Ylläpitodokumentti

Kivireki

Helsinki 17.12.2007

Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Kurssi

581260 Ohjelmistotuotantoprojekti (6 ov)

Projektiryhmä

Anu Kontio
Ilmari Helen
Olli Juvonen
Joonas Murtola
Teppo Niinimäki

Asiakas

Timo Aalto

Ohjaaja

Jari Suominen

Kurssin vastuuhenkilö

Kimmo Simola

Kotisivu

`http://www.cs.helsinki.fi/group/kivireki`

Versiohistoria

Versio	Päiväys	Tehdyt muutokset
0.1	3.12.2007	Pohja kopioitu suunnitteludokumentista
1.0	17.12.2007	Lopullinen versio

Sisältö

1 Johdanto

Tämä on ylläpitodokumentti syksyn 2007 Ohjelmistotuotantoprojekti-kurssilla toteutetulle kurssikirjatietojen hallintajärjestelmälle. Järjestelmän avulla opetushallinto ja luennoitsijat voivat koota tiedot järjestettävistä kursseista sekä niiden kurssikirjoista ja välittää tiedot kirjastolle. Kirjaston henkilökunta pystyy järjestelmän avulla hotamaan kurssikirjojen hankintaan ja yleiseen hallintaan liittyviä toimenpiteitä. Järjestelmä on alkuvaiheessa tarkoitettu Kumpulan tiedekirjaston ja Helsingin yliopiston Tietojenkäsittelytieteen laitoksen yhteiseen käyttöön. Myöhemmin sitä on mahdollista laajentaa koskemaan Helsingin yliopiston laitoksia ja kirjastoja.

Järjestelmä asennettiin kirjaston palvelimelle ja sitä käytetään WWW-selaimella. Ohjelmisto on toteutettu Java-kielellä ja tietojen tallennukseen käytetään MySQL-tietokantaa. Toteutus tehtiin MVC-mallin mukaisesti Apache Struts - ja Hibernate -sovelluskehysten avulla.

Tämä dokumentti kuvaa toteutetun järjestelmän ylläpitoon liittyviä asioita. Dokumentissa tarkastellaan muuttuneita suunnitelman osia ja esitetään parannusehdotuksia ratkaisuihin, erityisesti tietokannan toteutukseen. Dokumentissa myös kuvataan muutokset käyttöliittymä-suunnittelussa, esitellään sovelluksen rakennetta, listataan toteutetut ja toteuttamattomat vaatimukset, sekä kerrotaan koodin ja tietokannan ylläpitoon liittyviä asioita. Lopussa käsitellään myös testaustuloksia.

Dokumenttia on tarkoitus lukea yhdessä vaatimusdokumentin sekä ohjelmakoodin ja kommentoinnin kanssa. Tietokantaosiossa viitataan myös suunnitteludokumenttiin. Testausosiossa mainitut testitapaukset löytyvät testaussuunnitelman liitteenä. Nämä dokumentit tulisi siis olla luettavissa tätä dokumenttia käsiteltäessä.

2 Sanasto

2.1 Käytettävien tekniikoiden nimet

Apache Struts – Web-sovelluksille tarkoitettu sovelluskehys, joka perustuu sovelluksen toimintalogiikan jakamiseen MVC-mallin mukaiseksi. Struts abstrahoi J2EE:n Servlet rajapinnan ja helpottaa sovelluksen toteutuksen rutiininomaisten puuhien toteutuksessa.

Apache Tomcat – Web-palvelinohjelmisto, joka toteuttaa Javan Servlet-rajapinnan.

Hibernate – Avoimen lähdekoodin sovelluskehys tietokantaoperaatioita varten.

HQL – Hibernate Query Language on Hibernaten oma tietokanta-kyselykieli, jonka avulla tietokantoihin voidaan tehdä mm. hakuja ja muutoksia. Katso myös *SQL*.

InnoDB – Tietokannan tallennusmoottori, joka tuo hyödyllistä lisätoiminnallisuutta tietokantaoperaatioihin.

JavaScript – Web-selaimen oliopohjaimen komentosarjakieli.

JDBC – Java Database Connectivity, Javan tietokantarajapinta, joka määrittelee tavan jolla Java-sovellus voi ottaa yhteyden tietokantaan. Sovelluksen ja tietokannan välisen kommunikoinnin hoitaa käytännössä tietokantakohtainen tietokanta-ajuri.

JSP – JavaServer Pages on Javan Servlet-rajapintaa hyödyntävä tekniikka, joka mahdollistaa web-sivujen dynaamisen generoinnin selainohjelmille. JSP käyttää omaa merkkausmenetelmäänsä, joka mahdollistaa Java-koodin upottamisen HTML:n sekaan.

LDAP – Lightweight Directory Access Protocol, hakemistopalvelujen tietojen hallinnointiin tarkoitettu, TCP/IP:n päällä toimiva sovelluskerroksen verkkoprotokolla.

MVC – Model-View-Controller on ohjelmistotuotannossa käytettävä arkkitehtuurimalli, jossa ohjelmisto jaetaan kolmeen osaan: tietosisällön hallinta, käyttöliittymä sekä näitä hallinnoivaan kontrolliosio.

MySQL – Avoimen lähdekoodin olio-relaatiotietokantajärjestelmä. MySQL-kantaa voidaan hallinnoida SQL-kielellä.

SQL – Structured Query Language on tietokanta-kyselykieli, jonka avulla tietokantoihin voidaan tehdä mm. hakuja ja muutoksia.

XHTML – eXtensible Hypertext Markup Language on kuvauskieli, jota käytetään websivujen luomiseen. Erona perinteiseen HTML:n, XHTML täyttää XML-standardin muotovaatimukset.

2.2 Yleiset termit

arvioitu osallistujamäärä – Arvio kurssille osallistuvista opiskelijoista, johon uusien kirjojen hankinta suurelta osin perustuu. Tämä tieto syötetään järjestelmään laitoksen päässä. Kurssihallinto syöttää samanlaisen arvion myös TKT-laitoksen Ilmo-järjestelmään.

kirjasto – Kirjasto on järjestelmän tilaaja sekä toinen pääasiallinen käyttäjä. Kirjasto käyttää järjestelmää kurssikirjatietojen hankintaan.

kurssi – Säännöllisesti tai vähemmän säännöllisesti järjestettävä opintokokonaisuus. Kurssi voidaan luennoida/järjestää useita kertoja. Jokaisella kurssilla on kurssikoodi (kurssin yksilöivä tunnus) sekä nimi. Esimerkiksi “581326-3, Java-ohjelmointi”, “99501Käyt, English Academic & Professional Skills: Reading, Writing & Spoken Communication” tai “477628, PSY382 Varianssianalyysi”.

kurssihallinto – Kurssihallinolla tarkoitetaan dokumentissa opetushallintoa.

kurssikerta – Kurssin yksi järjestämiskerta. Esimerkiksi “Java-ohjelmointi, syksy 2007, periodi 2”.

laitos – Laitoksella viitataan järjestelmän toiseen käyttäjään – joko Tietojenkäsittelytieteen laitokseen tai yleisesti johonkin Helsingin yliopiston laitokseen. Laitoksen piiriin kuuluvat luennoitsija sekä kurssihallinto, jotka syöttävät järjestelmään tarpeelliset kurssi- ja kurssikirjatiedot.

sessio – Istunto. Yhtenäinen ajanjakso, jolloin käyttäjä on kirjautuneena järjestelmään.

tietokanta – “Kokoelma tietoja, joilla on yhteys toisiinsa.” Tässä dokumentissa tietokannalla viitataan käytettävään MySQL-relaatiotietokantaan.

3 Asennusohje

Tässä luvussa kuvataan sovelluksen asentamiseksi tarvittavat toimenpiteet. Järjestelmä on testattu Tomcat-sovelluspalvelimella mutta sen pitäisi toimia kaikissa Servlet-rajapinnan toteuttavissa WWW-palvelimissa. Ainoastaan viitteet tomcat-hakemistoihin tulee muuttaa vastaamaan käytettyä palvelinta.

Tässä ohjeessa oletetaan, että järjestelmässä, johon sovellus asennetaan on valmiiksi asennettuna seuraavat komponentit: Tomcat-palvelin, versio 5.5.17 MySQL-palvelin, versio 4.7 Java JRE -suoritusympäristö, versio 1.6.0_02

Järjestelmä on testattu yllä olevalla kokoonpanolla eikä sen toimivuutta komponenttien muiden versioiden kanssa taata. Tämä ohje ei kata näiden komponenttien asennusta.

Seuraavassa kuvataan tietokannan luominen MySQL-palvelimelle, tietokannan toimintaan tarvittavat asetustiedostojen muutokset sekä sovelluksen asentaminen Tomcat-palvelimelle.

3.1 Tietokannan luominen

Tämä aliluku kuvaa tietokannan luomisen käyttäen MySQL-palvelimen mukana tulevaa hallintatyökalua. Kannan voi luoda myös käyttäen muita työkaluja.

