades võiks arvestada.

5. Lahenduse struktuur

Süsteemi failid on jagatud antud konkreetse teemaga seotud kataloogidesse. Näiteks klientide lehe failid on paigutatud kataloogi *localhost/kliendid* ja töötajad on paigutatud kataloogi *localhost/tootajad*. Veel on olemas ka näiteks CSS stiilifail (*stiil.css*) ja keele Javascript teek JQuery (*jquery.js*) ning veel mõned süsteemifailid, mis on paigutatud lahenduse juurkataloogi. JQuery teek on internetist allalaetav, aga juhuks, kui internetti ei ole, on see jäetud ka süsteemi failide hulka.

Süsteemi failide paigutus näeb välja järgmine:

Kataloogid/failid – Kataloogi failid

arve *– arve.php, arved.php, arvekustutamine.php, arvemakstuksmuutmine.php, arveteotsing.html, Kaup.php, kustuta.php, lisaarve.php, lisamine.php, muuda.php, otsingutulemused.php, salvestamine.php, uuendahinnad.php, uuendamakstud.php*
docs *– dokumendid.php, download.php, faililisamine.php, kustutafail.php, uploaddocument.php*
importijad *– ilisamine.php, importijaandmed.php, importijad.php, kustutai.php, lisai.html , muudaimportijat.php*
kaubad *– kaubad.php, kustuta.php, lisa.php, lisamine.php, muuda.php, muutmine.php, otsing.html, paring2.php, paring3.php, paring.php, server.php, tulemused.php, vaata.php,*

votapilt.php

kliendid *– kasklientolemas.php, kliendid.php, kmuutmine.php, kustutak.php, lisak.php, lisaklient.html, muudak.php, vaatak.php*

planeerija *– muutmine.php, planeerija.php*

s2tted *– kaibemaksuprotsent.php, madallaoseis.php, mintooteidarves.php, s2tted.html salvesta.php, soodprotsent.php*

soodus *– salvesta.php, soodeemaldamine.php, soodlisamine.php, soodprotsent.php, soodustused.php*

stat *– parimadtooted.php, statistika.html, suuremadostjad.php*

system *– info.php*

tootajad *– kastootajaolemas.php, kustutat.php, lisat.php, lisatootaja.html, muudat.php, tmuutmine.php, tootajad.php, vaatat.php*

tootenimed *– kustutatn.php, lisatn.php, tnlisamine.php, tootenimed.php*

tootjad *– kustutat.php, lisat.html, muudatootjat.php, tlisamine.php, tootjaandmed.php, tootjad.php, vaatat.php*

varad *– kasvaraolemas.php, kustutav.php, lisav.php, lisavara.html, muudav.php, varad.php, vmuutmine.php*

adminmenu.php, banner.html, haldus.php, index.php, jquery.js, kasutajamenu.php, logo.bmp, stiil.css, vajutuslogisisse.php, valjalogimine.php

6. Kasutusjuhend

Eeldame, et kasutajal on süsteemi sisenemiseks kas brauser Internet Explorer, Firefox või Opera. Laohaldussüsteemi aadressiks on *localhost/index.php*. Antud leht suunab kasutaja edasi lehele *localhost/haldus.php*. Antud leht teatab, et selle lehe turvalisuse sertifikaat pole loodud turvalise organi poolt. Sellegipoolest tuleks esmakordsel süsteemi sisenemisel kilkkida järgnevalt "*Continue to this website(not recommended)*". Kui see on tehtud, siis võib kindel olla, et veebilehel kasutatakse turvalist HTTPS protokolli. Selleks, et siseneda, peab kasutajal olemas olema ka kasutajanimi ning parool, milleks antud süsteemi tavakasutaja puhul hetkel on "*kasutaja*"

ning “*teineparool*”. Süsteemi sisenemisel on taas aadressiks *localhost/index.php.* Esmasisenemisel on all parempoolses raamis avatud toodete nimekirja lehekülg ning all vasakpoolses raamis on avatud menüü. Kui menüüst elemente valida, ilmuvad nendele vastavad leheküljed all parempoolsesse raami.

Laohaldussüsteemi tavakasutajal on võimalik valida järgmiseid menüüpunkte (Joonis 3):

1. *Toodete nimekiri*. Selle menüüpunkti all on võimalik lisada tooteid, muuta tooteid, kustutada tooteid ja vaadata tooteid detailsemalt. Toodetega kaasnevad järgmised andmed: toote nimi, kogus, tootja, hind, mis on kohustuslikud ja ostu hind, importija, parim enne, kaal, mõõtmed ja pilt, mis on valikulised. Toodete nimekirjas näidatakse punasega ära tooted, millel on madal laoseis. Tooted, mis on seotud arvega, näidatakse ära tärniga. Kustutada saab ainult neid tooteid, mis pole arvetega seotud.

2. *Toote nimed*. Seal on võimalik lisada ja kustutada toote nimesid, et oleks lihtsam toodete sisestamisel toote nimesid valida. Nimelt on võimalik toote nimesid valida neid sisse trükkides või rippmenüüst valides. Seal näidatakse punaselt ka ära toote nimed, mis on toodetega seotud. Neid kustutada pole võimalik.

3. *Importijad*. Selle punkti all on võimalik lisada, muuta ja kustutada importijaid. Importijaga kaasnevad järgmised andmed: nimi, mis on kohustuslik ja aadress, telefon, e-mail ja link. Importijad, mis on seotud mõne tootega, on tähistatud punaselt. Jällegi neid kustutada pole võimalik.

4. *Tootjad*. Tootjate puhul on võimalik neid lisada, muuta ja kustutada. Tootjaga kaasnevad järgmised andmed: nimi, mis on kohustuslik ja aadress, telefon, e-mail ning link. Punasega on tähistatud toodetega seotud tootjad. Neid ei ole võimalik kustutada, kui toodete alt vastavat toodet kustutatud ei ole.

5. *Toodete otsing*. Selle menüüpunkti all saab toote nime, tootja ja importija järgi otsida olemasolevaid tooteid. Otsing toimub selliselt, et võetakse arvesse otsingusse pandud esimesi täheühendeid. Näiteks, kui on otsingusse pandud importijaks ‘*lä*’, siis otsitakse kõiki tooteid, kus on importijaks “*lä*”-ga algav importija. Kui otsingusse ei ole pandud ühtegi täheühendit, siis antakse vastuseks, et tegemist on tühiotsinguga.

6. *Töötajate nimekiri*. Töötajaid on võimalik lisada, muuta ja kustutada. Töötajatega kaasnevad järgmised andmed: nimi, aadress, telefon, e-mail, muu

kontaktinfo ja kas töötaja omab autot. Töötaja sisestamisel peab olemas olema nimi ning peab arvestama, et see peab olema kahesõnaline ning peab olemas olema ka mingi kontaktinfo.

7. *Klientide nimekiri*. Kliente on võimalik lisada, muuta ja kustutada. Kliendiga kaasnevad järgmised andmed: nimi, telefon, e-mail ja muu kontaktinfo. Kliendi sisestamisel peab olemas olema vähemalt nimi. Kliente, kes on arvetega seotud, kustutada ei saa.

