Piia Viitapohja

WEB-POHJAINEN TUNTIKIRJAUSJÄRJESTELMÄ

Tietotekniikan koulutusohjelma
ohjelmistotekniikan suuntautumisvaihtoehto
2012

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

WEB-POHJAINEN TUNTIKIRJAUSJÄRJESTELMÄ

Viitapohja, Piia
Satakunnan ammattikorkeakoulu
tietotekniikan koulutusohjelma
huhtikuu 2012
Ohjaaja: Aarinen, Reino
Sivumäärä: 27
Liitteitä: 1

Asiasanat: Drupal, tuntikirjaus, PHP

Opinnäytetyön aiheena oli toteuttaa yrityksen intranettiin web-pohjainen työtuntien kirjausjärjestelmä. Järjestelmän tarkoituksena olisi helpottaa työntekijöiden osalta tuntien kirjaamista ja palkanlaskennan osalta tuntien laskemista ja raportoimista.

Opinnäytetyö toteutettiin käyttäen Drupal ohjelmointikehystä, PHP-ohjelmointikieltä ja MySQL tietokantaa Apache -ympäristössä. Tuntikirjaus-ohjelmaa varten kehitettiin oma ns. custom-moduuli.

Tässä opinnäytetyön kirjallisessa osuudessa pyritään selvittämään tuon moduulin suunnittelu- ja toteutusprosessia. Lisäksi käydään läpi yleisesti Drupalin toimintaa ja sovelluskehityksen perusteita Drupal-ympäristössä lähinnä koodiesimerkein ja lisäselityksin. Kohderyhmänä ovat Drupaliin vasta tutustuvat sovelluskehittäjät, joilla on perustiedot Internetin toiminnasta, PHP:stä ja HTML:stä.

WEB BASED WORK HOUR TRACKING SYSTEM

Viitapohja, Piia
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Information Technology
April 2012
Supervisor: Aarinen, Reino
Number of pages: 27
Appendices: 1

Keywords: Drupal, PHP

The purpose of this thesis was to develop a web application for a company that could keep a record of employees working hours and provide a simple input form to the company's employees.

The hands-on project was made using the Drupal programming framework, the PHP programming language and the MySQL relational database in a Linux environment. Practically a custom module was written for the Drupal that consists some input forms and a option to download reports to Excel charts.

This written part of the thesis concentrates on providing some basic information on how the Drupal works and also describes the process of designing and writing a custom module in Finnish.

SISÄLLYS

LYHENTEET

AJAX	Asynchronous JavaScript And XML
Apache	Avoimen lähdekoodin web-palvelin
API	Ohjelmointirajapinta
CMS	Content Management System (sisällönhallintajärjestelmä)
jQuery	Javascript-kirjasto
Moduuli	Laajennus Drupaliin
MySQL	Avoimen lähdekoodin tietokantaohjelmisto
PHP	Laajasti käytetty yleis käyttöinen skriptikieli, joka sopii esimerkiksi palvelinpuolen internetohjelmointiin
SQL	Structured Query Language, standardoitu kyselykieli relaatiotietokantojen käyttöön

1 JOHDANTO

Opinnäytetyön käytännön toteutusosan tarkoituksena oli toteuttaa Neorem Magnets Oy:lle intranettiin web-pohjainen ohjelma, johon työntekijät voisivat kirjata työtuntinsa omilla tunnuksillaan, ja jotka ohjelma tarvittaessa näyttäisi siistinä taulukkona työnseurantaa ja palkanlaskua varten.

Neorem Magnets Oy on ulvilalainen, vuonna 1995 perustettu teollisuusmagneetteja valmistava yritys, jolla on työntekijöitä noin 70.

Ohjelman tavoitteena oli helpottaa tuntien kirjaamisen työn määrää, koska vanhalla tavalla kaikki merkinnät kirjattiin käsin taulukkolaskentaohjelmaan. Tämä oli sekä hidasta että virhealtista.

Ohjelmointikehykseksi valittiin Drupal, sillä sitä voisi tarvittaessa jatkokehittää esimerkiksi Digian toimesta. Drupalin perusteiden opiskelemiseen käytettiin John VanDykin kiroittamaa Pro Drupal Development kirjaa. Drupalilla on myös hyvin kattavat yhteisösivut osoitteessa drupal.org. Koska työssä käytettiin avoimeen lähdekoodiin perustuvia tekniikoita, materiaalia löytyi paljon esimerkiksi PHP:n ja MySQL:n virallisilta sivuilta, sekä erilaisilta foorumeilta ja blogeista.

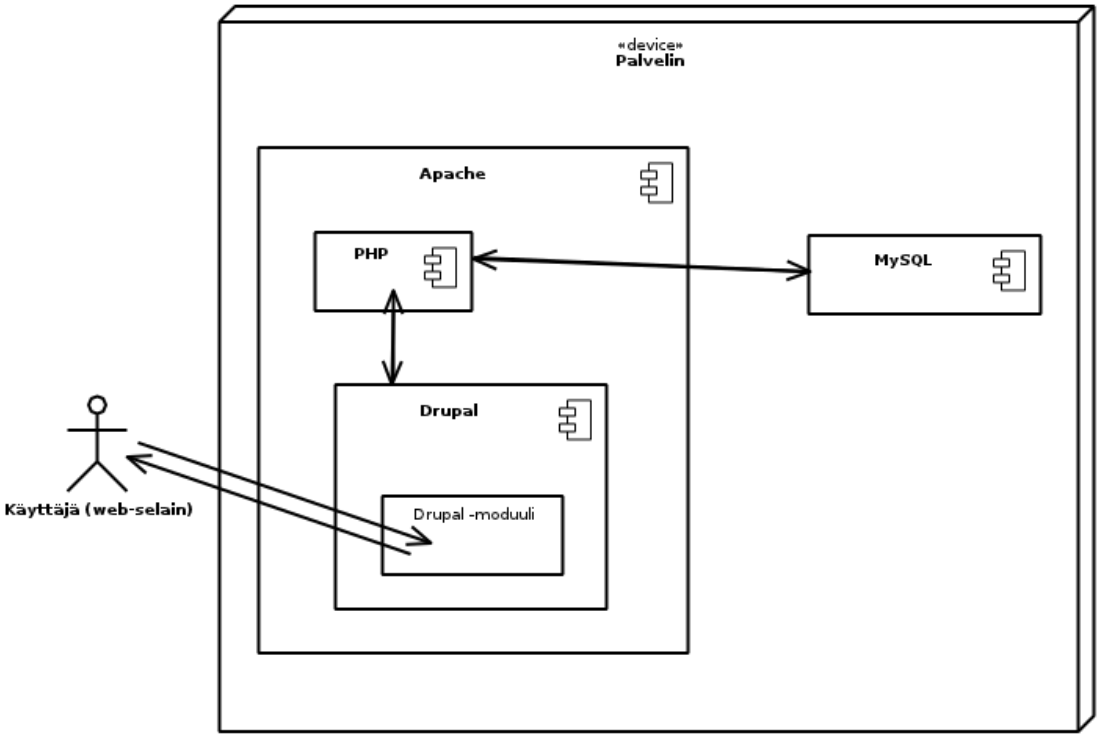
Tämän raportin tavoitteena on selventää lukijalle Drupal moduulin kehittämisprosessia ja valottaa Tuntikirjaus -moduulin toimintaa ja kehitysprosessia.

2 KÄYTETYT TEKNIIKAT JA TYÖYMPÄRISTÖ

2.1 Drupal

Drupal on lisenssimaksuton, avoimen lähdekoodin WWW-pohjainen sisällönhallintajärjestelmä ja sovelluskehitysalusta. Alunperin sen loi Dries Buytaert foorumijärjestelmäksi vuonna 2001. Nykyään Drupaliin perustuvia internetsivuja löytyy miljoonittain. Drupalia käyttävät mm. Obaman hallinto, Infoworld, YLE ja Uusi Suomi.,

Drupal pohjautuu PHP-ohjelmointikieleen ja toimii erilaisissa tietokantaympäristöissä, esimerkiksi MySQL:ssä. Drupalin toiminta perustuu moduuleihin eli laajennuksiin, jotka ovat erikoistuneet jonkin tietyn asian toteuttamiseen.



