TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma
Tietokonetekniikka

Tutkintotyö

Toni Ollinpoika

TENTTISOVELLUKSEN GRAAFINEN KÄYTTÖLIITTYMÄ

Työn ohjaaja Esa Kujansuu
Työn teettäjä Rakennusosasto, Tampereen Ammattikorkeakoulu
Tampere 2005

TAMPEREEN AMMATTIKORKEAKOULU
Tietotekniikka
Tietokonetekniikka
Ollinpoika, Toni Tenttisovelluksen graafinen käyttöliittymä
Tutkintotyö 30 sivua
Työn ohjaaja Esa Kujansuu
Työn teettäjä Rakennusosasto, Tampereen ammattikorkeakoulu
Syyskuu 2005
Hakusanat Java, graafinen käyttöliittymä

TIIVISTELMÄ

Tammikuussa 2005 Tampereen ammattikorkeakoulussa lähdettiin kehittämään sähköistä, koulun intranet-verkossa käytettävää tenttisovellusta. Sovelluksella opettajat pystyvät luomaan nopeasti ja helposti kysymyksiä ja kokoamaan niistä tenttejä. Opiskelijat suorittavat nämä tentit Internet-selainikkunassa. Tavoitteena on tarjota opettajille työkalu, jolla laajankin tentin tekee tarvittaessa nopeasti sekä nopeuttaa tenttien tarkistusprosessia. Ohjelmointikieleksi on valittu Java, koska graafisen käyttöliittymän tekeminen on sillä kätevintä. Syksyllä 2005 sovelluksen ensimmäinen versio on lähes valmis.

TAMPERE POLYTECHNIC
Computer Systems Engineering
Computer Engineering
Ollinpoika, Toni
Graphical User Interface of Examination Application
Engineering Thesis 30 pages
Thesis Supervisor Esa Kujansuu
Commissioning Company Construction Department, Tampere polytechnic
September 2005
Keywords Java, graphical user interface

ABSTRACT

In January 2005 at Tampere polytechnic started the development of the examination software which was used in schools Intranet. By using this software, teachers can easily create questions and collect them to examinations. Students carry out these examinations in Internet-browser window. Goal was to offer a tool to the teachers, which speeds up the process of making examinations and checking their results. Software will be implemented with Java-programming language because making of a graphical user interface with it is easiest. In autumn 2005 the software is almost ready.

ALKUSANAT

Tässä työssä käsiteltyä tenttisovellusta tehtiin Tampereen ammattikorkeakoulussa, tammikuun 2005 alusta lähtien. Aluksi työtä teki kolme opiskelijaa: Jyri Vuorinen, Timo Soininen ja allekirjoittanut. Toukokuun 2005 jälkeen työtä jatkoi ainoastaan Jyri Vuorinen. Tero Markkanen toimi projektin vetäjänä ja Esa Kujansuu oli teknisen toteutuksen vastuuhenkilö ja koordinoija.

Haluan tässä yhteydessä kiittää Esa Kujansuuta ja Tero Markkasta, että sain olla osana tekemässä tätä projektia.

Tampereella 22.10.2005

Toni Ollinpoika

SISÄLLYSLUETTELO

1 JOHDANTO

Tampereen ammattikorkeakoulun rakennusosastolla oli kehitelty Java-appletien päälle rakennettavaa tenttisovellusta jo vuonna 2004. Tämän työn tekijä tuli kyseiseen projektiin mukaan saman vuoden lopulla. Projekti lähti liikkeelle toden teolla 2005 vuoden tammikuussa. Vanha sovellus päätettiin hylätä ja aloittaa puhtaalta pöydältä eli ohjelman toiminnan määrittelystä. Tavoitteena oli luoda ohjelma, jolla pystyttäisiin pitämään tenttejä täysin sähköisessä muodossa ja jolla opettajat kykenevät luomaan näitä tenttejä helposti ja nopeasti. Tällöin aikaa säästyisi nimenomaan tentin sisällön miettimiseen ja muihin opettajan ydintehtäviin.

Aluksi projektia vietiin eteenpäin kolmen opiskelijan voimin. Timo Soininen teki opiskelijan käyttöliittymää, Jyri Vuorinen aloitti tietokannan suunnittelulla ja toteutuksella ja jatkoi siitä opettajan käyttöliittymän tenttieditori-osioon ja allekirjoittanut luonnosteli ensin opettajan käyttöliittymän ulkoasua ja siirtyi tekemään opettajan käyttöliittymän kysymyseditori-osiota.

Tässä raportissa ei käsitellä sovellukseen kuulunutta tietokantaa sen tarkemmin, vaan keskitytään ohjelman toiminnalliseen rakenteeseen sekä rakenteeseen käyttäjän kannalta. Ohjelman rakennetta tarkastellaan myös kehityksen kannalta eli alkuperäisiä suunnitelmia ja niistä muotoutunutta nykyistä versiota.

