KARELIA-AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

Simo Kunnari

RAKENNUKSEN YMPÄRISTÖLASKURI

Opinnäytetyö
Toukokuu 2014

Karelia AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Toukokuu 2014
Tietotekniikan koulutusohjelma

Karjalankatu 3
80200 JOENSUU
p. (013) 260 6800

Tekijä(t)

Simo Kunnari

Nimeke

Rakennuksen ympäristölaskuri

Toimeksiantaja

Metsäntutkimuslaitos

Tiivistelmä

Opinnäytetyön aiheena oli tehdä web-sovellus, jolla voi laskea rakennuksen ympäristövaikutukset sen materiaalien perusteella. Aihe tuli Metsäntutkimuslaitoksen tutkija Tarmo Rädyltä.

Sovellukseen piti suunnitella ja toteuttaa rakennusosatietokanta, jonne tallennetaan tiedot sovelluksen käyttäjien tekemistä rakennusosista. Tietokannan toteuttamiseen käytettiin MySQL-tietokantaa. Ympäristövaikutusten laskenta tehtiin suoritettavaksi Mysql-proseduureilla. Sovelluksen käyttöliittymä tehtiin Juha Ruokolaisen kanssa yhdessä. Sovellus kehitettiin pääosin PHP-ohjelmointikielellä käyttäen Yii-sovelluskehystä.

Sovelluksen testaamiseen käytettiin tietokoneita, joista tehtiin paikallisia web- ja tietokantapalvelimia sekä niihin asennettiin tarvittavat ohjelmistoympäristöt sen kehittämistä varten. Sovellus kehitettiin Metsäntutkimuslaitoksen tiloissa Joensuussa ja sen kehittämiseen käytettiin Scrum-projektinhallintamenetelmää. Sovelluksesta ei saatu tehtyä kaikkia suunniteltuja ominaisuuksia, mutta tärkeimmät ominaisuudet saatiin toteutettua.

Kieli	Sivuja 28
suomi	Liitteet 1
	Liitesivumäärä 36

Asiasanat

www-sivustot, relaatiotietokannat, ympäristövaikutukset

Karelia
UNIVERSITY OF APPLIED SCIENCES

THESIS
May 2014
Degree Programme in Information Technology
Karjalankatu 3
FI 80200 JOENSUU
FINLAND
Tel. +358 13 260 6800

Author(s)

Simo Kunnari

Title

Environmental Impacts Calculator for Buildings

Commissioned by

Finnish Forest Research Institute

Abstract

The goal of this thesis was to make an environmental impact calculator for buildings. The calculator was developed as a web application. The subject came from Tarmo Räty who works in the Finnish Forest Research Institute.

The topic was to design and implement a database for user-defined building elements and calculation of the environmental impacts. This database was implemented as a MySQL-database. The user interface of the application was made with Juha Ruokolainen. The application was mostly developed with PHP-programming language using Yii-framework.

The application was developed in a software development environment so that the testing and debugging the application was easy. The Scrum software development framework was used to manage the project tasks. The result of the project was that most of the tasks were done, but some tasks were not because there was not enough time to develop them and some of the tasks had to be prioritized.

Language	Pages 28
Finnish	Appendices 1
	Pages of Appendices 36

Keywords

web application, relational database, environmental impacts

Sisältö

Lyhenteet

Ajax	Asynchronous JavaScript and XML, menetelmä, jonka avulla web-sivun haluttua kohtaa voi muuttaa ilman, että koko sivua ei pidä ladata uudelleen ja se on toteutettu JavaScript- ja XML-ohjelmointikielillä.
CSV	Comma Separated Values, tiedostoformaatti, jossa tiedoston arvot on erotettuna pilkulla tai jollain muulla merkillä.
ISO	The International Organization of Standardization, kansainvälinen standardien määrittämisjärjestö.
LCA	Life Cycle Assessment, menetelmä, jolla määritetään ympäristövaikutuksen ominaisuudet sen elämänkaaren ajalta.
Metla	Metsäntutkimuslaitos, metsäntutkimiseen liittyviä ja muita eri palveluja tuottava suomalainen tutkimuslaitos.
MVC	Model View Controller, sovelluksen toteutustapa, jossa sovelluksen päätoiminnot on jaettu kolmeen osaan: malliin, näkymään ja kontrolleriin.
Yii	Yes It Is, PHP-sovelluskehys, jolla voi tehdä web-sovelluksia ja siinä on valmiiksi toteutettuna web-sovelluksen perustoimintoja.

1 Johdanto

Opinnäytetyön aiheena oli tehdä rakennuksen ympäristölaskuri, jolla voi laskea rakennuksen materiaalien ympäristövaikutukset rakennuksen elinkaaren ajalta. Aihe oli tullut koululle Metsäntutkimuslaitokselta (Metla) ja sitä oli aloittanut tehdä Juha Ruokolainen. Ruokolainen ja aiheen antaja Tarmo Räty huomasivat, että työ oli Juhalle liian suuri ja kysyivät minua mukaan (Räty 2012).

Työn tarkoituksena oli tehdä web-sovellus rakennuksen rakennusosien luontiin ja niiden käytettyjen materiaalien ympäristövaikutusten laskemiseen. Tässä vaiheessa ympäristövaikutusten laskeminen ja rakennusosien lisääminen oli tarkoitus tehdä vain elinkaaren vaiheista rakentamisen ajalle. Työn pohjana käytettiin Tarmo Rädyn Excel-taulukkoon tekemää ympäristölaskuria.

Minun aiheena oli suunnitella ja tehdä tietokanta sovelluksen käyttäjän rakennusosille sekä tehdä laskuri ympäristövaikutusten laskemiseen. Aiheeseen kuului myös sovelluksen käyttöliittymän suunnittelu ja toteutus, joka tehtiin Ruokolaisen kanssa yhdessä. Ruokolaisen pääkohteena oli materiaalitietokannan suunnittelu ja toteutus. Ruokolaisen opinnäytetyö on nimeltään Rakentamisen ympäristölaskuri (Ruokolainen 2014).

Sovellus tehtiin Metsäntutkimuslaitokselle, josta käytetään lyhennettä Metla. Metla on tutkimus ja asiantuntijaorganisaatio. Se kehittää tuotteita ja eri palveluja liittyen metsien hoitoon. (Metsäntutkimuslaitos 2014)

2 Ympäristölaskuri

2.1 Ympäristövaikutukset

Ympäristövaikutusten määrittelyä käytetään tuotteiden kestävyyden arvioinnin apuna. Aluksi tätä määrittelyä on käytetty mm. teknisten muutosten tekemiseen tuotteessa ja samankaltaisten materiaalien vertailussa. Ajan myötä se on laajentunut isompien käsitteiden arviointiin, kuten biopolttoaineiden ja rakentamisen ympäristövaikutusten määrittelyyn. Ympäristövaikutusten määrittäminen perustuu LCA:n (Life Cycle Assessment) menetelmiin. (Curran 2012, 1)

LCA-menetelmiin ISO (The International Organization of Standardization)-standardijärjestö on tehnyt oman ISO 14040-standardisarjan. Standardit on lajiteltu eri vaiheisiin seuraavasti:

- Määränpään ja laajuuden määrittäminen
- Inventaarioanalyysi
- Vaikutuksen elinkaaren määrittäminen
- Elinkaaren tulkitseminen.

Ympäristövaikutuksen määränpään ja laajuuden määrittämisessä keskitytään ympäristövaikutuksen liittyviin kysymyksiin ja asiayhteyteen. Tässä määritetään myös mistä vastauksia haetaan. Esimerkkejä määränpään kysymyksille ovat mm. tarkoitetut kohteet, missä niitä käytetään ja syyt, miksi tutkimusta jatketaan. Tässä vaiheessa ei vielä kerätä tietoja ja lasketa tuloksia. Laajuuden määrittämisessä on tärkeää määrittää ympäristövaikutuksen funktionaalinen yksikkö. Funktionaalinen yksikkö määrittää ympäristövaikutuksen toiminnan ja näin voidaan vertailla ympäristövaikutuksen suorituskykyä. Ympäristövaikutuksen laajuus asettaa pääalueet ympäristövaikutukselle, joita tarkennetaan muissa vaiheissa. (Curran 2012, 15–18.)

Inventaarioanalyysissä määritetään ympäristövaikutus ja tutkitaan vaikutuksen syötteet ja tulokset koko elinkaaren ajalta. Tässä analyysissä tapahtuu suurin osa laskemisesta. Elinkaaren inventaarioanalyysi perustuu yksikköprosessiin, jossa syötteet ja tulokset pitää määritellä tarkasti, jotta tulokset voidaan laskea ja ymmärtää oikein. Yksikköprosesseja on monenlaisia ja jotkut niistä linkittyvät toisiinsa muodostaen erilaisia tapahtumaketjuja. (Curran 2012, 18–19.)

Vaikutusten elinkaaren määrittämisessä tavoitteena on ymmärtää ja tarkastaa vaikutuksen laajuus ja tärkeys sen elinkaaren ajalta. Tämän määrittämisessä tarkastellaan inventaarioanalyysin eri prosessien tuloksia, joiden avulla määritettään mihin ympäristövaikutuskategoriaan vaikutus kuuluu. (Curran 2012, 22–23.)

Elinkaaren tulkitsemisessa vertaillaan aiempien tutkimusten tuloksia määritettyihin määränpäähän ja laajuuteen. Vertailuun valitaan inventaarioanalyysi tai vaikutuksen elinkaaren määrittäminen tai molemmat tutkimukset. Tästä saa-

daan lopputulokset ja merkittävät asiat. Tulkitsemisessa tehdään myös tiedonlaadunmäärittely ja tarkkuusanalyysi ja näin selviää tiedon rajoitukset. (Curran 2012, 27–28.)

2.2 Ympäristövaikutusten laskenta

Rakennuksen ympäristövaikutuksilla on suuret vaikutukset energian käyttöön, ilmastonmuutokseen ja rakennuksen käyttäjiin jne. Rakennuksen materiaalien ympäristövaikutusten laskemisella voidaan arvioida koko rakennuksen elinkaaren ympäristövaikutuksia. Arvioinnin perusteella voidaan vähentää haitallisia ympäristövaikutuksia ja rakentaa rakennuksen käyttäjille terveellisimpiä sekä vähemmän energiaa käyttäviä rakennuksia. (Curran 2012, 311–312.)

Sovelluksessa käytettävän elinkaaren vaiheet ovat rakentaminen, kunnossapito, kiinteistöhuolto, purkaminen ja koko elinkaaren aika. Tässä sovelluksessa ympäristövaikutukset saatiin materiaalitietokannan perusteella ja tässä sovelluksessa käytetään saksalaista Ökobau-materiaalitietokantaa (Informationsportal Nachhaltiges Bauen des BMUB, 2014). Rakennusmateriaalien ympäristövaikutukset ovat uusiutumaton primäärienergia, uusiutuva primäärienergia, polttoaineiden uudelleenkäyttö, veden käyttö, louhintajätteet, yhdyskuntajätteet, ongelmajäte, elottomien ympäristötekijöiden ehtyminen, rehevöityminen, otsonikato, fotokemiallisen otsonin muodostuminen, ilmastonmuutos sekä maaperän ja veden happamoituminen. (Räty 2012)

Sovelluksessa käytettävä laskentakaava on aika yksinkertainen, mutta sen laskemisessa pitää olla erittäin tarkka, koska rakennusosien hierarkiassa pitää osata erottaa pää- ja alirakennusosat. Rakennusosa on päärakennusosa, kun sille on määrätty yksi tai useampi rakennusosa aliosaksi. Aliosana voi olla myös jokin materiaali ja tällöin laskentavaiheessa sen arvo pitää hakea materiaalitietokannasta. Sovelluksen laskentaan käytettiin laskentakaavaa 1. Laskenta aloitetaan rakennusosahierarkian alimmalta tasolta nousten taso kerrallaan valittuun rakennusosaan. Kun laskenta alkaa alimmalta tasolta, niin ensimmäisten rakennusosien tulokset tulevat materiaalien perusteella ja näitä tuloksia käytetään ylempien tasojen rakennusosien ympäristövaikutusten laskentaan. (Räty 2012)

$$I^i = q_j^i c_j^i I_j \quad (1)$$

missä

I^i = päärakennusosan ympäristövaikutus
I_j = alirakennusosan ympäristövaikutus
q_j^i = alirakennusosan käytettävä määrä pääosaan nähden
c_j^i = elinikä aliosan ja pääosan suhteen

2.3 Rakennusosaeditorin toiminnot

Ohjelman päänäkymä on rakennusosaeditori, josta ohjelman käyttäjä pääsee näkemään valitun rakennusosan tietoja ja laskemaan sen ympäristövaikutukset. Rakennusosaeditorissa näkyy, mistä osista rakennusosa koostuu ja nämä osat voivat olla muita rakennusosia ja/tai rakennusmateriaaleja. Rakennusosaeditorissa näkyy rakennusosien ja/tai rakennusmateriaalien laskentaan käytettävät parametrit. Rakennusosan ympäristövaikutuksia voidaan tarkkailla sen kaikkien elinkaaren vaiheilta. (Räty 2012)

Ohjelman käyttäjä voi laskea rakennusosan ympäristövaikutukset ja tulokset tulevat näkyviin tällä sivulla. Käyttäjä pääsee myös luomaan, muokkaamaan tai poistamaan rakennusosia rakennusosaeditorista. Lasketetut ympäristövaikutukset voidaan tallentaa tiedostoon ja käyttäjä voi tehdä niillä tiedoilla mitä haluaa, esimerkiksi käyttäjä voi luoda erilaisia taulukoita havainnollistamaan rakennusosan ympäristövaikutuksia paremmin. (Räty 2012)

2.4 Tietokannat

Sovellus käyttää kahta tietokantaa: rakennusosatietokanta ja sovelluksen hallintaan käytettävää tietokantaa. Sovelluksen hallintaan käytettävässä tietokannassa on käyttäjien tiedot, rakennusmateriaalien ominaisuudet ja sovelluksen sivujen oikeuksia määräävä tieto (Ruokolainen 2014, 14).

Sovelluksessa käyttäjä tekee erilaisia rakennusosia ja nämä rakennusosat tallennetaan sovelluksen rakennusosatietokantaan. Tähän tietokantaan tallennetaan ympäristölaskemiseen käytettävät rakennusosan parametrit, muut rakennusosan tiedot, yhteydet rakennusosan materiaaleihin ja rakennusosien hiearkia. Rakennusosien hierarkiassa nähdään, mitkä rakennusosat kuuluvat mihin-

kin rakennusosiin eli rakennusosa voi olla pääosa, jolla on aliosia. Samaa rakennusosaa ei voi laittaa hierarkiassa aliosaksi, joka on ylempänä hierarkiassa pääosana, jotta laskentavaiheessa sovellus ei joudu ikuiseen silmukkaan.

3 Käytetyt tekniikat

Sovelluksen käytettävät tekniikat olivat määräytyneet jo ennen kuin tulin projektiin mukaan. Sovelluksen eri toimintoja tehtäessä päätettiin käyttää JavaScriptiin pohjautuvaa Ajax-menetelmää ja ympäristövaikutusten laskennassa päädyttiin MySQL-proseduurien käyttöön.

Ajax-menetelmää käytettiin, koska Yii-sovelluskehyksestä ei löydetty, miten web-sivun tiettyä kohtaa voi muuttaa, joka on jo muutettu Yii:n Ajax-toiminnolla. Laskenta tehtiin MySQL-proseduureilla, koska laskennan aikana haetaan tietoa tietokannasta monta kertaa. Yhteyden ottaminen monesti tietokantaan voi rasittaa web-palvelimen käyttöä aika paljon, jos laskenta olisi tehty PHP:lla.

3.1 PHP

PHP on yleiskäyttöinen ohjelmointikieli, jossa tietokoneen tiedostoon kirjoitetaan PHP:n käskyjä ja toimintoja, joita tietokone osaa toteuttaa PHP-tulkin avulla Web-selaimeen (Valade 2004, 10). PHP on rekursiivinen lyhenne nimelle PHP: Hypertext Preprocessor (The PHP Group 2014).

PHP on oliopohjainen ohjelmointikieli, jossa class-sanan avulla määritetään PHP-luokkia ja näiden luokkien toiminnot toteutetaan muuttujilla ja funktioilla. Funktiot määritetään funktion-nimellä ja muuttuja saadaan käyttöön, kun sen nimen eteen kirjoitettaan $-merkki. Kuvassa 1 näkyvät, miten luokkia, funktioita ja muuttujia muodostetaan sekä miten luokista tehdään olioita new-sanan avulla.

```php
<?php

class Hei {
    function sanoHei($nimi) {
        echo "Hei " . $nimi;
    }
}

$hei = new Hei();
$hei->sanoHei('php !');
```

Kuva 1. Esimerkki PHP:n toiminnoista

3.1.1 Model-View-Controller (MVC) ohjelmistoarkkitehtuuri

MVC-ohjelmistoarkkitehtuuri viittaa siihen, miten ohjelman toiminnot on eritelty toisistaan. Tässä erotetaan ohjelman tiedon hallinnointi malleina (Model), tiedon näyttäminen näkyminä (View) ja ohjelman logiikka kontrollereihin (Controller). (Lecky-Thompson, Eide-Goodman & Nowicki 2005, 242)

Nämä kolme osaa kommunikoivat keskenään niin, että kontrolleri hakee ja vie mallilta halutut tiedot. Sitten kontrolleri tekee tiedolle tarvittavat muutokset ja vie tiedon näkymään esille. Näkymästä ei oteta koskaan suoraa yhteyttä malliin, vaan se tehdään kontrollerin kautta. (Lecky-Thompson ym. 2005, 242)

Ohjelman ollessa rakennettuna eri komponenteista, näin niitä voidaan vaihtaa, kun sille on tarvetta, ja koko ohjelman rakennetta ei tarvitse muuttaa. Ohjelman virheiden etsiminen on myös helpompaa monesta pienestä komponentista kuin suuremmista komponenteista, vaikka niitä olisi vähemmän. MVC-arkkitehtuurin avulla on myös helpompaa käyttää uudelleen eri ohjelmakomponentteja ja näin samaa koodia ei tarvitse kirjoittaa uudelleen. (Lecky-Thompson ym. 2005, 242)

3.1.2 Yii-sovelluskehys

Yii:n nimi on lyhenne sanoille Yes, it is. Yii on kirjoitettu PHP:lla ja se komponenttipohjainen web-sovelluksen tekoon tarkoitettu sovelluskehys. Yii-sovellus käyttää MVC-arkkitehtuuriin pohjautuvaa sovellusrakennetta. (Winesett 2010, 8)

Yii:llä voi tehdä Ajax-menetelmää käyttäviä web-sivuja, liittää web-palveluja ja tehdä tietokantakyselyitä monella tapaa. Yii:ssä on hierarkkinen ja roolipohjai-

nen sivujen käyttöoikeuksien kontrollointi sekä sivujen kansainvälistäminen ja lokalisointi. Yii:llä voi kehittää ja ylläpitää isompia web-sovelluksia ja siihen voi tehdä itse lisäosia. (Winesett 2010, 8–9.)

