Suunnitteludokumentti

Kohahdus

Helsinki 9.10.2006

Ohjelmistotuotantoprojekti

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Kurssi

581260 Ohjelmistotuotantoprojekti (6 ov)

Projektiryhmä

Taro Morimoto, Projektipäällikkö
Tuomas Palmanto, Vaatimusmäärittelyvastaava
Mikko Kinnunen, Suunnitteluvastaava
Markus Kivilä, Koodivastaava
Jari Inkinen, Testausvastaava
Paula Kuosmanen, Dokumenttivastaava

Asiakas

Teemu Kerola

Johtoryhmä

Sanna Keskioja

Kotisivu

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

Versiohistoria

Versio	Päiväys	Tehdyt muutokset
1.0	21.1.2004	Ensimmäinen versio

Sisältö

1 Johdanto

TitoTrainer on järjestelmä automaattisesti tarkastettavien TTK-91-konekielen harjoitustehtävien luomiseen ja ratkaisemiseen. Järjestelmä on tarkoitettu käytettäväksi opetuksen tukena, opetettaessa Tietokoneen toiminta -kurssia. Tietojenkäsittelytieteen opettajat voivat tehdä järjestelmään uusia tehtäviä ja määritellä kuinka ne tarkastetaan automaattisesti. Tietokoneen toiminta -kurssin opiskelijat ja kurssin tehtävistä kiinnostuneet itseopiskelijat voivat ratkaista tehtäviä ja saada palautetta niiden onnistumisesta.

Painopiste projektissa on opettajan ja opiskelijan käyttöliittymillä. Käyttöliittymistä tehdään mahdollisimman selkeät ja helppokäyttöiset. Opettajan käyttöliittymän avulla määritellään tehtävät parametreineen ja kuinka opiskelijan ratkaisun oikeellisuus tarkistetaan, sekä nähdään statistiikkatietoja opiskelijoiden suorituksista. Opiskelijan käyttöliittymän avulla opiskelija valitsee tehtäviä, syöttää niiden ratkaisut ja saa palautetta vastauksensa oikeellisuudesta.

Projekti käyttää valmiina olevaa Titokone-simulaattoria harjoitustehtävien ratkaisemiseen. Projekti rakennetaan valmiina olevaan eAssari-kehykseen, joka sopii geneeriseen automaattisesti tarkistettavien tehtävien määrittelyyn ja toteutukseen.

Järjestelmä toteutetaan Java-kielellä ja järjestelmän käyttö vaatii, että selain tukee JavaScript-kieltä ja CSS-tyylitiedostoja. Järjestelmä toimii uusimmilla Firefox ja Internet Explorer-selaimilla.

2 Sanasto

TTK91=Auvo Häkkisen kehittämä ohjelmointikieli, joka läheisesti muistuttaa symbolista konekieltä.

KOKSI=Auvo Häkkisen kirjoittama konekielisimulaattori, joka toteuttaa TTK-91-kielen.

Kohahdus=Syksyn 2006 ohjelmistotuotantoprojekti (tämä projekti)

TitoTrainer=Projektimme tuotos

Järjestelmä=Projektimme tuotos

Ohjelma=Opiskelijan kirjoittama TTK91-ohjelma, eli vastaus johonkin tehtävään

eAssari=Tietokantapohjainen ympäristö ohjelmallisesti tarkastettavien harjoitus- ja koetehtävien suorittamiseen

Titokone=Koski-nimisen Ohjelmistotuotantoprojektiryhmän vuonna 2004 rakentama järjestelmä konekielisten ohjelmien kääntämiseen ja suorittamiseen.

Koski=Vuoden 2004 Ohjelmistotuotantoprojekti joka rakensi konekielen simulaattorin ja debug-ympäristön, eli Titokoneen

Koskelo=Vuoden 2004 Ohjelmistotuotantoprojekti, joka integroi Titokoneen ja eAssari-kehyksen yhteen. Ratkaisusta ei tullut kuitenkaan käyttökelpoista, eikä sitä ole otettu käyttöön.

Kriteeri=Sääntö jonka mukaan tehtävän oikeellisuus tarkistetaan. Kriteereitä voi olla monta yhdelle tehtävälle.

Aihepiiri=Tehtävälle täytyy määritellä aihepiiri, johon tehtävä kuuluu.