Ohjelma oli aluksi tarkoitus toteuttaa Java-appleteilla, mutta todennäköisesti ohjelmisto tullaan toteuttamaan perinteisellä palvelimella ajettavalla ratkaisulla. Appleteista olisi etua verrattuna palvelinkoneella ajettavaan sovellukseen, mutta tässä tapauksessa ongelmaksi muodostuu tietoturva. Appletit perustuvat ideaan, jossa ohjelmakoodi ainoastaan noudetaan palvelimelta, mutta ohjelma suoritetaan käyttäjän tietokoneessa. Etuna on se, että palvelimen sijasta kuormitetaan käyttäjän tietokonetta, jolloin täysin vuorovaikutteisten verkkosovellusten rakentaminen on mahdollista. Lisäksi ohjelman käynnistymisviive on ainoa, joka kuluttaa käyttäjän aikaa. Käynnissä ollessaan ohjelma toimii kuten paikallinen sovellus. /1, s. 12 − 13/

Turvallisuus on tärkeä asia, jota vaaditaan verkosta haettavilta ohjelmakoodeilta. Palvelinkoneella ajettava ohjelma ei ole käyttäjän koneelle riski, sillä se ei pääse käsiksi käyttäjän tietokoneen tiedostoihin tai keskusmuistiin. Asiakaskoneella ajettavalla ohjelmalla taas on ilman rajoitteita täydet oikeudet tehdä mitä tahansa asiakaskoneella, joten ohjelma voi esimerkiksi tuhota tiedostoja tai tartuttaa viruksia. Java-appletien kohdalla nämä ongelmat eivät kuitenkaan ole arkipäivää, koska Java käyttää turvallisuusmanageria rajoittamaan ohjelman mahdollisesti aiheuttamia vaarallisia tilanteita asiakaskoneella. Turvallisuusmanageri ei päästä Java-appleteja käsiksi käyttäjän tietokoneen tiedostojärjestelmään tai siellä oleviin ohjelmiin. Java-appletien turvallisuus taataan seuraavasti: /1, s. 12 − 13/

- Appletit eivät voi käsitellä järjestelmässä olevia tiedostoja.
- Appletit eivät voi suorittaa järjestelmässä olevia ohjelmia.
- Appletit eivät pääse lähettämään tietoa muualle Internetiin kuin sille palvelinkoneelle, josta ohjelma on ladattu, tai käsittelemään jotain muuta suojattua resurssia.

Ongelmaksi tämän kaltaisen järjestelmän kanssa muodostuukin palvelinkoneen tietoturvallisuus. Apletin ohjelmakoodi ladataan palvelimelta kokonaisuudessaan käyttäjän koneelle ohjelman suoritusta varten, ja sopivalla ohjelmalla Javan class-päätteiset ajettavat tiedostot pystyy dekoodaamaan takaisin java-päätteisiksi lähdekooditiedostoiksi. Näistä käyttäjä pystyy tarkastelemaan ohjelman rakennetta ja toiminnoista vastaavia luokkia. Javaan perehtynyt henkilö kyllä löytää tietokantayhteydestä vastaavat luokat, joissa sijaitsevat tunnukset ja salasanat tietokannan käyttöön. Tällöin käyttäjä pystyy tarkastelemaan suoraan tietokannan sisältöä ja muokkaamaan sitä. Tämä ei tietenkään ole hyväksyttävää ja siksi apletien käytöstä joudutaan tämän sovelluksen osalta mitä luultavimmin luopumaan.

2 OHJELMISTON TOIMINNALLINEN MÄÄRITTELY

Ohjelmiston toiminnallinen määrittely tehtiin osittain koodaamisen rinnalla ja sitä tarkennettiin kun ongelmia ilmeni. Sovelluksella tulisi olemaan neljä käyttäjäryhmää, joista kaksi tärkeintä ovat opiskelijat ja opettajat. Opiskelijat suorittavat ohjelmalla tenttejä ja opettajat luovat kyseiset tentit tietokannassa olevista kysymyksistä ja huolehtivat niiden järjestelyistä. Kysymyksien luominen tietokantaan on myös opettajien tehtävä. Jälkikäteen huomattiin tarve myös ylläpito-käyttäjälle, jolla on laajennetut oikeudet ohjelman käyttöön.

2.1 Opiskelijan käyttöliittymä

Opiskelijan käyttöliittymä sisältää vain vähän toimintoja: Sisään ja uloskirjautumisen, tentin valinnan, tenttiin vastaamisen ja vastauksien lähettämisen. Sovelluksen on tarkoitus pystyä keskustelemaan Tampereen ammattikorkeakoulun opiskelijarekisterin, WinhaVillen kanssa. Kun opiskelija kirjautuu tenttisovellukseen sisälle, Winhasta tarkistetaan, millä kursseilla hän on läsnä ja mitä tenttejä hänellä on oikeus suorittaa.

2.2 Opettajan käyttöliittymä

Opettajan käyttöliittymä on huomattavasti monimutkaisempi kuin opiskelijan, koska opettajalla on paljon enemmän tehtäviä tenttisovelluksen käytössä. Opettajan tehtäviä ovat kysymyksien ja tenttien luominen sekä tenttien tarkistaminen. Sekä kysymyksiä että tenttejä voi tehdä joko vanhoista muokkaamalla tai luoda kokonaan uusia.

2.2.1 Kysymyseditori

Opettajat luovat tietokantaan kysymyksiä, joista he kokoavat tentit. Kysymyksestä tehdään joko monivalinta- tai esseekysymys. Esseekysymyksen vastausalueen koon voi määritellä kysymystä tehtäessä. Kysymykseen voi liittää kuvan, tätä ei tosin toteuteta vielä ensimmäisessä versiossa. Tietokannassa jo olevia kysymyksiä pystyy selaa-

maan. Selausnäkymässä kysymyksiä pystyy lajittelemaan ainakin tekijän ja tekoajan perusteella. Selausnäkymässä kysymyksien vastauksien ei tarvitse näkyä. Kun opettaja on valinnut yksittäisen kysymyksen selausnäkymästä, hän voi tarkastella kysymystä esikatselunäkymässä, jossa kysymys näkyy kokonaisuudessaan vastauksineen. Esikatselunäkymässä kysymykset näyttävät täysin samalta kuin lopullisessa muodossaan varsinaisessa tentissä opiskelijalle. Valitusta kysymyksestä voida ottaa kopion, jota voi käyttää sellaisenaan tai muokata.