Kuva 1, esimerkki Drupal ympäristöstä.

Drupalin asennuksen mukana tulee tärkeimmät ydinmoduulit, kuten esimerkiksi käyttäjän hallinnasta ja toiminnasta huolehtiva ”user”-moduuli. Lisäksi on saatavissa paljon yhteisön kehittämiä moduuleita (kirjoitushetkellä yli 9 500).

Erilaisia valmiita moduuleita on esimerkiksi WWW-kauppapaikkoja, kuvagallerioita, työnhallintaa, projektinhallintaa ja postituslistojen hallintaa varten. Drupalin asentamiseen ja käyttämiseen ei siis välttämättä tarvita ohjelmointitaitoja, jos käyttökohde on yleinen. Silloin riittää, että etsitään sopivat moduulit, säädetään asetukset ja käytetään.

Tässä opinnäytetyössä keskitytään vain Drupalin sovelluskehitysominaisuuksiin, eli kirjoitetaan oma moduuli. Omaa moduulia kutsutaan Drupal maailmassa ns. custom moduuliksi.

2.1.1 Tiedostorakenne, info ja module -tiedostot

Drupalin ydinmoduulit ovat drupal/modules/ hakemistossa. Näihin ei tulisi tehdä muutoksia, sillä esimerkiksi Drupalin päivityksen yhteydessä nekin voivat päivittyä ja omat muutokset pitäisi tällöin kirjoittaa aina uudelleen. Omat moduulit voidaankin turvallisesti sijoittaa hakemistoon drupal/sites/all/modules. Lisäkirjastot, kuten esimerkiksi JQuery sijoitetaan hakemistoon drupal/sites/all/libraries.

Jokainen moduuli on omassa hakemistossaan, ja sisältää .info ja .module -loppuiset tiedostot. Info tiedosto sisältää perustietoja, kuten esimerkiksi moduulin nimen, kehittäjän nimen, Drupal version, jossa moduuli toimii ja riippuvuudet muista moduuleista. Module tiedosto sisältää itse PHP koodin. Koodia voi jakaa myös pienempiin osasiin, jotka nimetään .inc loppuisiksi tiedostoiksi.

2.1.2 Koukkufunktiot (hooks)

Drupalin moduulijärjestelmä perustuu ns. ”koukkufunktioiden” käyttön. Koukkufunktiot ovat PHP funktoita, jotka on nimetty tyylillä foo_bar(), joissa ”foo” on moduulin nimi, ja ”bar” koukun nimi.

Koukkufunktiot ovat custom moduulissa tapa vaikuttaa Drupalin ytimen (core) toimintoihin. Esimerkiksi kun käyttäjä poistetaan, custom moduulista suoritetaan automaattisesti funktio omamoduuli_user_delete(), jos sellainen on moduulissa toteutettu.

Kaikki mahdolliset koukkufunktiot ovat selitetty api.drupal.orgin ”Hooks” osiossa. Myös omia koukkkufunktioita on mahdollista tehdä, mutta siihen ei kuitenkaan ollut tässä työssä tarvetta.

2.1.3 Tietokanta API

Drupal tarjoaa rajapinnan, jolla voi hallinnoida tietokantaa riippumatta siitä minkä tyyppinen se on. Rajapinta pyrkii säilyttämään SQL:n syntaksin, samalla kun se tarjoaa lisää tietoturvaa ja kääntää kyselyt asetustiedostossa määritellylle tietokannalle (esim. MySQL) sopivaksi. /1/

Tyypillisimmät kyselyt suoritetaan kutsumalla funktiota db_query() tai db_query_range(). Hyödyllinen funktio on myös pager_query(), jonka avulla saa helposti rakennettua sivutuksen.

Yksinkertainen kuvitteeellinen esimerkki tietokanta rajapinnan käyttämisestä olisi listata 10 kirjautuneena olevan käyttäjän tekemää artikkelin otsikkoa taulusta ”artikkelit”:

```
<?php
  $result = db_query_range('SELECT a.otsikko
    FROM {artikkelit} a WHERE n.uid = %d', $uid, 0, 10);
  while ($artikkeli = db_fetch_object($result)) {
    // Tuloksien käsittely: $artikkeli->otsikko
  }
?>
```

Esimerkki 2.1, yksinkertainen haku tietokannasta.

Huomioi myös aaltosulkeet taulun nimen ympärillä. Niitä käytetään lisäämään taulun nimeen etuliitteeksi tietokannan nimi.

Kaikki argumentit SQL-lauseelle merkitään %d tai %s merkeillä ja itse arvo annetaan db_query() funktion argumenteiksi. Drupalin tietokanta API osaa silloin varmistaa, ettei käyttäjä pääse antamaan haitallista koodia lauseen sisään. Tässä esimerkissä parametrit 0 ja 10 määrittävät montako arvoa tulostetaan ja mistä lähtien. Huomaa myös varsin yleinen malli tulosten käsittelyssä, jossa while-silmukassa iteroidaan tulosjoukkoa funktiolla db_fetch_object().

2.1.4 Valikot

Valikoita käytetään muodostamaan navigointilinkit sivulle, mutta sen lisäksi ne yhdistävät Drupalin sisäiset polut callback funktioihin, eli käytännössä mikä funktio suoritetaan, kun käyttäjä ohjaa selaimen tietylle osoitteelle. Valikoihin voi myös määrittää, mitä oikeuksia kyseiseen osoitteeseen pääseminen edellyttää. Drupal käsittelee valikoita PHP:n taulukoina.

Omaan moduuliin tehdään valikko toteuttamalla koukkufunktio hook_menu(), keräämällä halutut kohteet listaan ja palauttamalla lista (esimerkki 2.2.)

```
<?php
function hook_menu() {
  $items = array();
  $items['mypath/%object'] = array(
    'title' => 'Page title',
    'description' => 'Your description goes here.',
    'access arguments' => array('administer users'),
    'page arguments' => array(1),
    'page callback' => 'object_display',
    'file' => 'name_of_file.inc',
    'weight' => 0,
    'type' => MENU_NORMAL_ITEM,
  );
return $items
?>
```

Esimerkki 2.2, valikko.

Huomionarvoisia kohtia esimerkissä 2.2:

- title on sivun otsikko, joka on samalla kohteen nimi valikossa
- page arguments lähettää kutsuttavalle funktiolle parametriksi arvon, joka saadaan urlista, esimerkiksi, jos ohjaa selaimen osoitteeseen /poista/43, parametriksi tulisi 43.
- page callback on funktion nimi, johon kutsu ohjataan.
- access arguments määrittelee oikeudet, tässä jos käyttäjällä on oikeus hallinnoida käyttäjiä, hänellä on oikeudet sivuun.