3.2 HTML

HTML tulee sanoista Hypertext Markup Language ja tällä ohjelmointikielellä luodaan web-sivuja. HTML 5 on tämän kielen uusin versio ja HTML:ää ylläpitää World Wide Web Consortium -niminen organisaatio ja sen nimestä käytetään lyhennettä W3C. (Ducket & Larsen 2013, 2)

HTML-kielen varatut sanat ja ilmaisut laitetaan kulmasulkujen sisälle esim. <body></body> ja tämä on elementti, jossa on body-elementin aloitus- ja lopetusmerkkaus, joka tunnetaan nimellä ”tag”. Web-sivun näyttämiseen tarkoitettu dokumentti jäsennetään eri elementeillä ja web-sivujen näyttämiseen tarkoitetut ohjelmat osaavat näyttää web-sivut niin kuin ne on suunniteltu. (Ducket & Larsen 2013, 3–4.)

Web-sivut koostuvat yleensä kahdesta osasta, jotka ovat pää ja vartalo. Pääosaan laitetaan sivun otsikko ja kuvaus, jota käytetään kuvaamaan sivustoa. Vartaloon tulee sivun varsinainen sisältö, joka näytetään web-selaimessa. (Ducket & Larsen 2013, 6)

3.3 CSS

CSS-nimi tulee sanoista Cascading Style Sheets ja sitä käytetään web-sivun tyylien määrittämiseen. CSS-tyyleillä voi määrittää värin ja koon mm. tekstille ja eri viivoille sekä paikat ja välit eri elementtien välille web-sivulla. (Ducket & Larsen 2013, 191)

Tyylit määritetään CSS-sääntöjen (kuva 2) avulla, jossa ensimmäisenä näkyy mihin HTML-elementteihin sääntö on voimassa ja sen jälkeen säännön määrittely tulee aaltosulkujen sisälle. (Ducket & Larsen 2013, 192)

```
p {color: blue;}
```

Kuva 2. Esimerkki CSS-säännöstä, jossa määritetään p-elementin sisällön väri siniseksi

CSS-tyylejä voi lisätä web-sivuille kahdella eri tapaa. Yleisin tapa on kirjoittaa säännöt omaan tiedostoon ja linkittää se haluttuun HTML-tiedostoon tai kirjoittaa säännöt suoraan HTML-tiedoston styles-elementin sisälle. Molemmilla tavoille on omat hyötynsä, mutta sääntöjen kirjoittaminen omaan tiedostoon on yleensä parempi ratkaisu. Tyylien ylläpitäminen on helpompaa, kun ne ovat yhdessä paikassa. (Ducket & Larsen 2013, 197–198.)

3.4 JavaScript

JavaScript on oliopohjainen ohjelmointikieli, jossa nähdään HTML-tiedoston HTML-elementit ja selain eri objekteina ja sillä pystyy muokkaamaan web-sivuja JavaScript toimintojen avulla. JavaScriptin kehitti Netscape Communication Corporation ja se oli alun perin nimeltään LiveScript. Nimi muuttui JavaScriptiksi, kun Java-ohjelmointikieli alkoi saada suosiota, mutta Javalla ja JavaScriptillä ei ole kovin paljon yhteistä. (Ford 2008, 31–32.)

JavaScriptistä on luotu ECMAScript-standardi, jonka Netscape teki European Computer Manufacturing Association-järjestön kanssa. Standardi luotiin, koska Netscape kilpaili Microsoftin tekemän Jsciptin kanssa. ECMAScriptin valmistuttua suurin osa web-selaimista alkoi tukea JavaScriptiä paremmin, mutta vielä nykyään eri selaimista löytyy pieniä eroja miten JavaScriptiä suoritetaan. (Ford 2008, 32)

Ajax-menetelmän avulla web-sivun haluttuja kohtia voidaan päivittää saumattomasti ilman, että koko sivua ei pidä ladata uudelleen. Ajax-nimi tulee sanoista Asynchronous JavaScript and XML, koska se koostuu näistä tekniikoista. (Ford 2008, 3–5.)

Ajax-menetelmä perustuu XMLHttpRequest-luokan toimintoihin, jolla voidaan siirtää tietoa epäsynkronisesti web-selaimen ja web-palvelimen välillä. XMLHttpRequest-luokkaa käytetään JavaScriptin sisällä oliona ja se siirtää koostettua tietoa XML-formaatin mukaisesti web-selaimen ja web-palvelimen

välillä. Ajax-menetelmässä käytetään muita JavaScriptin toimintoja tiedon hallintaan web-sivulla. (Ford 2008, 6)

3.5 MySQL

MySQL-tietokannan kehittämisen aloitti ruotsalainen MySQL AB vuonna 1995 ja vuonna 2008 Sun Microsystems osti MySQL:n (Murphy & Cabral 2009, 4). Nykyään MySQL:ää kehittää Oracle Corporation, kun se osti Sun Microsystemsin vuonna 2010 (Oracle 2014).

MySQL-tietokannat käyttää relaatiotietokantamallia, jossa tietokannan tieto varastoidaan määrättyyn riviin ja sarakkeeseen, jotka muodostavat tietokantataulun. Eri taulujen välille voidaan määritellä monenlaisia yhteyksiä ja näin tietoa ei tarvitse tallentaa monta kertaa samaan tietokantaan. (Sheldon & Moes 2005, 6)

MySQL-tietokannassa voi myös suorittaa eri rutiineja eli aliohjelmia. Nämä rutiinit ovat liipaisimet, tapahtumat, proseduurit ja funktiot. Tapahtumat suoritetaan tiettyyn aikaan ja funktiot aloitetaan manuaalisesti kutsumalla sen nimeä. Proseduuri aloitetaan CALL-lauseella, jossa annetaan myös sen nimi ja mahdolliset parametrit. Rutiineilla voi tehdä eri SQL-käskyjä ja kyselyitä tietokantaan. (Murphy & Cabral 2009, 241)

Liipaisin eli englanniksi ”trigger” käynnistyy, kun tietokantaan tehdään jokin toiminto. Nämä toiminnot ovat lisäys-, päivitys- ja poistolauseet. Jokaisen liipaisimen alussa pitää vielä tarkentaa, että se suoritetaan ennen tai jälkeen tapahtuneen toiminnon. Tietokantataululla voi olla maksimissaan kuusi eri liipaisinta. (Murphy & Cabral 2009, 242)

3.6 Scrum-projektinhallintamenetelmä

Scrum ei ole vain ohjelmiston kehitystapa, vaan se on johtamisen kehys, jolla ohjelmistoa kehitetään. Scrum kuuluu niin sanottuihin ketteriin menetelmiin. Scrumin pohjana on mielekäs vuorovaikutus sääntöjen, järjestyksen, henkilökohtaisen vastuun, yhdessä ajattelemisen ja muiden auttamisen kanssa. (Opelt, Gloger & Pfarl 2013, 11–12.)

Projektin työntekijät ovat jaettu Scrumissa eri rooleihin, jotka ovat kehitystyöryhmä, tuotteenomistaja ja Scrum-mestari. Kehitystyöryhmä tekee varsinaisen tuotteen ja arvioivat mitä he voivat tehdä ja näin ovat vastuussa tuotteen laadusta. Tuotteenomistaja ohjaa kehitystyöryhmää mihin suuntaan tuotetta kehitetään ja on vastuussa, että tuote on sellainen kuin halutaan. Scrum-mestari auttaa kehitystyöryhmää sen määränpäitten saavuttamisessa ja varmistaa Scrum-prosessin toteutumisen, mutta hän ei anna suoria käskyjä miten tehdä töitä. (Opelt ym. 2013, 16)

Scrumilla työskentely tapahtuu ajanjaksossa, joka on nimeltään Sprint. Sprint on maksimissaan 30 päivän pituinen. Ennen sprintin alkua pidetään kaksi palaveria, joissa päätetään mitä sprintissä tehdään ja miten se tehdään. Sprintin ajan joka päivä pidetään palaveri, jossa työryhmän jäsenet kertovat mitä he ovat tekemässä ja mahdolliset ongelmat Scrum-mestarille. Tämän palaverin nimi on Daily Scrum. Daily Scrumin pituus maksimissaan on n. 15 minuuttia. Sprintin lopussa on sen päätöspalaveri, jossa esitetään sprintin tulokset. (Opelt ym. 2013, 18–19.)

3.7 CSV-tiedosto

CSV tulee sanoista Comma Separated Values eli pilkulla erotetut arvot. Arvot tallennetaan tavalliseen tekstitiedostoon ja sen voi avata esimerkiksi Notepad-ohjelmalla. (Backx & Gelineau 2012, 54)

CSV-tiedoston arvot tallennetaan sarakkeisiin ja nämä arvot erotetaan pilkulla. CSV-nimestä huolimatta arvojen erottimena voi olla jokin muu merkki ja muita yleisempiä erottimia ovat sarkain, välilyönti ja puolipiste. (McNeil 2010, 98).

4 Toteutus

4.1 Suunnittelu

Sovelluskehitys aloitettiin rakennusosatietokannan suunnittelulla. Suunnittelu tehtiin MySQL Workbench-ohjelmalla. Tietokannan suunnittelussa käytettiin pohjana Tarmo Rädyn Excel-taulukkoon tekemää laskuria, josta nähtiin mitä

tietoa käyttäjä antaa. Räty antoi myös palautetta ja lisäyksiä mitä tietokannassa pitää olla.

Rakennusosaeditorin käyttöliittymän suunnittelu tehtiin Ruokolaisen ja Rädyn kanssa yhdessä. Käyttöliittymä suunniteltiin ensimmäiseksi paperille, josta Räty antoi palautetta ja lisäysehdotuksia. Kun rakennusosaeditorin suunnittelusta päästiin yhteisymmärrykseen, niin sitten vasta aloitettiin koodaaminen Yii-sovelluskehyksen avulla.

Rakennusosan lisäykseen Rädyllä oli ideana, että käyttäjä voi tehdä uuden rakennusosan jo olemassa olevan osan pohjalta tai kokonaan alusta. Ensimmäiseksi suunniteltiin tämän valintaan tarvittava sivu ja seuraavaksi suunniteltiin uuden rakennusosan luonti tyhjältä pohjalta. Aluksi mallia otettiin Rädyn tekemästä Excel-taulukosta, josta nähtiin mitä arvoja ja tietoja käyttäjän pitää antaa ja tästä tehtiin ensimmäinen versio rakennusosan luonnille. Rakennusosan luonnin suunnittelu tapahtui periaatteessa kolmessa osassa. Ensimmäisenä oli rakennusosan perustietojen lisäys, toisena tuli alirakennusosien lisäys ja kolmantena rakennusmateriaalien lisäys.

Minä toteutin suunnittelun ohjelman toiminnoista rakennusosan luonnin, ympäristövaikutusten laskennan, rakennusosan poiston ja laskentatulosten tiedostoon viennin osalta. Muitten toimintojen suunnittelun ja toteutuksen teki Juha Ruokolainen. Jokainen toiminto tehtiin syklissä, jossa ensin Räty kertoi mitä toiminnon pitäisi sisältää ja tehdä ja sen perusteella suunniteltiin toiminta ohjelmaan MVC-arkkitehtuurin pohjautuen.

4.2 Toimintaympäristö

Sovelluskehitys toteutettiin Metlan tiloissa. Ruokolaisen kanssa saatiin yhteinen työhuone ja omat tietokoneet sovelluskehitystä varten.

Sovelluskehityksen hallintaan käytettiin Scrum-projektinhallintamenetelmää. Scrumista käytettiin aika riisuttua versiota. Ensimmäiseksi kirjattiin sovelluksen toiminnot ja sitten ne pilkottiin pienempiin osiin työtehtäviksi ja nämä kirjoitettiin PostIt-lapuille. PostIt-laput laitettiin näkyville Scrumin työtehtävätaululle, josta

nähtiin mitä tehtäviä oli jäljellä, mitä oltiin tekemässä ja mitä oli tehty. Lapuille kirjattiin kuka oli tehnyt tehtävän ja käytetty työaika.

Scrumista käytettiin näin riisuttua versiota, koska sovellus kehitettiin Ruokolaisen kanssa yhdessä. Projektin aikana kommunikointi pysyi hyvänä Ruokolaisen kanssa, koska näimme tehtävätaulusta mitä kumpikin oli tekemässä ja ongelmatilanteissa pystyttiin tulemaan nopeasti toisen avuksi.

4.3 Sovelluskehitysympäristö

Sovelluksen kehitykseen käytettiin paikallista tietokonetta, jolla pystyi kehittämään web-sovellusta. Koneelle asennettiin Apache web-palvelin, MySQL-tietokantapalvelin ja PHP-tulkki, jotta web-sovellusta voidaan testata tietokoneella. Tarkemmat versiot voi nähdä liitteen 1 kohdasta 2.5.

Sovelluskehitykseen käytettiin NetBeans-ohjelmointiympäristöä. Sovellus kirjoitettiin pääosin PHP-kielellä ja käytetimme PHP:lla toteutettua Yii-sovelluskehystä sovelluksen toteutukseen. NetBeansissa käytettiin PHP:n testaamiseen Xdebug-lisäosaa. Sovelluksen versionhallintaan käytettiin TortoiseSVN-ohjelmaa. Sovelluksen tietokannan kehitys tehtiin MySQL Workbench-ohjelmalla. JavaScriptin testaamiseen Mozilla Firefox-selaimessa käytettiin Firebug-lisäosaa.

4.4 Rakennusosatietokanta

Tietokannan toteutukseen käytettävässä MySQL Workbench-ohjelmassa on oma tietokannan suunnittelualue, jossa tehtiin rakennusosille tietokantamalli ja sen jälkeen tehtiin kyseisen mallin avulla .sql tiedosto, jolla luodaan varsinainen tietokanta tietokantapalvelimeen. Tietokantamallissa näkyy tietokannan taulut ja yhteydet. Yhteydet ovat taulujen vierasavainten määrittelyjen mukaisesti näkyvillä ja nämä yhteydet oli helppo määritellä MySQL Workbench-ohjelmalla. Rakennusosatietokannasta piti tehdä yhteydet sovelluksen toiseen tietokantaan, koska sieltä linkitetään käyttäjä ja käytettävät materiaalit rakennusosaan.

Ympäristövaikutusten laskenta tehtiin tietokannan puolelle, koska käyttäjän antamat arvot ja materiaalien ympäristövaikutusarvot ovat tietokannan puolella. Web-palvelin ei joudu muodostaan yhteyttä tietokantapalvelimeen monta kertaa, ja näin nopeuttaa sovelluksen toimintaa. Ympäristövaikutusten laskenta toteutettiin proseduureilla.

Muut tietokantakyselyt tehtiin PHP-kielellä Yii-sovelluskehyksen avulla. Yii-sovelluksessa on oma asetustiedosto, jonne laitettiin tietokantayhteyden muodostamista varten tarvittavat parametrit molemmille tietokannoille. MVC-arkkitehtuuriin pohjautuen Yii-sovelluksessa tietokantakyselyt tehtiin models-kansion alle. Sinne tehtiin jokaista tietokantataulua vastaava PHP-luokka, joihin tehtiin taulukohtaiset tietokantakyselyt. Yii:ssä on monia tapoja tehdä tietokantakysely, mutta kuvasta 3 näkee tavan, jolla tehtiin suurin osa kyselyistä.

```
$id = Yii::app()->db2->createCommand()
        ->select('structural_element_id as id')
        ->from(constant('ELEMENT_TABLE'))
        ->where("name = '$name' AND user_id = $user_id")
        ->queryRow();
```

Kuva 3. Esimerkki tietokantakyselystä

4.5 Käyttöliittymä

Sovelluksen käyttöliittymän näkymät toteutettiin Yii:n sovellushierarkiassa views-kansioon. Näkymät kirjoitettiin pääosin PHP-ohjelmointikielellä, mutta jossain kohdin oli helpompaa käyttää suoraan HTML-kieltä. Näkymien toteutuksessa käytettiin aika paljon Yii-sovelluskehyksen eri toimintoja, mutta erityisesti CHtml-luokan toimintoja (kuva 4).

```
echo CHtml::beginForm('ViewElement', 'post');
    echo CHtml::hiddenField('element', $tempParts['structural_element_id']);
    echo CHtml::hiddenField('calc', 1);
    echo CHtml::submitButton(Yii::t('translation', 'Calculate impacts'));
echo CHtml::endForm();
```

Kuva 4. CHtml-luokan toimintoja

Juha Ruokolaisen kanssa toteutettiin rakennusosaeditorin käyttöliittymä, jossa minä tein rakennusosan perustietojen ja muitten toimintojen painikkeiden sijoit-

tamisen editoriin ja Ruokolainen toteutti rakennusosaan kuuluvien osien ja niiden ympäristövaikutusten näyttämisen rakennusosaeditorissa.