2.2.2 Tenttieditori

Opettajat luovat tentit kokoamalla tietokannasta kysymyksiä ja lisäämällä ne tenttiin. Opettaja voi selata tietokannassa olevia kysymyksiä, kopioida jonkun toisen opettajan tekemiä kysymyksiä omaan käyttöönsä ja muokata niitä tai luoda kokonaan uusia kysymyksiä. Opettaja voi luoda tenttejä omista kysymyksistään. Kysymysten järjestystä tentissä pystyy vaihtamaan tentin tekovaiheessa. Samoin tenttiä tehtäessä on käytössä esikatselu, johon pääsee yhdellä napin painalluksella. Kysymyksille ja vastauksille voi asettaa pisteytyksen, jota käytetään oletusarvoisesti tentin kaikkiin kysymyksiin. Mahdollisuus yksittäisten kysymysten muista poikkeavaan pisteytykseen on kuitenkin olemassa.

2.2.3 Tenttien järjestäminen

Tentti on avoinna tietyn ajan, jona opiskelijat voivat käydä sen tekemässä. Opettaja määrittelee tentin aukioloajan tenttiä tehdessään. Tentti avautuu ja sulkeutuu aukioloaikansa mukaisesti, automaattisesti ilman käyttäjältä vaadittavia toimenpiteitä. Tentin tehnyt opettaja voi myös manuaalisesti avata ja sulkea tentin ennen määräaikaa.

2.2.4 Tentin tarkastaminen

Ohjelmisto tarkistaa monivalintakysymykset automaattisesti heti, kun opiskelija lähettää vastaukset tietokantaan. Monivalintakysymysten pisteet ja niistä saatavat maksimipisteet näytetään opiskelijalle myös heti tentin lähettämisen jälkeen. Opettajan teh-

täväksi jää mahdollisten esseevastauksien tarkastus ja pisteytys. Opettajan voi tarkastella tentin tuloksia sekä yksittäisen opiskelijan että koko tentin tehneen ryhmän osalta. Tulokset täytyy myös saada ulos tenttijärjestelmästä jossain helposti käytettävässä tiedostomuodossa. Mikäli kokeessa oli esseekysymyksiä, lopullisten tulosten esittely opiskelijoille on opettajan tehtävä.

2.3 Muut käyttäjät

Opettajien ja opiskelijoiden lisäksi ohjelmisto tarvitsee pari muutakin käyttäjäryhmää ylläpitoa ja hallinnointia varten. Koska kysymykset ja tentit jaotellaan aiheen ja kurssin mukaan, pitää sovelluksessa pystyä lisäämään uusia aiheita ja kursseja. Näiden lisäämiseen ei kuitenkaan ole oikeuksia tavallisella opettajalla, vaan ne kuuluvat esimerkiksi Tampereen ammattikorkeakoulussa osastosihteerin tehtäviin. Samoin osastosihteerin tulee voida siirtää mahdollisia väärän aiheen tai kurssin alle luotuja kysymyksiä ja tenttejä. Luonnollisesti näinkin laaja ohjelmisto tarvitsee myös ylläpitokäyttäjän, joka kykenee korjaamaan kaikenlaiset ohjelmiston virheellisestä käytöstä seuranneet poikkeustilanteet.

3 ALKUPERÄISET SUUNNITELMAT

Graafinen käyttöliittymä perustaa toimintansa ruudulla näkyville symboleille ja ohjeille. Käyttäjä reagoi näkemäänsä eikä hänen tarvitse muistaa ulkoa mitään mystisiä käskyjä. Käyttöliittymä ei saa hukuttaa käyttäjää turhaan tietoon. Hyvä käyttöliittymä ei näytä käyttäjälle tarpeetonta tietoa vaan se piilotetaan pois näkyviltä. /2/

Käyttäjän elämän helpottamiseksi käyttöliittymän täytyy olla avulias mutta silti kokonaan käyttäjän ohjattavissa. Aivan kuten auto vastaa kuljettajan ohjausliikkeisiin niin myös käyttöliittymän täytyy reagoida välittömästi käyttäjän toimenpiteisiin. Mikäli sovellus tekee jotain aikaa vievää toimintaa, joiden tekeminen kestää kauemmin kuin muutaman sekunnin, täytyy käyttäjälle kuitenkin heti toiminnan aloituksen yhteydessä vastata ja pyytää odottamaan. Käyttäjää pitää myös informoida kysei-

sen toiminnan aloittamisesta ja raportoida sen edistymisestä, esimerkiksi prosenttilukemalla. /2/

Jos käyttäjä yrittää tehdä jotain, mikä voidaan tulkita erehdykseksi, kuten sulkea ohjelma tallentamatta tekemiään muutoksia, pitää käyttöliittymän varmistaa tilanne ja tarjota peruutusmahdollisuus.

Vuorovaikutteisuus ja tarinan kerronta ovat toisilleen vastakkaisia. Mitä vuorovaikutteisempi jokin sovellus on, sitä vähemmän sen avulla voi kertoa ennalta suunniteltua tarinaa. Hyvä vuorovaikutteinen käyttöliittymä taipuu eri käyttäjien erilaisiin tapoihin tehdä asiat ja toimii silti loogisesti. Esimerkiksi lomakkeen täyttämisessä vuorovaikutteisuus sallii eri kohtien täyttämisen vapaassa järjestyksessä. Avustava vuorovaikutteinen käyttöliittymä osaa myös neuvoa ja varoittaa käyttäjää. Tenttisovelluksessa ei ole pyritty korostamaan vuorovaikutteisuutta vaan käytön loogisuutta ja helppoutta. Vuorovaikutteisuutta on vain sen verran, ettei se rajoita mainittuja ominaisuuksia. Opiskelijan käyttöliittymän ainoa vuorovaikutteisuus on itse tentin teko, joka vastaa suurehkon lomakkeen täyttöä. Kysymyksiin voi vastata haluamassaan järjestyksessä. Opettajan käyttöliittymässä, varsinkin kysymysten tai tentin tekemisessä, on enemmän vuorovaikutteisuutta. /2/