3 Toiminnot

Tämä luku kuvaa järjestelmän HTML-sivut ja sen, mitä toimintoja ne tarjoavät käyttäjälle.

3.1 Rekisteröityminen

Rekiströintisivulla opiskelija rekisteröityy järjestelmän käyttäjäksi. Rekisteröinti vaatii seuraavat tiedot: etunimi, sukunimi, käyttökieli, sähköposti, opiskelijanumero tai henkilötunnus, sekä vapaavalintainen tunnus ja salasana

3.2 Käyttäjätietojen muokkaaminen

Käyttäjätietosivulla sisäänkirjautunut käyttäjä voi muokata rekisteröintitietojaan, poislukien tunnus joka on valittu rekisteröidyttäessä. Tämäm toiminnon ei tarvitse toimia opettajien tunnusten kanssa, mutta odotettavissa on, että opettajatunnusten tukeminen ei aiheuta minkäänlaista lisätyötä.

3.3 Kirjautuminen

Kirjautumissivulla käyttäjä syöttää tunnuksensa, salasanansa ja pudotusvalikosta kurssin nimi. Kurssivalinta määrittää, että minkä kurssin suorituksiin opiskelijan tekemät vastaukset tallennetaan. Opettajan ollessa kyseessä kurssivalinnalla ei ole merkitystä. Jos kirjautuminen onnistuu, näytetään tehtävälistaus. Jos kirjautuminen epäonnistuu, käyttäjä palautetaan kirjautumissivulle.

3.4 Kurssien ja tehtävien valinta (opettaja)

Kirjautumisen jälkeen opettajalle näytetään kurssilistaus josta opettaja voi siirtyä kurssin statistiikka-sivulle. Kurssilistauksesen alla on myös lomake uuden kurssin luomista varten.

Samalla sivulla näytetään myös tehtävälista. Listassa näytetään tehtävän kieli, aihepiiri ja nimi, sekä viimeinen muokkauspäivä ja muokkaaja. Listattuja tehtäviä voi muokata ja poistaa, sekä käynnistää uuden tehtävän luomisen.

3.5 Tehtävän määrittely (opettaja)

Tehtävänmäärittelysivulla opettaja voi luoda uuden tehtävän tai muokata olemassa olevaa tehtävää. Tehtävälle määritellään nimi, tehtävänanto, tarkastustyyppi (malliratkaisu/kiinteät arvot), tehtävätyyppi (täyttötehtävä/ohjelmointitehtävä), sekä tarkastuskriteerit ja kriteerien palautteet.

3.6 Kurssin statistiikka (opettaja)

Kurssin statistiikkasivulla opettajalle näytetään lista kaikkien kurssin tehtäviän suorittaneiden opiskelijoiden suorituksista. Listassa näytetään opiskelijan nimi, opiskelijanumero tai henkilötunnus, sekä onnistuneesti ratkaistujen tehtävien määrä. Lista on helposti tulostettavassa muodossa.

Listassa opiskelijan nimi toimii linkkinä ko. opiskelijan käyttäjätietosivulle.

Kurssin statistiikka toteutetaan toisessa kehitysiteraatiossa.

3.7 Käyttäjän tiedot (opettaja)

Käyttäjän statistiikkasivulla opettajalle näytetään opiskelijan tarkemmat tiedot. Listassa näkyy salasanaa lukuunottamatta kaikki rekisteröintitiedot.

Käyttäjän tiedot toteutetaan toisessa kehitysiteraatiossa.

3.8 Tehtävien valinta (opiskelija)

Kirjautumisen jälkeen opiskelijalle näytetään tehtävälista, josta opiskelija voi valita tehtävän ratkaistavaksi. Listassa näytetään tehtävän kieli, aihepiiri ja nimi, sekä tieto onko opiskelija yrittänyt jo tehtävän suoritusta ja onko yritys onnistunut.

3.9 Tehtävään vastaaminen (opiskelija)

Opiskelijan ratkaistavaksi valitsemasta tehtävästä näytetään nimi ja tehtävänanto, mahdollinen valmis koodi mikäli kyseessä on täyttötehtävä, sekä tekstialue johon opiskelija kirjoittaa vastauksensa. Lisäksi on linkki jolla pääsee takaisin tehtävälistaukseen.