2.1.5 Lomakkeet (forms) ja niiden käsittely

Drupalin ohjelmointirajapinta helpottaa lomakkeiden tekoa tyylittelemällä ne yhtenäisen tyylin mukaan ja tarjoamalla yksinkertaisen tavan määritellä ominaisuuksia elementeille (esimerkki 2.2.) Lisäksi validaatio ja tulosten käsittely automatisoituu, sillä Drupal kutsuu omavalikko_validate() ja omavalikko_submit()

funktioita, kun käyttäjä on painanut submit-nappia. Jos validaatiofunktiossa kutsutaan form_set_error() funktiota (esimerkki 2.3b rivi 3), Drupal keskeyttää lomakkeen lähetyksen ja antaa virheilmoituksen. Jos validaatio menee läpi, päästään submit funktioon (esimerkki 2.3c.) Lomakkeen lähettämiin arvoihin päästään kummassakin funktiossa käsiksi $form_state['values'] taulukossa käyttäen elementin nimeä avaimena.

```
<?php
function test_myform($form_state) {
  $form['name'] = array(
       '#type' => 'textfield',
       '#title' => t('Nimi'),
       '#size' => 30,
       '#maxlength' => 64,
       '#description' => t('Kirjoita nimesi'),
  );
  //Submit nappi
  $form['submit'] = array('#type' => 'submit',
                          '#value' => t('Tallenna'));
  //Palautetaan koko $form array
  return $form;

//Sivulla kutsutaan drupal_get_form() funktiota,
// parametrina "lomakefunktion" nimi
function test_page() {
  return drupal_get_form('test_myform');
}
?>
```

Esimerkki 2.3a, toimiva lomake, joka kysyy käyttäjältä nimeä. Huomaa myös, että lomakkeen elementtien ominaisuudet määritellään merkkijonona, joka alkaa ristikkomerkillä.

```
function test_myform_validate($form, &$form_state) {
  if ($form_state['values']['name'] == '') {
    form_set_error('', t('Et antanut nimeä!'));
  }
}
```

Esimerkki 2.3b, validaatio. Syötteen tarkastaminen on varsin helppoa; testataan toteutuuko jokin ehto, ja jos ei toteudu, kutsutaan Drupalin funktiota form_set_error, joka lopettaa käsittelemisen siihen, rakentaa lomakkeen uudelleen ja näyttää käyttäjälle virheviestin.

```
function test_myform_submit($form, &$form_state) {
  db_query("INSERT INTO {table} (name) VALUES ('%s')",
  $form_state['values']['name'], $form_state['values']);
  drupal_set_message(t('Tiedot tallennettu.'));
}
```

Esimerkki 2.3c, submit. Suoritetaan vain, jos validaatio meni läpi.

Drupal.orgista löytyy kattava referenssitaulukko nimellä ”Form API Reference”. Tätä kannattaa pitää esillä lomakkeita tehdessä. Liitteessä 1 on tarkempi kaavio Drupalin sisäisestä lomakkeenkäsittelystä.

2.1.6 Ajax ja Drupal (AHAH forms)

Ajax on joukko web-sovelluskehityksen tekniikoita, joiden avulla web-sovelluksista voi tehdä vuorovaikutteisempia. Alkuperäisessä merkityksessään AJAX:lla on alun perin viitattu tekniikkaan, jossa verkkosivulla JavaScript:llä asynkronisesti tehtävistä HTTP-pyynnöistä palautetaan XML-merkkausta. Nykyisin Ajax-tekniikoilla viitataan yleisesti samankaltaiseen toimintatapaan: Ajaxissa selainohjelma vaihtaa pieniä määriä dataa palvelimen kanssa taustalla niin, ettei koko verkkosivua tarvitse ladata uudelleen joka kerta käyttäjän tehdessä muutoksen. Tekniikan päämääränä on siis lisätä verkkopalvelun vuorovaikutteisuutta, nopeutta ja käytettävyyttä. /2/

Tähän käytetään Drupalissa omaa rajapintaansa. Ennen Drupalin versiota 7 rajapintaa kutsuttiin nimellä AHAH, koska haluttiin tehdä ero ”oikean” Ajaxin välille. Drupalin AHAH ei käytä XML:ää ollenkaan. Toisekseen Drupalin AHAH lomakkeissa ei ohjelmoijan tarvitse koskea Javascript-koodiin ollenkaan. Uusimmassa Drupalin versiossa eroa ei enää tehdä niin selväksi, koska koko Ajax terminkin määritys on laajentunut. Myös Ajaxin käyttöä on helpotettu uudemmissa versioissa.

AHAH toiminnallisuuden saa käyttöön lisäämällä lomakkeen elementille, esimerkiksi napille #ahah ominaisuuden. Sillä on myös omia lisämäärityksiä, kuten efektin (esimerkiksi häipyminen tai liukuminen) ja callback funktion nimen lisääminen. Kun käyttäjä sitten painaa tätä nappia, Drupal kutsuu callback funktiota, joka tutkii mitä käyttäjä on syöttänyt ja palauttaa ne JSON pakettina sitä kutsuvalle lomake-funktiolle.

2.2 Relaatiotietokannat ja MySQL

Relaatiotietokannassa tiedot esitetään tauluina (table), joita kutsutaan myös relaatioiksi. Yhtä riviä kutsutaan tietueeksi (record). Taulun jokaisella rivillä on yhtä monta tietoa eli kenttää (field). Jokaisella rivillä täytyy olla yksikäsitteinen

perusavain, joka vastaa jotakin reaalimaailman kohdetta. Kuhunkin kohteeseen liitetään vain siihen välittömästi liittyvät ominaisuudet. Kukin yksittäinen tieto relaatiotietokannassa voidaan hakea ainakin ilmoittamalla taulun nimi, perusavaimen kentän nimi ja avaimen arvo sekä haettavan tiedon kentän nimi. Lisäksi on olemassa lukemattomia muita tapoja hakea tietoa. Relaatiotietokannasta tietoa haetaan vain tiedon nimien ja arvojen perusteella, ei siis tiedon sijainnin tai järjestyksen mukaan. /4/

MySQL on suosituin avoimen lähdekoodin relaatiotietokantaohjelmisto. MySQL on saatavissa vapaalla GNU GPL -lisenssillä tai kaupallisella lisenssillä, mikäli asiakas ei halua käyttää GPL lisensoitua ohjelmistoa. /5/

2.3 Työympäristö ja muut apuvälineet

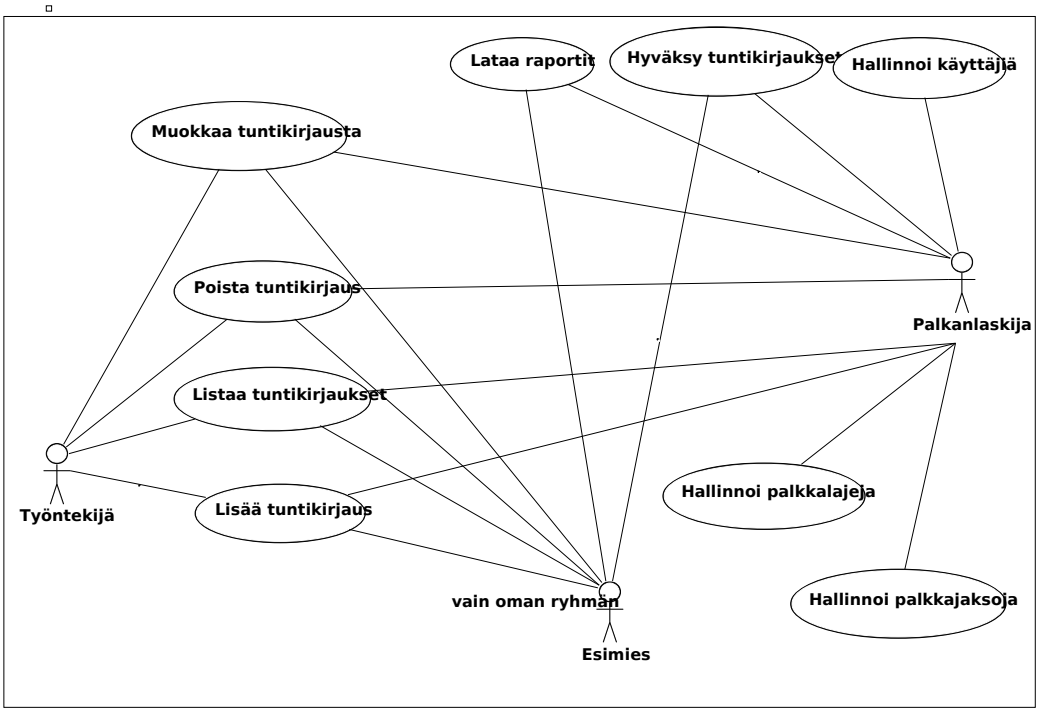
Työn tekemisessä käytettiin omaa virtuaalipalvelinta, joka sijaitsee yrityksen sisäverkossa. Virtuaalipalvelimessa on Linux -käyttöjärjestelmä, jossa Apachen web palvelin PHP tuella (versio 5.1) ja tietokantana MySQL. Drupal versioksi valittiin 6, joka on vanhempi, mutta myös vakaampi uusimpaan 7 versioon nähden.