8. *Arvete nimekiri*. Arveid on võimalik lisada, muuta ja kustutada. Arve lisamisel peavad kindlasti olemas olema klient, minimaalne kogus ostetavaid tooteid ning maksmise aeg. Juhul, kui on vähem, kui minimaalne kogus ostetavaid tooteid, siis kuvatakse veateade. Arves vajaminevat minimaalset kogust ostetavaid tooteid saab muuta administraator menüüelemendi "*Sätted*" all, millest on juttu edaspidi. Arvetel on olemas ka soodustused. Soodustusega saab tooteid osta klient, kes on lisatud soodustuse saajate listi. Juhul, kui arves on määratud maksmise ajaks "*hiljem*", siis saab hiljem selle arve arvete nimekirja lehel makstuks muuta.

9. *Arvete otsing*. Seal on võimalik otsida kliendi nime, toote nime, tootja, importija ja alguse ning lõpukuupäeva järgi arveid. Näiteks, kui märkida otsingus kliendi nimeks "*Pi*", siis on võimalik näha tulemusena kõiki arveid, kus kliendi nimi algab tähekombinatsiooniga "*Pi*". Näiteks, kui on olemas klient nimega "*Piret*", siis on võimalik näha kõiki tema arveid.

10. *Sätted*. Selle punkti all on võimalik tavakasutajal näha süsteemi administraatori poolt määratud sätteid. Nendeks säteteks on toodete hulk, millel on madal laoseis; minimaalselt tooteid arves; soodustuste protsent ja käibemaksu protsent.
Kui mingi toode on madala laoseisuga, siis näidatakse seda toodet punaselt. Arvetesse ei ole võimalik lisada tooteid vähem, kui minimaalne toodete arv arves ette näeb. Klientidel, kellele on lisatud soodustused, on võimalik soodustuste protsendi võrra odavamalt tooteid osta. Toodete ostmisel arvestatakse ka käibemaksu.

11. *Planeerija*. See on kalendri moodi ülestähenduste tegemise süsteemi osa. Planeerijal on näidatud algselt käesolev kuu ja tärniga käesolev kuupäev. Planeerijas on võimalik liikuda kuupäeva peale ja teha selles märkmeid. Kui mingil kuupäeval on tehtud märge, siis kuvatakse seda kuupäeva paksemalt. Tähistustega "<<" ja ">>" on vastavalt

võimalik liikuda üks kuu tagasi ja üks kuu edasi.

12. *Soodustused*. Soodustuste all on võimalik klientidele soodustusi lisada ja neid eemaldada. Soodustuste protsenti on võimalik määrata administraatoril oma enda menüüpunkti "*Sätted*" all.

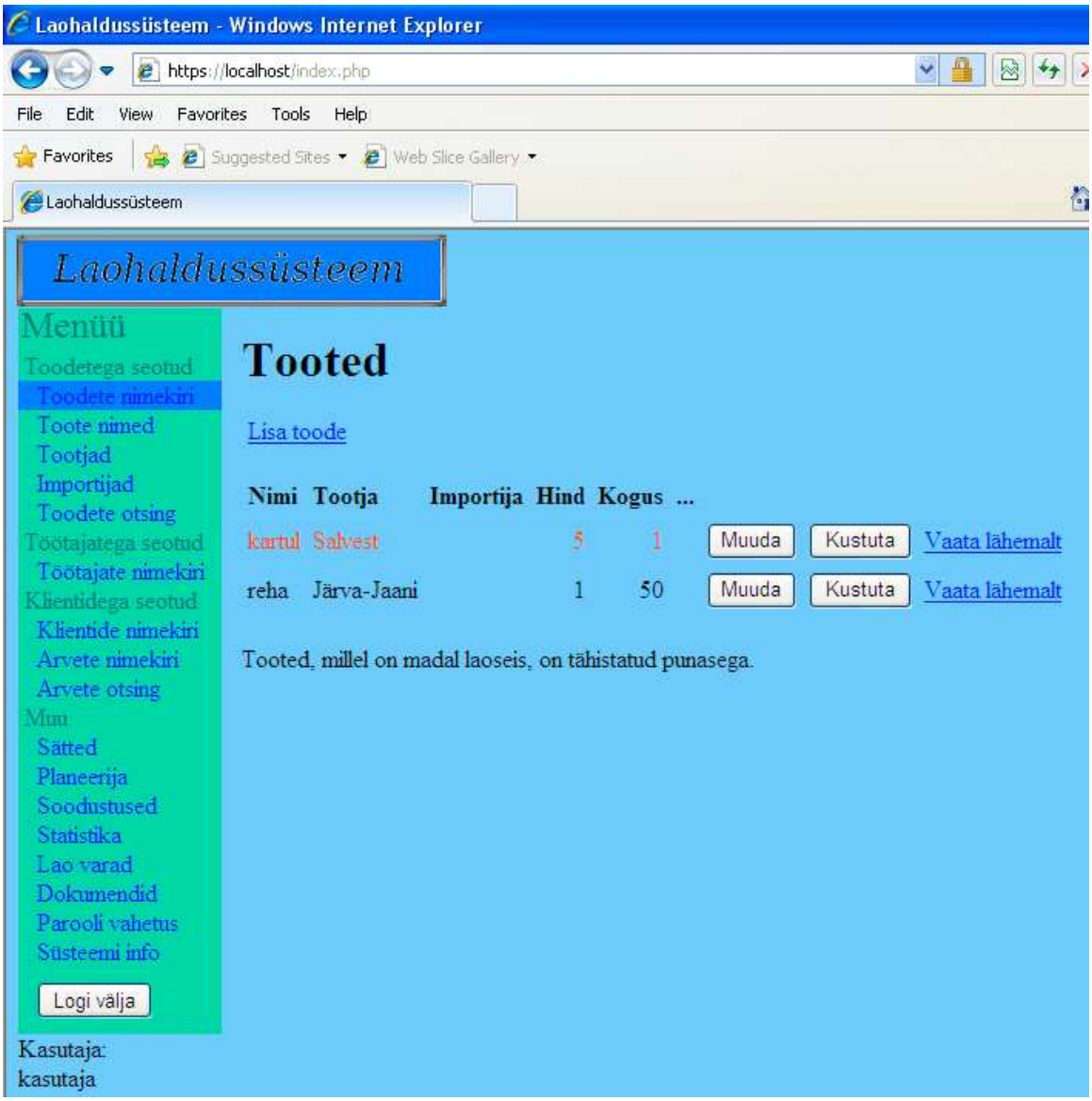
13. *Statistika*. Statistika all on olemas suurema summa eest tooteid ostnud kliendid, suurema läbimüügiga tooted ning kõige aktiivsemad süsteemi kasutajad. Kliente, kes tooteid ostnud pole, nimekirjas ei kajastata. Samuti ei kajastata tooteid, millel läbimüük puudub. Nii suurema summa eest tooteid ostnud klientide puhul, kui suurema läbimüügiga toodete puhul näidatakse summat koos käibemaksuga. Käibemaksu on võimalik muuta administraatoril.

14. *Lao varad*. Neid on võimalik lisada, muuta ja kustutada. Koguseks võib olla tükk, kilogramm, liiter või mõni muu koguse tüüp.

15. *Dokumendid*. Laohaldussüsteemis on võimalik ka laoga seotud dokumente andmebaasi laadida ja neid sealt vaadata ning kustutada. Maksimaalne üleslaetava dokumendi suurus on 5MB. Dokumentide nimekirjas näidatakse ka tüüp, mis liiki dokumendiga on tegemist ning dokumendi suurus baitides.

16. *Parooli vahetus*. Selle punkti all on võimalik süsteemi sisseloginud kasutaja parooli muuta.