1. Kirjautuminen MySQL-palvelimelle

   Tietokannan luomiseksi täytyy kirjautua MySQL-palvelimelle, tämä tapahtuu komennolla:

   ```
   mysql -u <käyttäjä> -h <host> -p
   ```

   Komento tulee suorittaa MySQL-tietokannan juurihakemistosta löytyvässä bin-hakemistossa. Komennossa `<käyttäjä>`-merkkijono korvataan käyttäjänimellä, jolla palvelimelle kirjaudutaan. `<host>` taas korvataan palvelimen osoitteella eli localhost mikäli palvelimelle ei olla kirjautumassa etänä.

   Mikäli kirjautuminen onnistui, komentorivillä pitäisi komennon suorittamisen jälkeen näkyä kehoite:

   ```
   mysql>
   ```

2. Tietokannan luominen

   Tietokannan luomiseksi kehoitteessa suoritetaan komento, tämä tapahtuu syöttämällä seuraava rivi kehoitteeseen ja painamalla enter:

   ```
   CREATE DATABASE kivireki;
   ```

   Tämän jälkeen siirrytään käyttämään luotua kantaa komentamalla:

```
USE kivireki;
```

3. Tietokannan taulujen luominen

   Tietokannan taulujen luomista varten sovelluksen asennus-cd:n hakemistosta `sql` löytyy tiedosto `sql_all.sql`. Tiedosto sisältää tarvittavat komennot taulujen luomiseksi ja se ajetaan suorittamalla MySQL-hallintatyökalun kehoitteessa komento:

   ```
   source <tiedoston polku>;
   ```

   `<tiedoston polku>`-merkkijono korvataan tiedoston polulla, esim. `C:\sql_all.sql`

   Tämän jälkeen asennuksen onnistumisen voi varmistaa komennolla:

   ```
   show tables;
   ```

   Komennon tulisi listata tietokannan taulut. Mikäli tauluja ei näy, on jokin mennyt pieleen.

3.2 Sovelluksen asentaminen Tomcat-palvelimelle

Järjestelmän asennus tapahtuu siirtämällä sovelluksen sisältävä war-tiedosto WWW-palvelimen ajettavat sovelluksen sisältävään hakemistoon. Tämän jälkeen tietokannan tiedot pitää päivittää asetustiedostoon.

1. Sovelluksen asentaminen Tomcat-palvelimelle

   Sovelluksen asennus-cd:ltä löytyy tiedosto `kivireki.war`, joka sisältää sovelluksen. Asennus tapahtuu siirtämällä tiedosto tomcat-palvelimen juurihakemistosta löytyvään `webapps`-hakemistoon. Mikäli palvelin on käynnissä, se asentaa automaattisesti sovelluksen tiedoston siirron jälkeen. Muuten palvelin tulee käynnistää ajamalla komento:

   ```
   tomcat/bin/startup
   ```

   Komennossa tomcat kuvaa Tomcat-palvelimen juurihakemistoa.

2. Tietokannan asetusten asettaminen

   Tietokannan asetukset asetetaan tiedostossa `hibernate.cfg.xml`, joka löytyy sovelluksen asennukset jälkeen `tomcat/kivireki/WEB-INF/classes`-kansiosta. Tiedoston voi avata millä tahansa tekstieditorilla. Alla alkuosa tiedostosta, jonka alla ohjeet tarvittavista muutoksista:

```
1 <?xml version="1.0" encoding="utf-8" ?>
2 <!DOCTYPE hibernate-configuration (View Source for full doctype...)>
3 <hibernate-configuration>
4 <session-factory>
5 <!--  Database connection settings -->
6 <property name="connection.driver_class">com.mysql.jdbc.Driver</property>
7 <property name="connection.url">jdbc:mysql://localhost/kivireki</property>
8 <property name="connection.username">kivireki</property>
9 <property name="connection.password">kivireki</property>
10 <property name="connection.zeroDateTimeBehavior">convertToNull</property>
11 <!--  JDBC connection pool (use the built-in)   -->
12 <property name="connection.pool_size">1</property>
13 <!--  SQL dialect   -->
14 <property name="dialect">org.hibernate.dialect.MySQLDialect</property>
15 <!--  Enable Hibernate's automatic session context management   -->
```

Rivillä kuusi asetetaan käytetyn JDBC-ajurin sijainti, käytettäessä MySQL-tietokantaa riviä ei tarvitse muokata.

Rivillä seitsemän asetetaan tietokannan sijainti, tähän tulee asettaa missä tietokanta sijaitsee. Mikäli tietokanta sijaitsee samalla palvelmella kuin Tomcat, riviä ei tarvitse muokata.

Rivillä 8 ja 9 asetetaan käyttäjätunnus ja salasana, joille MySQL-palvelimelle kirjaudutaan.

Rivillä 14 asetetaan käytetty SQL-kielen tulkintaluokka, mikäli käytössä MySQL, ei tarvitse muokata.

Lopuksi tiedosto tallennetaan ja Tomcat-palvelin pysäytetään ja käynnistetään uudelleen suorittamalla komennot:

```
/tomcat/bin/shutdown
```

ja

```
/tomcat/bin/startup
```

3. Sovelluksen käyttöönotto

   Sovellus toimii `/kivireki`-hakemistossa palvelimen url-osoitteessa, esim. mikäli tomcat kuuntelee osoitetta db.cs.helsinki.fi, järjestelmän url-osoite on: db.cs.helsinki.fi/kivireki

3.3 Käyttäjätunnusten luonti

Tämä aliluku kuvaa käyttäjätunnusten luomisen käyttäen MySQL-palvelimen mukana tulevaa hallintatyökalua. Järjestelmän käyttöä varten tarvitaan tunnukset kolmelle eri käyttäjäryhmälle. Nämä ovat luennoitsija, kurssihallinto ja kirjasto.

Käyttäjätunnusten luomiseksi täytyy kirjautua MySQL-palvelimelle, tämä tapahtuu komennolla:

```
mysql -u <käyttäjä> -h <host> -p
```

Komennossa `<käyttäjä>`-merkkijono korvataan käyttäjänimellä, jolla palvelimelle kirjaudutaan. `<host>` taas korvataan palvelimen osoitteella eli localhost mikäli palvelimelle ei olla kirjautumassa etänä.

Mikäli kirjautuminen onnistui, komentorivillä pitäisi komennon suorittamisen jälkeen näkyä kehoite:

```
mysql>
```

Tämän jälkeen komentoriville syötetään seuraavat kolme komentoa peräkkäin. Komentojen salasana merkkijono tulee korvata halutulla salasanalla.

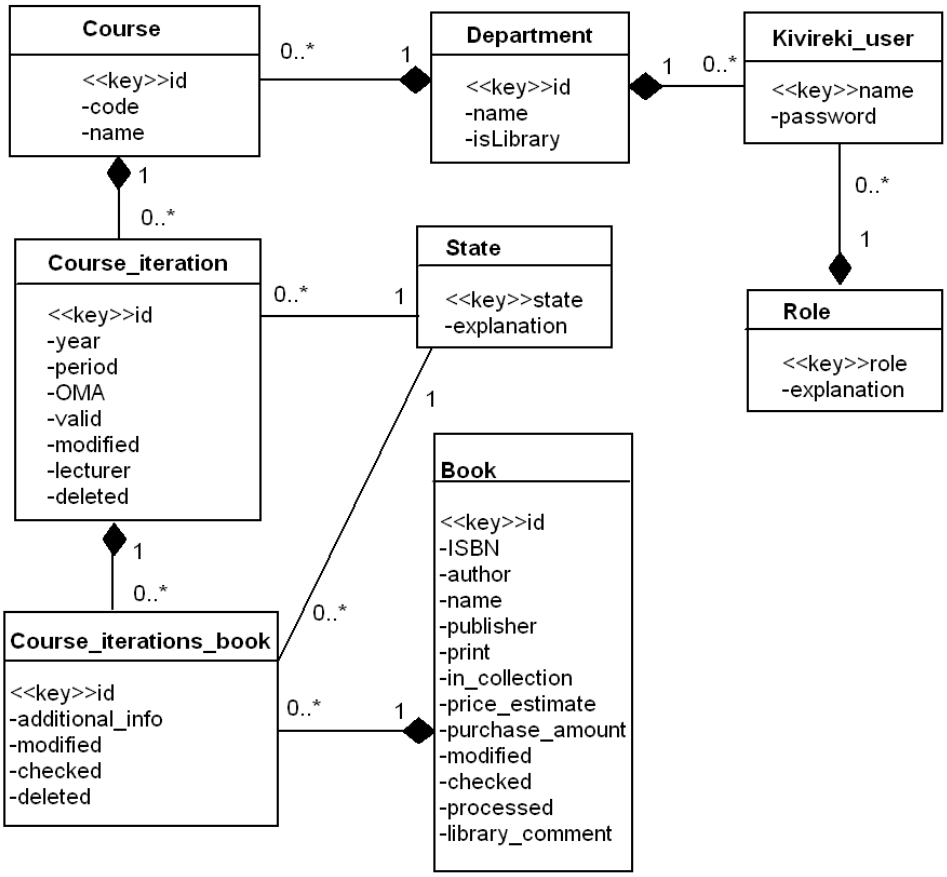
```
insert into kivireki_user (name, password, role, department_id)
values("luennoitsija", SHA1('salasana'), 2, 1);
insert into kivireki_user (name, password, role, department_id)
values("kurssihallinto", SHA1('salasana'), 4, 1);
insert into kivireki_user (name, password, role, department_id)
values("kirjasto", SHA1('salasana'), 1, 2);
```

Salasanan vaihtaminen tapahtuu jollakin seuraavista komennoista. Komento valitaan sen mukaan, minkä käyttäjäryhmän salasanaa halutaan muuttaa.

```
update kivireki_user set password='uusisalasana' where name='luennoitsija';
update kivireki_user set password='uusisalasana' where name='kurssihallinto';
update kivireki_user set password='uusisalasana' where name='kirjasto';
```

4 Muuttuneet vaatimukset

Vaatimusdokumentin osassa 6.3.1 (Toimintaympäristö) on muuttunut XHTML:n validointi Strict:stä Transitional:iin. Palvelinkoneen MySQL-versio ei ole 14.7, vaan 4.1.11 (vaatimusdokumentissa virhe).