Editorina käytettiin Vim-editoria, jossa oli muutamia hyödyllisiä plugineita, kuten esimerkiksi PHP-syntaksitarkistus. Drupal.org sivulla on hyvä opas Vim-editorin konfiguroimiseen Drupal käyttöön sopivaksi. /6/

3 MÄÄRITTELY JA SUUNNITTELU

3.1 Käyttötapaukset

Järjestelmässä on kolme erityyppistä käyttäjää ja riippuen käyttäjästä tuntikirjauksen, palkkajaksojen ja palkkalajien syöttämiseen ja listaamiseen liittyviä toimintoja. Käyttötapauskaavio on hyvä apuväline, kun pohditaan ohjelman eri ominaisuuksia hyvin yksinkertaisella tasolla, joten aloitin suunnittelemisen siitä.



Kuva 3.1, Tuntikirjaus- ohjelman käyttötapauskaavio.

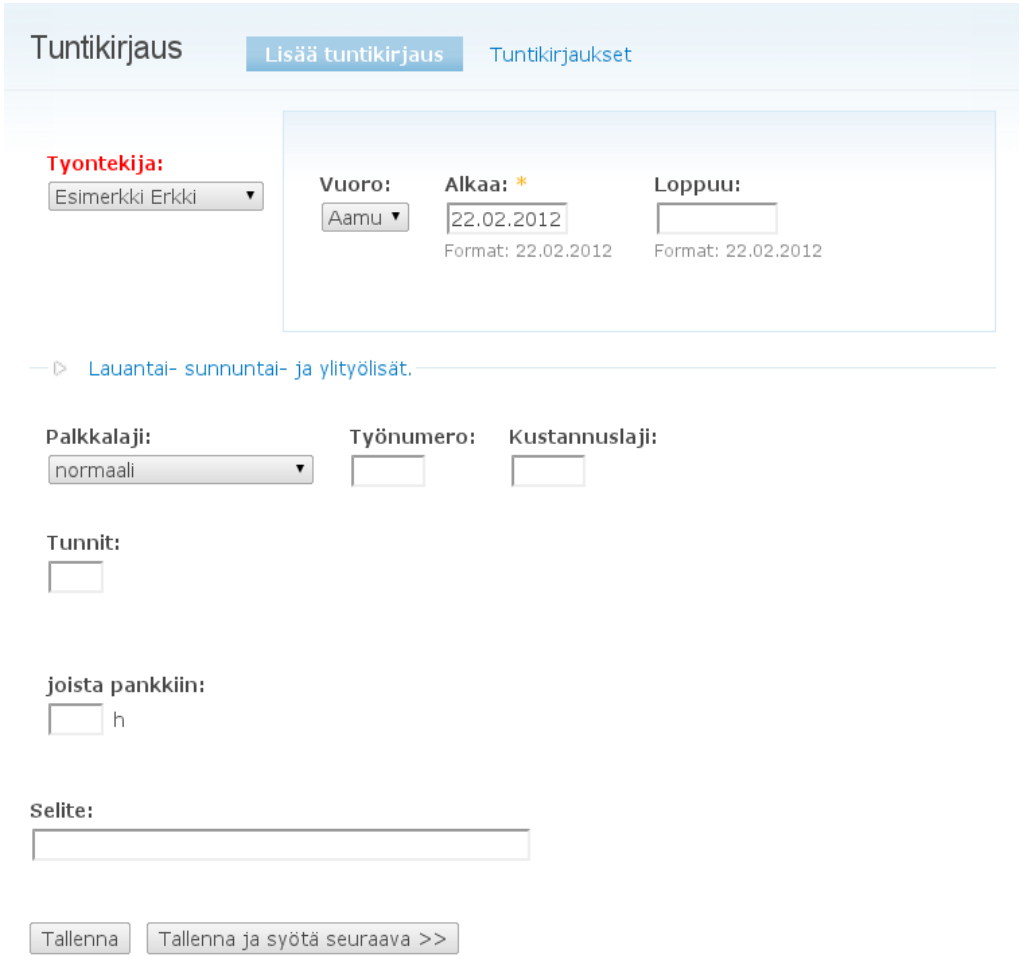
Tuntikirjaus tarkoittaa tässä tapauksessa tapahtumariviä, johon kirjataan mm. päivämäärä, tehdyt työtunnit, työkoodi ja mahdollisesti mitä palkkalajia se edustaa. Yhdellä päivällä voi samalla työntekijällä olla useampikin tuntirivi, jos työntekijä on kyseisenä päivänä tehnyt erityyppisiä työtehtäviä. Palkkalajit ovat palkanlaskennan kannalta erityisiä palkkatyyppejä, joita ovat esimerkiksi sairaslomat, pekkaset ja vuosilomat. Palkkajaksot taas ovat noin kahden viikon mittaisia ajanjaksoja, joiden välillä tulostetaan siltä ajalta yhteenvetoraportteja. Raporteista ilmenee paljonko ja minkä tyyppisiä työtunteja työntekijät ovat tehneet, ja niiden mukaan lasketaan palkat.

Työntekijä on käytännössä peruskäyttäjä, joka kirjautumisen jälkeen kirjaa tuntinsa, ja mahdollisesti myös muokkaa ja poistaa niitä. Muokkaaminen ja poistaminen on mahdollista vain, jos tunteja ei ole vielä hyväksytty. Näin voidaan estää mahdolliset sekaannukset ja väärinkäytökset. Esimies on käyttäjä, jolla on täysi kontrolli oman ryhmänsä käyttäjiin. Esimies voi tehdä samat toiminnot kuin alaisensakin, ja sen lisäksi hyväksyä tunteja. Lisäksi esimies voi ladata yhteenvetoraportteja tulostettavaksi.

Palkanlaskija on eräänlainen ”pääkäyttäjä”, jolla on mahdollisuus hallinnoida palkkajaksoja, palkkalajeja ja työntekijöitä. Tämän lisäksi on tietenkin kaikki samat toiminnot kuin esimiehellä, sillä erotuksella, että palkanlaskija näkee kaikkien ryhmien työntekijöiden tuntikirjaukset.

3.2 Käyttöliittymä

Ohjelmaa suunnitellessa erääksi tärkeimmistä ominaisuuksista muodostui käyttöliittymän helppokäyttöisyys ja selkeys, sillä käyttäjäkunta tulisi olemaan hyvin eritasoista tietotekniikan taitojen suhteen. Käyttöliittymän kaksi perusosaa ovat tuntirivin syöttösivu ja tuntien listaussivu. Syöttösivulla (ks. kuva 3.2a) tulisi olla päivämääräkentät ja vuorokentät, palkkalajien valinta ja tuntikentät. Syöttökenttien asettelua ja määrää piti harkita tarkasti, jotta syöttölomakkeesta tulisi mahdollisimman yksinkertainen. Drupal tarjoaa mahdollisuuden ryhmitellä kenttiä eri osa-aluiesiin (fieldset), joten ryhmittelin samankaltaiset asiat omiin alueisiinsa.