Kun opiskelija on kirjoittanut vastauksensa, hän painaa "Suorita"nappia joka vie takaisin tehtävävastaussivulle. Sivulla näytetään samat tiedot kuin alussa, mutta lisäksi näytetään palautetta tehtävän ratkaisun onnistumisesta/epäonnistumisesta.

Tehtävään vastaaminen toteutetaan toisessa kehitysiteraatiossa.

4 Arkkitehtuurisuunnitelma

Järjestelmä on jaettu kahteen osajärjestelmään, Titokone ja eAssari/TitoTrainer. Ideaalitapaus olisi ollut kolme osajärjestelmää, jossa eAssari on TitoTraineria käyttävä sovelluskehys. Tämä malli ei kuitenkaan ole mahdollinen johtuen eAssarin keskeneräisyydestä. Lopputuotteelle asettujen vaatimusten täyttäminen vaatii huomattavia lisäyksiä eAssariin, mm. puuttuvat kurssien ja käyttäjien hallinta täytyy tehdä. Kireän aikataulun vuoksi eAssaria ei yritetä säilyttää geneerisenä kehyksenä, vaan muokkauset tehdään vain Kohahdus projektin tarpeita ajatellen. Käytännössä eAssarista tehdään siis ns. code fork, eli synnytetään täysin TitoTrainer-spesifinen eAssari-versio joka ei ole yhteensopiva geneerisen eAssarin kanssa.

Ainoa osuus eAssarista jota pyritään käyttämään sen alkuperäisen arkkitehtuurin mukaan, on tehtävien tarkistus eli Answer-servlet ja Analyzer sekä Feedback rajapinnat. Tietokannan rakennetta ei myöskään muuteta, mutta kaikkia tauluja ei käytetä (kts. tämän dokumentin luku Tietokanta).

Titokonetta käytetään sen TTK91-rajapintojen kautta, joita muokataan tarpeen mukaan jotta vaatimuksissa esitetyt tehtävien tarkistukset ja suorituksenaikaiset tilastot saadaan toteutettua. Titokoneen käytön ja muokkauksen yksityiskohdat jätetään toisen iteraation asioiksi.

5 Tietokanta

Tietotokannan rakenteen määrää eAssari kehys. Rakenne on kuvattu liitteessä 1. Ajanpuutteesta johtuen module-tietokantataulua ei tulla käyttämään sen mahdollistamassa laajuudessa. Kyseisestä taulusta luodaan vain yksi rivi kantaan, johon sitten viitataan muista tauluista. Lisäksi jokainen tehtävä sisältyy jokaiseen kurssiin. Eli kun tehtävä luodaan, niin se liitetään jokaiseen kurssiin. Lisäksi kun kurssi luodaan, niin jokainen tehtävä lisätään kyseiseen uuteen kurssiin. Tämä linkitys tapahtuu "taskinmodule-tietokantataulussa.

Tietokantaa käytetään aina DBHandler-luokan kautta, joka piilottaa tietokantarakenteen yksityiskohdat. Eli SQL-lauseita tulee vain ks. luokkaan eikä muualle koodiin. DBHandler-luokasta voidaan luoda instanssi staattisen DBHandler.getInstance() -metodin avulla. Tämä tarkoittaa sitä, että ko. luokkaa voidaan käyttää virtuaalikoneen sisällä mistä vaan.

5.1 tasktype

Tasktype on eAssari-kehyksen tarvitsema taulu, joka määrittelee tehtävätyypin käyttämät Analyzer ja Displayer luokat. Tauluun olisi mahdollista tallentaa myös muuta tehtävätyyppikohtaista tietoa, mutta ainakaan ensimmäisen iteraation kohdalla tälle ole näkyvissä tarvetta.

Kuva 1: Tietokantakaavio

5.2 task

Task määrittelee tehtävän. Itse task-tauluun tallentaan lähinnä tehtävän nimi ja luojan nimi. Tehtävän kriteerit, kuvaus, palautteet yms lisätään yleiskäyttöiseen attributevalues-tauluun.

5.3 course

Määrittelee kurssin nimen ja tunnisteen. Tähän tauluun viitataan useista muista taluista, jotta esimerkikisi opiskelijoiden suoritukset voidaan kirjata tietylle kurssille.