Kuva 1: Tietokannan rakenne

5 Muuttuneet suunnitelman osat

5.1 Tietokanta

Tietokanta on muuttunut suunnitteludokumentissa esitetystä versiosta jonkin verran toteutusvaiheessa ilmenneiden muutospaineiden vaikutuksesta. Tässä luvussa selvitetään taulujen tietosisällöt aivan kuten suunnitteludokumentissa, taulukuvaukset on siis vain päivitetty ajantasalle. Jos tauluun on lisätty uusia sarakkeita sitten suunnitteluvaiheen, niistä mainitaan erikseen (poikkeuksena id-kentät). Mikäli taulusta on poistettu sarakkeita, niitä ei erikseen mainita. Suurin koko tietokannan käsittävä muutos koskee taulujen monisarakkeisten avaimien muuttamista UNIQUE-tyyppisiksi, ja vastaavien pääavaimien korvaamista yksikäsitteisillä id-kentillä. Tämä muutos on kokonaisvaltaisesti yksinkertaistanut kaikkien tietokantaoperaatioden toteuttamista, sekä selkeyttänyt kannan rakennetta. Lisähuomiona kurssikoodi-sarakkeen tietotyyppi on muutettu INT:stä VARCHAR:ksi ilmeisistä syistä. Uudistetun tietokannan sisältö ja rakenne on esitelty kuvassa 1.

5.1.1 Käyttäjä

`Kivireki_user`-taulu sisältää käyttäjien tiedot. `Role` tarkoittaa käyttäjäryhmää, ja `department_id` on laitos, jossa käyttäjä työskentelee (kirjasto on myös laitos).

```
CREATE TABLE kivireki_user(
       name                    VARCHAR(100) NOT NULL,
       password                VARCHAR(100) NOT NULL,
       role                    INT UNSIGNED NOT NULL,
       department_id           INT UNSIGNED NOT NULL,

       PRIMARY KEY             kivireki_user_key (name),
       FOREIGN KEY             (department_id) REFERENCES department (id),
       FOREIGN KEY             (role) REFERENCES role (role)
) ENGINE=InnoDB;
```

5.1.2 Kirja

`Book`-tauluun talletetaan kurssikirjojen tiedot. Kirjat yksilöidään automaattisella id-numerolla. Pakollisia tietoja kirjoista ovat kirjoittaja, kirjan nimi, kustantaja sekä painosnumero (`print`). Lisäksi kirjasta voidaan tallettaa tieto onko kirjaa kirjaston kokoelmassa (`in_collection`), hinta-arvio sekä tilattava määrä (`purchase_amount`). Kirjoittaja syötetään muodossa “sukunimi, etunimi” (ilman lainausmerkkejä). Jos kirjalla on useampia kirjoittajia, erotetaan ne toisistaan puolipisteellä. ISBN-numeron voi syöttää VARCHAR-tietotyypin rajoissa.

Lisätyt sarakkeet:

modified – Aikaleima, josta selviää, koska kirjan tietoja on viimeksi muutettu.

checked – Kirjan tilasta implisiittisesti kertova aikaleima, jonka perusteella koodissa määritetään, onko kirjaa koskien tapahtunut muutoksia (juuri lisätty, peruttu jne..). HUOM, Tämä sarake tulisi turhana poistaa, checked-tila on kurssi-iteraation mapatun kurssikerran kirjan (ei kirjan yleisesti) ominaisuus.

processed – Sisältää tiedon siitä, onko kirja viimeksi merkitty käsitellyksi vai ei-käsitellyksi.

library comment – Kommenttikenttä kirjalle.

```
CREATE TABLE book(
       id                          INT UNSIGNED NOT NULL AUTO_INCREMENT,
       ISBN                        VARCHAR(100) NOT NULL,
       author                      VARCHAR(100) NOT NULL,
       name                        VARCHAR(100) NOT NULL,
       publisher                   VARCHAR(100) NOT NULL,
       print                       INT NOT NULL,
       in_collection               INT,
       price_estimate              INT,
       purchase_amount             INT,
       modified                    TIMESTAMP NOT NULL,
       checked                     TIMESTAMP,
       processed                   BOOLEAN NOT NULL DEFAULT FALSE,
       library_comment             VARCHAR(255) DEFAULT '',

       PRIMARY KEY       book_key (id),
       CONSTRAINT        book_unique UNIQUE (author, name, publisher, print)
) ENGINE=InnoDB;
```

5.1.3 Kurssi

`Course`-taulussa eri kurssit yksilöidään kurssikoodin (`code`) ja laitoksen (`department_id`) avulla, sillä eri laitoksilla saattaa olla kursseja samalla koodilla. Kurssiin voidaan liittää myös nimi.

```
CREATE TABLE course(
       id                      INT UNSIGNED NOT NULL AUTO_INCREMENT,
       code                    VARCHAR(100) NOT NULL,
       name                    VARCHAR(100) NOT NULL,
       department_id           INT UNSIGNED NOT NULL,

       PRIMARY KEY             course_key (id),
       CONSTRAINT              course_unique UNIQUE (code, department_id),
       FOREIGN KEY             (department_id) REFERENCES department (id)
) ENGINE=InnoDB;
```

5.1.4 Kurssikerta

`Course_iteration`-taulu kuvaa yksittäistä kurssikertaa. Kurssikerta yksilöidään koodin, laitoksen, vuoden ja periodin avulla. Lisäksi kurssikerran tietoihin kuuluu osallistujamääräarvio (`OMA`), tila, luennoija sekä muokkauspäivä. Mikäli luennoijia on useita, syötetään kurssin vastuuhenkilön nimi. Luennoijan nimi syötetään muodossa “sukunimi, etunimi” (ilman lainausmerkkejä).

Lisätyt sarakkeet:

valid – Kenttä ilmoittaa, onko luennoitsija merkinnyt kyseisen kurssikerran kurssikirjat valmiiksi.

deleted – Kenttä ilmoittaa, onko kurssi-iteraatio tälle lukuvuodelle poistettu. Tällä mahdollistetaan ehdollinen listaus poistettujen ja ei-poistettujen kurssien välillä. Näin ollen mikäli kurssi-iteraatio päätetään poistamisen jälkeen lisätä uudestaan, voidaan suoraan päivittää jo poistetun kurssi-iteraation deleted-saraketta.

```
CREATE TABLE course_iteration(
       id                    INT UNSIGNED NOT NULL AUTO_INCREMENT,
       course_id             INT UNSIGNED NOT NULL,
       year                  INT UNSIGNED NOT NULL,
       period                INT UNSIGNED NOT NULL DEFAULT 0,
       OMA                   INT,
       state                 INT UNSIGNED,
       valid                 BOOLEAN NOT NULL DEFAULT FALSE,
       modified              TIMESTAMP NOT NULL,
       lecturer              VARCHAR(100),
       deleted               BOOLEAN NOT NULL DEFAULT FALSE,

       PRIMARY KEY          course_iteration_key (id),
       CONSTRAINT           course_iteration_unique UNIQUE (course_id, year, period),
       FOREIGN KEY         (course_id) REFERENCES course (id),
       FOREIGN KEY         (state) REFERENCES state (state)
) ENGINE=InnoDB;
```

5.1.5 Kurssikerran kirja

Taulu liittää yhden kirjan tiettyyn kurssikertaan. Yhdellä kurssikerralla voi olla useampia kirjoja. Kurssikerran kirja yksilöidään kuten kurssikerta, lisäyksenä kirjan id (`book_id`). Lisäksi on kenttä luennoijan antamille lisätiedoille (`additional_info`), kirjaston kommenttikenttä, kirjailmoituksen tila sekä muokkauspäivä.