Toteutin rakennusosaeditorin käyttöliittymään linkin rakennusosan luontiin ja hallintaan käytettävät painikkeet eli muokkaa ja poista. Tein rakennusosan poistoon JavaScriptillä varmistuskysymyksen popup-ikkunaan. Rakennusosan muokkaamiseen Ruokolainen teki uuden näkymän tekemäni rakennusosan luonnin pohjalta, mutta tällä sivulla tulee näkyviin myös rakennusosan tiedot, joita voi muokata.

Käyttöliittymän tyyli tehtiin CSS-tyylien avulla. Tyylejä käytettiin aika pitkälle Yii:n omien tyylien pohjalta. Värien käytön osalta Räty halusi käyttää Metlan värejä ja Räty selvitti meille värikoodit mitä käytetään. Ruokolainen teki CSS-tiedoston meidän tyylien toteutukseen ja minä tein omat toiminnot tähän tiedostoon ja näin kaikki tyyleihin liittyvät yksityiskohdat ovat omissa CSS-tiedostoissa.

4.6 Sovelluksen toimintojen toteutus

Sovelluksen toiminnot toteutettiin MVC-arkkitehtuurin pohjalta Yii sovelluksessa controllers-kansion alle. Tänne luotiin jokaiselle näkymälle oma kontrolleri PHP-luokkana. Sovelluksessa käytettävien dynaamisten alasvetovalikoiden hallintaan tehtiin oma kontrolleri, koska sovelluksessa käytetään aika monta alasvetovalikkoa.

Sovelluksen toiminnat kirjoitettiin PHP:n funktioihin kontrolleriin. Funktioihin tuli ohjelman looginen toiminta ja se mitä tarvitaan eri näkymien näyttämiseen.

Rakennusosan materiaalien dynaamiseen lisäykseen käytettiin Ajax-menetelmää ja nämä funktiot tulivat omaan JavaScript-tiedostoon. Yii:ssä voi käyttää Ajax-parametreja eri komponenttien yhteydessä Ajaxin toteuttamiseen. Yii:n Ajax-menetelmää ei saatu toimimaan oikein materiaalin lisäyksessä koska, koska Yii luo sivulle tulevat JavaScript-käskyt sivun ladatessa. Halusin lisätä Ajax-toimintoja vielä sivun lataamisen jälkeen, joten ne piti luoda itse ilman Yii:n apua.

5 Tulokset

5.1 Laskenta

Ympäristövaikutusten laskenta tapahtuu PHP:n puolella yhdellä käskyllä (kuva 5), joka ohjaa käskyn tietokantapalvelimen puolelle ja suorittaa calculate_main-proseduurin. Liitteen 1 kohdasta 3.3.5 näkee tarkemmin miten laskentaa etenee ja tapahtuu tietokannan proseduureissa. Tulokset siirretään niiden näyttämistä varten erilaiseen taulukko järjestykseen ja samalla ne pyöristetään ceil-funktiolla sekä muutetaan luvun 10 logaritmiin log10-funktiolla.

```
$result = Yii::app()->db2->CreateCommand("CALL calculate_main($id);")->queryAll();
$values = array();
$t = 0;
foreach ($result as $array) {
    $tmp = array();
    foreach ($array as $key => $value) {
        $tmp[$key] = round($value, ceil(0-log10(doubleval($value))+5));
    }
    $values[$t]=$tmp;
    $t++;
}
```

Kuva 5. Ympäristövaikutusten laskeminen ja tulosten pyöristäminen

Tulokset saadaan näkyville rakennusosa editorin näkymässä (kuva 6). Tulokset näytetään osakohtaisesti ja alimmalla rivillä on yhteenlasketut tulokset.

CO2 equiv	Embodied primary energy (MJ)			
GWP*	Nonrenewable	Renewable	Secondary fuels	Use of net fresh water
13.085	105.255	4.80258	0.00180985	16.9602
0	810228	1341020	98.348	49561200

Kuva 6. Laskentatulosten esimerkki

Lasketut tulokset voidaan tallentaa CSV-tiedostoon (kuva 7) ja tämä tiedosto voidaan avata esimerkiksi Excel-ohjelmalla. Tiedostoon tallentamista varten on tehty painike, joka tulee käyttöön, kun laskenta on tehty valitulle rakennusosalle rakennusosa editori näkymässä.

```
1341020;98,3498;49561200
```

Kuva 7. CSV-tiedosto tavallisena tekstitiedostona

5.2 Rakennusosatietokanta

Käyttäjän tekemät rakennusosat tallennetaan rakennusosatietokantaan, jonka tarkempi kuvaus on liitteen 1 kohdissa 3.1 ja 3.3. Tietokantakyselyt tehtiin suurimmaksi osaksi Yii-sovelluskehyksen avulla. Tietokantatauluista tehtiin vastaavat PHP-luokat, joissa näkyy tauluun liittyvät kentät, tauluun liittyviä sääntöjä ja tarvittavat tietokanta kyselyt.

Esimerkkinä katsotaan pena_slave_db-tietokannan structural_element-taulun ominaisuuksia. Kuvasta 8 nähdään, että tehty PHP-luokka laajentaa Yii:n CActiveRecord-luokkaa, koska näin voidaan myöhemmin hyödyntää Yii:n ominaisuuksia tiedon tallentamisessa tietokantaan. Kuvassa 8 näkyvät myös tarvittavat tietokannan kentät PHP-muuttujina.

```
class BuildingElement extends CActiveRecord{

    public $user_id;
    public $name;
    public $lifetime;
    public $talo80_element_id;
    public $talo2000_element_id;
    public $description;
```

Kuva 8. PHP-luokka structural_element-taululle

Tietokantatauluihin liittyviä rajoituksia laitettiin rules-funktion alle. Tätä funktiota käytetään ennen kun tiedot tallennetaan tietokantaan, eli sääntöjen perusteella tarkastetaan, että tietokantaan vietävät tiedot ovat sääntöjen mukaiset. Kuvassa 9 näkyy BuildingsElement-luokkaan liittyvät säännöt, jossa nähdään pakolliset kentät required-parametrilla ja match-parametrilla rajataan lifetime-kentän annettavat arvot numeroihin.

```
/**
 * Declares the validation rules.
 * @return array validation rules for model attributes.
 */
public function rules() {
    return array(
        array('name, lifetime', 'required'),
        array('lifetime', 'match', 'pattern' => '/^[0-9]/'),
    );
}
```

Kuva 9. BuildingsElement-luokan rules-funktio

Lopuksi tehtiin tauluun tarvittavat tietokantakyselyt PHP-funktioina. Kuvassa 10 näkyy yksi funktio, jossa structural_element-taulusta haetaan käyttäjän rakennusosien nimet käyttäjänumeron perusteella.

```
public function Load_element_names($user_id) {
    $names = Yii::app()->db2->createCommand()
            ->select('structural_element_id, name')
            ->from(constant('ELEMENT_TABLE'))
            ->where("user_id = $user_id")
            ->queryAll();
    Yii::app()->db2->setActive(false);
    return $names;
}
```

Kuva 10. Esimerkki PHP-funktioon tehdystä tietokantakyselystä

5.3 Rakennusosaeditori

Rakennusosaeditori tehtiin Ruokolaisen kanssa yhdessä. Ruokolainen teki valitun rakennusosan aliosien tietojen ja ympäristövaikutusten näyttämisen (Ruokolainen 2014, 19). Tämän näkymän selostus löytyy liitteen 1 kohdasta 4.2.4.

Rakennusosaeditorin näyttämiseen kontrollerin puolelta Yii käyttää render-funktiota (kuva 11). Siihen laitettiin näkymän nimi ilman .php-päätettä ja näytettävät tiedot erilliseen taulukkoon. Tämän funktion liitetyt tiedot pitää ensin hakea tietokannasta ja tehdä tarvittavat toiminnot, että tiedot näkyvät oikein ja halutusti.

```
$this->render('elementEditorView', array(
    'tempParts'=>$tempParts,
    'parameters'=>$parameters,
    'children'=>$children,
    'elements_data'=>$elements_data,
    'selected_element'=>$element,
    'result'=>$result,
    'save_button'=>$save_button,
    'auto_calculate'=>$auto_calculate,
));
```

Kuva 11. Esimerkki render-funktiosta

Näkymätiedostossa käytettäviä tietoja voidaan käyttää nimillä, jotka ovat vasemmalla heittomerkkien välissä, esimerkiksi element-muuttujan tieto saadaan näkyville selected_element-nimellä.

5.4 Rakennusosan luonti

Rakennusosan luonnin näkymän selostus näkyy liitteen 1 kohdassa 4.2.5. Rakennusosan luonnissa käytettiin eri ominaisuuksien määrittelyyn alasvetovalikoiden ohjaamiseen Ajax-menetelmää. Yii:ssä voi käyttää Ajax-parametria eri CHtml-toimintojen yhteydessä ja kuvassa 12 näkyy miten sitä käytettiin.

```
echo CHtml::dropDownList('talo80_1', '', $talo80_data,
        array(
            'prompt'=>'Element division 1',
         'ajax' => array(
         'type'=>'POST', //request type
         'url'=>CController::createUrl('DropDownlist/talo80_1'),
         'update'=>'#talo80_2', //selector to update
    )));
echo "<br/>\n";
echo CHtml::dropDownList('talo80_2','', array(),
        array(
```

Kuva 12. Ajaxin käyttö YII:llä

Yii:n Ajax-menetelmässä url-parametri ohjaa missä haluttu toiminta tehdään, kun alasvetovalikosta on valittu jokin arvo. Tässä tapauksessa toiminta on talo80_1-funktiossa (kuva 13), joka on DropDownlist-kontrollerissa. Tässä funktiossa haetaan talo80-määrittelyn toisen tason nimet valitun alasvetovalikon arvon perusteella. Toiminnan tulos viedään update-parametrin mukaisesti ”talo80_2” nimiseen alasvetovalikkoon.

```
public function actionTalo80_1() {
    $query = new Talo80Element();
    $data = $query->Load_2nd_level_names($_POST['talo80_1']);

    foreach ($data as $key => $value) {
        echo CHtml::tag('option',
                array('value'=>$key),  CHtml::encode($value),true);
    }
}
```

Kuva 13. Talo80_1-funktion toiminta

Rakennusosamateriaalien lisäys tehtiin myös Ajax-menetelmällä, mutta se tehtiin käsin JavaScript-kielellä. Materiaalien lisäyksessä käytettävien alasvetovalikoiden yhteydessä käytettiin onclick-parametria, joka ohjaa haluttuun JavaScript funktioon. Kuvassa 14 näkyy esimerkki funktiosta, joka toteuttaa Ajax-menetelmän.

```
function Gabi1(url, g1)
{
    var xmlhttp;
    if (window.XMLHttpRequest)
    {// code for IE7+, Firefox, Chrome, Opera, Safari
        xmlhttp=new XMLHttpRequest();
    }
    else
    {// code for IE6, IE5
        xmlhttp=new ActiveXObject("Microsoft.XMLHTTP");
    }
    xmlhttp.onreadystatechange=function()
    {
        if (xmlhttp.readyState==4 && xmlhttp.status==200)
        {
            var parent=document.getElementById('gabi_2');
            var child=document.getElementById("gabi2");

            var new_select = document.createElement('div');
            new_select.setAttribute('id', 'gabi2');
            new_select.innerHTML = xmlhttp.responseText;

            parent.removeChild(child);
            parent.appendChild(new_select);
        }
    }
    xmlhttp.open("GET",url+"?g1="+g1,true);
    xmlhttp.send();
}
```

Kuva 14. Esimerkki JavaScriptin Ajax-menetelmästä

5.5 Tietoturvatarkastukset

Käyttäjän antamat syötteet tarkistettiin Yii-sovelluskehyksen antamien tapojen avulla. Jokaisesta käyttäjän täyttämästä lomakkeesta tehtiin oma PHP-luokka Yii-sovellushierarkiassa models-kansioon.

Lomakkeesta tehtävä PHP-luokka laajentaa Yii:n CFormModel-luokkaa. Lomakkeessa käytettävät kentät toteutetaan PHP-muuttujina. Syötteiden tarkistukseen käytettävät säännöt tehdään omaan rules-funktioon, johon voi määrittää mm. mitkä kentät ovat pakollisia, mitä merkkejä kenttiin saa laittaa sekä syötteen maksimi- ja minimipituudet. Tämä rules-funktio muistuttaa hyvin paljon samalta kuin tietokantataulusta tehtyjen PHP-luokkien rules-funktiota, koska ne käyttävät samoja sääntöjä syötteiden tarkistamiseen.

Jotta syötteiden tarkistus tapahtuu oikein, niin lomakkeesta tehdyn PHP-luokan muuttujien nimiä pitää käyttää myös näkymätiedostojen syötekenttien niminä. Yii laittaa näkyviin muuttujan nimen lomakkeen syötekentän yläpuolelle ja muut-

taa ensimmäisen kirjaimen isoksi kirjaimeksi. Lomakkeen PHP-luokkiin tehtiin attributeLabels-funktio (kuva 15), johon laitettiin moniosaisten muuttujanimien vastaavat näkyviin tulevat syötekenttien nimet.

```
/**
* @return array customized attribute labels (name=>label)
*/
public function attributeLabels()
{
    return array(

        'talo80_element_id' => 'Talo80',
        'talo2000_element_id' => 'Talo2000',
        'service_life' => 'Service life (years)',
        'quantity_of_cycles' => 'Quantity of cycles'
    );
}
```

Kuva 15. Esimerkki attributelabels-funktiosta

6 Pohdinta

Sovellus tehtiin Metlan tiloissa ja siellä käytiin tekemässä töitä tiettymäärä tunteja viikossa. Aluksi oli jonkin verran sopeutumista, kun siinä samalla kävin myös koulussa. Työt pystyin kuitenkin järjestelemään koulun lukujärjestyksen ympärille ja näin ei tarvinnut jäädä pois koulun tunneilta.

Sovelluksen kehittäminen oli luontevaa eri ohjelmointiympäristöillä, joista pääasiassa tuli käytettyä NetBeans-ohjelmaa PHP-ohjelmointikielen hallinnointiin sekä MySQL Workbench-ohjelmaa tietokantojen hallinnointiin.

Suurimmat ongelmat sovelluksen kehittämisen aikana olivat ympäristövaikutusten laskemisen kehittämisen aikana, kun laskennan tulokset eivät täsmänneen käsin tehtyjen esimerkkilaskujen kanssa. Tulosten vertailussa näki, että ne olivat kuitenkin oikean suuruisia, mutta ei kuitenkaan täysin samanlaisia. Ongelma selvisi kun etsittiin eri lukujen tietotyyppejä MySQL:n lähdemanuaalista, kun siellä oli artikkeli tarkkuuslaskemisesta. Siitä selvisi, että käytetty float-tietotyyppi ei ole tarkoitettu tämänkaltaiseen tarkkuuslaskentaan, kun se pyöristää arvoja sille asetettujen sääntöjen perusteella. Tämän artikkelin ohjeiden mukaisesti käytettiin DECIMAL-tietotyyppiä, joka ei pyöristä lukuja ja laskut alkoivat näyttää oikeita tuloksia vertaillessa käsin tehtyihin esimerkkilaskuihin.

Toisena suurena ongelma oli sovelluksen siirtäminen Windows-käyttöjärjestelmästä Linux-käyttöjärjestelmään. Windows-järjestelmä oli kehitysympäristön alusta ja Linux-järjestelmä on tuleva julkaisualusta. Ongelmaksi paljastui tietokannan kyselyiden erilaiset käsittelyt MySQL-palvelimen osalta näiden kahden käyttöjärjestelmän kesken. Myöhemmin opittiin, että Windows-järjestelmässä MySQL pakottaa kaikkien tietokantojen ja taulujen nimien olevan kirjoitettu pienillä kirjaimilla, mutta kyselyissä kirjainkoolla ei ole väliä. Linux-järjestelmässä MySQL ei pakota tietokannan ja taulujen nimiä pieniin kirjaimiin, mutta kyselyissä kirjaintenkoolla on merkitystä. Meidän itsemme tekemät tietokantakyselyt olivat kaikki kirjoitettu pienillä kirjaimilla, mutta Yii-sovelluskehyksen tekemissä kyselyissä oli taulujen nimiä, joissa oli isoja ja pieniä kirjaimia. Yii:n lähdekoodista löydettiin missä nämä tietokannantaulujen nimet ovat ja ne muutettiin pieniksi kirjaimiksi ja sen jälkeen sovellus alkoi toimia Linux-järjestelmässä. Tuloksena tähän ongelmaan on kehitysympäristön asetusten laittaminen samaksi kuin sovelluksen julkaisuympäristön asetukset tai muuttamalla Yii:n lähdekoodia. Lähdekoodin muuttaminen ei kuullosta kovin järkevältä ratkaisulta, jos päätetään tulevaisuudessa päivittää Yii uudempaan versioon. Löysimme nämä eroavaisuudet näiden kahden käyttöjärjestelmän kesken aika myöhään sovelluksen kehittämisessä ja päädyimme muuttamaan Yii:n lähdekoodia.

Sovelluksen kehittämiseen olisi voinut ottaa vielä kolmannen henkilön, jos sen kehityksen olisi rajannut materiaalitietokantaan, rakennusosatietokantaan ja laskentaan sekä kolmas osuus olisi sovelluksen käyttöliittymä. Suunnitelmissa oli tehdä ns. taloeditori-näkymä, jossa näkyisi kaikki rakennusosat ja niiden materiaalit rakennusosien hierarkian mukaisesti. Tätä näkymää ei keritty koskaan aloittaa, koska muiden ominaisuuksien kehittämisessä kesti niin kauan ja rakennusosaeditorin kehittäminen oli tärkeämpää. Kolmannella henkilöllä olisi ollut näin aika paljon tekemistä sovelluksen käyttöliittymän kanssa.

Saimme myös palautetta Metlan IT-tuen puolelta, että sovellus ei olisi tarpeeksi turvallinen tietoturvan osalta. Niiden mielestä sovellus ei ollut turvallinen, koska se päästi tallentamaan tietoa palvelimelle, mutta käyttäjän rakennusosat pitää tallentaa tietokantaan. Ruokolaisen kanssa testattiin myös syötteiden tarkistuk-

sia ja SQL-injektiota. Sovelluksen kirjautumissivulta ei päässyt SQL-injektiolla kirjautumaan, mutta löysimme joitakin puutteita muiden lomakkeiden syötteiden tarkistuksessa ja ne korjattiin.