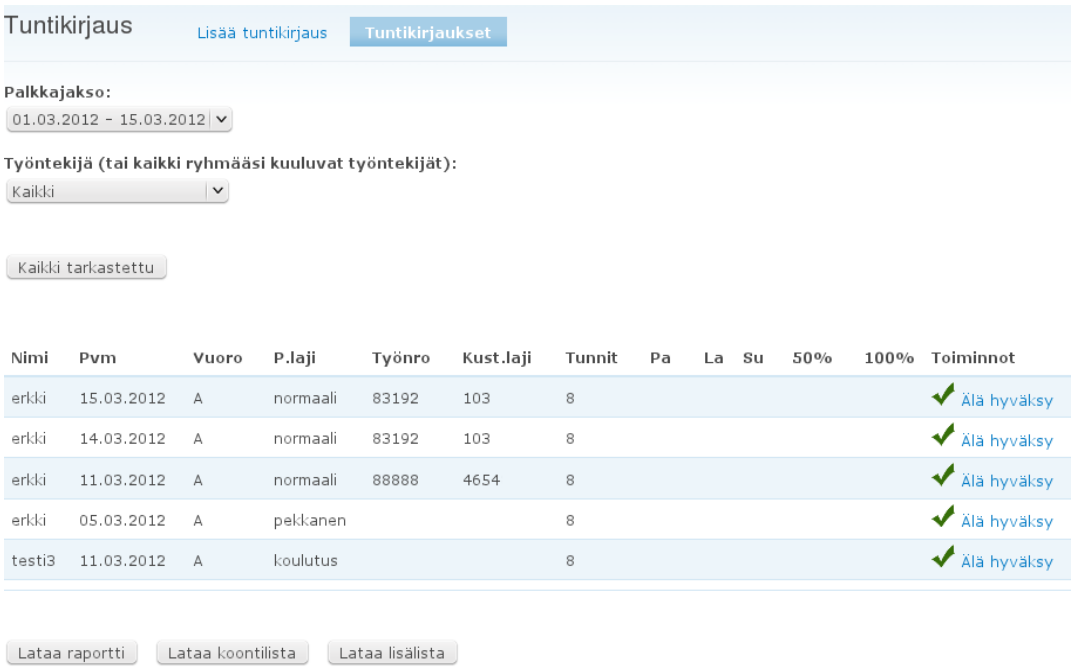


Kuva 3.2a, tuntirivin syöttölomake, esimiehen näkymä.

Määrittelyssä tuli myös ilmi, että erilaisille ylityö- ja viikonloppulisille tarvittaisiin monta lisäkenttää (ks. kuva 3.2b). Näitä kuitenkin käytetään käytännössä vain viikonloppuna, joten muina päivinä ne saattoi piilottaa linkin taakse.

Kuva 3.2b, tuntirivin syöttölomake, työntekijän näkymä. Lauantain tuntien lisäys ja palkkalajina palaveri.

Listausnäkymässä näytetään lisätyt tuntirivit ja voidaan ladata Excel raportit. Listan muodostumista ohjaavat alasvetovalikot palkkajakson ja työntekijän mukaan. Palkkajaksovalikosta voi valita haluamansa jakson, ja se tulostuu alapuolelle. Oletuksena tulostuu vain omat tuntirivit, tai jos sisäänkirjautunut on esimies, kaikkien työntekijöiden tuntirivit. Valintaa voi myös rajata työntekijävalikosta työntekijän mukaan (kuva 3.3).



Kuva 3.3, esimiehen listausnäkymä.

Listausnäkymän toiminnot:

- Alasvetovalikoista säädetään minkälainen lista tulostetaan (palkkalajin ja työntekijän mukaan). Työntekijäkäyttäjällä ei ole työntekijävalikkoa ollenkaan, koska työntekijä ei saa nähdä muiden työntekijöiden tunteja.
- Lista tuntikirjauksista tulostuu alasvetovalikoiden alle sen mukaan mitä valikoista on valittu. Jos kirjauksia tulostuu yli 20, sivutetaan tulokset useammalle sivulle. Näin vältetään tilanne, jossa lista toisi liikaa tuloksia, ja siten kestäisi kauan ladata.
- Mahdollisuus hyväksyä työtunteja.
- Napit raporttien lataamiselle. Näkyvät vain esimies ja palkanlaskija käyttäjille. Raportit tulostavat Excel raporttina tiedot siltä palkkajaksolta, mikä palkkajaksovalikosta on valittu.

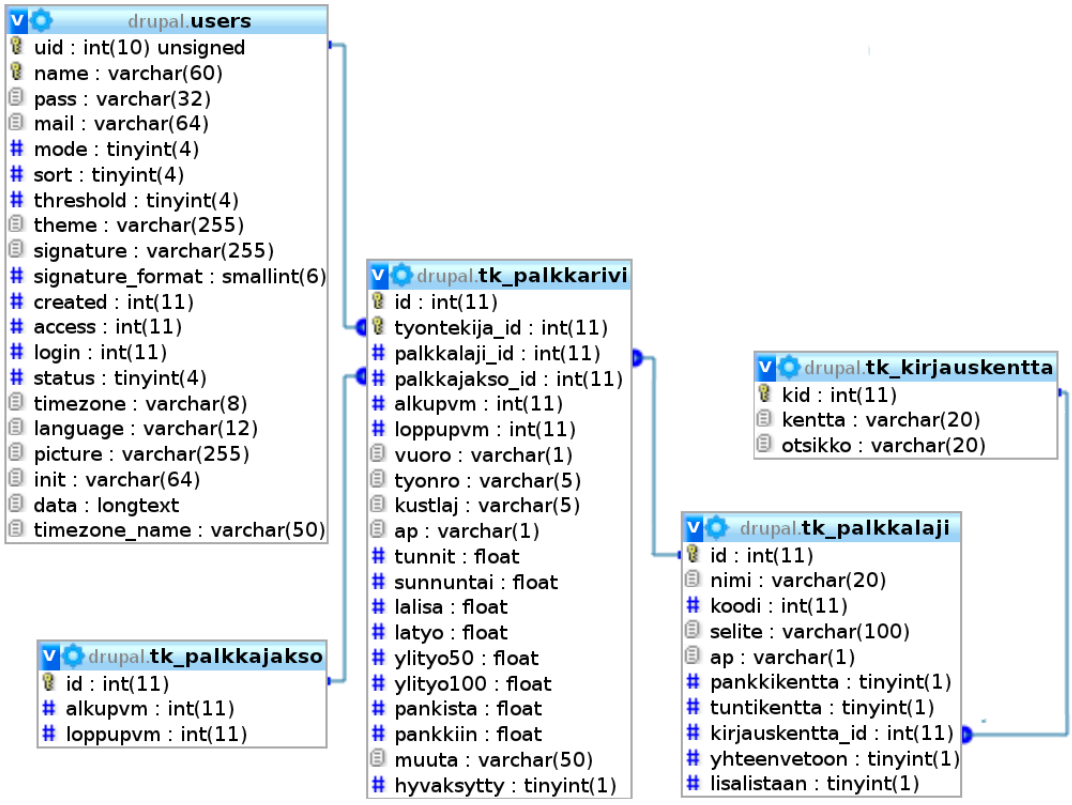
Valikoita suunniteltaessa pyrin myös yksinkertaistamaan mahdollisiman paljon, eli käytännössä kaikki Drupalin oletuslohkot ja -valikot pois. Näitä pystyi säätämään suoraan Drupalin hallintapaneelista.

4 TOTEUTUS

4.1 Tietokannan taulut ja rakenne

Tuntikirjaus ohjelmaa varten lisäsin neljä uutta tietokantataulua tietokantaan. Nimesin nämä tk_<taulun nimi> tyylillä, jotta ne erottuisivat muista tauluista (kuva 4.1.) Ohjelma käyttää myös joitakin Drupalin mukana tulevia tauluista, kuten esimerkiksi users-taulua.