Lisätyt sarakkeet:

course iteration id – Kentällä mapataan kurssikerran kirja oikeaan kurssi-iteraatioon.

checked – Kirjan tilasta implisiittisesti kertova aikaleima, jonka perusteella koodissa määritetään, onko kirjaa koskien tapahtunut muutoksia (juuri lisätty, peruttu jne..).

deleted – Kenttä ilmoittaa, onko kurssikerran kirja tietylle kurssi-iteraatiolle poistettu. Tällä mahdollistetaan ehdollinen listaus poistettujen ja ei-poistettujen kirjojen välillä. Näin ollen mikäli kurssikerran kirja päätetään poistamisen jälkeen lisätä uudestaan, voidaan suoraan päivittää jo poistetun kirjan deleted-saraketta.

```
CREATE TABLE course_iterations_book(
       id                          INT UNSIGNED NOT NULL AUTO_INCREMENT,
       course_iteration_id         INT UNSIGNED NOT NULL,
       book_id                     INT UNSIGNED NOT NULL,
       additional_info             VARCHAR(255) DEFAULT '',
       modified                    TIMESTAMP NOT NULL,
       checked                     TIMESTAMP,
       deleted                     BOOLEAN NOT NULL DEFAULT FALSE,

       PRIMARY KEY                 course_iterations_book_key (id),
       CONSTRAINT                  course_iterations_book_unique UNIQUE (course_iteration_id, book_id),
       FOREIGN KEY                 (book_id) REFERENCES book (id),
       FOREIGN KEY                 (course_iteration_id) REFERENCES course_iteration (id)
) ENGINE=InnoDB;
```

5.1.6 Laitos

`Department`-tauluun tallennetaan eri laitosten nimet. Laitokset yksilöidään automaattisella id-numerolla (järjestelmän mahdollinen laajennettavuus on otettu huomioon).

Lisätyt sarakkeet:

isLibrary – Kentällä erotetaan kirjasto(t) laitoksista. Jos listattu laitos on kirjasto, saa arvon TRUE.

```
CREATE TABLE department (
       id              INT UNSIGNED NOT NULL AUTO_INCREMENT,
       name            VARCHAR(100) NOT NULL,
       isLibrary       BOOLEAN NOT NULL DEFAULT FALSE,

       PRIMARY KEY       department_key (id)
) ENGINE=InnoDB;
```

5.1.7 Tila

`State`-Suunnitteluvaiheen alkuhämäristä mukaan unohtunut kummajainen. Suositeltavana ylläpitotoimenpiteenä tämän taulun voisi poistaa, kunhan siihen eri puolilla olevat viitteet korjataan myös asianmukaisesti.

```
CREATE TABLE state (
    state           INT UNSIGNED NOT NULL AUTO_INCREMENT,
    explanation     VARCHAR(100) NOT NULL,

    PRIMARY KEY state_key (state)
) ENGINE=InnoDB;
```

5.1.8 Rooli

`Role`-taulu toimii selittävänä tauluna. Siitä löytyy ohjelmoijan avuksi selitteet eri roolien numeroille sanallisessa muodossa. Käytetään käyttäjäryhmien identifioimiseen.

```
CREATE TABLE role(
    role            INT UNSIGNED NOT NULL,
    explanation     VARCHAR(100) NOT NULL,

    PRIMARY KEY role_key (role)
) ENGINE=InnoDB;
```

5.2 Luokat ja muut tiedostot

Järjestelmän luokkakaavio on kuvassa 2.

5.2.1 Action-luokat

Action-luokat sijaitsevat järjestelmän `src/kivireki/action`-hakemistossa. Järjestelmä sisältää seuraavat action-luokat:

BaseAction.java – Pohjaluokka, jonka muut action-luokat perivät. Sisältää näille yhteistä toiminnallisuutta. *(uusi)*

CourseManagementAction.java – Kurssihallinnon käyttöliittymän action-luokka.

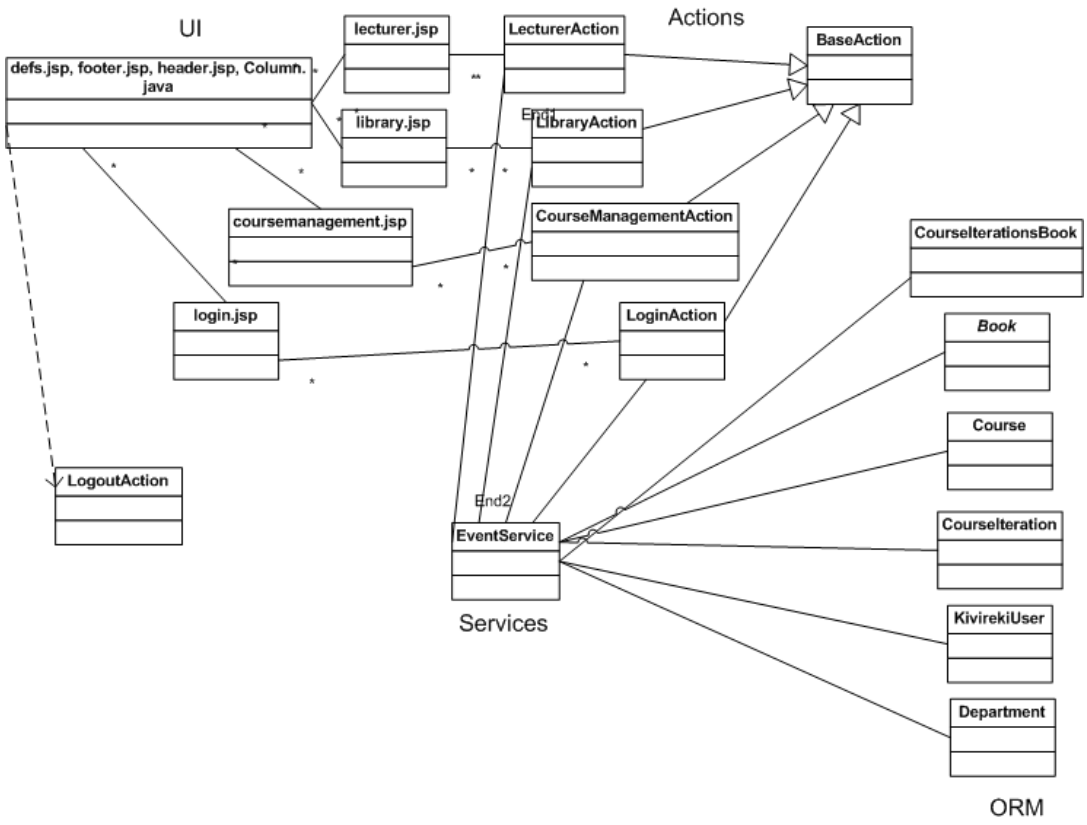
LecturerAction.java – Luennoitsijoiden käyttöliittymän action-luokka.

LibraryAction.java – Kirjaston käyttöliittymän action-luokka.

LoginAction.java – Sisäänkirjautumisen action-luokka.

LogoutAction.java – Uloskirjautumisen action-luokka. *(uusi)*

Column.java – Apuluokka, jota käytetään JSP-sivuilla järjestettävien tauluikoiden sarakkeiden otsikoiden toteuttamiseen. *(uusi)*



Kuva 2: Luokkakaavio

5.2.2 Service-luokat

Service-luokat sijaitsevat järjestelmän `src/kivireki/service`-hakemistossa. Järjestelmä sisältää seuraavat service-luokat:

EventService.java – Palvelu-luokka-rajapinta, joka suorittaa tiedon haun kannasta, tallentamisen kantaa ja jäsentämisen.

EventServiceImpl.java – EventServicen toteutus.

EventServiceUtil.java – Luokka, jolta voi pyytää EventServicen ilmentymää. *(uusi)*

HibernateUtil.java – Luokka, jolta voi pyytää SessionFactoryn ilmentymää. *(uusi)*

SessionService.java – Ei käytetä mihinkään.

SessionServiceImpl.java – Ei käytetä mihinkään.

PropertyReader.java – Asetustiedostojen lukuun tarkoitettu luokka. *(uusi)*

SimultaneousModificationException – Poikkeus, joka heitetään samanaikaisen muokkauksen tapahtuessa. *(uusi)*

5.2.3 Model-luokat

Model-luokat sijaitsevat järjestelmän `src/kivireki/model`-hakemistossa. Järjestelmä sisältää seuraavat model-luokat:

Book.java – Kannan book-taulua vastaava luokka.

Course.java – Kannan course-taulua vastaava luokka.

CourseIteration.java – Kannan course_iteration-taulua vastaava luokka.

CourseIterationsBook.java – Kannan course_iterations_book-taulua vastaava luokka.

Department.java – Kannan department-taulua vastaava luokka.

KivirekiUser.java – Kannan kivireki_user-taulua vastaava luokka.

Role.java – Kannan role-taulua vastaava luokka.

State.java – Kannan state-taulua vastaava luokka.

5.2.4 JSP-sivut

JSP-sivut sijaitsevat järjestelmän `WebContent/kivireki`-hakemistossa. Järjestelmä sisältää seuraavat JSP-sivut:

login.jsp – Sisäänkirjautumissivu, jossa käyttäjä syöttää käyttäjätunnuksen ja salasanan kirjautuakseen sisään.

coursemanagement.jsp – Kurssihallinnon näkymä, jossa näytetään ja muokataan kursseja ja kurssi-iteraatioita ja lähetetään ilmoitus kirjastolle muuttuneesta tilanteesta.

lecturer.jsp – Luennoitsijoiden näkymä, jossa näytetään ja muokataan kurssikirjatietoja sekä lisätään, muokataan ja poistetaan kirjoja järjestelmästä.

library.jsp – Kirjaston näkymä, jossa näytetään kirjatilaustiedot ja käyttäjät voivat lisätä tietoja kirjojen hinnasta ja tilasta sekä muokata tilausten käsittelystatusta.