Sovelluksen jatkokehitysajatuksia on mm. ohjelman ja laskennan laajentaminen muille elinkaaren vaiheille, taloeditorin suunnittelu ja toteutus. Muita tarpeita on materiaalitietokannan tietojen kääntäminen englanniksi ja suomeksi sekä ohjelman käyttöliittymän kääntäminen suomeksi ja saksan kielelle. Sovellukseen pitäisi saada myös paremmat käyttäjätietojen hallinto-ominaisuudet.

Lähteet

Backx, P. & Gélineau, D. 2012. ActionScript Graphing Cookbook. Olton, Birmingham, GBR. Packt Publishing Ltd.

Curran, M. 2012. Life Cycle Assessment Handbook : A Guide for Environmentally Sustainable Products. Somerset, NJ, USA. Wiley.

Ducket, J. & Larsen R. 2013. Beginning HTML and CSS. . Somerset, NJ, USA. Wiley.

Ford, J. 2008. Ajax Programming for the Absolute Beginner. Boston, MA, USA. Course Technology / Cengage Learning.

Informationsportal Nachhaltiges Bauen des BMUB. 2014. Ökobau.dat. Saatavilla: http://www.nachhaltigesbauen.de/baustoff-und-gebaeudedaten/oekobaudat.html. 13.5.2014.

Lecky-Thompson, E., Eide-Goodman, H. & Nowicki, S. 2005. Professional PHP5. Hoboken, NJ, USA. Wiley.

McNeil, J. 2010. Python 2.6 Text Processing: Beginners Guide. Olton, Birmingham, GBR. Packt Publishing Ltd.

Metsäntutkimuslaitos. 2014. Metsäntutkimuslaitos (Metla). Saatavuus: http://www.metla.fi/metla/. 30.4.2014.

Murphy, K. & Cabral, S. 2009. MySQL Administrator's Bible. Hoboken, NJ, USA. Wiley.

Opelt, A., Gloger, B. & Pfarl, W. 2013. Wiley Series in Systems Engineering and Management : Agile Contracts : Creating and Managing Successful Projects with Scrum. Somerset, NJ, USA. Wiley.

Oracle. 2014. Oracle and Sun Microsystems. Saatavuus: http://www.oracle.com/us/sun/index.htm. 8.5.2014.

Ruokolainen, J. 2014. Rakentamisen ympäristölaskuri. Karelia-ammattikorkeakoulu. Tietotekniikan koulutusohjelma. Opinnäytetyö. Käsikirjoitus.

Räty, T. 2012. Tutkija. Metsäntutkimuslaitos. Palaveri marraskuu 2012.

Sheldon, R, & Moes, G. 2005. Beginning MySQL. Hoboken, NJ, USA. Wiley.

The PHP Group. 2014. What is PHP?. Saatavuus: http://www.php.net/manual/en/intro-whatis.php. 6.5.2014

Valade, J. 2004. PHP 5 for Dummies. Hoboken, NJ, USA. Wiley.

Winesett, J. 2010. Agile Web Application Development with Yii1.1 and PHP5 : Fast-track Your Web Application Development by Harnessing the Power of the Yii PHP Framework. Olton, Birmingham, GBR. Packt Publishing Ltd.

KARELIA-AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

Simo Kunnari 0901618
Juha Ruokolainen 1301553

RAKENTAMISEN YMPÄRISTÖLASKURI

Toiminnallinen määrittely
Kevät 2014

Karelia-amk	Ohjelmistotekniikka	Rakentamisen ympäristölaskuri
Määrittelyn tekijät: Simo Kunnari ja Juha Ruokolainen		Tulostettu:
Jakelu: Simo Kunnari, Juha Ruokolainen ja Tarmo Räty		
Dokumentin tila: Valmis		Muokattu: 15.5.2014

VERSIOHISTORIA:

Versio	Päiväys	Tekijät	Selite (muutokset, korjaukset...)
0.1	19.12.2012	Kunnari, Ruokolainen	Luvut 1 ja 2
0.2	20.12.2012	Kunnari, Ruokolainen	Osa luvusta 3
0.3	21.12.2012	Ruokolainen	Luvulle 3 jatkoa
0.4	3.1.2013	Kunnari, Ruokolainen	Edelleen jatkoa luvulle 3
0.41	7.1.2013	Ruokolainen	Jatkoa luvulle 3
0.5	10.1.2013	Kunnari, Ruokolainen	Luku 3 loppuun
0.6	14.1.2013	Kunnari, Ruokolainen	Luvun 4 alkuosa
0.7	15.1.2013	Kunnari, Ruokolainen	Jatkoa luvulle 4
0.71	16.1.2013	Kunnari	Luku 4 loppuun
0.8	16.1.2013	Kunnari	Luvut 5, 6, 7 ja 8
0.81	4.2.2013	Kunnari	Päivityksiä lukuihin 6.3 ja 7.2
0.82	11.2.2013	Ruokolainen	Koko dokumentin oikoluku
0.85	12.3.2014	Ruokolainen	Pieniä korjauksia tekstiin ja muotoiluihin
1.0	15.5.2014	Kunnari, Ruokolainen	Koko dokumentin päivittäminen ajan tasalle

SISÄLTÖ

1 JOHDANTO

1.1 Tarkoitus ja kattavuus

Tässä dokumentissa kuvataan ’Rakentamisen ympäristölaskuri’ -nimisen web-sovelluksen toiminnallisuudet ja tietosisältö. Dokumentissa käydään läpi kaikki ohjelman toiminnot.

1.2 Tuote

Päämääränä on toteuttaa Metsäntutkimuslaitokselle sovellus, jolla uudisrakennuksen suunnittelija voi laskea rakennuksen tai rakennusosan ympäristövaikutukset koko rakennuksen tai rakennusosan elinkaaren ajalta. Ohjelma toteutetaan web-sovelluksena. Käyttäjän määrittelemät rakennusosat tulee voida tallentaa sovelluksen tietokantaan ja laskelmat tiedostoon.

1.3 Määritelmät, termit ja lyhenteet

Taulukko 1: Dokumentissa käytettävät merkintätavat

Lihavointi	välilehtien nimet toimintojen nimet valikon kohdat / nimet painikkeet tietokannan kenttien nimet
Kursivointi	tietokannan taulujen nimet tietokannan proseduurien nimet tiedostokansioiden nimet tiedostojen nimet
Lihavointi ja kursivointi	käyttäjän syötteet
alleviivaus	linkit
ISOILLA KIRJAIMILLA	tietovarastojen nimet
[hakasuluissa]	viittaukset

1.4 Yleiskatsaus dokumenttiin

Dokumentin ensimmäinen luku on johdanto määrittelydokumenttiin. Luku kertoo dokumentin tarkoituksen, määriteltävän tuotteen yleiskuvauksen ja käytetyt termit.

Luku 2 kuvaa järjestelmän toiminnan yleisellä tasolla: siihen kuuluvan laitteiston, käyttäjät, järjestelmän riippuvuudet ja rajoitukset.

Luvussa 3 kuvataan järjestelmän tietosisältö eli tietokannat ja tietovirrat.

Luvussa 4 määritellään järjestelmän toiminnot. Kukin toiminto on selitetty käytännön esimerkein ja kuvakaappauksin.

Luku 5 kertoo järjestelmän ulkoiset liittymät, eli laitteiston, tietoliikenteen ja ohjelmistoliittymät.

Lukuun 6 on kuvattu järjestelmän ei-toiminnalliset ominaisuudet, kuten suorituskyky, vasteajat, käytettävyys ja ylläpidettävyys.

Lukuun 7 on kirjattu suunnitteluun vaikuttavat rajoitteet, kuten standardit sekä ohjelmisto- ja laitteistorajoitteet.

Luku 8 on varattu jatkokehitysajatuksille.

2 YLEISKUVAUS

2.1 Ympäristö

Rakentamisen ympäristölaskuri -sovellus asennetaan Metlan web-palvelimelle. Ympäristölaskuri toteutetaan PHP-ohjelmointikielellä, joten se vaatii toimiakseen HTTP-palvelinohjelman ja PHP-tulkin. Metlan palvelimella on Apachen versio 2.2.10 ja PHP:n versio 5.3.10. Ohjelmisto tallentaa tietoa MySQL-tietokantaan, josta Metlalla on käytössä 5.5.16-versio. Lisäksi ohjelmisto käyttää Yii-sovelluskehyksen versiota 1.1.12, joka on tällä hetkellä kehyksen uusin käytettävissä oleva versio.

2.2 Toiminta

Ohjelmalla on tarkoitus voida laskea rakennusosan tai jopa kokonaisen talon ympäristövaikutukset koko elinkaaren ajalta. Tämän lisäksi ohjelman pitää voida eritellä vaikutukset myös rakentamisen, kunnossapidon, kiinteistönhuollon ja purkamisen osalta.

Valmiin ohjelmiston pitäisi pystyä tallentamaan käyttäjän määrittelemiä rakennusosia tietokantaan mahdollista myöhempää käyttöä varten. Osat eivät kuitenkaan saa olla julkisia, joten niiden on tarkoitus näkyä vain käyttäjälle itselleen. Käytössämme on saksankielinen Ökobau-materiaalitietokanta, jossa on 13 erilaista ympäristövaikutusta kullekin rakennusmateriaalille. Rakennusosan sisältämät materiaalit tulee voida tallentaa tietokantaan. Rakennusosia pitäisi pystyä myös muokkaamaan ja poistamaan.

Ohjelmaan on tarkoitus tehdä kaksi editoria: osaeditori ja taloeditori. Osaeditorilla käyttäjän pitäisi pystyä luomaan omia rakennusosia materiaaleista tai muista rakennusosista tai valitsemalla valmiita osia esimerkiksi pudotusvalikoista. Taloeditorissa osista pitäisi pystyä näkemään koko rakennuksen osien hierarkia.

Ohjelmistossa on tarkoitus käyttää kahta tietokantaa, jotka on nimetty PENA_MASTER_DB- ja PENA_SLAVE_DB-tietokannoiksi. PENA_MASTER_DB-tietokantaan tallennetaan käyttäjien tiedot, talo80- ja talo2000-rakennusosamäärittelyt sekä rakennusmateriaalien ympäristövaikutukset eli Ökobausta luetut tiedot. PENA_SLAVE_DB-tietokantaan tallennetaan käyttäjien määrittelemiä rakennusosia ja niiden ominaisuuksia.

2.3 Käyttäjät

Ohjelmistolla on kahdentyyppisiä käyttäjiä: ylläpitäjät ja tavalliset käyttäjät. Ainoastaan ylläpitäjillä on oikeudet muokata PENA_MASTER_DB-tietokantaa poikkeuksena uuden käyttäjän rekisteröityminen ja oman salasanan vaihto. Ylläpitäjien käyttöliittymään on tarkoitus tehdä ylläpito-välilehti tai -välilehtiä, joissa on vain heille näkyviä toimintoja, kuten PENA_MASTER_DB-tietokantaan kohdistuvia toimenpiteitä.

Tavalliselle käyttäjälle tulevat olemaan näkyvissä ainoastaan osaeditori- ja taloeditori- sekä salasananvaihto-välilehti.

Mahdollisena käyttäjänä voi olla kuka tahansa, joka rekisteröi itsensä sivustolle.

Ylläpitäjänä tulee toimimaan Metlan tutkija Tarmo Räty.

2.4 Yleiset rajoitteet

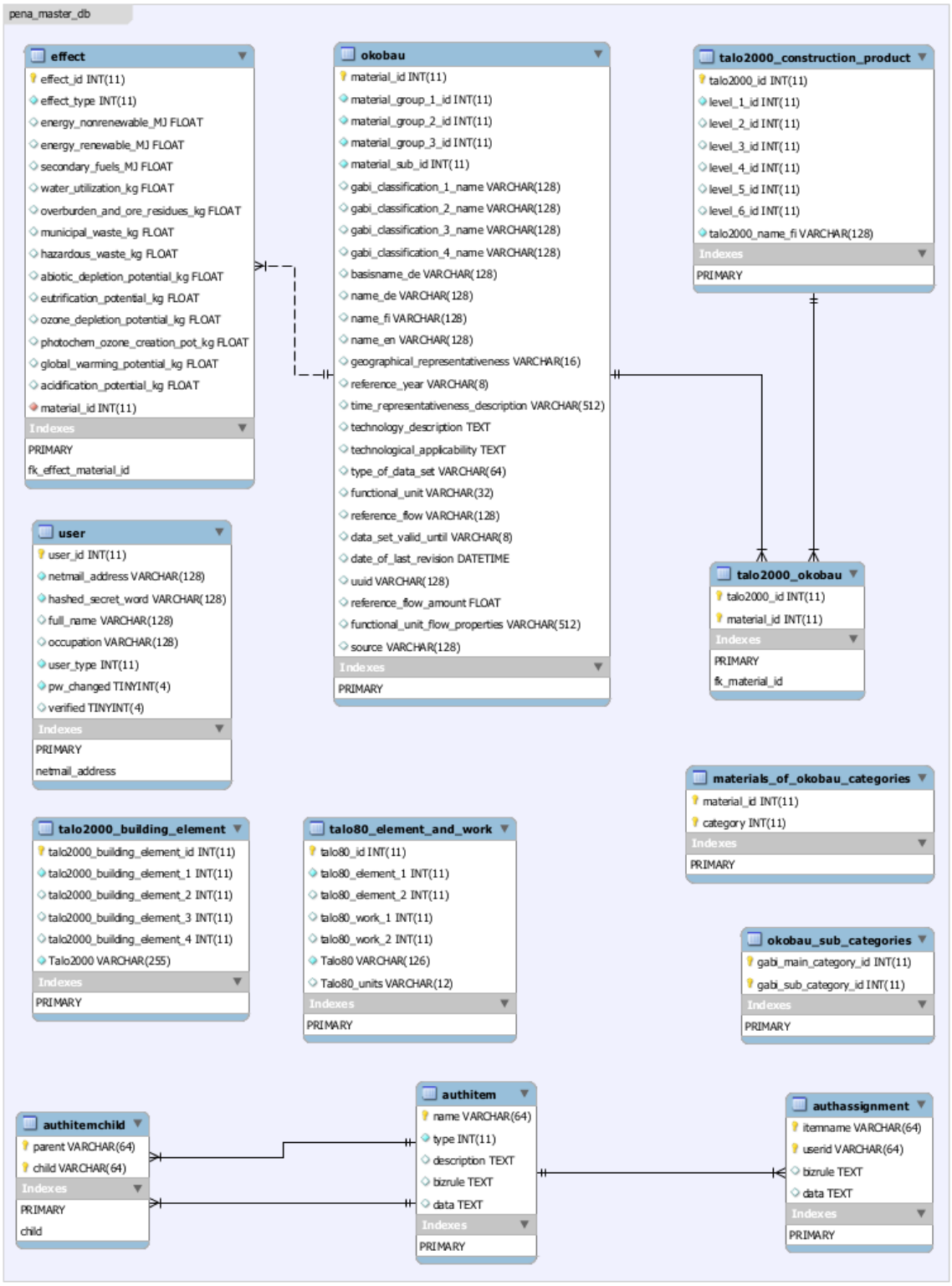
Sovelluksen käyttämiseksi tarvitaan tietokone ja Internet-yhteys sekä selain. Selaimessa pitää olla Javascript sallittu, että sovellus toimisi oikein. Sovellukselle ei ole suunnitteilla käyttöliittymää mobiililaitteille. Sovellus testataan yleisimmillä selaimilla ennen käyttöönottoa.

2.5 Oletukset ja riippuvuudet

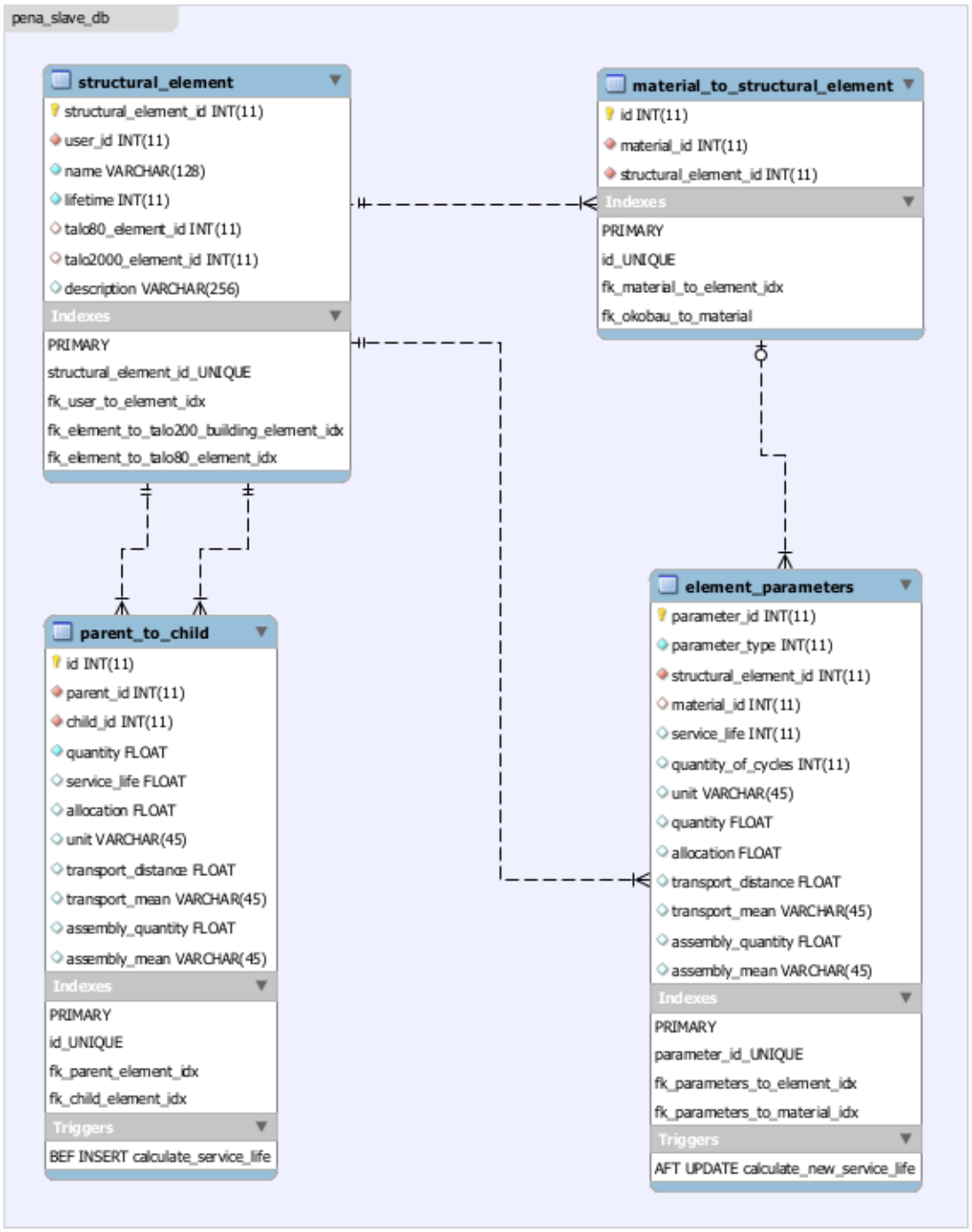
Sovellus vaatii PHP:n version 5.3 tai uudemman toimiakseen palvelimella. Sovellus on testattu Apachen web-palvelimen versiolla 2.2.22 sekä MySQL-palvelimen versiolla 5.5.27. Lisäksi sovellus vaatii Yii-sovelluskehyksen version 1.1.12. Sovellus pyritään testaamaan tarvittaessa myös muilla em. ohjelmistojen versioilla. Määrittely ei ole voimassa mobiililaitteille.