17. *Süsteemi info*. Selles on võimalik näiteks "*system*" all ära näha, mis operatsioonisüsteemi kasutab laohaldussüsteemi server ning mis tüüpi arvutit ta kasutab. Samuti on näha Apache serveri ja PHP versioon sektsiooni "*Apache Environment*" alampunti "*SERVER_SOFTWARE*" all. Peale selle saab näha andmebaasisüsteemi MySQL versiooni sektsiooni "*mysql*" alampunkti "*Client API version*" alt. Peale sektsiooni "*HTTP Headers Information*" on ära näha kõik moodulid, mida Apache server veel kasutab, kuni sektsioonini "*Additional Modules*".



Joonis 3. Laohaldussüsteem tavakasutaja vaates.

Administraatoril on siis õigus veel lisaks nendele menüüpunktidele muuta punkti "*Sätted*" all süsteemi sätteid. Vaikimisi on need sätted järgmised: toodete hulk, millel on madal laoseis: 3; minimaalne toodete arv arves: 1; : soodustuste protsent: 2,1 ja käibemaksu protsent: 20 (Joonis 4).

Joonis 4. Laohaldussüsteem administraatori vaates.

Peale laohaldussüsteemis töö lõpetamist tuleks kindlasti süsteemist välja logida. Väljalogimine toimub menüü allosas paikneva nupu "*Logi välja*" abil.

7. Süsteemi katsetus

Süsteemi testimiseks genereeriti PHP faili “*TestingSystem.php*” abil 1500 suvalise nime, tootja, hinna ja kogusega toodet. Andmete genereerimiseks valiti ainult toote jaoks kohustuslikud väljad. Operatsioon võttis aega kaks minutit ja 3 sekundit. Selle tulemusena genereeriti 1183 unikaalset toodet, sest 317 toodet olid genereerimisel korduvad ja neid ei arvestatud.

Laohaldussüsteemis toodete nimekirja lahti võtmisel ilmnes, et ei oldud arvestatud sellega, et väga pika toodete nimekirja puhul läheb laadimiseks palju aega ja see võib kasutajale tüütuks muutuda. Kui see nüanss oli avastatud, siis tehti süsteemis suuremate nimekirjade puhul lehekülgedeks jaotamine selliselt, et nimekiri muudeti osadeks, kus igasse ossa kuulub maksimaalselt 100 kirjet. Suvaliste kirjetega nimekirjade kuvamisel süsteemis muid puudusi ei esinenud.

Kokkuvõte

Antud bakalaureusetööga saavutati järgmised tulemused:

1. Valmis dünaamiline ja kiire laohaldussüsteem, milles on võimalik hallata tooteid, tootjaid, importijaid, töötajaid, kliente, arveid, soodustuste saajaid, lao varasid ja laoga seotud dokumente.
2. Süsteem kajastab ka statistikat, kus on võimalik näha suurema summa eest tooteid ostnud kliente, suurema läbimüügiga tooteid ja aktiivseimaid kasutajaid.
3. Laohaldussüsteem täidab ka turvalisuse nõudeid. Kogu informatsioon, mis kasutaja ja serveri vahel liigub, on krüpteeritud, nii et neid andmeid pole võimalik kõrvalistel isikutel pealt kuulata.
4. Laohaldussüsteem on lihtsasti kasutatav igaühel, kes veebibrauserit kasutada oskab.
5. On kasutatud tänapäevaseid graafika loomise tehnoloogiaid ja tänapäevaseid programmeerimise tehnoloogiaid.

Antud bakalaureusetöö tulemusena valmis ladudele mõeldud haldussüsteem, mida on võimalik soovi korral edasi arendada, kuid mis peaks olema laialdaselt kasutatav ka ilma selleta. Selline süsteem loob kindasti parema korra laopidamises ja võimaldab lihtsamini laomajanduses muudatusi teha.

Süsteemis kasutatavad tehnoloogiad olid peale PHP programmeerimiskeele järgmised: MySQL, Javascript, HTML5, CSS, HTTPS protokoll ja JQuery. Eraldi ära märkimist vääriksid kindlasti ka objektorienteeritud programmeerimine PHP keeles ja AJAX kasutamine. Kõigi nende tehnoloogiate koostoimimise tulemusena valmis turvaline, dünaamiline ja kiire laohaldussüsteem.

Üheks süsteemi võimalikuks edasiarenduseks võiks olla näiteks teistesse keeltesse tõlkimine. Teine edasiarendus võiks olla näiteks toodete tellimise võimalus internetist kliendile. See aga eeldab, et veebiserver on ülemaailmses veebiruumis registreeritud, aga selle registreerimine võib laole kulusid tekida.

Secure warehouse management system programmed in PHP

Bachelor Thesis (6 ECTS)

Kristjan Robam

Summary

The aim of the present thesis was to create a simple warehouse management system in PHP. The reason of this was that there is a relatively big demand for this kind of systems at the moment.

In addition to PHP in the bachelor thesis were also used technologies like MySQL, Javascript, HTML5, CSS, HTTPS, AJAX and JQuery. As a web server Apache 2.2 was used.

The practical results of this bachelor work are the following:

1. The dynamic and fast warehouse management system, in which is possible to manage products, manufacturers, importers, workers, clients, bills, beneficiaries, warehouse properties and warehouse documents has been developed.

2. The warehouse system enables also to view the statistics – it is possible to see the most sold products and the clients who have bought the most from the warehouse.

3. In the warehouse system information moves between the user and the server in a crypted way.

4. The system is easily usable, so it does not expect the knowing of programming languages and is usable to everybody who can use a web browser.

5. The created warehouse management system uses modern graphics and programming technologies.

Viited

[1] PHP programmeerimiskeelest.
http://en.wikipedia.org/wiki/PHP (Viimati vaadatud 23. juuni 2012)

[2] PHP õpetus.
http://www.w3schools.com/php/php_intro.asp (Viimati vaadatud 23. juuni 2012)

[3] HTML5 õpetus.
http://www.w3schools.com/html5/html5_intro.asp (Viimati vaadatud 23. juuni 2012)

[4] CSS õpetus.
www.w3schools.com/css (Viimati vaadatud 23. juuni 2012)

[5] JQuery õpetus.
http://www.w3schools.com/jquery/default.asp (Viimati vaadatud 23. juuni 2012)

[6] AJAX tehnoloogiast.
http://en.wikipedia.org/wiki/Ajax_(programming) (Viimati vaadatud 23. juuni 2012)

[7] AJAX tehnoloogia õpetus.
http://www.w3schools.com/ajax/default.asp (Viimati vaadatud 23. juuni 2012)

[8] HTTPS protokoll.
http://en.wikipedia.org/wiki/HTTP_Secure (Viimati vaadatud 23. august 2012)

[9] PHP objektorienteeritud programmeerimisest.
http://www.techotopia.com/index.php/PHP_Object_Oriented_Programming (Viimati vaadatud 23. juuni 2012)

[10] XML keelest.
http://en.wikipedia.org/wiki/XML (Viimati vaadatud 23. juuni 2012)

[11] HTTPS installeerimine.
http://www.neilstuff.com/apache/apache2-ssl-windows.htm (Viimati vaadatud 23. juuni 2012)

Lisad

Lisa 1. PHP programmeerimiskeelest

PHP on serveripoolne skriptimiskeel, mis on loodud aastal 1995 Rasmus Lerdorfi poolt ja mida on arendatud tänapäevani. Esimene PHP tõsiseltvõetav versioon kandis nime PHP 3. Tänaseks on kasutusel versioon PHP 5, millel on edasijõudnud objekt-orienteeritus ja muidki häid omadusi. Hetkel on ka arendamisel täiesti uudne PHP 6, mis pole veel piisavalt hea, et seda tõsiseltvõetavaks nimetada, kuid selle kallal käib hetkel tõsine töö. Üheks suureks plussiks PHP puhul on see, et ta on vabavara. Peale selle töötab ta mitmetel platvormidel nagu nt Windows, Linux, Unix jne ja on kokkusobiv mitmete veebiserveritega nagu nt. Apache, IIS jne.