5.2.5 JSP-apusivut

Apusivut eivät ole itsenäisiä sivuja vaan niitä käytetään tavallisten sivujen rakentamiseen. JSP-apusivut sijaitsevat järjestelmän `WebContent/kivireki/include`-hakemistossa. Järjestelmä sisältää seuraavat apusivut:

defs.jsp – Sisältää sivuille yhteisiä määrittelyjä. *(uusi)*

header.jsp – Sisältää sivuille yhteisen alkuosan. *(uusi)*

footer.jsp – Sisältää sivuille yhteisen loppuosan. *(uusi)*

sortableheader.jsp – Sisältää koodin, joka generoi HTML-taulukolle otsikkorivin, jonka avulla taulukon sisällön voi järjestää eri sarakkeiden mukaan. *(uusi)*

5.2.6 CSS-tyylitiedostot

Tyylitiedostot sijaitsevat järjestelmän `WebContent/kivireki/styles`-hakemistossa. Järjestelmä sisältää seuraavat tyylitidostot:

main.css – Pääasiallinen tyylitiedosto.

print.css – Sivuja tulostettaessa käytettävä lisätyylitiedosto. *(uusi)*

5.2.7 JS-skriptitiedostot

Skriptitiedostot sijaitsevat järjestelmän `WebContent/kivireki/scripts`-hakemistossa. Järjestelmä sisältää seuraavat JavaScriptillä kirjoitetut skriptitiedostot:

formfuncs.js – Lomakkeiden käsittelyyn liittyviä skriptejä. *(uusi)*

5.2.8 Kuvatiedostot

Kuvatiedostot sijaitsevat järjestelmän `WebContent/kivireki/images`-hakemistossa. Järjestelmä sisältää seuraavat kuvat:

icon_asc.png – Ikoni taulukon sarakkeen nousevalle järjestykselle. *(uusi)*

icon_desc.png – Ikoni taulukon sarakkeen laskevalle järjestykselle. *(uusi)*

icon_delete.png – Ikoni elementin (tällä hetkellä vain kurssikirjan) poistamiselle. *(uusi)*

5.3 Käyttöliittymä

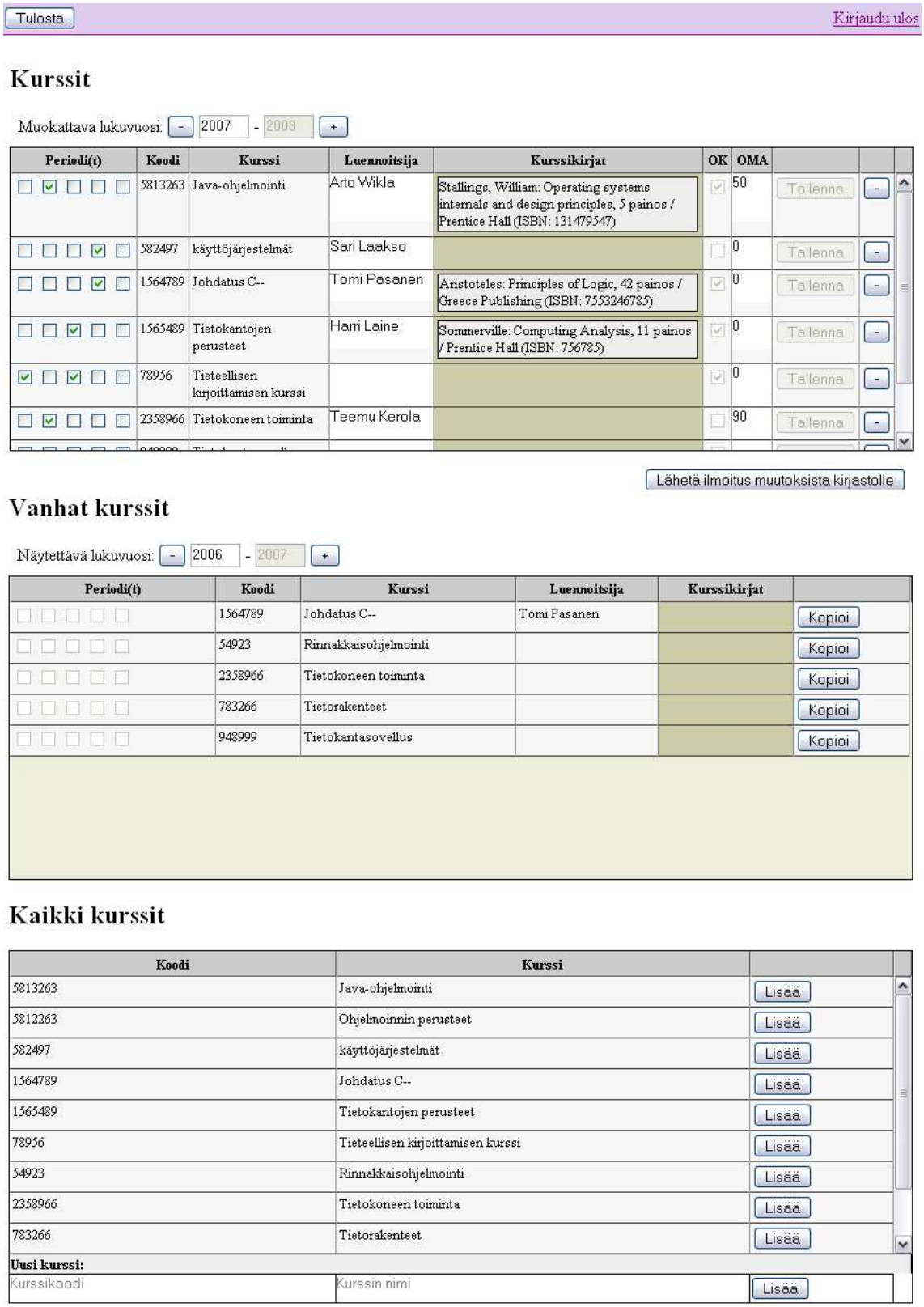
Käyttöliittymät ovat hieman muuttuneet suunnitteluvaiheesta. Toteutusteknisistä syistä jotain on lisätty ja jotain jätetty pois. Tarkemmat kuvaukset käyttöliittymäkuvien yhteydessä.

Kurssihallinnon sivu näkyy kuvassa 3. Kurssihallinnon käyttöliittymään on lisätty tallennusnappi, jotta kaikki tallennettavat tiedot voidaan lähettää samalla kertaa ja näin vältytään ylimääräisiltä sivun latauksilta. Lisäksi kurssihallinnolta poistettiin mahdollisuus poistaa kurssikirjoja sekä niiden asettaminen käsitellyiksi. Suodattavaa hakua ei ole taulukoihin toteutettu ajanpuutteen vuoksi.