Käyttäjän vapaus tuo ongelmaksi paikan havaitsemisen. Isossa syöttöpohjassa tai laajalla työpöydällä liikkuvan ihmisen näkökykyä rajoittaa näytön ikkunan koko. Ihmisen luonnollinen tapa havainnoida ympäristöään perustuu oman olinpaikan muistamiseen ja havainnointiin tästä tunnetusta paikasta käsin. Käyttöliittymän pitää myös tarjota kulkureitin ilmaisevaa tietoa tai kartaksi sopivan apuvälineen paikantamista varten. Alkeellisin paikan osoitin on x-y-koordinaattiajatteluun perustuva vierityspalkki. /2/

Se ei kuitenkaan auta vähääkään silloin, kun on mahdollista liikkua dokumenttipinossa eri syvyydellä. Selattujen dokumenttien luettelo auttaa seuraamaan kuljettua reittiä taaksepäin. Tenttisovelluksessa on kuitenkin aina vain yksi ikkuna auki, joten selattujen dokumenttien luetteloa ei tarvita.

Käyttöliittymän suunnittelun kannalta on tärkeää ikkunan rakenneosien sijoittelu käyttöliittymän sääntöjen mukaan. Tärkeintä on noudattaa johdonmukaisesti ennalta sovittuja marginaaleja ja linjata rakennusosat riveihin ja sarakkeisiin. Graafista silmää kannattaa käyttää myös painikkeiden ja muiden ohjauspintojen teossa. Selkeys auttaa käyttäjää hahmottamisessa. Joskus valitettavasti näkee sovelluksia, joissa painikkeet on sijoiteltu erimittaisina satunnaisiin paikkoihin. Painikkeiden oikea järjestys tekee käytön turvalliseksi. Käyttäjä oppii suhteelliset paikat nopeasti, jolloin tekstejä ei ole koko ajan pakko lukea. Ihminen tunnistaa asioita usein sijaintipaikan mukaan. Tästä syystä on viisasta sijoittaa esimerkiksi Ok-painike aina samaan kohtaan ja jättää käyttämätön paikka tyhjäksi. /2/

Hyvä graafinen käyttöliittymä käyttää myös värejä signaalien välittämiseen käyttäjälle. Ihminen tulkitsee eri värejä eri tavalla ja käyttöliittymässä tämä pitää huomioida tai jättää värien käyttö pois. Punainen yhdistetään vaaraan ja sitä kautta pysähtymiseen ja tarkkaavaisuuteen, aivan kuten liikennevaloissa. Vihreä on punaisen vastakohta, ja se viestittää kaiken olevan ok, toimintaa voi jatkaa. Sinistä voi käyttää taustavärinä harmaan ja valkoisen ohella, kunhan sävy pidetään himmeänä. Tenttisovelluksessa on tulevaisuudessa tarkoitus ottaa värien käyttöä mukaan ainakin joissain käytön avainkohdissa tulevissa viesteissä ja niiden yhteydessä olevissa painikkeissa. /2/

3.1 Graafinen käyttöliittymä ja Java /1, s. 296 - 307/

Javan graafinen käyttöliittymä pohjautuu säiliöihin, joiden sisälle kaikki näkyvät komponentit sijoitetaan. Komponenttien järjestelyyn säiliö käyttää sijoittelumanageria. Säiliöiden koko on muokattavissa, samoin jokainen säiliöluokka voi sisältää toisia säiliöitä, jotka puolestaan sisältävät komponentteja. Sijoittelumanageri pitää huolen siitä, että komponentit skaalautuvat muutettaessa ikkunakokoa. Samalla tavalla sijoittelumanageri pitää komponentit järjestyksessä. Komponenttien väliset mittasuhteet voivat muuttua ja paikat vaihtua, mutta järjestys säilyy samana.

Kiinteällä x-y-koordinaatistolla komponentit on helppo sijoitella juuri haluamaansa kohtaan, mutta siitä seuraa muutamia ongelmia. Mikäli käyttäjä muuttaa ikkunan kokoa tai käyttää eri resoluutiota kuin ohjelman suunnittelija on tarkoittanut, saattaa osa

komponenteista jäädä toisten alle tai ulos näytöstä. Tämän takia Javaan on kehitelty erilaisia sijoittelumanagereita, jotka käyttävät komponenttien sijoitteluun suhteellista koordinaatistoa. Tenttisovelluskin päätettiin tehdä heti alusta lähtien sijoittelumanagereiden avulla, kiinteän x-y-koordinaatiston sijaan. Monet Java-sovelluskehittimet sisältävät sijoittelumanagereja, jotka ovat viralliseen spesifikaatioon sisältyviä versioita kehittyneempiä. Koska tenttisovellusta mahdollisesti myydään tulevaisuudessa muillekin oppilaitoksille, ei sen koodi saa kuitenkaan sisältää mitään sellaisia ratkaisuja, joiden käyttö voidaan tulkita tekijänoikeuslakien rikkomiseksi.