PHP ülesseadmisest operatsioonisüsteemis Windows XP on juttu punktis "*Vajamineva tarkvara installeerimise juhend*" (Lisa 2).

PHP failinimede laienditeks on kas "*.php*", "*.php3*" või "*.phtml*". Kõige sagedamini kasutatakse laiendit "*.php*". Antud laohaldussüsteemi puhul on ka kasutatud ainult laiendit "*.php*".

Kõige lihtsam PHP programm näeks välja järgmine:

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

See kuvab kasutajale ekraanile teksti "Hello World!".

Võtame vaatluse alla PHP süntaksi. Muutujad algavad märgiga "`$`" ja koodi laused lõpevad semikooloniga. Näiteks defineerib järgmine PHP lause muutuja nimega "summa" väärtusega "null": "`$summa=null;`". Muutujaid kasutatakse arvuliste väärtuste, teksti või objektide hoidmiseks.

Keeles PHP kasutatakse kommenteerimise alustamiseks märki "`/*`" ja lõpetamiseks märki "`*/`". Üherealise kommentaari puhul kasutatakse aga märki "`//`". PHP keeles on kasutatavad näiteks mõlemad kommenteerimise viisid nagu

```
/*
tere hommikust
```

`*/`

ja `//tere hommikust`.

PHPs defineeritakse funktsioone järgmiselt:

```
function minuFunktsioon($param1,$param2,...)
{
 // funktsiooni kood
}
```

Selles funktsioonis on muutujad `$param1` ja `$param2` funktsiooni parameetrid ehk argumendid. Parameetrid võib ka ära jätta, kui nendeks vajadust ei ole.

Programmeerimiskeelel PHP on neli muutujate skoopi: lokaalne, globaalne, staatiline ja parameetriline.

Et kasutada globaalse skoobiga muutujat funktsioonis, tuleks kasutada võtmesõna "`global`", näiteks: "`global $m;`". Staatilise skoobiga muutuja defineeritakse järgmiselt "`static $muutuja=v22rtus;`". Staatilise skoobiga muutuja defineeritakse ainult üks kord. Kui PHP funktsioon, mis defineeris staatilise muutuja, jälle välja kutsuda, siis jätab PHP staatilise muutuja väärtuse meelde.

Keeles PHP ühendatakse tekste punktiga. Stringide pikkust on võimalik määrata stringifunktsiooniga `strlen()`. Näiteks annab käsk "`echo strlen("tere tulemast");`" väärtuseks 13. Alamstringi asukoha määramiseks kasutatakse keeles PHP funktsiooni `strpos($string, $alamstring)`. Näiteks annab `strpos("tere tulemast", "tulemast")` väärtuseks 5.

Keeles PHP on olemas järgmised aritmeetilised operaatorid "`+`", "`-`", "`*`", "`/`", "`%`" ja "`-` märk". Lisaks on veel olemas ülesannete operaatorid nagu näiteks "`x=y`", "`x+=y`", mis on võrdne järgmise avaldisega "`x=x+y`" ja "`x-=y`", mis on võrdne järgmise avaldisega "`x=x-y`" jne. Siis on veel olemas operaator "`x++`", mis liidab x-le juurde 1. Peale nende on olemas veel võrdlusoperaatorid nagu "`==`", mis tähendab võrdsed, "`===`", mis tähendab identsed, "`!=`" ja "`<>`" mis tähendavad mitte võrdsed, "`!==`" mis tähendab mitte identsed, "`>`", "`>=`" jne. Veel on olemas ka loogilised operaatorid nagu näiteks "`and`", "`or`", "`&&`", "`||`" ja "`!`". Loomulikult on olemas veel operaatoreid, aga neid antud töös ei käsitleta.

Muidugi on programmeerimiskeeles PHP olemas ka tingimuslause, mida saab kasutada järgmiselt: `if(tingimus) tegevus; else teinetegevus;`.

Edasi on olemas veel *switch* direktiiv, mis näeb välja järgmine:

```
switch (n)
 {
 case label1:
   kood, mis käivitatakse, kui n=label1;
   break;
 case label2:
   kood, mis käivitatakse, kui n=label2;
   break;
 default:
   kood, mis käivitatakse, kui n pole ei label1 ega label2;
 }
```

Keeles PHP on võimalik luua ka numbrilisi, assotsiatiivseid ja mitmedimensionaalseid järjendeid.

Numbrilisi järjendeid on saab keeles PHP luua näiteks järgmiselt: `$autod=array("Ford","Mustang","Toyota");.`
Järjendi teise elemendi saab kätte järgiselt: `echo $autod[1];`.

Assotsiatiivseid järjendeid on võimalik keeles PHP luua järgmiselt: `$kanalid = array(“Discovery”=>14,”CNN”=>10,”ETV”=>1,”Kanal 2”=>5);`. Näiteks “ETV”-le vastav kanali number on võimalik järjendist kätte saada järgmiselt: `echo $kanalid[‘ETV’];`.

Mitmedimensionaalseid järjendeid on võimalik luua keeles PHP järgmiselt: `$tiimid = array(“Hakkajad”=>array(“Peeter”, “Tõnis”, “Heiko”), “Võitjad”=>array(”Jaan”, “Oliver”, “Kristo”));`. Näiteks tiimi “Hakkajad” teine liige on võimalik kätte saada järgmiselt: `echo $tiimid[‘Hakkajad’][1];`.

Loomulikult on programmeerimiskeeles PHP olemas ka tsüklioperaatorid. *While* ja *for* tsüklid näevad tavapäraselt välja järgmised:

```
while(tingimus)
{
  tegevused;
}
```

ja

```
for(initsialiseerimine; tingimus; järeltegevus)
{
  tegevused;
}
```

Et PHP võiks kasutajalt ka dünaamiliselt sisendit saada, selleks on olemas ka vormide ja PHP vahelise suhtluse võimalus. Selleks määratakse keele HTML vormis ära, millisele PHP failile temas olevad muutujad on vaja anda, nagu näiteks järgmises vormis:

```
<form action="tervitus.php" method="post">
 Nimi: <input type="text" name="nimi" />
 <input type="Sisesta" />
 </form>
```

Kui kasutaja klikib nuppu "*Sisesta*" siis läheb brauser lehele "*tervitus.php*" ja võtab vormilt kaasa kõik seal olevad muutujad. Antud näite korral on võimalik saada vormilt sisend kastist "*nimi*" ja see saadakse PHP failis kätte muutujast `$_POST["nimi"]`. Selle näite korral on võimalik näiteks keeles PHP kuvada kirje "Tere Jaan!", kui kastist "*nimi*" on saadud sisendiks "Jaan".

Meetodite *get* ja *post* vahe vormilt saatmisel on selles, et *get* meetodiga saadetud info on nähtav kõigile, aga *post* meetodiga saadetud info ei ole ning *get* meetodiga saadetaval info hulgal on piirangud, aga *post* meetodiga saadetaval info hulgal need puuduvad.