5.4 module

Luotuja tehtäviä ei liitetä suoraan kursseihin, vaan moduuleihin ja moduulit liitetään kursseihin. Kurssiin voi liittää useita moduuleja, mutta yhtä moduulia ei voi lisätä useaan kurssiin. TitoTrainer ei käytä moduuleja siinä laajuudessa kuin tietokannan rakenne mahdollistaisi, vaan toteutuksen ja käyttöliittymien yksinkertaistamiseksi jokaiselle kurssille luodaan tarkalleen yksi moduuli johon lisätään kaikki tehtävät.

5.5 taskinmodule

Yhdistää tehtävän kursiin ja moduuliin. Normaalitapauksessa siis tehtävä lisättäisiin jonkin kurssin johonkin moduulin luomalla rivi taskinmodule-tauluun.

TitoTrainerin tapauksessa on määritelty, että kaikki tehtävät kuuluvat kaikkiin kursseihin, joten kun opettaja luo tehtävän, TitoTrainer luo automaattisesti yhtä monta taskinmodule riviä kuin on kursseja. Samoin kun luodaan uusi kurssi, sille luodaan automaattisesti moduuli ja kaikki olemassa olevat tehtävät lisätään ko. moduuliin. Näin ollen aina pitää paikkansa, että Taskinmodule rivien määrä = Task rivien määrä * Course rivien määrä.

5.6 attributevalues

Yleiskäyttöinen varasto jonne voi tallentaa mitä tahansa lisätietoa koskien mitä tahansa muuta taulua. Tänne tallennetaan tehtävien kriteerit, palautteet, aihepiiri, esimerkkiratkaisu, ja lähes kaikki muu tieto tehtävästä nimeä lukuunottamatta. Lisäksi voidaan tallentaa esimerkiksi lisätietoja kursseista jos tarvetta ilmenee.

5.7 eauser

Taulu kaikkien käyttäjätietojen tallentamiseen. Sekä oppilaat ja opettajat tallennetaan tänne, erona näillä on vain status-sarakkeen arvo.

5.8 storedanswer

Kun opiskelija vastaa tehtävään, vastaus sekä tiedot sen oikeellisuudesta tallennetaan tähän tauluun.

5.9 studentmodel

Jokaista tehtävän vastausyritystä kohden luodaan storedanswer-tauluun yksi rivi. Studentmodel-taluun tallennetaan yksi rivi per tehtävä, vaikka vastausyrityksiä olisi useampia. Rivillle tallennetaan tieto siitä, onko tehtävä suoritettu onnistuneesti ja kuinka monta yritystä tehtävään on käytetty. Nämä tiedot olisi mahdollista päätellä koostefunktioden avul-

la storedanswer-taulusta, mutta ilmeisesti tehokkuuden parantamiksi on luotu erillinen koostetaulu.

5.10 taskattributes

Task-attributes on eAssarissa tarkoitettu kuvaamaan tehtävien attribuuttien rakennetta. eAssari ei kuitenkaan missään vaiheessa itse hyödynnä tätä taulua, eikä TitoTrainerkään sitä tarvitse, joten taulu jää täysin käyttämättä.

5.11 pluginparameters

Tämän taulun tarkoitus on jäänyt täysin hämärän peittoon, eAssari ei missään kohtaa koodissaan viittaa mihinkään plugineihin. TitoTrainerillä ei myöskään ole tälle taululle mitään käyttöä, joten se jää täysin käyttämättä.

5.12 pluginparamattributes

Kuten pluginparameters.