3 TIEDOT JA TIETOKANTA

3.1 Tietosisältö



Kuva 1: Master-tietokanta



Kuva 2: Slave-tietokanta

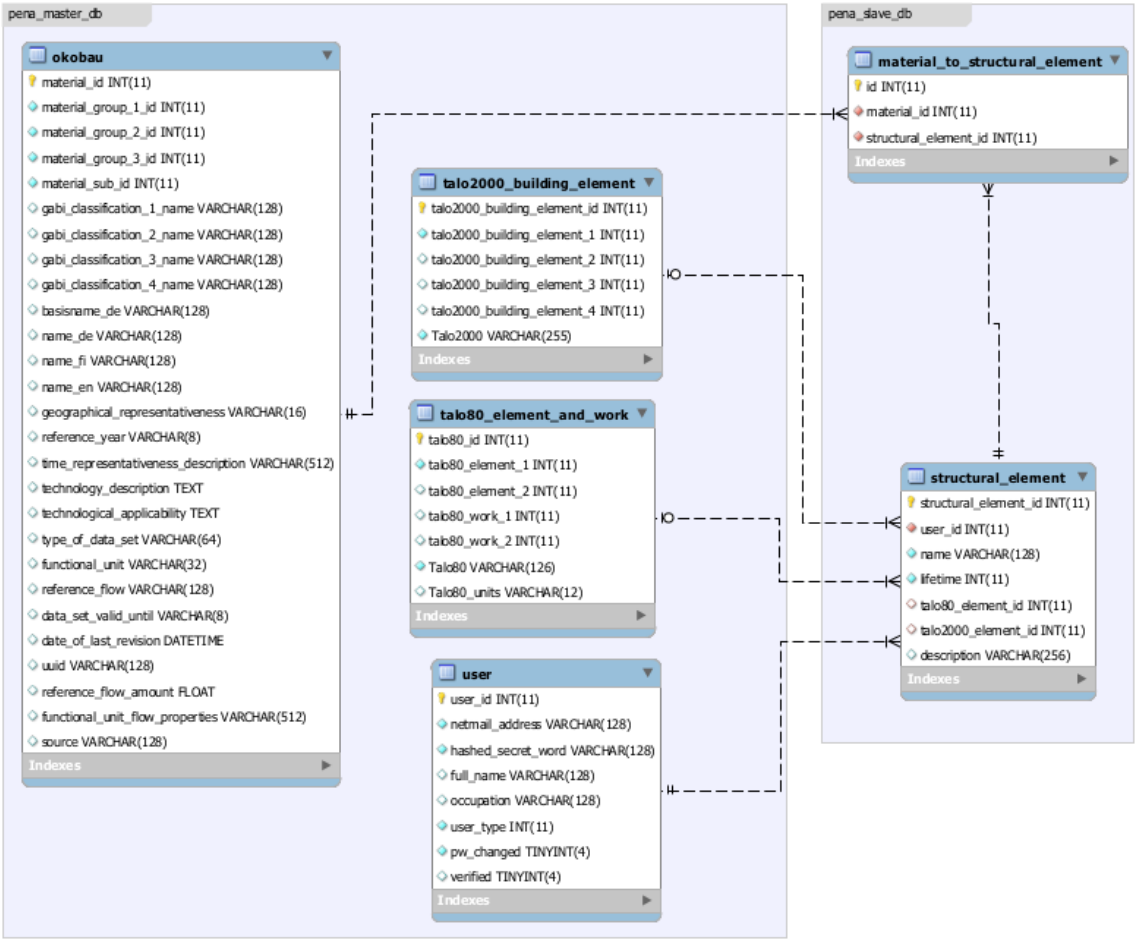
Sovellus käyttää kahta tietokantaa, jotka on nimetty PENA_MASTER_DB- ja PENA_SLAVE_DB-tietokannoiksi.

PENA_MASTER_DB-kanta sisältää *user*-taulun, johon tallennetaan rekisteröityneiden käyttäjien tiedot sekä ylläpitäjien tiedot, kuten esimerkiksi käyttäjätunnus, salasana ja käyttäjätyyppi. Käyttäjätunnuksena käytetään käyttäjän sähköpostiosoitetta.

PENA_MASTER_DB-kannassa on Yii:n omat *authitem-, authitemchild-* ja *authassignment*-taulut, joita Yii käyttää käyttöoikeuksien hallinnassa.

PENA_MASTER_DB-kantaan on tallennettu myös Ökobau-tietokannasta luetut rakennusmateriaalien tiedot sekä niiden ympäristövaikutukset *okobau-* ja *effect*-tauluihin. Tauluun *talo2000_construction_products* on tallennettu kotimaiset luokitukset rakennusmateriaaleille. Taulujen *okobau* ja *talo2000_construction_products* väliin on luotu *talo2000_okobau*-taulu monimoneen-yhteyden purkamiseksi.

Lisäksi master-kannassa on *talo80_element_and_work-* ja *talo2000_building_elements*-taulut, jotka sisältävät vastaavasti talo80- ja talo2000-luokituksia rakennusosille. Näitä käytetään ohjelmassa apuna uusien rakennusosien luomisessa ja määrittämisessä.



Kuva 3: yhteydet tietokantojen välillä

Slave-tietokanta on sovelluksen käyttäjien käytössä. Sinne tallennetaan käyttäjän määrittelemiä rakennusosia *structural_element*-tauluun. Kentät **ta-**

lo2000_element_id ja **talo80_element_id** viittaavat PENA_MASTER_DB-kannan *talo2000_building_elements*- ja *talo80_element_and_work*-taulujen id-kenttiin. Kenttä **user_id** taasen viittaa PENA_MASTER_DB-kannan *user*-taulun **user_ID**-kenttään. Rakennusosissa käytettävät materiaalit linkitetään PENA_MASTER_DB-kannan *okobau*-taulun **material_id-**kentästä PENA_SLAVE_DB-kannan *material_to_structural_element*-taulun **material_id**-kenttään.

Rakennusosat voivat sisältää toisia rakennusosia, jotka puolestaan voivat sisältää lisää rakennusosia. Tämä hierarkia tallennetaan *parent_to_child*-tauluun.

3.2 pena_master_db-tietokanta

3.2.1 Taulu user

Kenttä **user_id** on juokseva yksilöllinen numero, joka on samalla myös taulun pääavain. Kenttä **netmail_address** sisältää käyttäjätunnuksen, joka on käyttäjän validi sähköpostiosoite. Kenttä **hashed_secret_word** sisältää käyttäjän salasanan md5-koodin. Kentässä **full_name** on käyttäjän koko nimi ja kentässä **occupation** ammatti. **full_name** ja **occupation** ovat valinnaisia kenttiä ja voivat näin ollen sisältää myös NULL-arvon ellei käyttäjä ole niitä antanut. Kenttä **user_type** sisältää käyttäjän tyypin, joka voi olla joko 0 tai 1. 0 tarkoittaa ylläpitäjää (admin) ja 1 tavallista käyttäjää (user): Taulussa on vielä **password_changed**-kenttä, josta nähdään onko käyttäjä vaihtanut salasanansa. NULL-arvo tarkoittaa, että ei, ja kaikki muut arvot tarkoittavat kyllä. Oletusarvona kentässä on NULL. Kenttä **verified** on suunniteltu mahdollista käyttäjätunnuksen sähköpostivahvistusta varten ja se ei ole tällä hetkellä käytössä.

3.2.2 Taulu okobau

Kenttä **material_id** on taulun pääavainkenttä. Se on nk. surrogaatti eli keinotekoinen yksilöllinen juokseva numero. Kenttä **material_group_1_id** sisältää Ökobau-tietokannassa käytetyn nk. gabi-hierarkian ylimmän tason eli pääryhmän. Kenttä **material_group_2_id** sisältää kyseisen hierarkian seuraavan tason. Kenttä **material_group_3_id** puolestaan sisältää gabi-hierarkian kolman-

nen tason, jonka alla ovat itse materiaalit. Kenttä **material_sub_id** on juokseva numero, joka yksilöi kyseiset materiaalit.

Seuraavaksi tulevat Ökobau-tietokannasta luetut gabi-luokitusten nimet **gabi_classification_1_name**, **gabi_classification_2_name**, **gabi_classification_3_name** ja **gabi_classification_4_name**. Ne ovat pääosin samoja kuin gabi hierarkian **basisname_de** kentässä olevat nimet ja ne ovat tietokannassa ainoastaan materiaaleilla. Kentässä **basisname_de** on materiaalin tai gabi-hierarkiassa ylempänä olevan ryhmän saksankielinen nimi (kuten esimerkiksi ”Mineralische Baustoffe”). Kenttä on luettu suoraan Ökobau-tietokannasta ja ainakin materiaaleilla sisältää usein myös hierarkian numeroinnin. Kenttä **name_de** sisältää **basisname_de**-kentästä luetun nimen ilman numerointia ja kentissä **name_fi** ja **name_en** ovat suomen- ja englanninkieliset käännökset kyseiselle nimelle.

Taulun loput kentät ovat suoraan Ökobau-tietokannasta luettuja eikä niitä tätä kirjoitettaessa juurikaan ohjelmassa käytetä. Kenttien nimet on käännetty suoraan Ökobau-tietokannan vastaavista saksankielisistä nimistä, ja ovat järjestyksessä seuraavat: **geographical_representativeness**, **reference_year**, **time_representativeness_description**, **technology_description**, **technological_applicability**, **type_of_data_set**, **functional_unit**, **reference_flow**, **data_set_valid_until**, **date_of_last_revision**, **uuid**, **reference_flow_amount**, **functional_unit_flow_properties** ja **source**. Näistä mainittakoon erikseen **functional_unit**, joka sisältää materiaalille käytettävän yksikön.

3.2.3 Taulu effect

Kenttä **effect_id** on taulun pääavainkenttä. Se on nk. surrogaatti eli keinotekoinen yksilöllinen juokseva numero. Kentässä **effect_type** kerrotaan ympäristövaikutuksen tyyppi, joka voi olla kokonaisluku väliltä 0-4. 0 tarkoittaa rakentamista, 1 kunnossapitoa, 2 kiinteistönhuoltoa, 3 purkamista ja 4 koko elinkaarta.

Seuraavaksi taulussa ovat eri ympäristövaikutukset, jotka on luettu suoraan Ökobau-materiaalitietokannasta. Kentässä **energy_nonrenewable_MJ** on uusiutumattoman energian määrä megajouleissa, kenttä **energy_renewable_MJ** sisältää uusiutuvan energian määrän megajouleissa, **secondary_fuels_MJ** polttoaineen uudelleenkäytön niin ikään megajouleissa, **water_utilization_kg**

veden käytön kilogrammoina, **overburden_and_ore_residues_kg** luohintajätteet kilogrammoina, **municipal_waste_kg** yhdyskuntajätteet kilogrammoina, **hazardous_waste_kg** ongelmajätteet kilogrammoina, **abiotic_depletion_potential _kg** elottomien ympäristötekijöiden ehtymisen kilogrammoina, **eutrification_potential_kg** rehevöitymisen kilogrammoina, **ozone_depletion_potential_kg** otsonikadon kilogrammoina, **photochem_ozone_creation_pot_kg** fotokemiallisen otsonin muodostumisen kilogrammoina, **global_warming_potential_kg** ilmastonmuutoksen kilogrammoina ja **acidification_potential_kg** maaperän ja veden happamoitumisen kilogrammoina.

Viimeisenä taulussa on viiteavainkenttä **material_id**, joka viittaa *okobau*-taulun vastaavaan kenttään.

3.2.4 Taulu talo2000_construction_product

Sisältää Talo2000-tuotenimikkeistön mukaisen 6-portaisen hierarkian sovelluksessa käytettäville rakennusmateriaaleille.

Kenttä **talo2000_id** on taulun pääavain. Se on nk. surrogaatti eli keinotekoinen yksilöllinen juokseva numero.

Kentät **level_1_ID – level_6_ID** sisältävät alkuperäisen tuotenimikkeistön hierarkian tasot 1-6, jotka on luettu suoraan Talo2000-tuotenimikkeistöstä.

Lopuksi taulussa on tuotteen suomenkielinen nimi, **talo2000_name_fi**. Koska alkuperäinen nimikkeistö on suomeksi, emme tässä vaiheessa katsoneet mielekkääksi kääntää sitä muille kielille.

3.2.5 Taulu talo2000_okobau

Kyseessä on ns. linkkitaulu *okobau*- ja *talo2000_construction_products*-taulujen välillä. Tänne tallennetaan *talo2000_construction_products*-taulun materiaaleja vastaavat *okobau*-taulun materiaalien tunnisteet (id) ja päinvastoin.

Taulussa on ns. yhdistetty pääavain, joka koostuu **talo_2000_id-** ja **material_id**-kentistä. Nämä toimivat myös viiteavaimina *okobau*- ja *talo2000_construction_products*-taulujen vastaaviin kenttiin.

3.2.6 Taulu talo2000_building_element

Sisältää Talo2000-rakennusosanimikkeistön mukaisen hierarkian sovelluksessa apuna käytettäville valmiille rakennusosille.

talo2000_building_element_id-kenttä on surrogaatti pääavain. Kentät **talo2000_building_element_1 – talo2000_building_element_4** muodostavat alkuperäisessä rakennusosanimikkeistössä käytetyn hierarkian tasot 1-4. **Talo2000**-kenttä sisältää osan luokan tai nimen.

3.2.7 Taulu talo80_element_and_work

Sisältää Talo80-rakennus- ja suoritusosanimikkeistön mukaisen hierarkian sovelluksessa apuna käytettäville valmiille rakennusosille.

talo80_id-kenttä on surrogaatti pääavain. Kentät **talo80_element_1** ja **talo80_element_2** sisältävät alkuperäisen Talo80-rakennusosanimikkeistön mukaisen 2-portaisen hierarkian valmiille rakennusosille ja **talo80_work_1** ja **talo80_work_2** Talo80-suoritusosanimikkeistön mukaisen niin ikään 2-portaisen hierarkian osiin liittyville suoritteille. **talo80_element_1**- ja **talo80_element_2**-kentät määrittelevät siis rakennusosan ja **talo80_work_1**- ja **talo80_work_2**-kentät osaan liittyvän suoritteen tai työn.

Talo80-kenttä sisältää rakennusosan nimen ja **Talo80_units** rakennusosaan liittyvän yksikön. Hierarkia, suoritteet, rakennusosien nimet ja yksiköt on otettu suoraan Talo80-nimikkeistöstä.

3.2.8 Taulu materials_of_okobau_categories

Tämä taulu sisältää linkityksen ökobaun materiaalien ja ökobaun gabi-kategorioiden välillä eli mitkä materiaalit kuuluvat mihinkin gabi-kategoriaan.

Kenttään **material_id** tulee materiaalin yksilöllinen numero ja kenttään **category** tulee gabi-kategorian numero.

3.2.9 Taulu okobau_sub_categories

Tämä taulu sisältää ökobaun gabi-kategorioiden järjestyksen siten, että tässä näkyy mitkä gabi-alikategoriat kuuluvat mihinkin gabi-pääkategoriaan.

Kenttään **gabi_main_category_id** tulee gabi-pääkategorian numero ja kenttään **gabi_sub_category_id** tulee gabi-alikategorian numero.

3.2.10 Taulut authitem-, authitemchild- ja authassignment

Yii käyttää kyseisiä tauluja käyttöoikeuksien hallinnassa. Taulut ovat Yii:n luomia.