Keeles PHP on võimalik kasutada teise PHP faili muutujaid, funktsioone ja objekte lisades selleks faili algusesse rea "`include 'failinimi';`" või "`require 'failinimi';`". *Include* ja *require* vahe on selles, et *require* annab vea korral fataalse vea (*E_COMPILE_ERROR*) ja peatab skripti, aga *include* annab vea korral ainult hoiatuse (*E_WARNING*) ja skript jätkab töötamist.

Keeles PHP luuakse objekte sarnaselt teistele keeltele nagu näiteks Java. Olgu meie PHP objektide loomise näiteks järgmine:

```
<?php
class Raamat {
     public $nimi;
     public $pealkirjad;
     public function __construct($n) {
         $this->nimi=$n;
```

```
          $this->pealkirjad=array();
     }
     public function lisapealkiri($pealkiri) {
           if(!in_array($pealkiri,$this->pealkirjad))
array_push($this->pealkirjad,$pealkiri);
     }
     public function __toString() {
      $var="Raamatu nimi: ".$this->nimi.". ";
         $var=$var."Pealkirjad: ";
      for($i=0; $i<count($this->pealkirjad); $i++) {
           if($i!=count($this->pealkirjad)-1)
$var=$var.$this->pealkirjad[$i].", ";
           else $var=$var.$this->pealkirjad[$i].".";
         }
      return $var;
     }
}
$raamat1=new Raamat("Ellujäämise õpetus III");
$raamat1->lisapealkiri('Ohtlik tarkus');
$raamat1->lisapealkiri("Iseenda leidmine");
echo $raamat1;
?>
```

See kood loob objekti nimega *Raamat*, millele antakse ka nimi. Siis lisatakse sellele raamatule ka raamatu pealkirjad ning seejärel kuvatakse raamatu nimi ning selle pealkirjad objekti *Raamat* meetodi `toString` abil. Keeles PHP on olemas ka privaatsed objekti muutujad, neid tähistatakse selles nagu paljudes muudeski programmeerimiskeeltes märgiga "*private*". Kui näiteks mõnes klassis on defineeritud privaatne muutuja, siis väljaspoolt objekti sellele ligi ei pääse. Keeles PHP on olemas ka pärilus ja ülekatmine. Kui meil on näiteks vaja täpsustada klassi *Raamat*, siis on seda võimalik teha järgmiselt:

```
<?php
class Opik extends Raamat {
public $klass;
public function __construct($n, $k) {
         $this->nimi=$n;
```

```
        $this->klass=$k;
        $this->pealkirjad=array();
}
public function __toString() {
        $var="Klass: “.$this->klass. ". ";
        $var=$var."Raamatu nimi: ".$this->nimi.". ";
        $var=$var."Pealkirjad: ";
     for($i=0; $i<count($this->pealkirjad); $i++) {
          if($i!=count($this->pealkirjad)-1)
$var=$var.$this->pealkirjad[$i].", ";
          else $var=$var.$this->pealkirjad[$i].".";
               }
        return $var;
}
}
$opik=new Opik("Matemaatika","5");
$opik->lisapealkiri("Pythagorase teoreem");
$opik->lisapealkiri("Algarvud");
echo $opik;
?>
```

Selle koodi käigus luuakse objekt *opik* tüübist *Opik* pealkirjaga “Matemaatika” ja klassiga, kellele see ette nähtud on, “5”. Uus objekt pärib raamatu isendimeetodi `lisapealkiri` ning uuele objektile lisatakse pealkirjad “Pythagorase teoreem” ja “Algarvud” ning objekti *opik* kuvamiseks kasutatakse ülekatte tõttu klassi *Opik* `toString` meetodit.

Lisa 2. Vajamineva tarkvara installeerimise juhend

Esiteks, et süsteem töötaks, peaks olema arvutisse eelnevalt installeeritud Apache server (soovitavalt 2.2.22) koos programmeerimiskeelega PHP (soovitavalt 5.3.10) ja andmebaasikeelega MySQL (soovitavalt 5.5.20). Järgnevalt kirjeldame, kuidas neid programme operatsioonisüsteemis Windows XP installeerida.

Esmalt tuleb installeerida Apache server. Selle saab kätte näiteks lehelt http://httpd.apache.org/ nime alt “*Win32 Binary including OpenSSL 0.9.8t (MSI*

Installer)”.

Seda installeerides tuleks panna võrgu domeeniks: *localhost*, serveri nimeks samuti *localhost*, administraatori e-mailiks enda e-maili aadress. Pordi numbriks tuleb panna 80. Peale seda tuleks võtta *typical* installeerimise valik ja kõigi valikutega nõus olla. Siis tuleks avada start menüüst “*Edit the Apache httpd.conf Configuration File*” ning muuta *httpd.conf* faili järgmiselt: *Documentroot* rida tuleb muuta järgmiseks *DocumentRoot "C:/public_html"*. Siis tuleks ülejärgmine *Directory* rida muuta järgnevaks <*Directory "C:/public_html"*> ning ongi Apache installeeritud. Peale seda tuleks tegevused loomulikult ka salvestada.

PHP installeerimiseks tuleks algul see alla laadida. Selle saab kätte lehelt http://windows.php.net/download, kui võtta kõige uuema *thread safe* versiooni zip fail (näiteks *php-5.4.4-Win32-VC9-x86.zip*). See tuleks lahti pakkida kataloogi “*C:\php*”.
Siis tuleks failist “*php.ini-development*” teha koopia ja see ümber nimetada siis failiks “*php.ini*”. Seejärel tuleks see fail mõnes tekstiredaktoris lahti võtta. Siis tuleks vaadata, et rea “*short_open_tag*” eest oleks ära võetud semikoolon ja et see rida näeks välja järgmine: “*short_open_tag = Off*”. Samamoodi tuleks vaadata, et oleks olemas rida “*display_errors = On*”. Veel tuleks vaadata, et rea *extension_dir* eest oleks ära võetud semikoolon ning et see rida näeks välja järgmine: “*extension_dir = "C:\php\ext"*”. Siis tuleks veel ridade “*extension=php_mysql.dll*” ja “*extension=php_mysqli.dll*” eest ära võtta semikoolonid. Seejärel tuleks fail ka salvestada. Järgnevalt tuleks lahti võtta Apache serveri konfigureerimise fail nimega *httpd.conf*. Sealt tuleks vaadata, et, et faili lõpus oleks rida “*LoadModule php5_module "c:/php/php5apache2_2.dll"*”. Seejärel tuleks vaadata, et sektsioonis “*<IfModule mime_module>*” oleks ka rida “*AddType application/x-httpd-php .php*”. Veel tuleks faili lõppu lisada rida “*PHPIniDir "c:/php"*”. Seejärel tuleks fail salvestada ja Apache server uuesti tööle panna. Pärast seda võib vabalt igasuguseid PHP koode käivitada.

Et antud serveris oleks võimalik ka andmebaasi MySQL kasutada, peaks selle samuti serverisse installeerima. Viimase saab kätte näiteks lehelt http://dev.mysql.com/downloads/mysql/ laadides sealt alla versiooni “*Windows (x86, 32-bit), MSI Installer*”. Selle kiiremaks allalaadimiseks tuleks valida “*No thanks, just take me to the downloads!*” peale mida võib valida juba parima koha kust seda alla laadida.