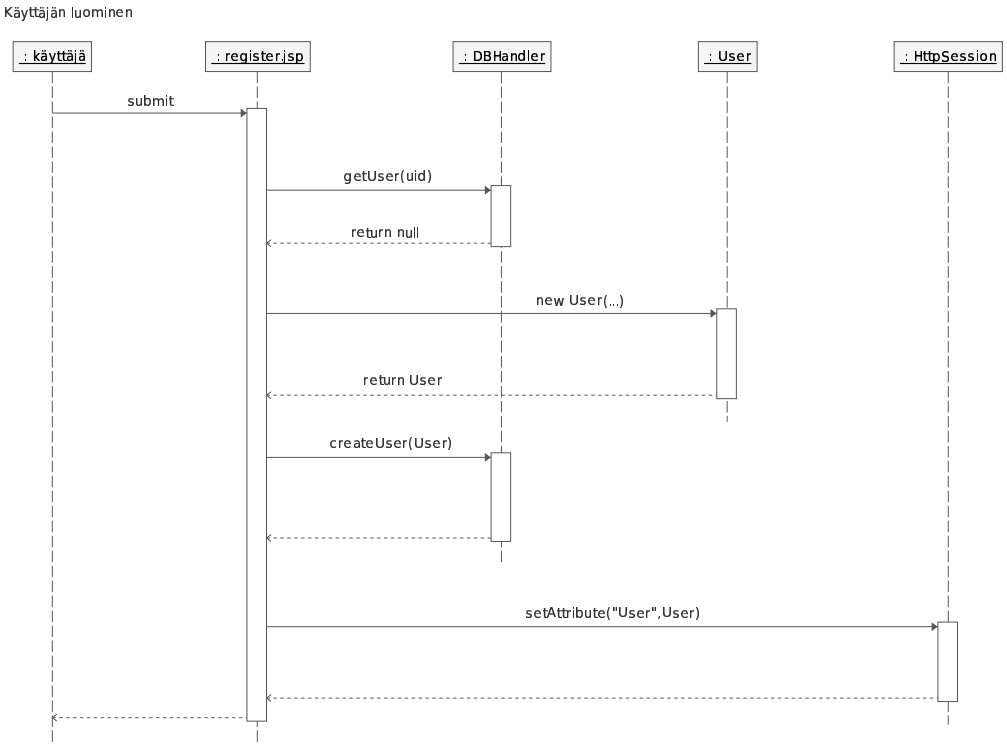
6 Luokat

DBConnectionPool Analyzer Displayer Feedback Composer +jsp:t eri toiminnoille

7 JSP sivujen sekvenssikaaviot

Tämä luku esittelee toteutettavien JSP-sivujen toimintalogiikan. Sekvenssikaaviot esittelevät tärkeimmät eri toiminnoissa käytetyt luokat ja metodit, menemättä kuitenkaan aivan samalle tasolle kuin lopullinen ohjelmakoodi. Myöskään kaikkia toimintoja ei kuvata, esimerkiksi erilaisten listausten näyttäminen katsotaan niin suoraviivaiseksi, että ne onnistuvat ilman erillistä suunnitelmaa.

Sekvenssikaavioit esittävät miten TitoTrainer/eAssari käsittelee tapahtuman, jossa käyttäjä on täyttänyt lomakkeen ja lähettänyt sen palvelimen käsiteltäväksi. Vain onnistuneet tapahtumat on kuvattu, esimerkiksi väärän salasanan syöttäminen kirjautumislomakkeeseen katkaisee kuvatun sekvenssin ja palauttaa käyttäjälle virheilmoituksen. Käyttäytyminen poikkeustilanteissa kuvataan tarkemmin käyttöliittymäsuunnitelmassa (Liite 1).