3.3 pena_slave_db-tietokanta

3.3.1 Taulu structural_elements

Tässä taulussa pääavaimena on **structural_element_id**-kenttä ja kun uusi tietue lisätään tauluun niin tämä kenttä kasvaa automaattisesti yhdellä numerolla. Käyttäjän antamista rakennusosan tiedoista nimi tallennetaan **name**-kenttään ja käyttöikä **lifetime**-kenttään. Nämä rakennusosan tiedot ovat pakollisia. Lisäksi käyttäjä pystyy antamaan kuvauksen rakennusosalle **description**-kenttään. Kuvaus on vapaaehtoinen ja ellei käyttäjä anna sitä, kenttä jätetään tyhjäksi. Kenttään **user_id** tallennetaan käyttäjän yksilöllinen tunniste, joka saadaan PENA_MASTER_DB-tietokannan *user*-taulun vastaavasta **user_id**-kentästä. Näiden kahden **user_id**-kentän yhteys on määritelty viiteavaimella **fk_user_to_element**. Kenttä **talo80_element_id** on vapaaehtoista Talo80-rakennusosamäärittelyä varten ja sen arvo saadaan PENA_MASTER_DB-tietokannasta *talo80_element_and_work*-taulun **talo80_id**-kentästä. Jos käyttäjä on tehnyt Talo80 määrittelyn omalle rakennusosalle, viiteavaimella **fk_element_to_talo80_element** määritetään kyseisten kenttien yhteys. Tauluun on tehty vastaava rakenne myös vapaaehtoista Talo2000-rakennusosamäärittelyä varten. Kenttä **talo2000_element_id** saadaan PENA_MASTER_DB-tietokannan *talo2000_building_elements*-taulun **talo2000_rakennusosa_id**-kentästä. Jos käyttäjä on tehnyt talo2000 määrittelyn omalle rakennusosalle, kyseisten kenttien yhteys on määritetty **fk_element_to_talo2000_building_element**-viiteavaimella.

3.3.2 Taulu material_to_structural_element

Tässä taulussa yhdistetään materiaalit rakennusosaan. Pääavaimena on **id**-kenttä. Kenttään **structural_element_id** tallennetaan tietyn rakennusosan yksi-

löllinen tunniste, joka saadaan *structural_element*-taulun **structural_element_id**-kentästä. Viiteavaimella **fk_material_to_element** määritetään yhteys taulujen *material_to_structural_element* ja *structural_element* välillä. Kenttään **material_id** tallennetaan materiaalin yksilöllinen tunniste, joka saadaan PENA_MASTER_DB-tietokannan *okobau*-taulun vastaavasta **material_id**-kentästä. Kenttien yhteys on määritetty viiteavaimella **fk_okobau_to_material**. Mikään taulun kentistä ei saa NULL arvoja.

3.3.3 Taulu element_parameters

Rakennusosan parametrit tallennetaan tähän tauluun. Taulun pääavaimena on **parameter_id**-kenttä. Rakennusosan parametrit voidaan tallentaa viidelle eri ajanjaksolle ja tietyn ajanjakson yksilöllinen tunniste tallennetaan **parameter_type**-kenttään. Kenttään **structural_element_id** tallennetaan rakennusosan yksilöllinen tunniste, joka saadaan *structural_element*-taulun **structural_element_id**-kentästä. Kenttien yhteys on määritelty viiteavaimeen **fk_parameters_to_element_idx**. Materiaalin parametrit tallennetaan myös tähän tauluun. Kenttään **material_id** tallennetaan materiaalin tunniste, joka saadaan *material_to_structural_element*-taulun **material_id**-kentästä. Tämä yhteys on määritelty viiteavaimeen **fk_parameters_to_material_idx.** Käyttäjä antaa seuraavat tiedot: 1) elinikä, 2) elinikien lukumäärä, 3) määrä yksikköä kohden ja 4) allokaatio, jotka tallennetaan vastaavasti kenttiin 1) **service_life**, 2) **quantity_of_cycles**, 3) **quantity** ja 4) **allocation**. Yksikkö tallennetaan **unit**-kenttään. Se saadaan rakennusosille käyttäjältä ja materiaaleille PENA_MASTER_DB-tietokannan *okobau*-taulun **functional_unit**-kentästä materiaalin lisäyksen aikana. Rakennusosan ja materiaalin parametreihin kuuluvat myös kentät **transport_distance** (kuljetusmatka), **transport_mean** (kuljetustapa), **assembly_quantity** (valmistusmäärä) ja **assembly_mean** (valmistustapa). Nämä kentät eivät ole vielä (2014-05-15) käytössä ohjelmassa. Ne ovat lähinnä tulevia ominaisuuksia varten valmiiksi tehdyt.

Tähän tauluun on liitetty lisäksi liipaisin (trigger) **calculate_new_service_life**, joka toteutetaan aina tauluun kohdennetun päivitystoiminnon jälkeen. Liipaisimessa lasketaan uudelleen eliniän kerroin pää- ja alirakennusosan suhteen. Esimerkiksi, jos pääosan elinikä on 100 v. ja aliosan 50 v. niin kertoimeksi tulee 2. Päivitetyn tiedon perusteella laskettu arvo (kerroin) tallennetaan *pa-*

rent_to_child-taulun **service_life**-kenttään. Uusi arvo lasketaan kyseisessä taulussa kaikille riveille, joissa päivitetty rakennusosa esiintyy. Laskettu arvo ei saa olla koskaan alle 1 ja se pyöristetään aina ylempään kokonaislukuun.

3.3.4 Taulu parent_to_child

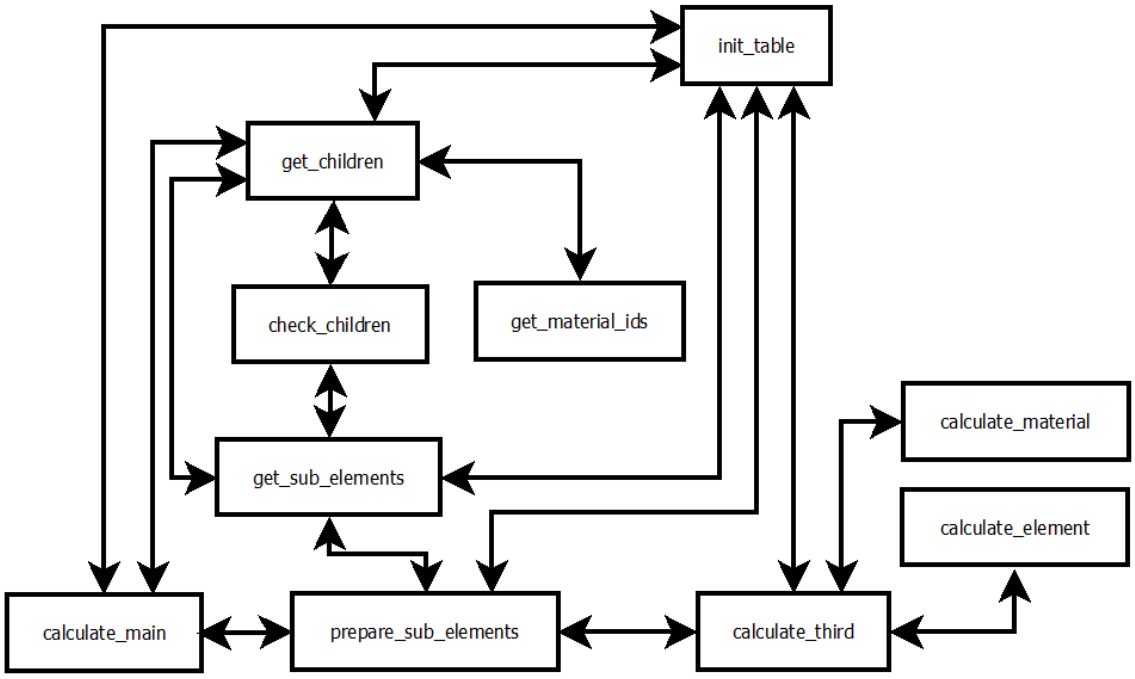
Talon rakennusosilla pitää olla jonkinlainen hierarkia, jotta talon ympäristövaikutukset voidaan laskea oikein. Tämä hierarkia tallennetaan tähän tauluun. Taulun pääavaimena on **id**-kenttä. Hierarkiassa päärakennusosa tallennetaan **parent_id**-kenttään ja sen alirakennusosa **child_id**-kenttään. Molempien kenttien arvot saadaan *structural_element*-taulun kutakin rakennusosaa vastaavasta **structural_element_id**-kentästä. Viiteavaimella **fk_parent_element** on määritelty **parent_id**-kentän yhteys *structural_element*-taulun **structural_element_id**-kenttään ja viiteavaimella **fk_child_element child_id**-kentän yhteys ja samaiseen **structural_element_id**-kenttään.

Ympäristövaikutusten laskemista varten käyttäjä antaa **quantity**-kenttään määrän kuinka monta yksikköä aliosaa käytetään pääosassa. Esimerkiksi, jos aliosan yksikkö on kilogramma (kg), niin kenttään laitetaan kuinka monta kiloa aliosaa käytetään pääosassa. Rakennusosan elinikä laitetaan **service_life**-kenttään. Rakennusosan yksikkö tallennetaan **unit**-kenttään ja se saadaan *element_parameters*-taulun **unit**-kentästä rakennusosan linkityksen aikana. Kenttään **allocation** tulee käyttäjän antama numero. Kuljetusmatkan pituus tulee **transport_distance**-kenttään ja kuljetustapa **transport_mean**-kenttään. Valmistusmäärä tulee **assembly_quantity**-kenttään ja valmistustapa **assembly_mean**-kenttään.

Lisäksi tauluun on liitetty liipaisin (trigger) **calculate_service_life**, joka toteutetaan aina tauluun kohdennetun uuden rivin lisäys -toiminnon yhteydessä, juuri ennen rivin lisäystä. Liipaisimessa haetaan lisättävän rivin pääosan ja aliosan eliniät *element_parameters*-taulusta ja lasketaan eliniän kerroin pääosan ja aliosan suhteen. Laskettu arvo asetetaan uuden lisättävän rivin **service_life**-kenttään. Laskettu arvo ei saa olla koskaan alle 1 ja se pyöristetään aina ylempään kokonaislukuun.

3.3.5 Laskentaproseduurit

Sovelluksen ympäristövaikutusten laskenta suoritetaan MySQL:n proseduureilla. Kuvassa 4 nähdään missä järjestyksessä proseduurit suoritetaan ja samalla nähdään niiden yhteydet.



Kuva 4: Laskenta proseduurien yhteydet

Laskenta aloitetaan kutsumalla *calculate_main*-proseduuria. Proseduurissa *calculate_main* annetaan parametrina rakennusosan numero, jolle lasketaan ympäristövaikutukset. Tässä proseduurissa haetaan rakennusosan aliosat ja mahdolliset materiaalit sekä *prepare_sub_elements*-proseduurilla haetaan mahdolliset aliosat ja materiaalit jne. laskettavan rakennusosan aliosille. Lopuksi lasketaan ympäristövaikutukset yhteen ympäristövaikutus kerrallaan ja tulokset palautetaan sovelluksen ”**Element Editor**”-välilehden ”**Impact summary**”-riville.

Laskennassa käytetään apuna MySQL:n väliaikaisia tauluja. Niihin tallennetaan aliosien ja materiaalien tunnistusnumerot sekä laskentatulokset. Näiden taulujen luomiseen käytettiin *init_table*-proseduuria, jolle annetaan parametrina väliaikaisen taulun numero.

Proseduurissa *prepare_sub_elements* etsitään annettujen aliosien mahdolliset aliosat ja materiaalit rakennusosahierarkian mukaisesti kutsumalla

get_sub_elemets-proseduuria. Laskentaan edetään *calculate_third*-proseduurilla.

Rakennusosahierarkian selvitys laskettavan rakennusosan aliosille tehdään *get_sub_elements*-proseduurissa. Tähän proseduuriin annetaan parametreina tarkistettavan aliosan numero ja sen pääosan numero. Tässä proseduurissa kutsutaan *check_children*-proseduuria, jolla tarkastetaan onko rakennusosalla aliosia ja materiaaleja.

Aliosien tarkistus tehdään *check_children*-proseduurilla. Tämä proseduuri palauttaa rakennusosan aliosien sekä mahdollisten materiaalien lukumäärän.

Rakennusosan aliosat haetaan *get_children*-proseduurilla. Sen parametreina annetaan rakennusosan id-tunnus, pääosan id-tunnus sekä numero, jolla määrätään mistä väliaikaisesta taulusta aliosien lukumäärä haetaan. Proseduuri palauttaa aliosien lukumäärän.

Rakennusosan materiaalit haetaan *get_material_ids*-proseduurilla. Sen parametreina annetaan rakennusosan id-tunnus, pääosan id-tunnus sekä numero, jolla määrätään mihin tauluun materiaalin tiedot viedään. Rakennusosan id-tunnus ja sen pääosan id-tunnus pitää antaa, koska samaa materiaalia voi olla käytössä myös jollakin toisella rakennusosalla ja näin ne yksilöidään muista samanlaisista materiaaleista.

Laskennan suoritus aloitetaan *calculate_third*-proseduurilla. Tässä selvitetään onko aliosa rakennusosa vai materiaali ja sen perusteella kutsutaan oikeata proseduuria laskemiseen.

Materiaalien ympäristövaikutusten laskeminen tehdään *calculate_material*-proseduurissa. Tässä proseduurissa annetaan parametreina eliniän ajanjakson tyyppi, rakennusosan id-tunnus, materiaalin id-tunnus ja pääosan id-tunnus. Laskentaan käytettävissä muuttujissa tehdään DECIMAL-tietotyypin muuttujia, koska kyseinen tietotyyppi ei pyöristä lukuja kesken laskennan. Materiaalin ympäristövaikutusten lukuarvot haetaan PENA_MASTER_DB-tietokannan *effect*-taulusta materiaalin id-tunnuksen perusteella. Laskennan tulokset viedään väliaikaiseen tauluun.

Rakennusosan ympäristövaikutukset lasketaan *calculate_element*-proseduurissa. Sen parametreina annetaan eliniän ajanjakson tyyppi, rakennusosan id-tunnus, pääosan id-tunnus, määrä, eliniän kerroin, etäisyys ja valmistusmäärä. Etäisyys ja valmistusmäärä parametreja ei ole vielä otettu mukaan laskennassa. Laskennassa käytetään DECIMAL-tietotyypin muuttujia, koska kyseinen tietotyyppi ei pyöristä lukuja kesken laskennan. Ensimmäisenä lasketaan yhteen aiemmin laskettujen materiaalien ja rakennusosien ympäristövaikutukset omiin muuttujiin. Seuraavaksi ne kerrotaan annettujen laskentaan tarkoitettujen parametrien kanssa ja tulokset tallennetaan väliaikaiseen tauluun.

3.4 Käyttöintensiteetti

Sovelluksella voisi periaatteessa olla lähes rajattomasti yhtäaikaisia käyttäjiä, koska se tulee Internetiin vapaaseen käyttöön. Sovellusta tehtäessä on arvioitu, että se selviytyy tehokkaalla palvelinkoneella ainakin 100 yhtäaikaisesta käyttäjästä. Tätä tosin ei ole testattu käytännössä.

Tietokantaan on tehty joitakin indeksointeja suorituskyvyn parantamiseksi.

Tilannetta, jossa kaksi käyttäjää muokkaa samoja tietoja, ei periaatteessa voi tulla, koska käyttäjien omat rakennusosat ja muut tiedot näkyvät ainoastaan käyttäjille itselleen.

Ohjelmistoa voi käyttää päivittäin, jossa ei ole kellonaika- eikä viikonpäivärajoitetta.

3.5 Kapasiteettivaatimukset

Palvelin tarvitsee vähintään 200 megatavua ohjelmakoodille, Yii-kirjastolle ja ökobau.dat2011-XML-tietokannalle yhteensä.

MySQL-tietokanta vaatii vähintään 500 megatavua, mutta jos käyttäjämäärä kasvaa suureksi, kasvaa tilan tarve myös vastaavasti. Suositeltavaa olisi varata levytilaa tietokannalle vähintään 10 gigatavua, että tietokannalla olisi kasvunvaraa.

Palvelimella tulisi olla myös suhteellisen tehokas suoritin mahdollisesti useiden yhtäaikaisten tietokantahakujen varalta.

Keskusmuistin osalta on vaikeaa arvioida sovelluksen tarvitsemaa vähimmäismäärää, koska se riippuu myös palvelimella mahdollisesti pyörivistä muistakin prosesseista sekä palvelimen käyttäjämääristä. Pelkästään laskuri-sovelluksen ajamiseen kohtuullisella käyttäjämäärällä riittänee noin gigatavu keskusmuistia.

4 TOIMINNOT

4.1 Yleistä

Sovellusta voi käyttää tavallisella PC-koneella, jossa on hiiri ja näppäimistö sekä Internet-yhteys. Ohjelmisto on suunniteltu käytettäväksi lähinnä hiirellä, mutta hätätilassa sitä voi käyttää myös näppäimistöltä käsin. Ohjelmiston käyttämiseksi tarvitaan lisäksi Internet-selain.

Sovelluksessa on käytössä skandinaavisten merkkien tuki. Isot ja pienet kirjaimet on eroteltu toisistaan. Ohjelma on tätä kirjoitettaessa suomen- ja englanninkielinen sekä tietokannan osalta sekoitus suomea, englantia ja saksaa. Valmiin ohjelman on tarkoitus tukea englantia, suomea ja saksaa.

4.1.1 Käyttöönotto

Ennen sovelluksen asennusta palvelimelle, siellä pitää olla luvussa 2.5 määritetyt sovellukset asennettuna. Sovellus asennetaan Apache web-palvelimen määrittelemään *htdocs*-kansioon ja sovelluksen index.php:ssä oleva polku Yii-sovelluskehyskansioon pitää olla oikea. Sovelluksen tietokanta pitää tuoda tietokantapalvelimelle dump-tiedostosta, jossa on molemmat tietokannat sovelluksen käyttämiseen. Yii:n asetustiedostossa *web-root\calculator\protected\config\main.php* pitää olla molempiin tietokantoihin oikeat yhteysasetukset, jotta niihin voi ottaa sovelluksesta yhteyttä.

Käyttäjät pääsevät käyttämään ohjelmaa Internet-selaimen kautta.