Peale allalaadimist tuleb see installeerida, mida tehes tuleks esiteks parameetriks *setup type* valida *typical*. Siis tuleks valida "*Launch the MySQL Instance Configuration Wizard*". Sealt edasi tuleks valida "*Detailed Configuration*" ning serveri tüübiks tuleks panna "*Developer Machine*". Database kasutuseks tuleks panna "*Multifunctional Database*" ning *InnoDB datafile'*i asukohaks tuleks panna "*C:\MySQL InnoDB Datafiles*". Siis tuleks panna ligikaudseks samaaegsete ühenduste arvuks 20, millele vastab valik "*Decision Support (DSS)/OLAP*". Peale seda tuleks niikaua *next*i vajutada, kui tuleb ette valik "*install as Windows Service*". Sinna tuleks linnuke panna, kuid linnuke tuleks ära võtta kasti "*launch the MySQL Server automatically*" eest. Peale seda tuleks valida *root* kasutaja parool (meie süsteemi puhul olgu selleks „*diivan*") ning vajutada nupule "*Execute*".

Nüüd tuleks minna "*Start->Control Panel->Administrative Tools->Services*" ja avada MySQL. Kui aken on avanenud, siis tuleks valida "*Stop*" ja seejärel "*OK*".

Nüüd tuleks luua töölauale MySQL serveri käivitamiseks link. See tuleks teha järgmiselt. Tuleks vajutada hiire paremat klahvi ja valida *New* ning *Shortcut*. Siis tuleks panna lahtrisse "*Type the location of the item*" järgmine tekst: "*"C:\Program Files\MySQL\MySQL Server 5.5\bin\mysqld.exe" --defaults-file="C:\Program Files\MySQL\MySQL Server 5.5\my.ini" --general_log=1 --general_log_file= "C:\localhost.log""*". Siis tuleks valida "*Next*" ja seejärel "*Finish*". Sellega on loodud MySQL serveri käivitamiseks link nimega "*mysqld.exe*", mille abil käivitatakse MySQL server nii, et salvestatakse ka süsteemi logi (faili "*C:\localhost.log*"). Kui MySQL serverit ei käivitata sealt, siis võivad mõned kasutajate tegevused mitte logituks jääda ning siis pole võimalik välja selgitada, milline kasutaja on süsteemi valesti kasutanud.

Veel tuleks installeerida veebiserverile HTTPS protokoll. Selle installeerimisest on kirjutatud ka aadressil http://www.neilstuff.com/apache/apache2-ssl-windows.htm, kuid kirjeldame siin seda lühidalt.

Esmalt tuleks alla laadida viimane OpenSSL. Selle saab kätte näiteks lehe http://slproweb.com/products/Win32OpenSSL.html lingilt http://slproweb.com/download/Win32OpenSSL_Light-1_0_1c.exe. Eeldame nüüd, et see installeeritakse järgnevalt kataloogi "*C:\OpenSSL-Win32*" ning valikuks "*copy OpenSSL DLLs to*" pannakse "*The OpenSSL binaries(/bin) directory*". Kui see on installeeritud,

siis on veel vaja faili *openssl.cnf*, mille saab aadressilt http://www.neilstuff.com/apache/apache2-ssl-windows.htm lingi alt http://tud.at/programm/openssl.cnf või lingi alt http://www.neilstuff.com/apache/openssl.cnf**.** See tuleb paigutada samasse kataloogi, kuhu OpenSSL on installeeritud.

Vajamineva sertifikaadi valmistamine toimub järgmiselt. Tuleb minna käsureale aadressile "*C:\OpenSSL-Win32\bin*" ja sisse trükkida rida: "*openssl req -config ..\openssl.cnf -new -out blarg.csr -keyout blarg.pem*". Järgnevalt küsitakse palju küsimusi, kuid kõik peale *PEM pass phrase* ja *Common Name* võib tühjaks jätta. Parameetriks *Common Name* võib panna nt. "localhost". Järgnevalt tuleks sisestada rida "*openssl rsa -in blarg.pem -out blarg.key*". Peale seda rida "*openssl x509 -in blarg.csr -out blarg.cert -req -signkey blarg.key -days 365*". Sellega on sertifikaat loodud.

Kuna Apache on installeeritud SSL mooduli toega, siis tuleb järgmisena see aktiivseks muuta. Aktiveerimiseks tuleks lahti võtta serveri Apache konfiguratsiooni fail nimega "*apache.conf*". Sealt tuleks võtta eest ära kommentaari märk realt: "`LoadModule ssl_module modules/mod_ssl.so`". Siis tuleks olla kindel, et järgmine koodijupp oleks ka kuskil olemas:

```
<IfModule mod_ssl.c>
    Include conf/extra/httpd-ssl.conf
</IfModule>
```

Edasi tuleks kataloogis "*C:\Program Files\Apache Software Foundation\Apache2.2\conf*" teha kataloog nimega *ssl* ning sinna kopeerida eelnevalt loodud failid "*blarg.key*" ja "*blarg.cert*". Peale seda tuleks lahti võtta fail "*conf/extra/httpd-ssl.conf*" ja muuta sealt seest ära read SSLCertificateFile ja SSLCertificateKeyFile vastavaks:

```
SSLCertificateFile "C:/Program Files/Apache Software
Foundation/Apache2.2/conf/ssl/blarg.cert"
```

ja `SSLCertificateKeyFile "C:/Program Files/Apache Software Foundation/Apache2.2/conf/ssl/blarg.key"`

Veel tuleks ka *DocumentRoot* rida muuta järgmiseks: `DocumentRoot "C:/public_html"`.

Lisa 3. Süsteemi lähtekood

Süsteemi lähtekood asub kaasasoleva CD-plaadi failis "*lahtekood.zip*".

Lisa 4. Süsteemi ülesseadmine

Eeldatakse, et kasutatakse arvutit, mis lubab korraga töötada Apache serveril, PHP programmeerimiskeelel ja MySQL andmebaasikeelel. Arvuti protsessoriga: Pentium 4 1,7GHz, mäluga 256MB ja kõvakettaga 20GB peaks selleks sobima ideaalselt. Operatsioonisüsteemiks eeldatakse olevat Windows XP. Et süsteem toimiks, peaksid olema kõigepealt laohaldussüsteemi failid kopeeritud veebiserveri juurkataloogi, milleks on antud näites "*C:\public_html*". Failide lahtipakkimisel tuleks kustutada või tõsta eraldi kataloog "*systemtest*", kus asub süsteemi testimiseks vajalik PHP fail.

Peale selle tuleks luua süsteemile andmebaas ja kasutajad ning anda kasutajatele õigused ning tuleks sisestada vajaminevad süsteemi tabelid, trigerid ja protseduurid. Kui kasutajale anda root kasutaja õigused, siis võib ta korda saata palju ebavajalikku, näiteks kustutada mõne root kasutaja tähtsa tabeli. Nende tegevuste edukaks sooritamiseks tuleks kõigepealt sisse logida keele MySQL root kasutajaga. *Root* kasutajaga*("root")* on võimalik sisse logida andmebaasikeele MySQL käsureale järgmiselt. Tuleks võtta lahti "*All Programs->MySQL->MySQL Server 5.5->MySQL 5.5 Command Line Client*" ja sisestada parool, mis on serverarvutis MySQLis *root* kasutaja puhul antud juhul „*diivan*".