Kuva 2: Käyttäjän rekisteröityminen

7.1 Register.jsp

7.2 Login.jsp

7.3 TeacherTaskList.jsp

7.4 TaskList.jsp

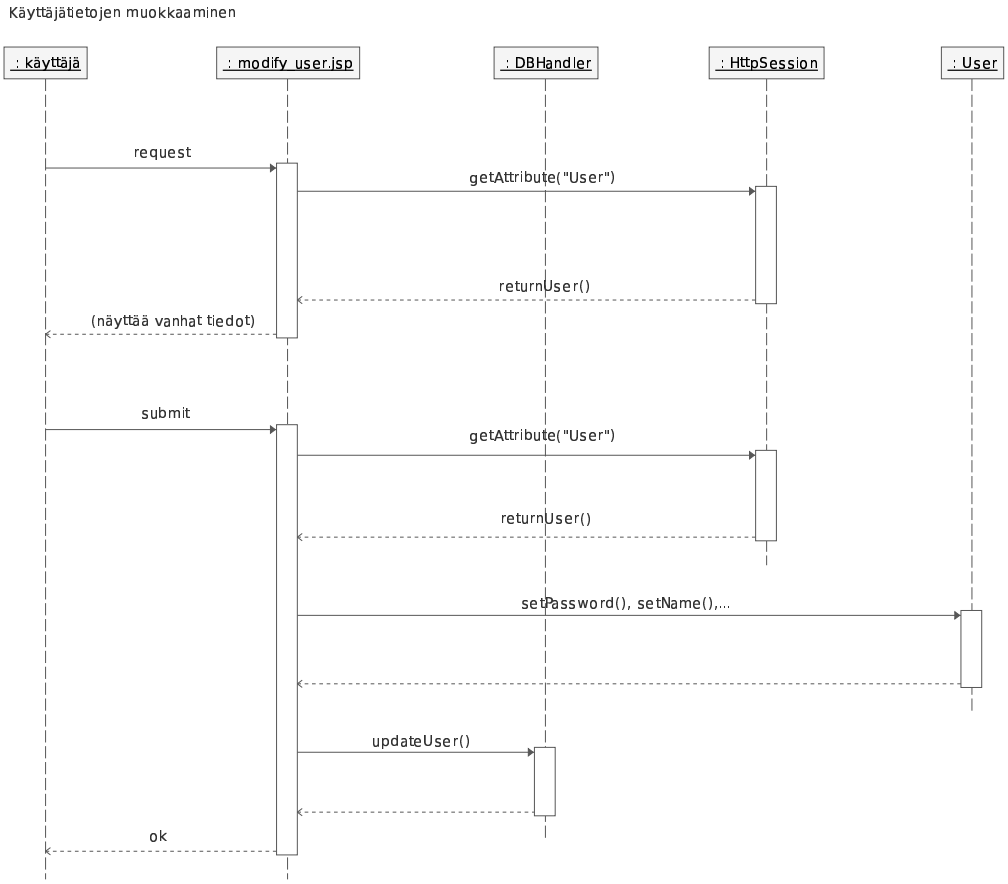
Ei kaaviota. Sisältää vain opiskelijan tehtävälistauksen.