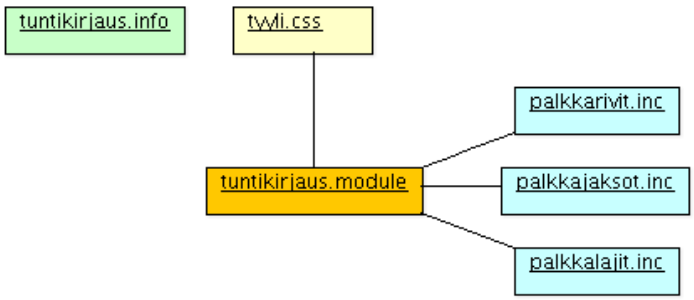
Tauluista tk_palkkarivi toimii ikäänkuin varastona tapahtumankirjauksille. Jokainen lisätty tuntikirjaus tallennetaan tauluun, joka sisältää myös viittaukset (id-numerolla) käyttäjä, palkkajakso ja palkkalaji tauluihin.



Kuva 4.1 Users taulu on Drupalin mukana tuleva, muut syötettiin tietokantaan manuaalisesti.

4.2 Tuntikirjaus moduulin sisäinen rakenne ja päävalikko

Custom moduulia rakennettaessa pakolliset tiedostot ovat .info ja .module. Lisäksi erottelin palkkariveille, palkkajaksoille ja palkkalajeille omat tiedostonsa selkeyden vuoksi.



Kuva 4.2, Tuntikirjaus-moduulin rakenne.

Infotiedosto sisältää lähinnä moduulin nimen ja version. Tuntikirjaus.module tiedostossa on toteutettu hook_menu() funktio (ks. esimerkki 2.2), joka muodostaa päävalikon ja ohjaa ohjelmansuoritusta oikeaan tiedostoon. Lisäksi se lisää css tiedoston tyyli.css, jossa on omia määrityksiä div-elementeille.

```
$items['palkkalajit/palkkalaji_add'] = array(
  'title' => 'Lisää palkkalaji',
  'path' => 'tuntikirjaus/palkkalaji_add',
  'page callback' => 'tuntikirjaus_palkkalaji_add',
  'type' => MENU_LOCAL_TASK,
  'access arguments' => array('administer users'),
  'file' => 'palkkalajit.inc',
);
```

Kuva 4.3, palkkalajin lisäyssivun valikkoelementti.

Kuvassa 4.3, kun käyttäjä pyytää palkkalajin lisäyssivua, Drupal etsii valikosta kohteen palkkalajit/palkkalaji_add ja toimii sen mukaan.

'file' attribuutti määrittää tässä mistä tiedostosta löytyvät palkkalajien toiminnot ja 'page callback' kertoo mikä funktio suoritetaan. 'type', eli tyyppi kertoo minkätyyppinen kohde on. Tässä esimerkissä se on MENU_LOCAL_TASK, joka tarkoittaa että Drupal muodostaa toiminnosta välilehden sivun ylälaitaan. Huomaa myös 'access arguments' määritys, joka varmistaa, että käyttäjällä on oltava oikeudet 'administer users', ennenkuin hän voi jatkaa tälle sivulle.

4.2.1 Tuntikirjauksen syöttölomake

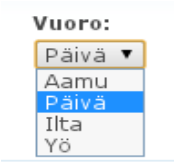
Syöttölomake (kuva 3.2a ja b) muodostuu Drupal lomakkeesta, validaatiosta ja submit osasta.

Koska uuden tuntirivin syöttäminen ja vanhan tuntirivin muokkaaminen toimivat pitkälti samalla lomakkeella, kumpikin toiminto käyttää funktiota tuntikirjaus_form, jolle annetaan parametriksi &$form_state, ja valinnaisena $edit_id, jossa on editoitavan tuntirivin id.

```
//Jos editoidaan haetaan editoitavan rivin tiedot idn perusteella.
if ($edit_id)
  $palkkarivi = db_fetch_object(db_query('SELECT * FROM {tk_palkkarivi}
                                          WHERE id = %d', $edit_id));
```

Kuva 4.4, jos $edit_id on annettu, haetaan sen tiedot tietokannasta, jotta voidaan täyttää valmiiksi vanhat arvot syöttökenttiin.

Syöttökentät kerätään $form arraylistaan, kuten kappaleessa 2.1.5 todettiin. Kaikki kentät tehdään lähes kaavan mukaan, joten otan esimerkiksi ”vuoro” alasvetovalikon.




```
$form['group1']['vuoro'] = array(
  '#type' => 'select',
  '#title' => t('Vuoro'),
  '#options' => array('A' => 'Aamu', 'P' => 'Päivä',
                      'I' => 'Ilta', 'Y' => 'Yö'),
  '#default_value' => ($palkkarivi ?
                       $palkkarivi->vuoro :
                       _get_last_vuoro($user->uid)),
  '#prefix' => '<div class="form-inline">',
  '#suffix' => '</div>',
);
```

Kuva 4.5a, vuoro-alasvetovalikko Drupalin muodostamana ja vuorokentän määrittely lähdekoodissa.

Vuoro on alasvetovalikko, jossa on neljä arvoa (aamu, päivä, ilta ja yö), joten määritellään avain-arvo-pareiksi '#type' => 'select' ja #options viittaa listaan, jossa alkukirjain vastaa vuoron nimeä. Huomaa kuvassa 4.5b, miten Drupal on rakentanut tästä listasta HTML koodia, jossa alkukirjaimesta on tullu options elementin ”value”. Kentälle saadaan otsikko #title määreellä.

Oletusarvo eli #default_value määräytyy sen mukaan, ollaanko editoimassa vanhaa tuntiriviä vai tekemässä uutta. Tässä käytetään PHP:n ehtolausetta tutkimaan onko

muuttuja $palkkarivi objekti olemassa (se haettiin, jos muuttuja $edit_id oli annettu, ks kuva 4.5a). Jos on, laitetaan kentän oletukseksi editoitavan tuntirivin vuoro. Jos taas ei ole, käytetään omaa funktiota get_last_vuoro hakemaan edellinen käyttäjän lisäämä vuoro. Tämä helpottaa sitä tilannetta, jos käyttäjällä on monta samaa vuoroa peräkkäin. Silloin ei tarvitse aina valita vuoroa valikosta, vaan se on jo valmiina.

#prefix ja #suffix määreillä voidaan antaa lisämäärityksiä, jotka Drupal lisää lomake-elementin alkuun ja loppuun. Tässä lisätään DIV tägi, jolla puolestaan on CSS tiedostossa kerrottu, kuinka lomake-elementti tulisi näyttää. Tyylitiedostossa on tässä esimerkissä määritelty se, että vuoro-syöttökenttä menee vaakatasossa peräkkäin seuraavan syöttökentän kanssa.

```
<div class="form-inline"><div class="form-item" id="edit-vuoro-wrapper">
<label for="edit-vuoro">Vuoro: </label>
 <select name="vuoro" class="form-select" id="edit-vuoro" >
   <option value="A" selected="selected">Aamu</option>
   <option value="P">Päivä</option>
   <option value="I">Ilta</option>
   <option value="Y">Yö</option>
</select>
</div>
```

Kuva 4.5b, ”Vuorokenttä” Drupalin tulostamana HTML:änä.

Loputkin kentät tehdään samantyylisesti. Kenttiä suunnitellessa drupal.orgin ”Forms API” on hyödyllinen pitää esillä, koska siinä on selitetty kaikki määreet ja miten niitä tulee käyttää. Lopuksi tehdään vielä submit-nappi, (käsitelty kappaleessa 2.1.5), jota painamalla käyttäjä lähettää tiedot käsiteltäväksi eteenpäin.

4.2.2 Syötteen tarkistukset ja tuntirivin lisäys tietokantaan

Kun käyttäjä painaa lomakkeen submit-nappia, Drupal ohjaa täytetyt kentät ensin validaatio-funktiolle, ja mikäli virheitä ei siellä tule, päästään eteenpäin submit-funktioon ja lopulta lisäämään tuntirivi tietokantaan. Syötteentarkistus eli validaatiofunktio on tässä tapaukseksessa nimeltään tuntikirjaus_form_validate. Kappaleessa 2.1.5 on selitetty miten validaatiofunktiota käytetään.

```
if ($form_state['values']['tunnit']<=0)
  form_set_error('', t('Tarkista tunnit'));
```

Kuva 4.6, tarkastetaan, että käyttäjä on antanut tunteja enemmän kuin nolla.