BorderLayout-sijoittelumanageri pyrkii järjestämään käyttöliittymän komponentit viiteen erilliseen lokeroon. Nämä ovat ilmansuunnat etelä, pohjoinen, itä, ja länsi sekä keskialue. BorderLayout-sijoittelumanageri voi hallita ainoastaan viittä komponenttia kerrallaan. Se voi kuitenkin venyttää ja kutistaa näitä komponentteja, jotta ne sopivat niille varattuun tilaan. Useimmiten BorderLayout-manageria käytetäänkin jakamaan näyttöalue selkeisiin lohkoihin, joihin sijoitetaan paneeleja, jotka sisältävät vasta varsinaiset näkyvät komponentit. Nämä paneelit käyttävät puolestaan jotakin toista sijoittelumanageria komponenttiensa järjestelyyn. Tenttisovelluksessa kunkin näkymän pohjalla käytetään BorderLayout-manageria, koska sillä on helppo jakaa näytön alue loogisiin kokonaisuuksiin, jolloin samankaltaiset toiminnot saadaan sijoiteltua samalle alueelle.

FlowLayout-manageri on sijoittelumanagereista yksinkertaisin, se pyrkii järjestämään komponentit määritellyn kokoisina vasemmalta oikealle sekä ylhäältä alas. Komponenttien sijoittelu tapahtuu samalla tavalla kuin tekstin kirjoitus tekstinkäsittelyohjelmalla. Komponentit voi tasata joko vasemmalle, keskelle tai oikealle. Tenttisovelluksessa FlowLayout-manageria on käytetty lähinnä otsikko-komponenttien sijoittamiseen apletin ylälaidan paneeliin.

Java 2:n myötä käyttöön tuli monipuolinen BoxLayout-sijoittelumanageri. Se jakaa käyttöliittymäkomponentit laatikoihin, joiden pituus ja leveys voi vaihdella. Komponentit voidaan lisätä laatikoihin vaaka- tai pystysuunnassa. Lisäksi laatikoiden väleihin voi lisätä kiinteän kokoisia tyhjiä alueita. Tenttisovelluksessa BoxLayout-manageria on käytetty runsaasti, koska sillä on suhteellisen yksinkertaista luoda taulukon omaisia rivejä ja sarakkeita esimerkiksi painikkeille tai tekstikentille.

Hieman muista sijoittelumanagereista erottuva CardLayout-manageri on ehkä kuitenkin tämän kaltaisessa sovelluksessa se tärkein. Paneeli, jonka sijoittelumanagerina on CardLayout, toimii säiliönä pelkästään toisille paneeleille. Idea on se, että CardLayout-manageri ei järjestele näitä paneeleja vierekkäin tai allekkain, vaan päällekkäin aivan kuten pelikortit ovat korttipakassa. Vain päällimmäinen paneeli näkyy käyttäjälle. CardLayout-luokan metodeilla vaihdellaan päällimmäistä eli käyttäjälle näkyvää paneelia tarpeen mukaan.

Koko tenttijärjestelmän graafinen käyttöliittymä pohjautuu CardLayout-sijoittelumanagerin käyttöön. Opiskelijan käyttöliittymä on oma aplettinsa ja opettajan käyttöliittymä omansa, mutta muuten kaikki eri näkymät ovat lopulta saman apletin päällä olevan paneelin lapsikomponentteja saman CardLayout-managerin hallinnassa. Niiden näkyvyyttä hallitaan painikkeiden tapahtumankäsittelijöiden ohjelmakoodissa.

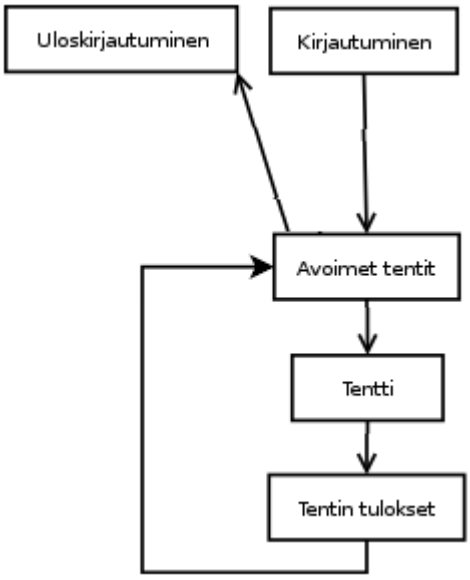
Tenttijärjestelmän ohjelmakoodia ei ajeta jatkuvasti, vaan suurimman osan ohjelman käyttöajasta sovellus odottaa käyttäjän toimenpiteitä. Näitä toimenpiteitä ovat näppäimistöltä jonkin napin painaminen, hiiren liikuttaminen tai hiiren näppäimien painallus. Korkean tason ohjelmointikielellä, tässä tapauksessa Javalla, toteutettavan sovelluksen tekijän ei onneksi tarvitse itse tehdä hiiren liikuttamisen tai näppäimistön painelun toteuttavaa ohjelmakoodia, vaan siitä huolehtivat tietokoneen käyttöjärjestelmä ja Javan virtuaalikone. Esimerkiksi tekstikenttiin pystyy syöttämään tekstiä, kunhan hiirellä vain on aktivoitu kyseinen tekstikenttä. Tällaisesta ei sovelluksen tekijän tarvitse murehtia. Sen sijaan tekstikenttään syötetyn tekstin käsittely on sovelluksen tekijän ongelma, samoin kuin käsittelyn aloittava toimenpide eli liipaisu. Tenttisovelluksessa suurin osa näistä liipaisuista on toteutettu painikkeilla. Kun käyttäjä painaa tallenna-painiketta, sovellus tarkistaa, että kaikissa tarvittavissa kohdissa on tietoa, lukee nämä tiedot ja tallentaa ne tietokantaan.