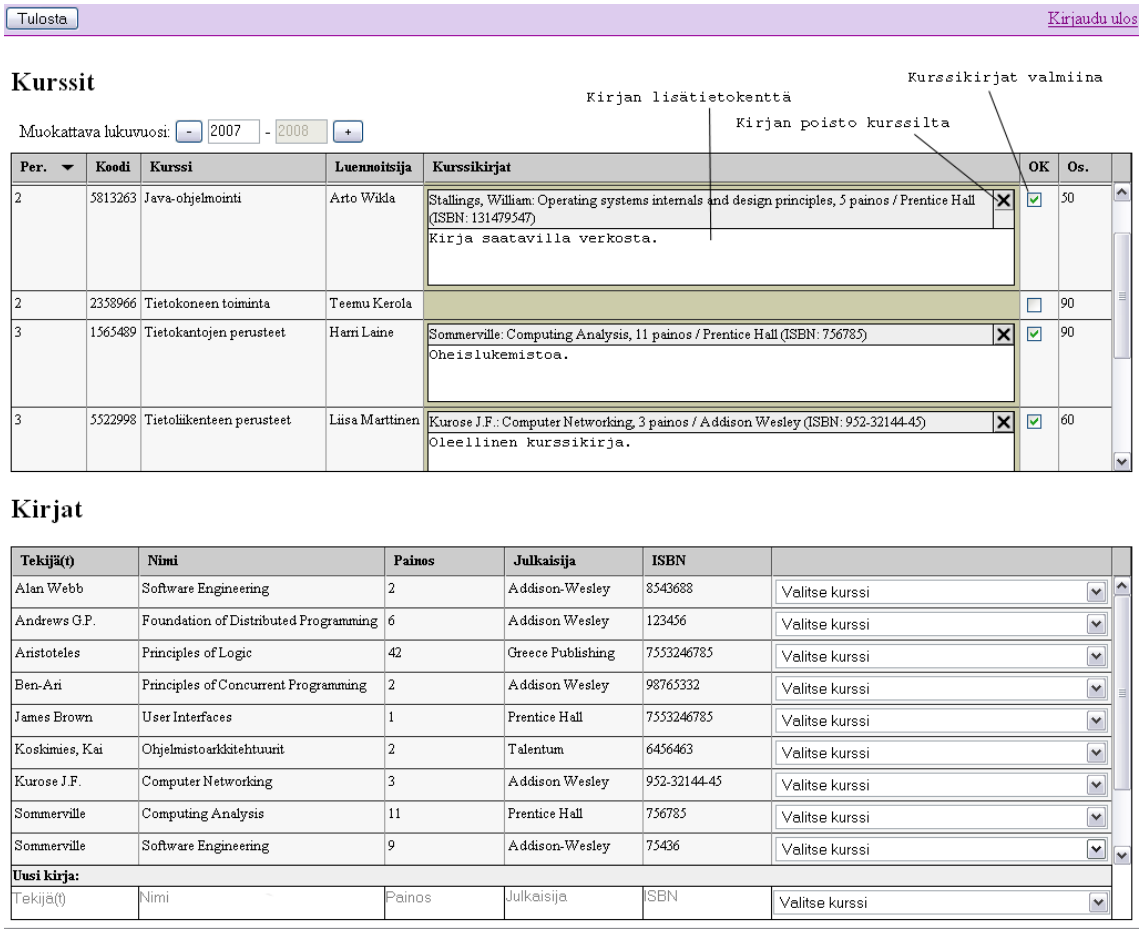
Vanhojen kurssikertojen listaukseen lisättiin samanlainen vuosivalitsin kuin uusien kurssikertojen listaukseen. Näin saatiin vuosisarake poistettua taulukosta ja kerralla näytettävien kurssien määrä pienemmäksi. Lisäksi alimmaiseksi lisättiin taulukko kaikista kursseista siksi, että on mahdollista, että jostakin kurssista ei ole olemassa ainuttakaan kurssiiteraatiota ja kurssin lisääminen uudestaan ei ole enää mahdollista (duplikaattien takia).

Luennoitsijoiden sivu näkyy kuvassa 4. Luennoitsijoiden käyttölittymää muutettiin siten, että periodit näkyvät luennoitsijoille vain numeroina (esim. 1,2), koska luennoitsija ei kuitenkaan voi periodeja muuttaa. Samoilla perusteilla on poistettu kurssikerran poistonappi ja osallistujamäärä- arvio ja luennoitsijan nimi eivät ole enää tekstikenttiä. Tähänkään käyttöliittymään ei suodatustoimintoja toteutettu.

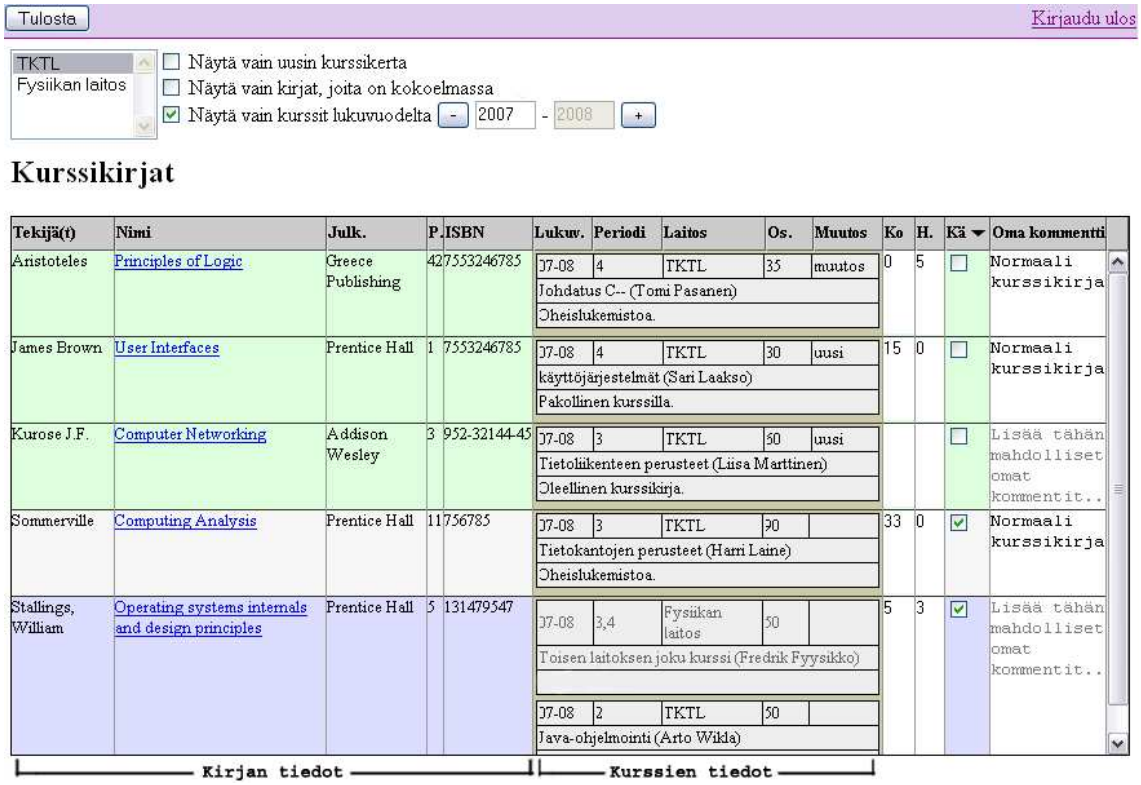
Kirjaston sivu näkyy kuvassa 5. Myös kirjaston käyttöliittymään on lisätty samanlainen vuosivalitsin kuin muissakin käyttöliittymissä on. Valitsimen saa kuitenkin tarvittaessa pois päältä poistamalla ruksin valitsinta edeltävästä ruudusta. Kirjataulukosta poistettiin sarakkeet hinnalle ja yhteissummalle, koska niitä ei ehditty toteuttaa. Tämän lisäksi näkymään ei toteutettu toimintoa, jolla järjestelmä näyttäisi kurssit, joille ei ole vielä liitetty kirjoja.



Kuva 3: kurssihallinnon käyttöliittymä



Kuva 4: luennoitsijoiden käyttöliittymä



Kuva 5: kirjaston käyttöliittymä

6 Toteutumattomat vaatimukset ja suunnitelman osat

Prio	Nro	Nimi	Toteutuminen
1	JV1	Sisään- ja uloskirjautuminen	**Toteutui.**
1	JV2	Luennoitavien kurssien listaus	**Toteutui.**
1	JV3	Uuden kurssin / kurssikerran lisääminen	**Toteutui.**
1	JV4	Kurssikerran poistaminen	**Toteutui.**
1	JV5	Kurssikirjojen listaus kirjastolle	**Toteutui.**
1	JV6	Kirjojen listaus luennoitsijoille	**Toteutui.**
1	JV7	Vanhojen kurssikertojen listaus	**Toteutui.**
1	JV8	Kurssikerran tietojen kopiointi	**Toteutui.**
1	JV9	Arvioidun osallistujamäärän syöttäminen	**Toteutui.**
1	JV10	Kirjatietojen merkitseminen päivitetyiksi	**Toteutui.**
1	JV11	Kurssikerran tietojen päivitys	**Toteutui.**
1	JV12	Järjestelmässä olevan kirjan lisäys kurssille	**Toteutui.**
1	JV13	Uuden kirjan lisäys kurssille	**Toteutui.**
1	JV14	Kirjan poistaminen kurssilta	**Toteutui.**
1	JV15	Kurssikirjan lisätiedot	**Toteutui.**
1	JV16	Tiedotus muutoksista	**Toteutui.**
1	JV17	Kurssikerran poistaminen	**Toteutui.**
2	JV18	Kokoelmassa olevien kirjojen määrä	**Toteutui.**
2	JV19	Tilauksen tilatiedot	**Toteutui.**
2	JV20	Tieto vanhoista kursseista	**Toteutui.**
2	JV21	Kurssien ehdollinen listaus	**Ei toteutunut.**
2	JV22	Laitokset	**Toteutui.**
2	JV23	Lokitiedot	**Ei toteutunut.** (Joitain asioita kyllläkin logitetaan.)
3	JV24	Kurssin tietojen muokkaus	**Ei toteutunut.**
3	JV25	Automaattinen tiedotus muutoksista	**Ei toteutunut.**
3	JV26	LDAP-autentikointi	**Ei toteutunut.**
-	JV27	Kurssitietojen syöttäminen	**Toteutui.**
-	JV28	Käytettävyys	**Toteutui.**
-	JV28	Kapasiteetti	Ei testattu tarpeeksi.
-	JV28	Luotettavuus	**Toteutui.**
-	JV28	Tehokkuus	**Toteutui.**
-	JV28	Skaalautuvuus	**Toteutui.**

7 Koodin ylläpito

Tässä luvussa käsitellään koodin jatkokehityksen kannalta ollennaisia seikkoja, erityisesti virheitä tai puuttuvaa toiminnallisuutta, jota ei olla ehditty toteuttaa. Yleisesti ottaen jatkokehityksen ensimmäinen tehtävä olisi suorittaa järjestelmälle dokumentoitua testausta vieläkin tarkempi testaus. Jako läpikäytävien ongelmakohtien välillä on tehty käyttäjänäkymittäin (lukuunottamatta kaikille yhteistä Login ja sessiot-osuutta). On huomioitava, että kaikki listatut osakokonaisuudet hyödyntävät luokkia BaseAction (jonka kaikki action-luokat perivät) ja EventServiceImpl (tietokantahaut implementoiva luokka, toteuttaa EventService.java-rajapintaluokan määrittämät palvelut).