Tuntiriviä lisättäessä pitää siis ottaa huomioon, että käyttäjä ei voi antaa mitä tahansa syötettä. Virheellisiä syötteitä ovat esimerkiksi:

- Negatiiviset tuntimäärät (kuva 4.6)
- Viikonloppulisien merkkaaminen arkipäiväksi
- Loppupäivämäärä isommaksi kuin alkupäivämäärä.
- Väärät työkoodit

Jos validaatio menee läpi, seuraava vaihe on tuntirivin lisääminen tietokantaan.

```
db_query(
  "INSERT INTO {tk_palkkarivi} (alkupvm, loppupvm, tunnit,
   pankkiin, sunnuntai, latyo, lalisa, ylityo50, ylityo100,
   tyonro, kustlaj, vuoro, ap, muuta, palkkajakso_id,
   tyontekija_id, palkkalaji_id)
     VALUES (%d, %d, %f, %f, %f, %f, %f, %f, %f, %d, %d, '%s',
             '%s', '%s', %d, %d, %d )",
   $apvm,
   $lpvm,
   preg_replace('/,/','.',$form_state['values']['tunnit']),
   preg_replace('/,/','.',$form_state['values']['pankkiin']),
   preg_replace('/,/','.',$form_state['values']['sunnuntai']),
   preg_replace('/,/','.',$form_state['values']['latyo']),
   preg_replace('/,/','.',$form_state['values']['lalisa']),
   preg_replace('/,/','.',$form_state['values']['ylityo50']),
   preg_replace('/,/','.',$form_state['values']['ylityo100']),
   $form_state['values']['tyonro'],
   $form_state['values']['kustlaj'],
   $form_state['values']['vuoro'],
   $ap,
   $form_state['values']['muuta'],
   $form_state['values']['pjid'],
   $tyontekijaid,
   $form_state['values']['palkkalaji']
);
drupal_set_message(t('Palkkarivi lisätty'));
```

Kuva 4.7, uuden tuntirivin lisäys tietokantaan.

4.2.3 Tuntikirjausten listausnäkymä

Listausnäkymä (kuva 3.3) on myös teknisesti ottaen lomake, sillä siinä on kaksi alasvetovalikkoa ja napit raporttien lataamiselle. Kuvassa 4.8. on työntekijävalikon määritys, joka muodostetaan vain, jos sisäänkirjautunut on esimies tai palkanlaskija.

```
if ((in_array("palkanlaskija",$user->roles)) || (in_array("esimies",$user->roles))) {
  $form['ajax_holder']['tyontekija_dropdown'] = array(
    '#type' => 'select',
    '#title' => 'Työntekijä (tai kaikki ryhmääsi kuuluvat työntekijät)',
    '#options' => $tyontekija_initial_options,
    '#default_value' => $valitun_tyontekijan_id,
    '#ahah' => array(
      'path' => 'tk_callback',
      'wrapper' => 'palkkajakso-dropdown-wrapper',
    ),
    '#attributes' => array('class' => 'tyontekija-dropdown'),
  );
```

Kuva 4.8. Työntekijävalikko. Huomaa myös #ahah attribuutti, sillä listaus suoritetaan Ajaxilla samantien kun työntekijä on valittu.

Itse tuntirivit listataan merkkauksena #markup elementiksi lomakkeen sisään. Tämän olisi voinut toteuttaa muullakin tavalla (listaus lomakkeen ulkopuolelle), mutta koska Ajax -toiminnallisuus oli helpointa sijoittaa suoraan lomakkeeseen ja siitä löytyi samantyyppisiä koodiesimerkkejä, päädyin lopulta tällaiseen ratkaisuun.

```
$form['ajax_holder']['taulukko'] = array(
  '#type' => 'markup',
  '#value' => !empty($form_state['values']['tyontekija_dropdown']) ?
              tuntikirjaus_list($valitun_palkkajakson_id, $valitun_tyontekijan_id) :
              tuntikirjaus_list($valitun_palkkajakson_id),
);
```

Kuva 4.9, lista tulostetaan taulukkoon merkkauksena.

Listauksen logiikka toimii siten, että jos työntekijävalikosta on valittu yksi käyttäjä, $form_state['values']['tyontekija_dropdown'] muuttujan arvo on työntekijän id, joten kutsutaan funktiota tuntikirjaus_list(palkkajakso, tyontekija). Jos taas työntekijää ei ole valittu, muuttujan $form_state['values']['tyontekija_dropdown'] arvo on tyhjä, ja kutsutaan funktiota tuntikirjaus_list(palkkajakso). Palkkajakson arvo ei siis ole koskaan tyhjä. Funktio tuntikirjaus_list kasaa lopulta annetuista arvoista varsinaisen listan ja palauttaa sen valmiiksi muotoiltuna lomakkeeseen.

4.2.4 Palkkajaksot ja palkkalajit

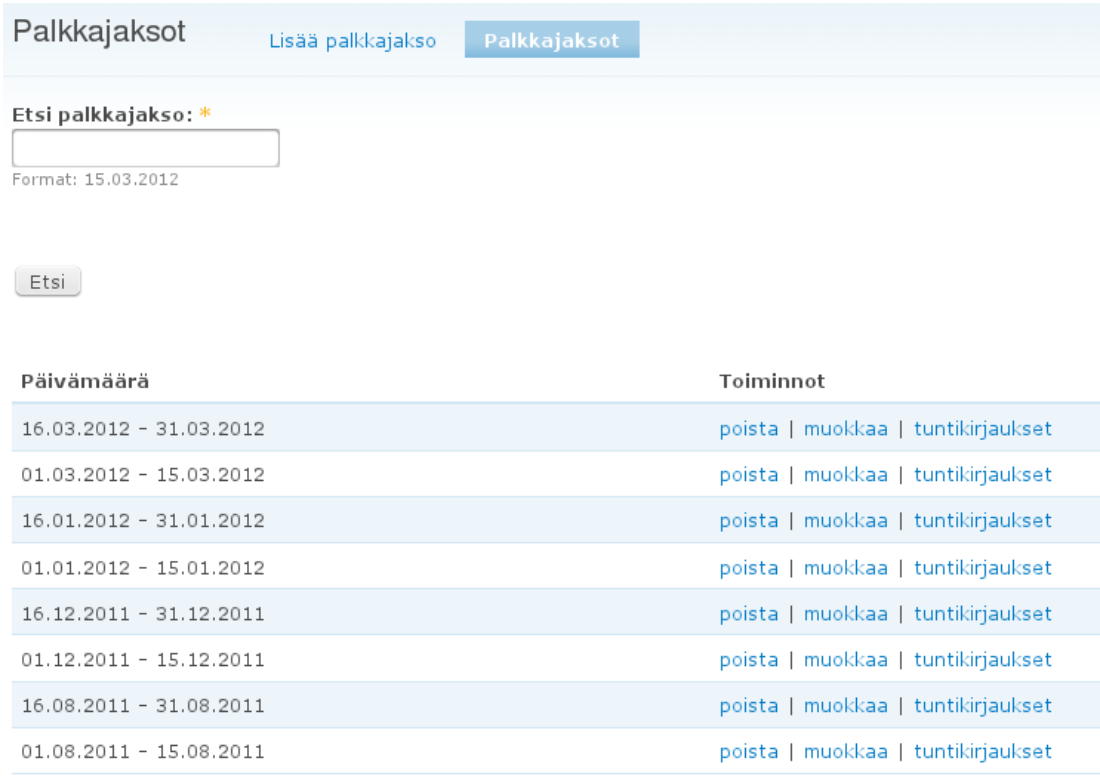
Kahdesti kuukaudessa työtunnit lasketaan, tarkastetaan ja tulostetaan taulukoksi. Tätä varten työtunnit piti jaotella palkkajaksoihin. Tuntiriviä lisättäessä ohjelma tarkistaa mihin palkkajaksoon kyseinen tuntirivi kuuluu, ja lopulta lisää sen palkkajakson id:n palkkariville, kuten kappaleessa 4.2.2 käsitellystä submit-funktioista käy ilmi. Palkkalajit taas ovat erilaisia työtuntityyppejä, kuten kuvasta 4.12 voikin päätellä.

```
$form['ajax_holder']['taulukko'] = array(
  '#type' => 'markup',
  '#value' => !empty($form_state['values']['tyontekija_dropdown']) ?
              tuntikirjaus_list($valitun_palkkajakson_id, $valitun_tyontekijan_id) :
              tuntikirjaus_list($valitun_palkkajakson_id),
);

$row =  db_fetch_object(db_query("SELECT id, loppupvm FROM {tk_palkkajakso}
                                  where alkupvm <= %d and loppupvm >= %d",
                                  strtotime($form_state['values']['apvm']),
                                  strtotime($form_state['values']['apvm'])));
if (!$row)
  form_set_error('', t('Tähän palkkajaksoon kirjaamista ei ole vielä sallittu.'));

else if ((isset($form_state['values']['lpvm']) && ($form_state['values']['lpvm'])>$row->loppupvm)))
  form_set_error('', t('Tuntikirjaus ei saa jatkua seuraavalle palkkajaksolle'));
else
$form_state['values']['pjid']=$row->id;
```