Java 1.1:ssä tapahtumien käsittely on toteutettu siten, että luokat, jotka aikovat käsitellä jonkin tapahtuman, rekisteröidään seuraamaan kyseistä tapahtumaa. Tällaisella luokalla voi olla useita tapahtumia, joita se ”kuuntelee”. Samoin yhden tapahtuman käsittelemiseksi voi asettaa useita luokkia. Systeemin hienous on se, että tapahtumaa seurataan vain, jos sille on rekisteröity käsittelijä. Koska sovelluksessa liikkuvat ainoastaan sellaiset tapahtumaviestit, joille on käsittelijä, rekisteröimättömiä tapahtumia ei käsi-

tellä ja säästetään resursseja. Tätä mallia kutsutaan tapahtuman delegointimalliksi. Komponentit laukaisevat tapahtumat, jolloin ne toimivat tapahtuman lähteinä. Komponentti, joka kuuntelee jotain tiettyä tapahtumaa, on tapahtuman kuuntelija. Tapahtuman ilmetessä tieto tapahtumasta välitetään kuuntelijalle erillisellä tapahtumaoliolla.

3.2 Opiskelijan käyttöliittymä

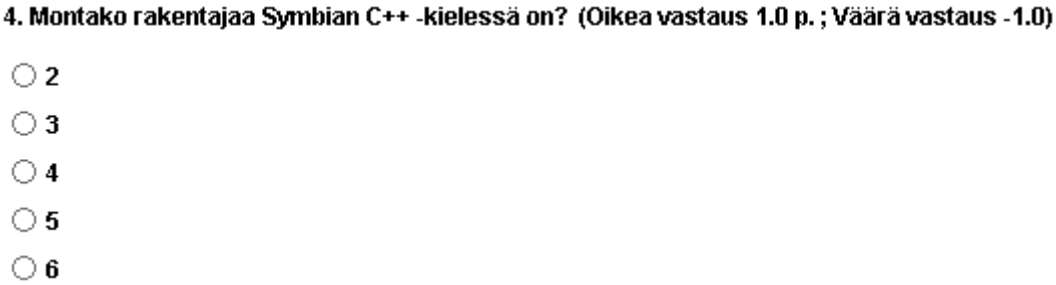
Kuvassa 1 on esitetty opiskelijan käyttöliittymän lohkokaavio. Kun opiskelija avaa tenttisovelluksen, hän tulee sivulle, josta kirjaudutaan sisään. Tässä opiskelija syöttää käyttäjätunnuksen ja salasanan. Tenttisovellus tarkistaa, että kyseisellä käyttäjätunnuksella löytyy Winhasta opiskelija, ja että salasana on oikea.



Kuva 1. Opiskelijan käyttöliittymän lohkokaavio /3/

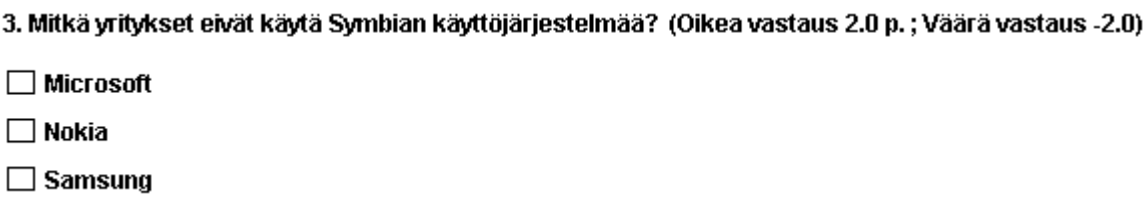
Seuraavassa vaiheessa opiskelijalla näytetään avoimet tentit eli tentit, jotka hänen on mahdollista sillä hetkellä suorittaa. Valittuaan tentin opiskelijan näytölle tulostuu varsinainen tentti, jonka kysymyksiin vastataan.

Yksittäisellä kysymyksellä on kolme mahdollista tyyppiä. Ensimmäinen niistä on monivalintakysymys, johon on vain yksi oikea vastaus ja vähintään yksi väärä vastaus. Tällaisen kysymyksen vastauksilla on jokaisella oma radio nappi, jolloin ei ole mahdollista vastata kuin yhteen kohtaan. Kuvassa 2 on esillä tällainen kysymys.



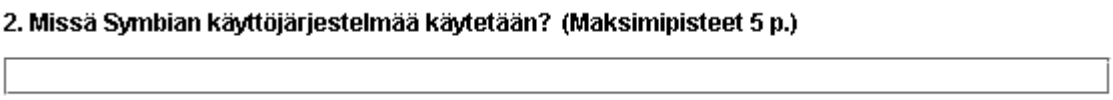
Kuva 2. Monivalintakysymys radionapeilla

Toinen kysymystyyppi on monivalintakysymys, jolla voi olla useampi oikea vastaus ja johon voi vastata useampaan kuin yhteen kohtaan. Nämä kysymykset näkyvät tentissä valintalaatikoilla toteutettuna. Kuvassa 3 on esimerkki tästä kysymystyypistä.



Kuva 3. Monivalintakysymys valintalaatikoilla

Kolmas kysymystyyppi on esseekysymys. Tässä yhteydessä esseekysymyksellä tarkoitetaan kaikkia niitä kysymyksiä, joihin ei ole annettu valmiita vastausvaihtoehtoja vaan tietokoneen ruudulla kysymyksen alla on joko tekstikenttä tai tekstialue, johon vastaus kirjoitetaan. Vastauksen ei siis välttämättä tarvitse olla kuin yksi sana tai jopa vain yksi numero, oleellista on se, että sitä ei ole valmiiksi annettu. Kuva 4 esittelee tämän kysymystyypin.