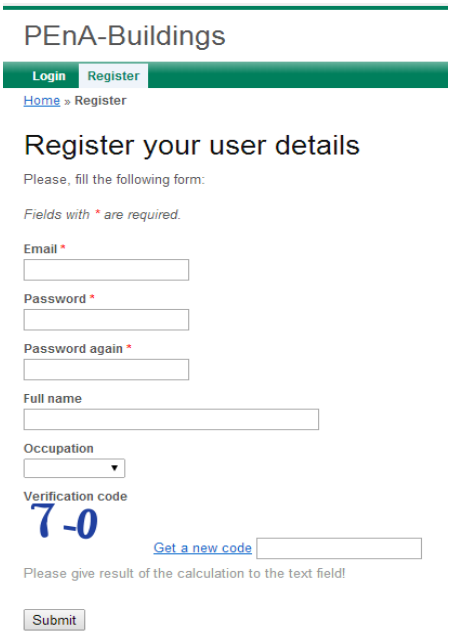
4.2 Järjestelmän toiminnot

4.2.1 Sovelluksen käynnistäminen

Sovellusta käytetään Internet-selaimen kautta ja sovellus käynnistetään kirjoittamalla web-sivun url-osoite selaimen osoitekenttään tai klikkaamalla linkkiä esim. Metlan sivuilta.

4.2.2 Rekisteröityminen

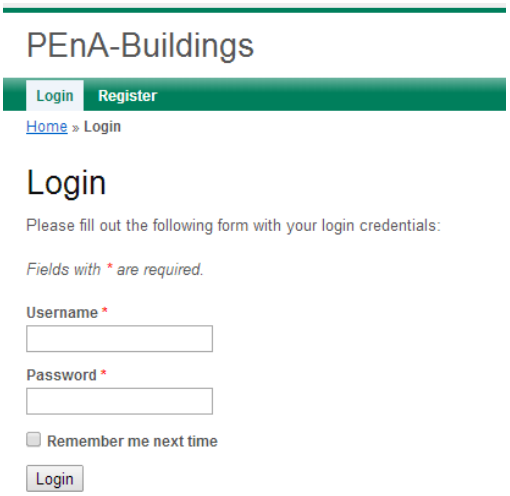
Käyttäjä voi rekisteröityä sivuille haluamallaan sähköpostiosoitteella ja salasanalla valitsemalla ”**Register**”-välilehden ja täyttämällä rekisteröitymislomakkeen (kuva 5) sekä lähettämällä sen palvelimelle ”**Submit**”-nappulan avulla. Sähköpostiosoitteen on oltava uniikki, joten samalla sähköpostiosoitteella ei voi rekisteröityä kuin yhden kerran. Salasanaksi käy 7-14 merkkiä pitkä, isoja tai pieniä kirjaimia, numeroita ja ala- tai väliviivoja sisältävä merkkijono. Lomakkeessa on myös kaksi vapaaehtoista kenttää: koko nimi ja ammatti. Ammatin voi valita pudotusvalikon neljästä vaihtoehdosta, jotka ovat: arkkitehti, suunnittelija, opiskelija ja utelias. ”**Register**”-välilehti näkyy ainoastaan kirjautumattomille käyttäjille. Rekisteröitymislomakkeessa on myös tarkistuskohta, jossa käyttäjän pitää tehdä helppo laskutoimitus annetun kuvan perusteella ja antaa vastaus viereiseen kenttään. Tällä pyritään varmistamaan, että rekisteröitymässä on ihminen eikä kone.



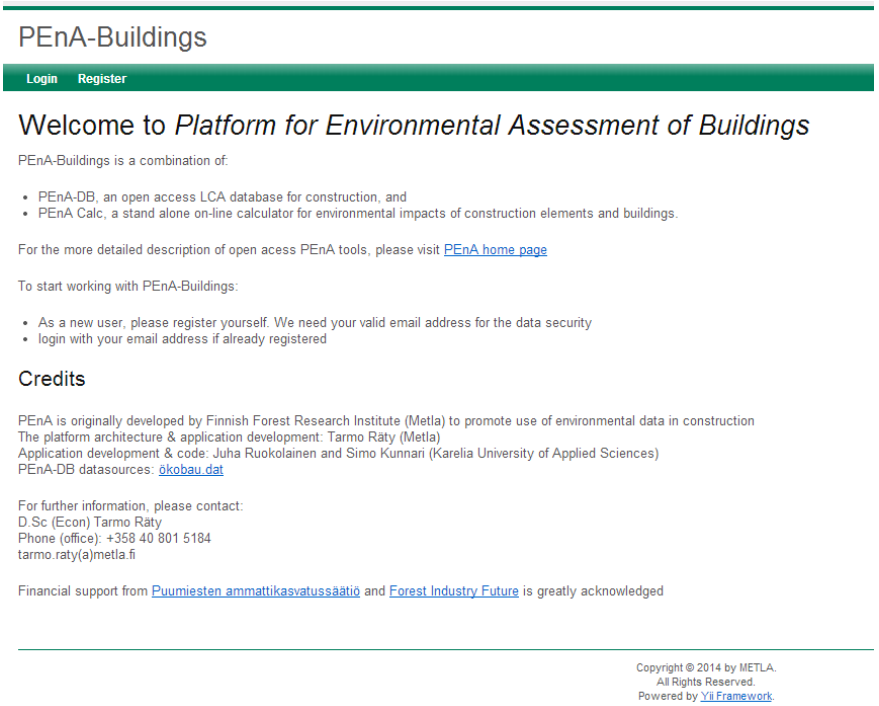
Kuva 5. Rekisteröitymislomake

4.2.3 Kirjautuminen

Käyttäjä syöttää ***käyttäjätunnuksen*** ja ***salasanan*** kirjautumisnäkymään ja painaa ”**Login**”-painiketta (kuva 6). Jos käyttäjätunnusta ei löydy palvelimelta tai salasana on väärä, niin siitä tulee virheilmoitus. Jos käyttäjätunnus ja salasana täsmäävät palvelimella oleviin käyttäjätietoihin, sovellus kirjaa käyttäjän sisään ja käyttäjälle avautuu ”**Home**”-näkymä. ”**Home**”-näkymä on muutenkin ensimmäinen sivu, jonka käyttäjä näkee tullessaan ensimmäistä kertaa tähän sovellukseen (kuva 7).



Kuva 6: Kirjautumislomake

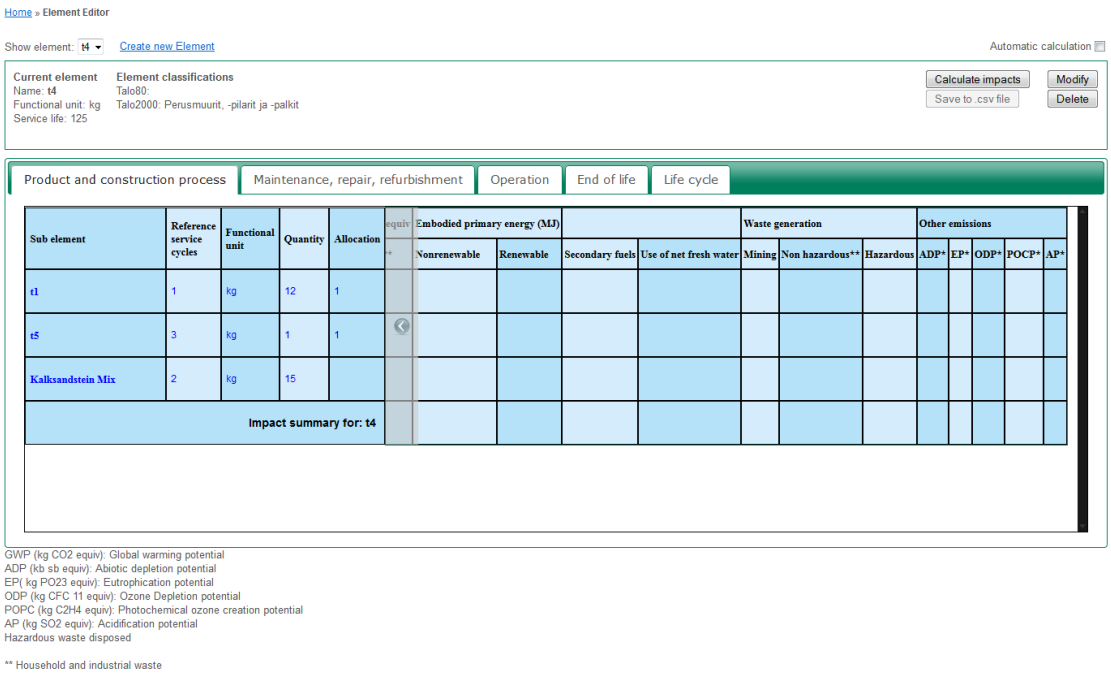


Kuva 7: Etusivu kirjautumattomalle käyttäjälle

4.2.4 Osaeditori

Osaeditoriin pääsee valitsemalla välilehtipalkista ”**Element Editor**”-välilehden. Osaeditorista avautuu tällöin päänäkymä.

Päänäkymä koostuu kahdesta osasta, joista ylimpänä ovat rakennusosan tiedot ja muut toiminnot sekä alempana rakennusosan alielementit ja niiden ympäristövaikutukset (kuva 8).



Kuva 8: Osaeditorin päänäkymä

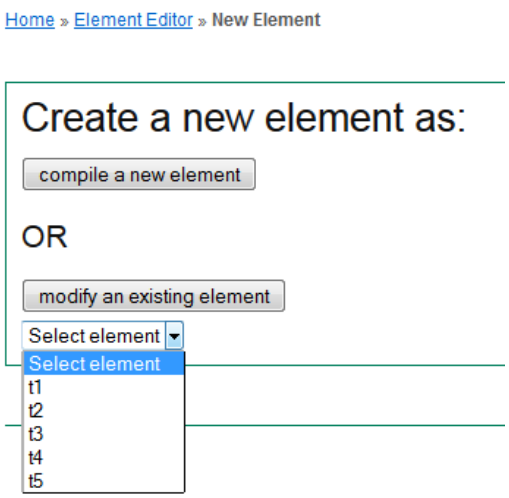
Ylimpänä, eli rakennusosan tiedoissa, näkyy pudotusvalikko halutun rakennusosan valitsemiseksi sekä linkki uuden rakennusosan luomiseen (ks. kappale 4.2.5). Niiden alla ovat rakennusosan nimi, yksikkö ja elinikä sekä mahdollisesti Talo80- ja Talo2000-rakennusosaluokitukset. Lisäksi oikealla on valintaruutu rakennusosan ympäristövaikutusten automaattiselle laskemiselle sekä neljä painiketta: ”**Modify**” (ks. 4.2.6), ”**Delete**”, ”**Calculate impacts**” ja ”**Save to .csv file**” (ks. 4.2.7). ”**Delete**”-painikkeella voi poistaa rakennusosan kokonaan järjestelmästä. Alielementtejä ei poisteta sen mukana, vaan ne pitää poistaa tarvittaessa erikseen. ”**Calculate impacts**”-painikkeesta pystytään laskemaan ympäristövaikutukset manuaalisesti, jollei valintaruutua automaattiselle laskemiselle ole valittu. Automaattinen laskeminen tapahtuu sivun avaamisen aikana.

Keskellä näkyvät rakennusosan alielementtien parametrit ja lasketut ympäristövaikutukset. Ympäristövaikutuksia tarkastellaan viideltä eri jaksolta, jotka näkyvät editorissa välilehtinä. ”**Product and construction process**”-välilehdellä näkyvät rakentamisen ympäristövaikutukset, ”**Maintenance, repair, refurbishment**”-välilehdellä kunnossapidon vaikutukset, ”**Operation**”-lehdellä kiinteistöhuollon, ”**End of life**”-lehdellä purkamisen vaikutukset ja ”**Life cycle**”-lehdellä ympäristövaikutukset koko rakennuksen elinkaaren ajalta. Jokaisen välilehden näkymä on samanlainen, ainoastaan arvot ovat erilaisia. Alielementtien parametreina näkyy ”**Sub element**” eli alielementin nimi, ”**Reference service cycles**” eli eliniän kerroin pääosaan nähden, ”**Functional unit**” eli yksikkö, ”**Quantity**” eli määrä ja ”**Allocation**” eli allokointi. Näiden jälkeen näkyvät elementin lasketut ympäristövaikutukset, joita on 13 kappaletta. Ne on kuvattu tarkemmin liitteessä 1. Näkymässä on käytetty apuna Yii:n lisäosaa ”hoverscroll”, jolla näkymää voidaan vierittää horisontaalisesti viemällä hiiren kursori vieritysnuolten päälle.

4.2.5 Uusi rakennusosa

4.2.5.1 ”New Element”-näkymä

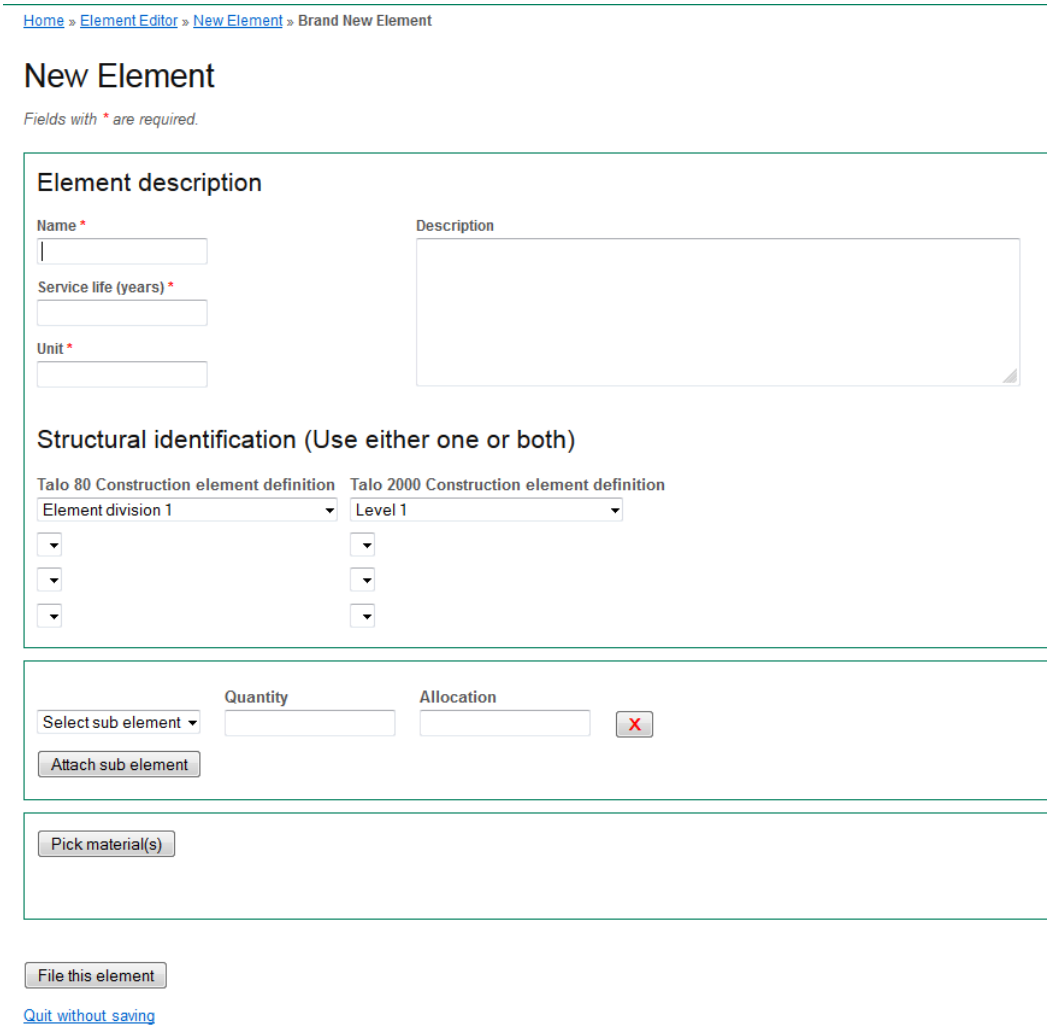
”**New Element**”-näkymään pääsee “**Element Editor**”-näkymän linkistä “New Element”. Näkymässä pitää valita tekeekö uuden rakennusosan toisen rakennusosan pohjalta painikkeesta ”**modify an existing element**” vai tehdäänkö ihan uusi rakennusosa ”**compile a new element**”-painikkeesta (kuva 9).



Kuva 9: Uusi rakennusosa-näkymä

4.2.5.2 ”Brand New Element”-näkymä

Näkymä koostuu neljästä kohdasta: rakennusosan tiedot, alirakennusosat, materiaalien määrittelyt ja rakennusosan tallennus (kuva 10).



Kuva 10: Ihan uusi rakennusosa –näkymä

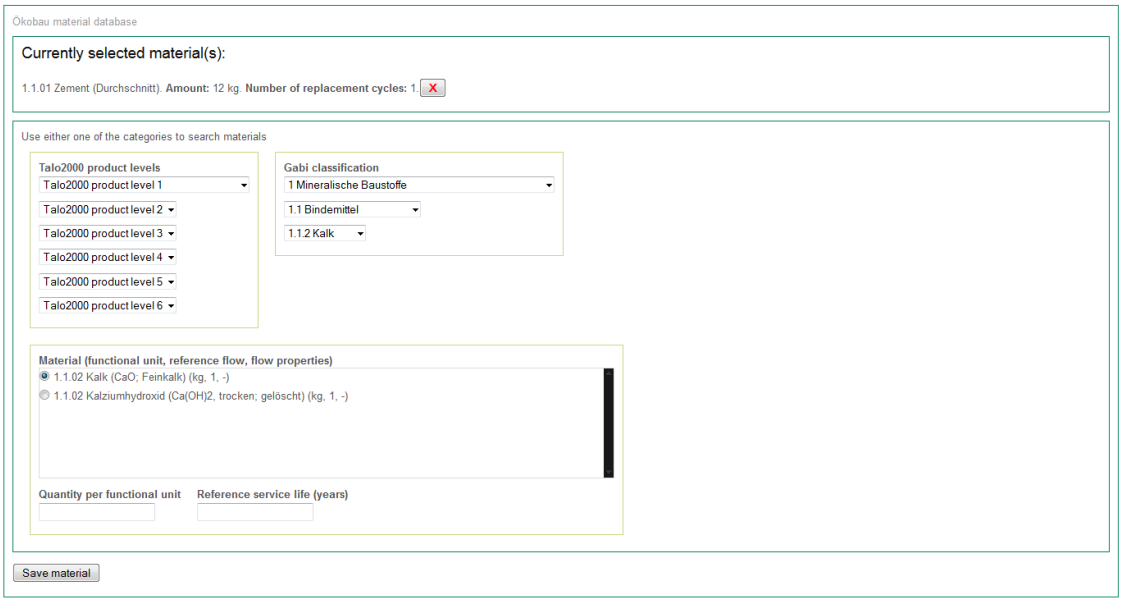
”**Element description**” -kohdassa määritetään rakennusosan tiedot, eli annetaan rakennusosalle nimi ”**Name**”-kenttään, elinikä ”**Service life (years)**”-kenttään, yksikkö ”**Unit**”-kenttään ja mahdollinen kuvaus ”**Description**”-kenttään. Kuvaus on vapaaehtoinen. Niiden alle tulevat mahdolliset Talo80- ja Talo2000-määrittelyt, jotka ovat myös vapaaehtoisia. Molemmat Talo80- ja Talo2000-rakennusosamäärittelyt tehdään omilla neljällä alasvetovalikolla, joista aina ylempi määrää seuraavan, alapuolella olevan valikon sisällön.