Kuva 4.10, validaatiofunktiossa tutkitaan, löytyykö lisättävälle tuntiriville oikea palkkajakso.



Kuva 4.11, palkkajaksojen listausnäkymä.

Palkkajaksoille ja palkkalajeille piti tehdä hallintamahdollisuudet, joten rakensin myös molemmille listaus- ja lisäysnäkymät. Tässä vaiheessa pystyi jo hyödyntämään uudelleen paljon koodia tuntirivien lisäys ja listausnäkymistä, koska periaate oli kaikissa sama (kappale 4.2).

Kuva 4.12, palkkalajien listausnäkymä.

4.2.5 Raporttien tulostusta

Ohjelmassa on mahdollisuus tulostaa raportit taulukkolaskentaohjelman ymmärtämään muotoon. Tämän toteutin hakemalla tarvittavat tiedot tietokannasta, keräämällä taulukkoon, laskemalla tuloksia ja muotoilemalla Excel-ohjelmalle sopivaksi. Taulukkolaskentaohjelmassa solujen erottimiksi kävi tabulaattorimerkki ”\t” ja rivinvaihdoksi rivinvaihtomerkki ”\n”. Desimaalieroitin piti vaihtaa pisteestä pilkuksi, sillä muuten Excel tulkitsi desimaaliluvut päivämääriksi. Kuvassa 4.13 on runko tämän toteuttamiseen ja kuvassa 4.14 miltä lopputulos näyttää taulukkolaskentaohjelmassa.

```
function tk_raportti_submit($form, &$form_state) {

  $filename =date($format="Ymd")."_raportti.xls";
  drupal_set_header('Content-type: application/ms-excel');
  drupal_set_header("Expires: 0");
  drupal_set_header('Content-Disposition: attachment; filename='.$filename);

  $query = "SELECT <sql lause>...";
  $query = db_query($query);
  $data = array();
  $i = 0;

  while ($row = db_fetch_array($query)) {
    $data[$i] = $row;

    /* Muotoillaan exceliin sopivaksi, jottei tyhjiin kenttiin tule nollaa
      ja vaihdetaan desimaalieroitin pisteestä suomalaisittain pilkuksi.
    */
    foreach ($data[$i] as $kentta => $arvo) {
      if (empty($data[$i][$kentta]))
        $data[$i][$kentta] = '';
      else
        $data[$i][$kentta] = preg_replace('/[.]/',',',strval($data[$i][$kentta]));
    }

    $contents .= implode("\t",$data[$i]) . "\n";
    $i++;
  }

  print utf8_decode($contents);
}
```

Kuva 4.13, mallifunktio tietojen tulostamiseksi Excelille sopivaan formaattiin.

	A	D	E	G	H	I	J	K	L	M	N	O	P	Q
1	16.03.2012 - 31.03.2012													
2														
3	Tyontekija	Normaalit	Muut	Sa	Pe	Pankista	Pankkiin	Lalisa	Su	50%	100%	Iltav	Yov	Yht
4	Esimerkki Erkki	48						2						48
5	Testataan Ääkköset	72					8	4	2					72
6														
7	YHT:	120	0	0	0	0	8	6	2	0	0	0	0	120
8														

Kuva 4.14, ohjelman tulostamaa raporttitaulukkoa.

5 TESTAUSTA JA LOPPUPÄÄTELMIÄ

Tuntikirjaus-ohjelmaa tehdessä huomattavana etuna oli se, että jo ennen työn aloittamistakin työskentelin yrityksessä, joka ohjelmaa tarvitsi ja kirjasin manuaalisesti työtunteja paperilta Exceliin. Sen vuoksi minulla oli jo jonkinlainen käsitys siitä, miltä ohjelman tulisi näyttää ja miten sen pitäisi toimia. Silti käyttäjien palaute oli tärkeintä työn onnistumisen kannalta.

Työn alussa yritin suunnitella ohjelman perusrakenteen valmiiksi, ja siinä mielessä se onnistuikin, että perustoimintaan ei tarvinnut tehdä isompia muutoksia. Ehdottomasti suurin osa ajasta menikin hienosäätöön, yllättäen mieleen tulleiden lisäominaisuuksien ja käyttäjien kasvavien vaatimusten toteuttamiseen.

Kuten web-ohjelmien kehityksessä tyypillisesti tapahtuu, testaus oli tässäkin työssä jatkuva prosessi siinä missä ohjelmointikin. Käytännössä uudet ominaisuudet testattiin sitä mukaa kun ne saatiin valmiiksi. Jatkuvien muutosten lisäksi tehtiin myös väliversioita ja pyydettiin niistä mielipiteitä työn edetessä.

Ohjelman tekijä on kuitenkin aina huono testaamaan omaa ohjelmaansa, joten ennen käyttöönottoa ohjelmaa testataan vielä käyttäjien toimesta ja otetaan käyttöön vähitellen. Ensimmäinen testauskierros on jo ohi, ja se suoritettiin pyytämällä osaa työntekijöistä ja esimiehistä syöttämään yhden palkkajakson tiedot järjestelmään ja tarkastamaan ne. Testauksesta saatavan palautteen mukaan ohjelman käyttöliittymä oli selkeä ja nopeasti omaksuttavissa. Lieviä käytettävyysongelmia aiheuttivat mm. pieniresoluutioiset näytöt, tuntien tarkastaminen päivämääräkohtaisesti ja muutamat muut pienet yksityiskohdat. Kuitenkaan mitään isompia virheitä ei löytynyt ja tarvittavat muutoksetkin ovat kohtuullisella vaivalla korjattavissa.

Tällä hetkellä ohjelma odottaa vielä lopullista käyttöönottoa, ja toivon mukaan se tulee korvaamaan nykyisen mekaanisen työni tuntikirjausten laskijana.

LÄHTEET

1. *http://drupal.org/node/310069 29.3.2012*

2. *http://fi.wikipedia.org/wiki/Ajax_(ohjelmointi) 29.3.2012*

4. *http://appro.mit.jyu.fi/doc/tiedonhallinta/tietokannat/index1.html 29.3.2012*

5. *http://fi.wikipedia.org/wiki/MySQL 29.3.2012*

6. *http://drupal.org/node/29325 29.3.2012*

Liite 1, vuokaavio Drupalin sisäisestä lomakkeenkäsittelystä (Drupal.org/2012)