7.5 Report.jsp

Toteutetaan ja suunnitellaan toisessa iteraatiossa

7.6 UserInfo.jsp

Toteutetaan ja suunnitellaan toisessa iteraatiossa



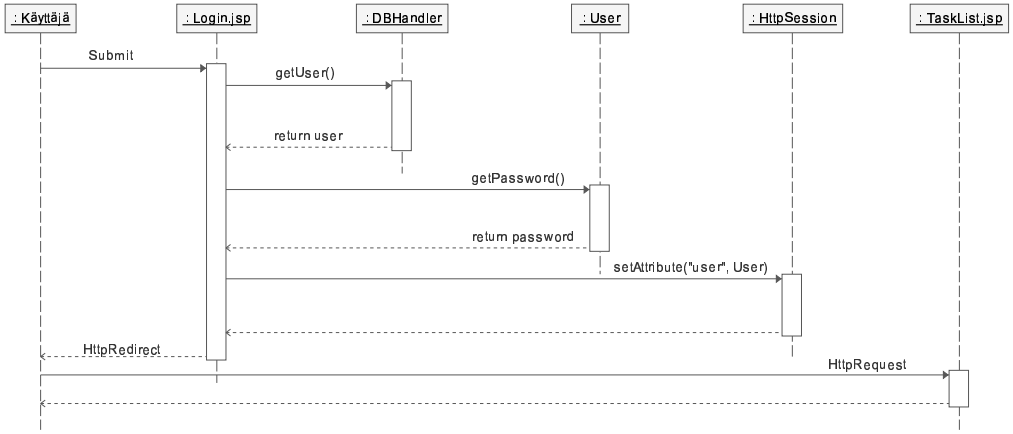
Kuva 3: Käyttäjätietojen muokkaaminen

7.7 Composer.jsp

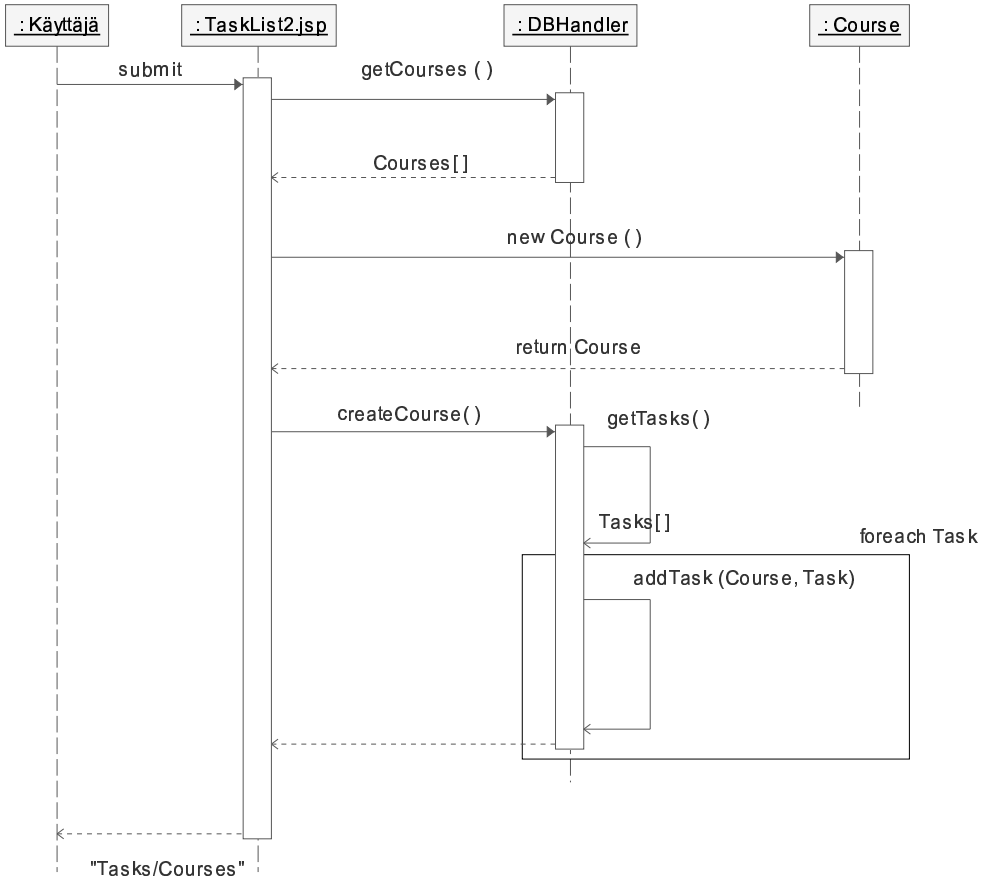
7.8 TeacherTaskList.jsp

7.9 Answer-servlet

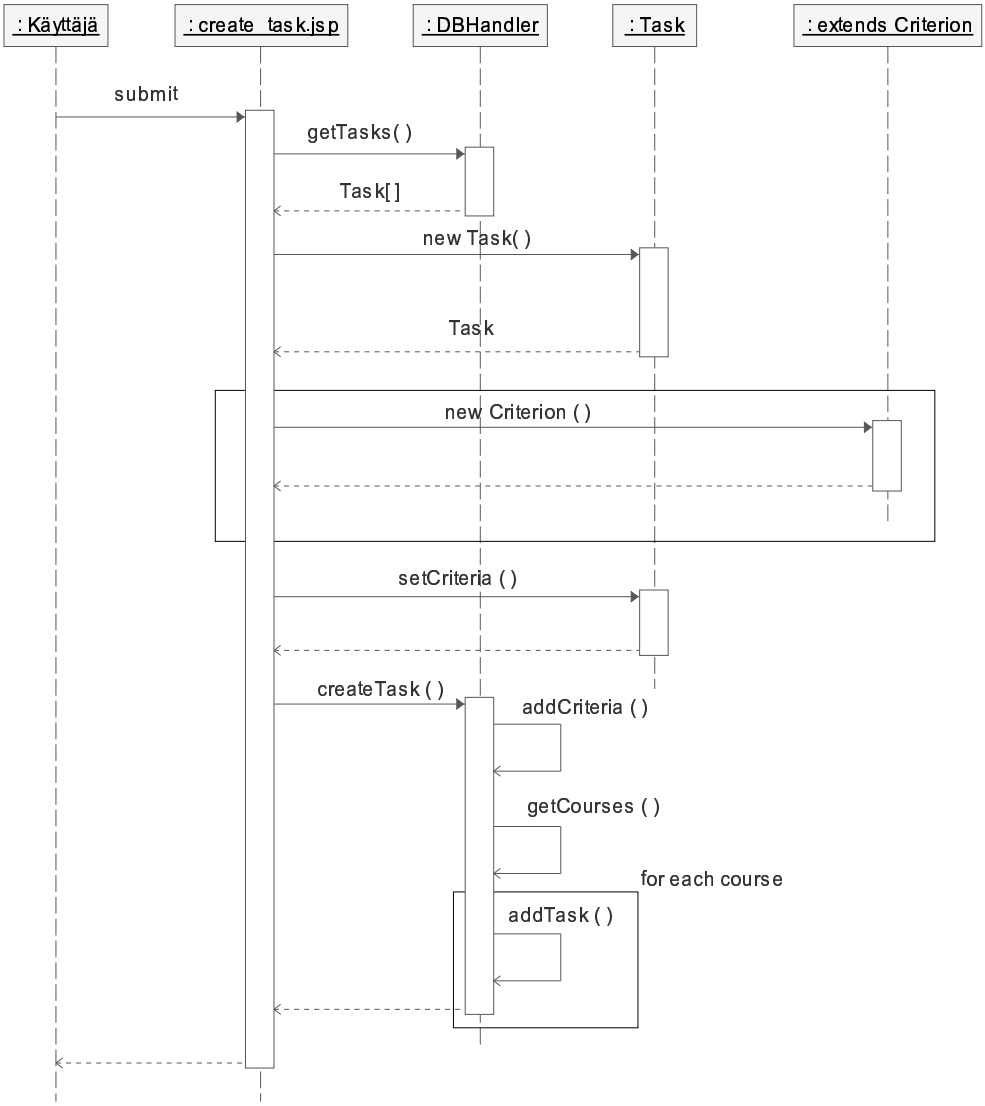
Toteutetaan ja suunnitellaan toisessa iteraatiossa