Rakennusosan mahdolliset alirakennusosat valitaan keskimmäisessä kohdassa. Alirakennusosa valitaan alasvetovalikosta, jossa näkyvät kaikki käyttäjän siihen asti määrittelemät rakennusosat. Aliosan määrä sen yksikköä kohden

annetaan ”**Quantity**”-kenttään ja allokaatio ”**Allocation**”-kenttään. ”**Attach sub element**”-painikkeesta käyttäjä voi lisätä lisää aliosia ja ”**X**”-painikkeesta kyseisellä rivillä oleva aliosa poistetaan tietoineen rakennusosan luonnista.

Tämän jälkeen on materiaalien lisäys rakennusosaan (kuva 11). Sen saa auki ”**Pick material(s)**”-painikkeesta. Käyttäjä voi lisätä materiaalin Ökobau materiaalitietokannan Gabi-luokkien avulla tai Talo2000-tuotemäärittelyn kautta. Gabi-luokat on määritelty neljään tasoon, jossa neljäs taso on materiaalitaso. Talo2000-tuotemäärittely on kuusitasoinen. Seitsemäs taso on materiaalitaso ja materiaalit tulevat Ökobau tietokannasta, mutta kaikille tuotteille tällä hetkellä löydy vastaavaa materiaalia, joten materiaaliosio voi jäädä tyhjäksikin sopivilla valinnoilla. Materiaalitasolta voi valita vain yhden materiaalin radiopainikkeilla. Sen määrä yksikköä kohti tulee ”**Quantity per functional unit**”-kenttään ja elinikä ”**Reference service life (years)**”-kenttään. ”**Save material**”-painikkeesta materiaali tallennettaan sovellukseen ja se tulee näkyviin ”**Currently selected material(s):**”-kohtaan, mutta materiaaleja ei vielä tallenneta tietokantaan.

Alimpana on ”**File this element**”-painike, josta tallennetaan rakennusosan, aliosien ja materiaalien tiedot tietokantaan sekä ”Quit without saving”-linkki, josta pääsee osaeditorin päänäkymään tallentamatta rakennusosan tietoja minnekään.



Kuva 11: materiaalin lisäys rakennusosaan

4.2.5.3 ”Compile new element from existing element”-näkymä

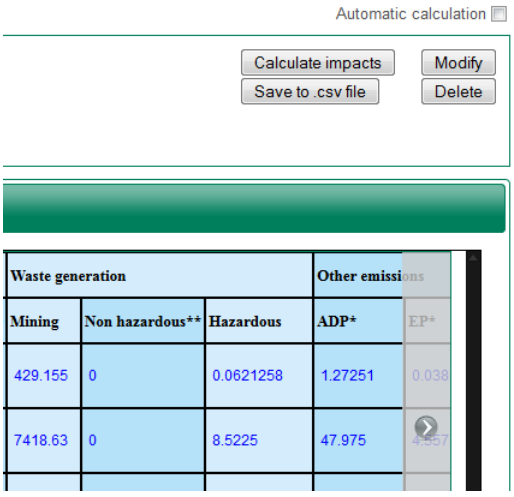
Tähän näkymään pääsee ”**New element**”-näkymästä, kun on ensin painanut ”**modify an existing element**”-painiketta ja valitsemalla sitten jonkun rakennusosan lähteeksi (ks. kappale 4.2.5.1). Tämä näkymä on muuten identtinen ”**Brand New Element**”-näkymän (ks. 4.2.5.2) kanssa, mutta kaikki kentät on täytetty pohjana olevan rakennusosan tiedoilla ja käyttäjä voi halutessaan muokata ne haluamillaan arvoilla. Ainoastaan ”**Name**”-kenttä on pakko muuttaa, koska käyttäjällä ei saa olla kahta samannimistä rakennusosaa. Ohjelma antaa virheilmoituksen, jos käyttäjä yrittää tallentaa muokatun osan jo olemassa olevalla nimellä.

4.2.6 Rakennusosan muokkaaminen

Tähän näkymään pääsee osaeditorin päänäkymän (ks. 4.2.4) ”**Modify**”-painikkeesta. Näkymä on muuten identtinen ”**Brand New Element**”-näkymän (ks. 4.2.5.2) kanssa, mutta kaikki kentät on täytetty muokattavan rakennusosan tiedoilla. Käyttäjä voi edelleen muokata kenttiä haluamillaan arvoilla ja lopuksi osaan tehdyt muutokset pitää tallentaa ”**Save changes**”-painikkeella, jolloin ne päivitetään tietokantaan. Myös rakennusosan nimen voi halutessaan vaihtaa.

4.2.7 Laskettujen ympäristövaikutusten tallentaminen tiedostoon

Osaeditorin päänäkymässä on ”**Save to .csv file**”-painike laskettujen ympäristövaikutusten tallentamiseksi .csv-tiedostoon. Tämä painike avautuu, kun ympäristövaikutukset on laskettu joko ”**Calculate impact**”-painikkeesta tai ne on laskettu automaattisesti, kun ”**Automatic calculation**”-valintaruutu on valittuna.



Kuva 12: Ympäristövaikutusten tallentaminen .csv-tiedostoon

4.2.8 Monen XML-tiedoston lukeminen

Tämän ominaisuuden käyttö lopetettiin, kun tietokannan kehityksessä alettiin käyttää MySQL dump-tiedostoja. Dump-tiedoston avulla materiaalitietokannan uudelleen tuominen tietokantaan on nopeampaa kuin XML-tiedostoista lukemalla. Jos Ökobaun materiaalitietokanta vaihdetaan uudempaan versioon, niin XML-tiedostojen luku pitäisi toimia, mutta päivityspainikkeen toiminta pitää suunnitella uudelleen, jos uudesta materiaalitietokannasta löytyy virheitä tai täydennettävää.

”**Read multiple XMLs**”-välilehti on näkyvissä vain järjestelmän ylläpitäjille eli järjestelmänvalvojan oikeuksin varustetulla käyttäjätunnuksella kirjautuneille käyttäjille. Valitsemalla kyseisen välilehden pääsee näkymään, jossa PE-NA_MASTER_DB-tietokantaan voi lukea tietoa Ökobaun XML-tietokannan XML-tiedostoista. Sivu sisältää ohjeet tiedostojen lukemiseksi sekä kaksi painiketta: ”**Save**” ja ”**Update**” (kuva 13). Kun ”**Save**”-painiketta painetaan, sovellus etsii XML-tiedostoja *webroot\files\okobau.dat2011\processes_renamed* -kansiosta. Jos XML-tiedostoja löytyi, ne luetaan tietokannan *okobau*- ja *effect*-tauluihin.

Huomaa, että Ökobaun 2011 XML-tiedostoissa on virhe, joka pitää korjata käsin yhteen tiedostoista ennen tiedostojen lukemista MySQL-tietokantaan. Virhe aiheuttaa ylimääräisen rivin tallentumisen tietokantaan, jolloin tietokannan *okobau*-taulun avainkenttä **material_id** ei ole kyseisestä rivistä alkaen ajan tasalla. Virhe tulee korjata *2.6.02_Foamglas_Perinsul_-_Pittsburgh_Corning.xml* -tiedoston kohtaan: *<baseName xml:lang="de">2.6.02 Foamglas Perinsul - Pittsburgh Corning</baseName>*. Kohta pitää muuttaa seuraavaan muotoon: *<baseName xml:lang="de">2.6.01 Foamglas Perinsul - Pittsburgh Corning</baseName>*. Eli gabi-hierarkian kakkonen pitää muuttaa ykköseksi.

”**Update**”-painikkeella päivitetään tietokantaan XML-tiedostoista puuttuneet tai virheelliset gabi-hierarkian kolmen ylimmän tason tiedot. Tällä toiminnolla tietokantaan lisätään myös kyseisten kolmen ylimmän tason suomennokset *okobau*-taulun kenttään **name_fi** sekä linkitys *okobau*- ja *talo2000_construction_products*-taulujen välille tauluun *talo2000_okobau*. Tiedot sijaitsevat osittain ohjelmakoodissa ja osittain *okobau_nimet_ja_koodit.txt* -

tiedostossa. Kyseinen tiedosto pitää olla *webroot\calculator* -kansiossa ennen kuin ”**Update**”-painiketta painetaan. Ohjelmakoodissa ei yleisesti ottaen oteta kantaa XML-tietokannassa oleviin virheisiin, vaan ne pitää korjata käsin joko XML-tiedostoihin ennen MySQL-tietokantaan lukemista tai sitten lukemisen jälkeen suoraan tietokantaan.

Näkymässä on lisäksi ”Show database content”-linkki. Siitä avautuu näkymä, jossa on *okobau*-taulun kenttien **material_group_1_id**, **material_group_2_id**, **material_group_3_id**, **material_sub_id** ja **basisname_de** sisältö. Tämä ominaisuus on jäänne sovelluksen kehitystyöstä, jolla sai tarkistettua nopeasti mitä tietokantaan on tiedostojen luvun myötä tallentunut.

Kuva 13: Monen XML-tiedoston lukeminen

4.2.9 Yksittäisen XML-tiedoston lukeminen

Tämän ominaisuuden käyttö lopetettiin, kun tietokannan kehityksessä alettiin käyttää MySQL dump-tiedostoja.

”**Read single XML**”-välilehti on näkyvissä vain järjestelmän ylläpitäjille eli järjestelmänvalvojan oikeuksin varustetulla käyttäjätunnuksella kirjautuneille käyttäjille. Valitsemalla kyseisen välilehden päästään näkymään, jossa on ainoastaan tiedostokenttä ja ”Show database content”-linkki (kuva 14). Tiedostokentän avulla voidaan valita haluttu XML-tiedosto käyttäjän koneen tiedostojärjestelmästä ja lukea se MySQL-tietokantaan ”**Save**”-painikkeen avulla. ”Show database content”-linkistä avautuu näkymä, jossa on OKOBAU-taulun kenttien **material_group_1_id**, **material_group_2_id**, **material_group_3_id**, **material_sub_id** ja **basisname_de** sisältö. Tämä näkymä on jäänne sovelluksen

kehitystyöstä, jolla sai tarkistettua nopeasti mitä tietokantaan on tiedoston luvun myötä tallentunut.

Kuva 14: Yksittäisen XML-tiedoston lukeminen

4.2.10 Käyttäjien hallinta

Käyttäjien hallinta välilehti on tarkoitettu järjestelmän ylläpitäjälle ja se näkyy vain hänelle. Tässä näkymässä on hakukenttä, lista sovellukseen rekisteröityneistä käyttäjistä ja käyttäjien hallinnointiin tarkoitetut painikkeet.

Hakukentällä voidaan hakea käyttäjiä heidän sähköpostiosoitteen tai nimen avulla ja kun painetaan ”Hae”-painiketta, niin hakutulokset tulevat alas näkyviin.

Tällä hetkellä (2014-05-14) käyttäjien hallintaa ei ole vielä tehty.

4.2.11 Taloeditori

Taloeditorissa voi tarkastella koko talon rakennusosien hierarkiaa. Tällä hetkellä (2014-05-14) taloeditoria ei ole vielä tehty.

5 ULKOISET LIITTYMÄT

5.1 Laitteistoliittymät

Jos käyttäjän koneelle on asennettu tulostin, niin sitä voi käyttää Internet-selaimen tulostus-toiminnon kautta. Käyttäjällä pitää olla Internet-yhteys, joka vaatii liittymäkohtaisesti omat laitteet, jotka ovat yhteydessä käyttäjän tietokoneeseen. Palvelimessa, jossa sovellus on asennettuna, pitää myös olla Internet-yhteys.

5.2 Ohjelmistoliittymät

Sovellus käyttää MySQL-palvelinta tietokantojen hallinnointiin. Sovellus on toteutettu pääosin PHP:lla, mutta joitain osia on tehty CSS:llä ja Javascriptillä.

5.3 Tietoliikenneliittymät

Jotta käyttäjä pääsee käyttämään sovellusta, tarvitsee hän toimivan Internet-yhteyden.

6 MUUT OMINAISUUDET

6.1 Suorituskyky ja vasteajat

Sovellus toimii palvelimella, johon käyttäjä on yhteydessä tavallisella Internet-selaimella. Käyttäjiä voi olla yhteydessä palvelimeen useita yhtä aikaa.

Sovelluksen vasteajat riippuvat paljolti palvelimen ja Internet-yhteyden nopeudesta, joten niitä ei voi määrittää. Lisäksi vasteajat eivät ole oleellisia, koska kyseessä ei ole reaaliaikajärjestelmä.

6.2 Käytettävyys, toipuminen, turvallisuus, suojaukset

Sovelluksessa ei oteta kantaa toipumiseen, elpymiseen, eikä sähkökatkoksiin palvelimessa. Vikatilanteessa menetetään kaikki palvelimen massamuistiin tallentamattomat tiedot.

Käyttäjät, joilla on sovelluksen käyttöön tarvittavat tunnukset ja salasanat, voivat syöttää virheellistä tietoa tietokantaan.

Käyttäjän antama salasana salakirjoitetaan tietokantaan. Itse selkokielistä salasanaa ei tallenneta palvelimella erikseen mihinkään.

6.3 Ylläpidettävyys

Sovelluksen ylläpitäjällä on omat toiminnallisuudet heidän kirjauduttua ylläpitäjän käyttäjätunnuksella sovellukseen ja ne on kerrottu luvusta 4.2.8 lukuun 4.2.10.

6.4 Siirrettävyys/kannettavuus, yhteensopivuus

Sovellus on yhteensopiva suosituimpien Internet-selainten uusimpien versioiden kanssa. Tällä hetkellä suosituimpia Internet-selaimia ovat: Mozilla Firefox, Internet Explorer ja Google Chrome.

6.5 Operointi

Luvussa 4.1.1 kerrotaan mitä pitää tehdä sovelluksen asentamiseksi palvelimelle.

7 SUUNNITTELURAJOITTEET

7.1 Standardit

Sovelluksessa on käytetty PHP:n 5.3.16 versiota ja se on testattu Apachen www-palvelimen versiolla 2.2.22. Kehitystyössä on lisäksi käytetty apuna MySQL-tietokantapalvelimen versiota 5.5.27.

7.2 Laitteistorajoitteet

Laitteesta tulee löytyä yhteys Internetiin.

Suositeltava näytön resoluution leveys on vähintään 1400 pikseliä.

7.3 Ohjelmistorajoitteet

Luvussa 2.5 on palvelimessa tarvittavien ohjelmien versiot. Käyttäjän käyttämissä Internet-selaimissa pitää olla vähintään versiot: Mozilla Firefox 18, Internet Explorer 9 ja Google Chrome 24.

8 JATKOKEHITYSAJATUKSIA

Sovellukseen olisi tarkoitus lisätä tuki suomen ja saksan kielelle. Sovelluksen ympäristölaskurin pitäisi sisältää kaikki elinkaaren vaiheet. Sovelluksen CSV-tiedostoon viennin asettelua pitäisi muuttaa paremmaksi.

Salasana pitäisi voida vaihtaa. Tätä varten voisi olla erillinen välilehti, jossa voi antaa uuden salasanan. Välilehti näkyisi vain kirjautuneille käyttäjille.

Myös unohtuneen salasanan palautustoiminto olisi hyvä olla olemassa. Se voisi toimia siten, että käyttäjän sähköpostiin lähetettäisiin kertakäyttöinen salasana, jolla pääsisi kirjautumaan yhden kerran sovellukseen. Käyttäjää ohjeistettaisiin sähköpostissa vaihtamaan salasana heti kirjautumisen jälkeen salasananvaihtovälilehdellä.

Ympäristövaikutukset Ökobau-tietokannassa Toiminnallinen määrittely. Liite 1

1	Primärenergie nicht regenerierbar	Primary energy non-renewable	Uusiutumaton primäärienergia
2	Primärenergie regenerierbar	Primary energy renewable	Uusiutuva primäärienergia
3	Sekundärbrennstoffe	Secondary fuels	Polttoaineiden uudelleen käyttö
4	Wassernutzung	Water utilization	Veden käyttö
5	Abraum und Erzaufbereitungsrückstände	Overburden and ore processing residues	Louhintajätteet
6	Hausmüll und Gewerbeabfälle	Municipal waste	Yhdyskuntajätteet
7	Sonderabfälle	Hazardous waste	Ongelmajäte
8	Abiotischer Ressourcenverbrauch (ADP)	Abiotic Depletion Potential (ADP)	Elottomien ympäristötekijöiden ehtyminen
9	Eutrophierungspotential (EP)	Eutrification potential (EP)	Rehevöityminen
10	Ozonabbaupotential (ODP)	Ozone Depletion Potential	Otsonikato
11	Photochem. Oxidantienbildungspot. (POCP)	Photochemical Ozone Creation Potential (POCP)	Fotokemiallinen otsonin muodostuminen
12	Treibhauspotential (GWP 100)	Global Warming Potential (GWP)	Ilmastonmuutos
13	Versauerungspotential (AP)	Acidification potential (AP)	Maaperän ja veden happamoituminen