7.1 Suorituksen eteneminen

Aluksi kerrotaan yleiskuva suorituksen etenemisestä tavallisimmissa skenaarioissa.

7.1.1 Normaalin sivun lataaminen

Normaalilla sivun lataamisella tarkoitetaan tilannetta, jossa käyttäjän selain suorittaa sivupyynnön lähettämättä mitään ylimääräistä informaatiota palvelimelle päin. Tällainen tapahtuu siis kun käyttäjä esimerkiksi kirjauduttuaan järjestelmään ohjataan luennoitsijoiden, kirjaston tai kurssihallinnon omalle sivulle.

1. Käyttäjän selain lähettää sivupyynnön.
2. Palvelin vastaanottaa sivupyynnön. Jos pyyntö koski suoraan esim. jsp-tiedostoa, siirtyy kohtaan 5. Jos pyyntö oli action-pyyntö (muotoa .../jotain.action), selvittää `src/kivireki.xml`-tiedostosta pyyntöä vastaavan action-luokan ja suoritettavan metodin (oletuksena `execute`) ja luo luokasta olion.
3. Palvelin suorittaa actioniin assosioidun action-olion metodin. Normaalisti metodin rakenne on suunnilleen seuraava:
   (a) Alustaa istunnon.
   (b) Tarkistaa, että käyttäjällä on tarvittavat oikeudet.
   (c) Hakee istunnosta tarvittavat tiedot ja/tai päivittää ne sinne.
   (d) Hakee näytettävän tiedoston sisällön kannasta EventServicen avulla.
   (e) Palauttaa paluuarvon, jonka perusteella päätetään, mikä jsp-sivu näytetään tai minne actioniin uudelleenohjataan tms.
4. Palvelin selvittää paluuarvon perusteella, mikä jsp-sivu näytetään. (Tai vaihtoehtoisesti, minne actioniin tms ohjataan, ja ohjaa sinne.)
5. Palvelin suorittaa jsp-sivun, joka generoi selaimelle lähetettävän HTML-koodin. Siinä tehdään yleensä suunnilleen seuraavaa:

   (a) Sisällytetään `header.jsp`, jossa tulostetaan kaikille sivuille yhteinen alkuosa.
   (b) Tulostetaan mahdolliset virheilmoitukset.
   (c) Haetaan tarvittava data action-luokan muuttujista ja generoidaan niiden avulla sivun sisältö.
   (d) Tulostetaan lomakkeiden tarkastuksessa tarvittavat JavaScript-funktiot ja -komennot.
   (e) Sisällytetään `footer.jsp`, jossa tulostetaan kaikille sivuille yhteinen loppuosa.
6. Palvelin lähettää generoidun HTML-sivun käyttäjän selaimelle.
7. Selain vastaanottaa HTML-sivun ja siinä viitattava kuvat ja skriptitiedostot (jne.), ja generoi HTML-koodin perusteella näytettävän sivun.
8. Selain suorittaa sivun lopussa määritellyt JavaScript-osiot:
   (a) Määrittelee validointifunktiot halutuille lomakkeiden kentille.
   (b) Asettaa elementit, joiden vierityspalkkien asento halutaan säilyttää, kun sivu ladataan uudestaan lomaketta lähetettäessä.
   (c) Asettaa lomakkeiden kentät, joiden sisältö halutaan validoida tai joissa halutaan näyttää lisätietoteksti/kehote. Tässä yhteydessä mm. asetetaan määritellyt kehotetekstit mahdollisiin tyhjiin kentiin.
   (d) Sovittaa tekstikentät sopiviksi taulukoiden soluihin (niin, että ne venyttyvät koko solun kokoisiksi).
   (e) Kaventaa taulukon sisältöä niin, ettei sivuttaisvierityspalkkia pitäisi tulla.
9. Selain näyttää sivun käyttäjälle.

7.1.2 Lomakkeen lähettäminen

Kun käyttäjä muuttaa tietoja sivuilla lähettää selain muuttuneet tiedot palvelimelle ja lataa sivun uudestaan. Tämä tapahtuu myös kun esimerkiksi taulukkko järjestetään uusiksi jonkin sarakkeen mukaan. Alkutilanne on siis se, että sivulla on lomake, joka sisältää nimettyjä tietokenttiä (joissa muuttuneita tai muita tietoja), ja käyttäjä painaa nappia tai muuten laukaisee lomakkeen lähettämisen.

1. Suoritetaan lähettämiseen liittyvä JavaScript (tryToSubmit/beforeSubmit)
   (a) Tarkastaa kenttien sisällön oikeellisuuden. Jos eivät valideja, näyttää virheet ja keskeyttää.
   (b) Selvittää määriteltyjen vierityspalkkien asennot ja tallentaa ne piilokenttinä kyseiseen lomakkeeseen.
   (c) Lähettää lomakkeen.

2. Käyttäjän selain lähettää sivupyynnön ja siinä yhteydessä lomakkeen tiedot.
3. Palvelin vastaanottaa sivupyynnön. Jos pyyntö koski suoraan esim. jsp-tiedostoa, siirtyy kohtaan 5. Jos pyyntö oli action-pyyntö (muotoa .../jotain.action), selvittää `src/kivireki.xml`-tiedostosta pyyntöä vastaavan action-luokan ja suoritettavan metodin (oletuksena `execute`) ja luo luokasta olion.
4. Palvelin kopioi lähetetyn lomakkeen kenttien sisällöt samannimisiin muuttujiin action-olioon.
5. Palvelin suorittaa actioniin assosioidun action-luokan metodin. Normaalisti metodi suoritettaa seuraavat asiat:
   (a) Alustaa istunnon.
   (b) Tarkistaa, että käyttäjällä on tarvittavat oikeudet.
   (c) Päivittää mahdolliset muuttuneet tiedot tietokantaan EventServicen avulla.
   (d) Hakee istunnosta tarvittavat tiedot ja/tai päivittää ne sinne.
   (e) Hakee näytettävän tiedoston sisällön kannasta EventServicen avulla.
   (f) Palauttaa paluuarvon, jonka perusteella päätetään, mikä jsp-sivu näytetään ja näytetäänkö virheilmoitus tai minne actioniin uudelleenohjataan tms.
6. Loput kuten edellä kohdasta 4 eteenpäin.

Seuraavat ovat selaimessa suoritettavaa JavaScript-toiminnallisuutta:

7.1.3 Lomakkeen kenttään saapuminen

Kun käyttäjä kohdistaa kursorin sellaiseen lomakkeen kenttään, joka on alustettu `initField`-funktiolla, suoritetaan `focusField`, joka tekee seuraavaa:

1. Poistaa kentässä mahdollisesti olevan kehotetekstin.
2. Poistaa kentässä mahdollisesti olevan virheviestin.

7.1.4 Lomakkeen kentästä poistuminen

Kun käyttäjä poistuu (tai painaa enteriä, jos kyseessä on yksirivinen tekstikenttä) sellaisesta lomakkeen kenttästä, joka on alustettu `initField`-funktiolla, suoritetaan `blurField`, joka tekee seuraavaa:

1. Validoi kentän sisällön tarvittaessa ja asettaa mahdollisen virheilmoituksen.
2. Yrittää lähettää lomakkeen, jos initField-alustuksessa on määritelty näin.
3. Asettaa tarvittaessa kehotetekstin.

7.2 Kurssihallinto

Kurssihallinon toiminnallisuuden toteuttavat yhdessä `CourseManagementAction.java` action-luokka, sekä `coursemanagement.jsp` JSP-sivu. Seuraavia ongelmia/huomioita havaittu:

- Taulukot eivät ole järjestettävissä toisin kuin muissa käyttöliittymissä. Kannattaisi toteuttaa.
- Osaa lomakkeista ei tarkisteta ollenkaan. Kannattaisi ottaa validointi käyttöön myös niissä.
- Lomakkeen lähettävästä Muuta-napista voisi hankkitua eroon ja lähettää tiedot aina kun kenttiä muutetaan, kuten muissakin käyttöliittymissä.
- Vierityspalkkien asennot eivät säily kaikkia lomakkeita lähetettäessä. Tämän voisi korjata (huom. korjautuu edellisen kohdan tekemällä automaattisesti).
- Action-luokan toiminta ei ole täysin yhtenäistä muiden kanssa. Voisi yhtenäistää.
- Olemassaolevien kurssien (ei siis kurssikertojen) tietoja olisi hyvä pystyä muokkaamaan / vahinkokursseja pitäisi ehkä pystyä poistamaan.