Kõigepealt tuleks luua süsteemile andmebaas. Selle loomine näeb käsureal välja järgmine: "`CREATE DATABASE minudb;`". Selle andmebaasiga on võimalik siduda süsteemi kasutajad, kelle lisamist kirjeldatakse edaspidi.

Järgmisena tuleks luua süsteemi andmebaasi vajaminevad tabelid. Nende loomiseks tuleks esmalt sisestada andmebaasi kasutamise rida: „`use minudb;`". Siis tuleks võtta kõik järmised tabelid blokki, need kopeerida ja sisestada keele MySQL käsureale.

```
create table kaubanimed(
id int auto_increment primary key,
```

```
nimi varchar(20) not null,
unique key(nimi)
);
create table tootjad(
id int auto_increment primary key,
nimi varchar(20) not null,
aadress varchar(60),
telefon varchar(20),
email varchar(60),
link varchar(60),
unique key(nimi)
);
create table importijad(
id int auto_increment primary key,
nimi varchar(20) not null,
aadress varchar(60),
telefon varchar(20),
email varchar(60),
link varchar(60),
unique key(nimi)
);
create table kaubad(
id int not null auto_increment primary key,
nimi_id int not null,
tootja_id int not null,
importija_id int,
kaal double,
hind double not null,
ostuhind double,
kogus int not null,
mootmed varchar(30),
parimenne date,
pilt mediumblob,
kommentaarid varchar(300),
unique key(nimi_id, tootja_id),
foreign key (nimi_id) references kaubanimed(id),
```

```
foreign key (tootja_id) references tootjad(id),
foreign key (importija_id) references importijad(id)
);
create table tootajad(
id int auto_increment primary key,
nimi varchar(30) not null,
ametinimetus varchar(30) not null,
telefon varchar(20),
email varchar(20),
aadress varchar(60),
muukontaktinfo varchar(20),
omabautot tinyint,
unique key(nimi)
);
create table kliendid(
id int auto_increment primary key,
nimi varchar(20) not null,
telefon varchar(20),
email varchar(60),
muukontaktinfo varchar(20),
unique key(nimi)
);
create table arved(
id int auto_increment primary key,
klient_id int not null,
kuupaev date not null,
makstud boolean not null default true,
foreign key (klient_id) references kliendid(id)
);
create table arvekaupseos(
id int auto_increment primary key,
arve_id int not null,
kaup_id int not null,
kogus int not null,
foreign key (arve_id) references arved(id),
foreign key (kaup_id) references kaubad(id)
```

```
);
create table soodustused(
id int auto_increment not null primary key,
klient_id int not null,
unique key(klient_id),
foreign key (klient_id) references kliendid(id)
);
create table s2tted(
mintooteid int,
madlaoseis int,
soodprotsent double,
kaibemaks double
);
create table upload(
id int not null auto_increment primary key,
nimi varchar(30) not null,
tyyp varchar(30) not null,
suurus int not null,
sisu mediumblob not null
);
create table planeerija(
kuupaev date,
tekst varchar(100),
unique key(kuupaev,tekst)
);
create table varad(
id int not null auto_increment primary key,
nimi varchar(20) not null,
kogus double not null,
unique key(nimi)
);
create index kaubad_indeks on kaubad(id);
create index tootjad_indeks on tootjad(id);
create index varad_indeks on varad(id);
create index kaubanimed_indeks on kaubanimed(id);
create index importijad_indeks on importijad(id);
```

```
create index arvekaupseos_indeks on arvekaupseos(id);
create index arved_indeks on arved(id);
create index tootajad_indeks on tootajad(id);
create index kliendid_indeks on kliendid(id);
create index soodustused_indeks on soodustused(id);
create index upload_indeks on upload(id);
insert into s2tted(madlaoseis) values (null);
```