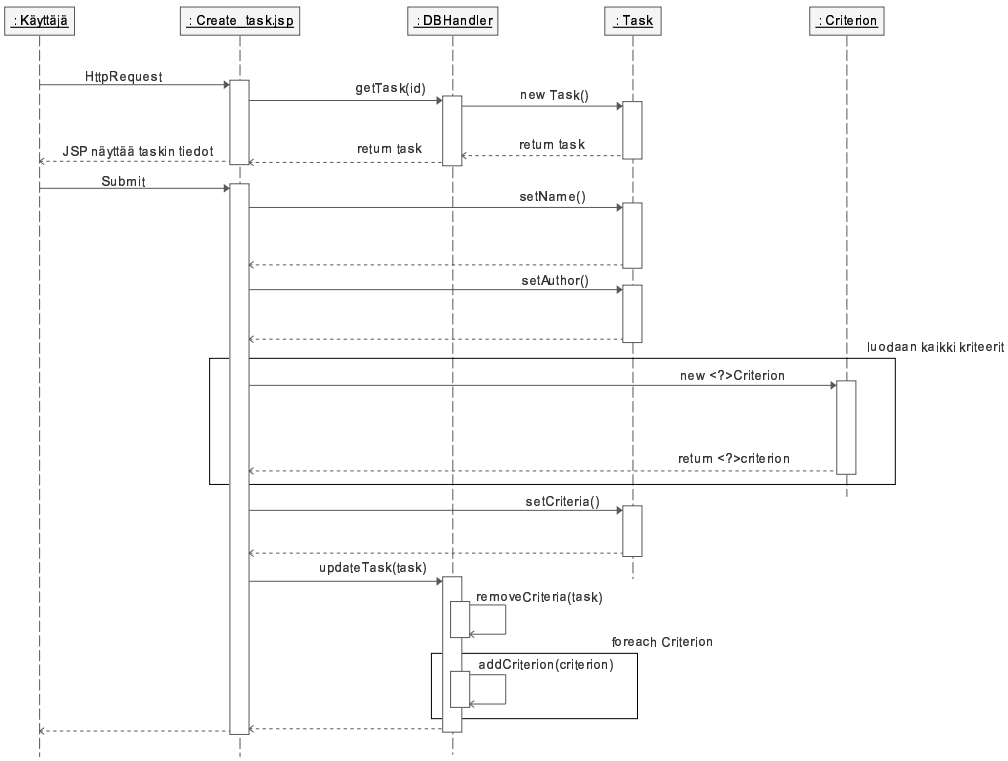
Kuva 4: Kirjautuminen



Kuva 5: Kurssin luominen



Kuva 6: Tehtävän luominen



Kuva 7: Tehtävän muokkaaminen