7.3 Luennoitsijat

Luennoitsijoiden toiminnallisuuden toteuttavat yhdessä `LecturerAction.java` action-luokka, sekä `lecturer.jsp` JSP-sivu. Seuraavia ongelmia/huomioita havaittu:

- Kirjan tietoja pitäisi pystyä muokkaamaan / virhekirja pystyä poistamaan.
- Uutta painosta varten ei ehkä tarvitsisi syöttää kaikkia kirjan tietoja uusiksi.
- Kirjalle pitäisi pystyä syöttämään useita ISBN-numeroita (esim. eri ISBN kova- ja pehmeäkantisilla versioilla). Lisäksi ISBN-kentän ei *ehkä* olla pakollinen.
- Järjestelmä voisi havaita älykkäämmin onko kirja jo olemassa järjestelmässä (vaikka tiedot olisi syötetty hieman eri tapaan).

7.4 Kirjasto

Kirjaston toiminnallisuuden toteuttavat yhdessä `LibraryAction.java` action-luokka, sekä `library.jsp` JSP-sivu. Seuraavia ongelmia/huomioita havaittu:

- Taulukkossa ei näy kursseja, joita ei ole vielä merkitty valideiksi. Tämän voisi toteuttaa vaatimusdokumentissa esitetyllä tavalla.
- Toteuttamatta jääneet yksikköhinta- ja kokonaishintasarakkeet kannattaisi toteuttaa.

- Kirjojen tietoja olisi hyvä päästä muokkaamaan myös kirjastosta käsin. Vahingossa luotuja duplikaattiversioita pitäisi pystyä myös yhdistämään.
- Kirjatukkuihin voisi lisätä hakulinkit kirjan nimellä samaan tapaan kuin Helkaan.
- Kirjaston päätä voisi muutenkin laajentaa niin, että koko hankintaprosessin vois hoitaa sillä.

7.5 Login

Kirjautumis-toiminnallisuuden toteuttavat yhdessä `LoginAction.java` ja `LogoutAction.java` action-luokat, sekä `login.jsp` JSP-sivu. Seuraavia ongelmia/huomioita havaittu:

- Koodia kannattaisi siistiä ja yhtenäistää muiden kanssa.
- Turhan roolin tallentamisen sessioon voisi poistaa, kunhan on ensin varmistanut, että se ei riko mitään.
- LoginAction ja LogoutAction voisivat hyödyntää myös BaseAction-luokkaa.

7.6 Yleistä

Seuraavassa on listattu yleisiä ongelmia tai muita huomioita:

- Järjestettävät taulukot kannattaisi toteuttaa järkevämmin. Paras tapa lienee kehittää oma struts-tagi tätä varten. Tätä varten on olemassa myös valmiita toteutuksia, joita voi ehkä käyttää hyväksi, ellei pysty pelkästään hyödyntämään.
- Nykyiset lomakkeiden tarkistukset on tehty omalla JavaScript-viritelmällä; Saattaisi olla järkevää yrittää kuitenkin käyttää Strutsin omaa validointisysteemiä. (Tästä luovuttiin joidenkin ongelmien takia, mutta ne ovat varmaankin ratkaistavissa. Jollei muuten niin kehittämällä Strutsin koodia eteenpäin.)
- Lomakkeiden lähetys tietoja muutettaessa kannattaisi ehdottomasti toteuttaa AJAX:lla. Tällöin voitaisiin päivittää aina vain tarvitta osa sivusta. Struts tarjoaa tähänkin jotain omia työkaluja.
- Taulukoiden sarakkeiden leveyksiä olisi mukava pystyä säätämään hiirellä. Tämän voisi toteuttaa JavaScriptillä ja tallentaa tiedot istuntoon.
- Samoin taulukoiden korkeutta voisi olla mukava pystyä säätämäään.
- Ehkäpä sopivasti tiivistämällä kaikkien sivujen sisällöt saisi mahtumaan kerralla ruutuun, jolloin edes pystysuuntaan ei tarvitsisi vierittää. Eri taulukot voisivat näkyä esimerkisi samaan tapaan kuin päällekkäiset kehykset.

- Column-luokan voisi siirtää action-paketista ehkä jonnekin järkevämpään paikkaan. Minne?
- Mahdolliset samanaikaisuuden aiheuttamat ongelmat kannattaisi käydä vielä läpi ajatuksen kanssa. Ainakin kirjaston päästä tietoja muokatessa näitä ei ehkä tarkisteta tarpeeksi hyvin.
- Kanta-kyselyjä voisi yrittää yksinkertaistaa (etenkin EventServiceImpl:n findBooks-metodin järkälekyselyä).
- EventServicessä ja EventServiceImpl:ssä on on suuri määrä käyttämättömiä metodeja, jotka kannattaa poistaa.

8 Tietokanta

Seuraavassa on listattu toimenpiteitä ja huomionarvoisia seikkoja tietokannan ylläpidon ja jatkokehityksen kannalta.

Library-taulu – Kirjastot jakavat nykyisellään department-taulun laitosten kanssa, ero kirjaston ja laitoksen välillä tehdään isLibrary-sarakkeen perusteella. Kirjastot voitaisiin kenties sijoittaa omaan tauluunsa, johon voitaisiin myös sijoittaa kirjastokohtaista tietoa kuten kirjaston sähköpostiosoite. Näin nykyisellään kovakoodattu sähköpostiosoite (opetushallinon sähköposti-ilmoitusta varten ks. JV16) voitaisiin kohdentaa tiedekunta- ja kirjastokohtaisiksi.

State-taulu – Suositeltavana ylläpitotoimenpiteenä tämän taulun voisi poistaa, kunhan siihen eri puolilla olevat viitteet korjataan myös asianmukaisesti.

Department-taulu – Laitokset ja kirjastot voisi assosioida keskenään, jolloin kirjaston kälissä voitaisiin suoraan valita automaattisesti ne laitokset, joiden kurssikirjoja kyseisessä kirjastossa säilytetään.

CourseIteration ja CourseIterationsBook -taulut – monimutkaisista checked- ja deleted-sarakkeista voisi päästä kokonaan eroon kopioimalla/päivittämällä taulujen kaikki tiedot kirjaston omiin vastaaviin tauluihin aina kun kirjastosta käsin luetaan kyseisiä tietoja / asetetaan näihin liittyvä kirja käsitellyksi. Tällä saavutettaisiin lisäksi kaksi muutakin etua:

- Muuttuneet tiedot voitaisiin selvittää vertaamalla näiden taulujen tietoja, ja näyttää siten käyttäjälle.
- Jokaisella kirjastolla voisi olla omat taulunsa. Nykyisellääänhän tiedot ovat yhteisiä ja käsittely-sarakkeen asettaminen vaikuttaa yhtälailla kaikkiin kirjastoihin.

9 Järjestelmän kehitys

Järjestelmä on kehitetty käyttäen Eclipse Europa -sovelluskehitysympäristöä. Järjestelmän `kivireki.war`-tiedostosta saa luotua automaattisesti projektin Eclipseen. Tämä tapahtuu valitsemalla ylävalikosta File->Import->Web->WAR file. Tämän jälkeen valitaan `kivireki.war` ja käytettävä tietokanta.

Mikäli tietokantaa ei ole asennettuna Eclipsessä, se täytyy tehdä valitsemalla ylävalikosta File->New->Other->Server ja asettamalla hakemisto, missä palvelin sijaitsee. Lisäohjeita löytyy Eclipsen Helpistä hakusanalla server.

Järjestelmän saa takaisin WAR-tiedostoksi valitsemalla File->Export->Web->War file.

10 Testaus

10.1 Yksikkötestaus

Yksikkötestauksessa käytettiin JUnit-kirjastoa apuna. Yksikkötestauksessa ei saavutettu tarvittua 80% haaraumakattavuutta ajan loppumisen vuoksi. Lisäksi metodeja ei testattu arvoalueen ääriarvoilla, nollakohdilla ja null-viitteillä. Yksikkötesteissä testattiin kuitenkin kaikki olennainen toiminallisuus.

Testit löytyvät
`test`-hakemistosta. Hakemistossa on `testLuokanNimi`-nimisiä luokkia, jotka testaavat nimessä kuvatun luokan. Kaikki testit voi suorittaa ajamalla kansiosta löytyvän `AllTests`-luokan.

10.2 Järjestelmä- ja integrointitest

Toiminnalliset testitapaukset suoritettiin suurimmaksi osaksi onnistuneesti.

10.3 Löydetyt bugit

Epänumeerista dataa OMA-kentässä – Jos kurssihallinnon sivulla syöttää kurssi-iteraation OMA-kenttään epänumeerista dataa ja painaa tallenna-painiketta, järjestelmä näyttää virheilmoituksen oikein muttei listaa (muita) kurssi-iteraatiota enää.

Kirjan lisääminen mahdotonta jos kursseja ei ole – Kirjaa ei voi lisätä ilman että se lisätään jollekin kurssille, ts. jos kursseja ei ole, ei voi lisätä kirjoja.