Peale nende tabelite sisestamist andmebaasi "*minudb*" tuleks ka sisestada süsteemi vajalikud trigerid ja protseduurid. Nende lisamiseks tuleb võtta samuti järgmised read blokki, need kopeerida ja sisestada keele MySQL käsureale.

```
DELIMITER $$
CREATE TRIGGER trigger1
BEFORE INSERT ON s2tted
FOR EACH ROW
BEGIN
  set @var1 = (SELECT COUNT(*) FROM s2tted);
  IF ( @var1> 0) THEN
    set NEW = NULL;
  END IF;
  If(NEW.mintooteid<=0) THEN
    Set NEW = NULL;
  END IF;
  If(NEW.madlaoseis<0) THEN
    Set NEW = NULL;
  END IF;
  If(NEW.soodprotsent<0) THEN
    Set NEW = NULL;
  END IF;
 If(NEW.kaibemaks<0) THEN
    Set NEW = NULL;
  END IF;
END
$$
CREATE TRIGGER trigger2
BEFORE UPDATE ON s2tted
```

```
FOR EACH ROW
BEGIN
  If(NEW.mintooteid<=0) THEN
    Set NEW=NULL;
  END IF;
  If(NEW.madlaoseis<0) THEN
    Set NEW=NULL;
  END IF;
  If(NEW.soodprotsent<0) THEN
    Set NEW=NULL;
  END IF;
 If(NEW.kaibemaks<0) THEN
    Set NEW=NULL;
  END IF;
END
$$
CREATE TRIGGER trigger3
BEFORE INSERT ON tootajad
FOR EACH ROW
BEGIN
  set @var1 = LENGTH(NEW.nimi) - LENGTH(REPLACE(NEW.nimi, '
', ''))+1;
  IF ( @var1 = 1 OR @var1 > 4) THEN
       SET NEW=NULL;
  END IF;
  IF(not isnull(NEW.telefon) || not isnull(NEW.email) || not
isnull(NEW.aadress) || not isnull(NEW.muukontaktinfo)) THEN
   SET NEW=NULL;
  END IF;
END
$$
CREATE TRIGGER trigger4
BEFORE UPDATE ON tootajad
FOR EACH ROW
BEGIN
  set @var1 = LENGTH(NEW.nimi) - LENGTH(REPLACE(NEW.nimi, '
```

```
', ''))+1;
  IF ( @var1 = 1 OR @var1 > 4) THEN
       SET NEW=NULL;
  END IF;
  IF(not isnull(NEW.telefon) || not isnull(NEW.email) || not
isnull(NEW.aadress) || not isnull(NEW.muukontaktinfo)) THEN
   SET NEW=NULL;
  END IF;
END
$$
CREATE TRIGGER trigger5
BEFORE UPDATE ON kaubad
FOR EACH ROW
BEGIN
  set @var1=(select count(*) from arvekaupseos where
kaup_id=OLD.id);
  IF(@var1>0) THEN
     Set @pilt=(SELECT pilt FROM kaubad where id=OLD.id);
     set @id = (SELECT id FROM kaubad where id=OLD.id);
     set @nimi_id = (SELECT nimi_id FROM kaubad where
id=OLD.id);
     set @tootja_id = (SELECT tootja_id FROM kaubad where
id=OLD.id);
     set @importija_id = (SELECT importija_id FROM kaubad
where id=OLD.id);
     set @kaal = (SELECT kaal FROM kaubad where id=OLD.id);
     set @hind = (SELECT hind FROM kaubad where id=OLD.id);
     set @ostuhind = (SELECT ostuhind FROM kaubad where
id=OLD.id);
     set @mootmed = (SELECT mootmed FROM kaubad where
id=OLD.id);
     set @parimenne = (SELECT parimenne FROM kaubad where
id=OLD.id);
     set @kommentaarid = (SELECT kommentaarid FROM kaubad
where id=OLD.id);
    IF (
```

```
NEW.id!=@id||NEW.nimi_id!=@nimi_id||NEW.tootja_id!=@tootja_i
d||NEW.importija_id!=@importija_id||NEW.kaal!=@kaal||NEW.hin
d!=@hind||NEW.ostuhind!=@ostuhind||NEW.mootmed!=@mootmed||NE
W.parimenne!=@parimenne||NEW.kommentaarid!=@kommentaarid||NE
W.pilt!=@pilt&&NEW.pilt!=NULL) THEN
      set NEW=NULL;
    END IF;
  END IF;
  If(NEW.kogus<0) THEN
    Set NEW=NULL;
  END IF;
  If(NEW.kaal<=0) THEN
    Set NEW=NULL;
  END IF;
  If(NEW.hind<0) THEN
    Set NEW=NULL;
  END IF;
 If(NEW.ostuhind<0) THEN
    Set NEW=NULL;
  END IF;
END
$$
CREATE TRIGGER trigger6
BEFORE INSERT ON kaubad
FOR EACH ROW
BEGIN
  If(NEW.kogus<0) THEN
    Set NEW=NULL;
  END IF;
  If(NEW.kaal<=0) THEN
    Set NEW=NULL;
  END IF;
  If(NEW.hind<0) THEN
    Set NEW=NULL;
  END IF;
 If(NEW.ostuhind<0) THEN
```

```
    Set NEW=NULL;
  END IF;
END
$$
CREATE TRIGGER trigger7
BEFORE INSERT ON varad
FOR EACH ROW
BEGIN
  If(NEW.kogus<=0) THEN
    Set NEW=NULL;
  END IF;
END
$$
CREATE TRIGGER trigger8
BEFORE UPDATE ON varad
FOR EACH ROW
BEGIN
  If(NEW.kogus<=0) THEN
    Set NEW=NULL;
  END IF;
END
$$
CREATE TRIGGER trigger9
BEFORE INSERT ON arvekaupseos
FOR EACH ROW
BEGIN
  set @var1=(select count(*) from arvekaupseos where
arve_id=NEW.arve_id AND kaup_id=NEW.kaup_id);
  IF(@var1>0) THEN
       SET NEW=NULL;
  END IF;
  If(NEW.kogus<=0) THEN
    Set NEW=NULL;
  END IF;
END
$$
```

```
CREATE TRIGGER trigger10
BEFORE UPDATE ON arvekaupseos
FOR EACH ROW
BEGIN
If(NEW.kogus<=0) THEN
    Set NEW=NULL;
  END IF;
END
$$
CREATE PROCEDURE koikkaubad(page int)
BEGIN
 DECLARE algus INT;
 set algus=(page-1)*50;
 select kaubanimi,tootja, nimi as importija, hind, kogus
from (select kaubad.id as kauba_id,kaubanimed.nimi as
kaubanimi,tootjad.nimi as tootja, kogus,hind,importija_id
from kaubad join (kaubanimed, tootjad) on
(kaubad.nimi_id=kaubanimed.id AND
kaubad.tootja_id=tootjad.id)) as t1 left join importijad on
(t1.importija_id=importijad.id) ORDER BY kaubanimi,tootja
limit algus, 50;
END$$
CREATE PROCEDURE koikarved(page int)
BEGIN
 DECLARE algus INT;
 set algus=(page-1)*50;
 select arved.id,nimi as klient,if(arved.makstud,
"makstud","maksmata") as makstus,arved.kuupaev from arved
join kliendid on arved.klient_id=kliendid.id order by
nimi,arved.id limit algus, 50;
END$$
CREATE PROCEDURE tooteidv2hem()
BEGIN
set @mintooteid=(select mintooteid from s2tted);
IF(@mintooteid IS NULL) THEN
   set @mintooteid=1;
```

```
END IF;
set @v2hem=(select count(*) from (SELECT arve_id, SUM(kogus)
AS sum_kogus FROM arvekaupseos GROUP BY arve_id
 HAVING sum_kogus <@mintooteid) as t1);
IF(@v2hem>0) THEN
   SELECT "Süsteemis mõnes arves tooteid vajaminevast
kogusest vähem." as V2ljund;
   ELSE
   SELECT “Arvetes pole tooteid minimaalsest vähem.” As
V2ljund;
END IF;
END$$
delimiter ;
```

Veel tuleks luua ka laohaldussüsteemi administraator kasutaja. Selle loomine toimub järgmiselt. Tuleks sisse trükkida read:

```
GRANT SELECT,INSERT,UPDATE,DELETE ON minudb.* TO 'admin'@'%'
IDENTIFIED BY 'mingiparool' WITH GRANT OPTION;
grant select on mysql.user to 'admin'@'%';
```

Kasutajal “*admin*” parooliga “*mingiparool*” on sellega olemas kõik õigused andmebaasil “*minudb*” koos privileegide andmise õigusega teistele kasutajatele. Süsteemi jaoks tuleks luua ka üks tavakasutaja. Järgnevalt kirjeldame, kuidas on võimalik luua süsteemile tavakasutaja nimega “*kasutaja*” ja parooliga “teineparool”. Tuleb olla “*root*” kasutajaga keele MySQL käsureal ning võtta blokki järgmised read ja need kopeerida ja kleepida andmebaasikeele MySQL käsureale:

```
grant SELECT,INSERT,UPDATE,DELETE on minudb.arved to
'kasutaja'@'%' identified by 'teineparool' with grant
option;
grant select on mysql.user to 'kasutaja'@'%';
grant SELECT,INSERT,UPDATE,DELETE on minudb.arvekaupseos to
'kasutaja'@'%' with grant option;
grant SELECT,INSERT,UPDATE,DELETE on minudb.importijad to
'kasutaja'@'%' with grant option;
grant SELECT,INSERT,UPDATE,DELETE on minudb.kaubad to
'kasutaja'@'%' with grant option;
grant SELECT,INSERT,UPDATE,DELETE on minudb.kaubanimed to
```

```
'kasutaja'@'%' with grant option;
grant SELECT,INSERT,UPDATE,DELETE on minudb.kliendid to
'kasutaja'@'%' with grant option;
grant SELECT,INSERT,UPDATE,DELETE on minudb.planeerija to
'kasutaja'@'%' with grant option;
grant select on minudb.s2tted to 'kasutaja'@'%' with grant
option;
grant SELECT,INSERT,UPDATE,DELETE on minudb.soodustused to
'kasutaja'@'%' with grant option;
grant SELECT,INSERT,UPDATE,DELETE on minudb.tootajad to
'kasutaja'@'%' with grant option;
grant SELECT,INSERT,UPDATE,DELETE on minudb.tootjad to
'kasutaja'@'%' with grant option;
grant SELECT,INSERT,UPDATE,DELETE on minudb.upload to
'kasutaja'@'%' with grant option;
grant SELECT,INSERT,UPDATE,DELETE on minudb.varad to
'kasutaja'@'%' with grant option;
```

Selliselt on loodud nii süsteemile kasutaja, kui ka antud talle teatud kogus õigusi andmebaasil "*minudb*".

Antud süsteem on sellega nüüd üles seatud ning kasutamisvalmis.

Lisa 5. Süsteemi testi lähtekood

Süsteemi testi lähtekood asub lahtipakituna kaasasoleva CD plaadi faili "*lahtekood.zip*" kataloogis "*systemtest*".