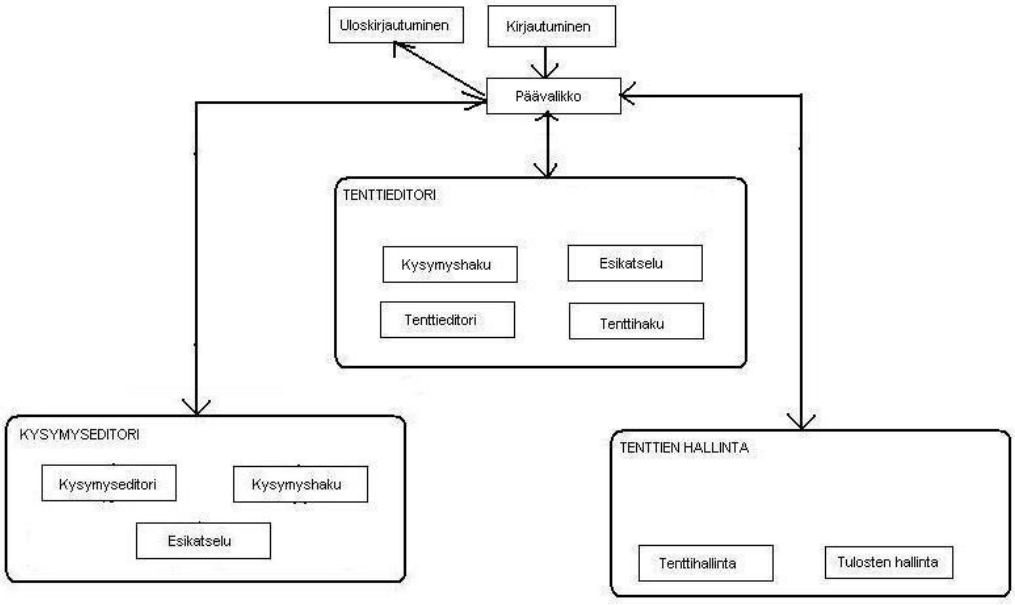
Kuva 4. Esseekysymys

Tentin suoritus lopetetaan lähetä painiketta painamalla, jonka jälkeen opiskelijalta kysytään vielä varmistus sille, että hän todellakin on saanut tentin valmiiksi ja haluaa lähettää vastaukset. Ohjelma tarkistaa tentin ja näyttää saman tien reaaliajassa opiskelijalle tentin tulokset. Poikkeuksena edellä mainittuun ovat esseekysymykset, jotka vaa-

tivat tentin tehneen opettajan pisteytyksen. Kun opiskelija on nähnyt tulokset ja kuitannut ne, ohjelma siirtyy takaisin avoimet tentit -näkymään. Tästä näkymästä on myös uloskirjautuminen, koska ei ole sallittavaa, että kesken tentin teon opiskelija vahingossa kirjautuu ulos sovelluksesta.

3.3 Opettajan käyttöliittymä

Opettajan käyttöliittymä päätettiin jakaa ohjelmiston tekovaiheessa kolmeen loogiseen osaan: kysymyseditoriin, tenttieditoriin ja tenttien hallintaan. Tähän päädyttiin henkilöresurssien sekä projektin loogisen etenemisjärjestyksen takia. Timo Soininen toteutti opiskelijan käyttöliittymää, jolloin allekirjoittaneen ja Jyri Vuorisen vastuulle jäi opettajan käyttöliittymän toteutus. Allekirjoittanut alkoi suunnitella ja koodata kysymyseditoria ja Jyri Vuorinen tenttieditoria. Tässä vaiheessa ne toteutettiin vielä erillisinä appleteina, jotka käyttivät saman tietokannan palveluita. Suunnittelussa oli alusta asti tavoitteena jakaa toteutus pieniin apuluokkiin, jotta tiedostojen koot pysyisivät mahdollisimman pieninä ja siten helpompina ylläpitää ja muokata. Toinen ääripää olisi ollut kaiken toiminnallisuuden sijoittaminen itse apletin sisälle, jolloin uusien ominaisuuksien lisääminen ja ylläpito ovat todella työlästä toimintaa.

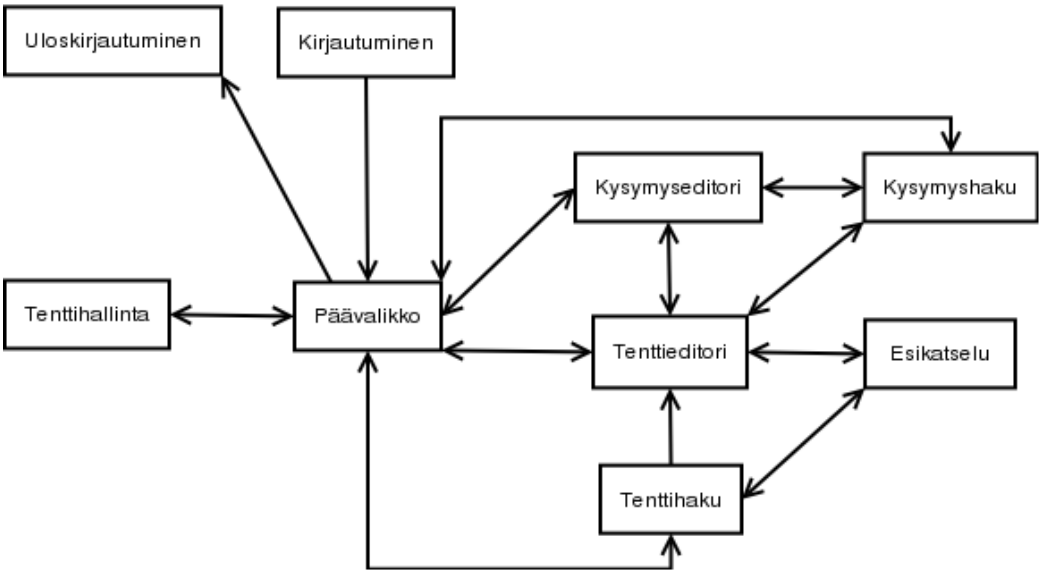


Kuva 5. Opettajan käyttöliittymän lohkokaavio

Kuvassa 5 on esitetty opettajan käyttöliittymän lohkokaavio työnjaon kannalta. Opettajat kirjautuvat aluksi sisään, jonka jälkeen näytöllä näkyy ohjelman päävalikko. Uloskirjautumisen lisäksi päävalikosta voi siirtyä johonkin kolmesta eri lohkosta:

- kysymyseditoriin
- tenttieditoriin
- tenttien hallintaan

Kysymyseditorissa käyttäjä voi luoda uuden kysymyksen tai selata tietokannassa olevia kysymyksiä ja muokata niistä uusia. Sekä editorista että selausnäytöstä pääsee esikatseluun tarkastelemaan kysymystä. Tenttieditorissa käyttäjä voi luoda uuden tentin ja koota siihen kysymykset tietokannasta tai hän voi selata tietokannassa jo olevia tenttejä ja valita niistä jonkin muokattavaksi. Tenttien hallinta on jaettu kahteen osa-alueeseen. Tenttihallinta käsittää tenttien järjestämisen ja tulosten hallinta nimensä mukaisesti tulosten tarkastelun, mahdollisten essee-kysymyksien tarkastuksen ja tulosten siirron tenttisovelluksesta johonkin muuhun järjestelmään.

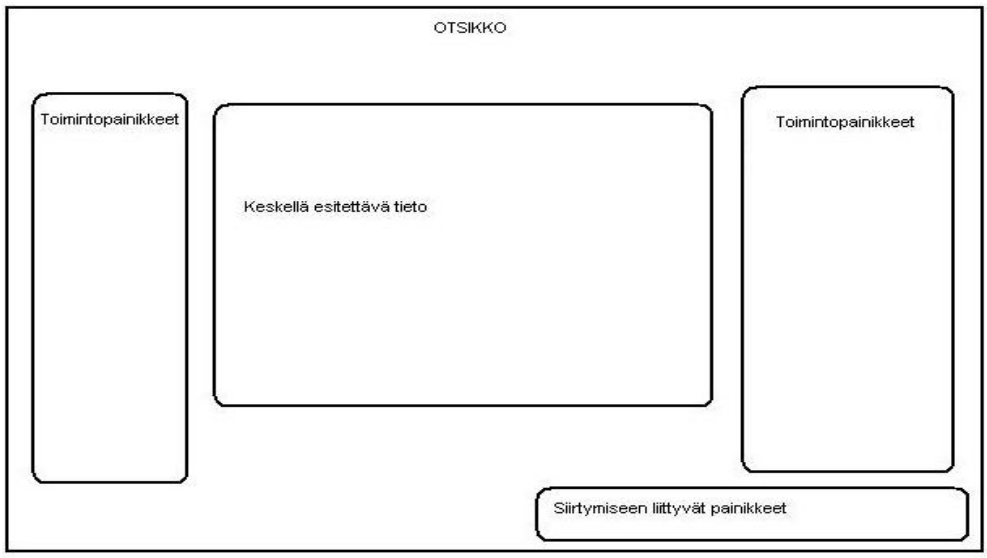


Kuva 6. Opettajan käyttöliittymän lohkokaavio /3/

Kuva 6 esittää opettajan käyttöliittymää ohjelmalohkojen kannalta. Kuvassa näkyy myös siirtymiset lohkojen välillä. Tenttihallinta näkyy kuvassa vielä yhtenä lohkona. Tämä johtuu siitä, että sen sisältämiä moduuleita ei alussa mietitty sen tarkemmin, vaan päätettiin toteuttaa ensin kysymys- ja tenttieditorit toimivaan kuntoon. Näin saa-

taisiin selkeää näyttöä työn etenemisestä projektin tulevaisuudesta päättäville tahoille, sekä saataisiin selkeämpi kuva ongelmista, mitä ohjelmistoa kehitettäessä mahdollisesti ilmenisi. Kun ohjelmisto on siinä vaiheessa, että pystytään tekemään uusia kysymyksiä ja muokkaamaan vanhoja sekä luomaan tenttejä, on paljon helpompi suunnitella ja toteuttaa niiden tenttien hallintaan vaadittavat ohjelmalohkot.

Yksi lohko kuvassa esittää yhtä ohjelmamoduulia ja samalla yhtä näyttöä, eli kun käyttäjä luo uutta kysymystä, kaikki siihen kyseisellä hetkellä tarvittavat painikkeet ja tekstikentät näkyvät ruudulla. Kun käyttäjä haluaa tarkastella luomaansa kysymystä esikatselussa, siirrytään toiseen näyttöön, jossa kysymys näkyy samanlaisena kuin lopullisessa muodossaan opiskelijan koneen ruudulla. Näyttöjen toiminnat riippuvat hieman siitä, mistä kyseiseen näyttöön on tultu. Mikäli esimerkiksi kysymyseditoriin on tultu tenttieditorista muokkaamaan kysymystä, niin sieltä on mahdollista palata takaisin tenttieditoriin. Jos taas kysymyseditoriin on tultu päävalikosta, ei siellä tarvita paluumahdollisuutta tenttieditoriin. Esikatselu toimii siten, että sieltä on paluumahdollisuus ainoastaan siihen näkymään, mistä sinne tultiin, ei muualle. Tämä on puhtaasti käytön yksinkertaistamiseksi, vaikkakin se vaatii ohjelman taustalle enemmän koodia erilaisten ehtolauseiden muodossa.



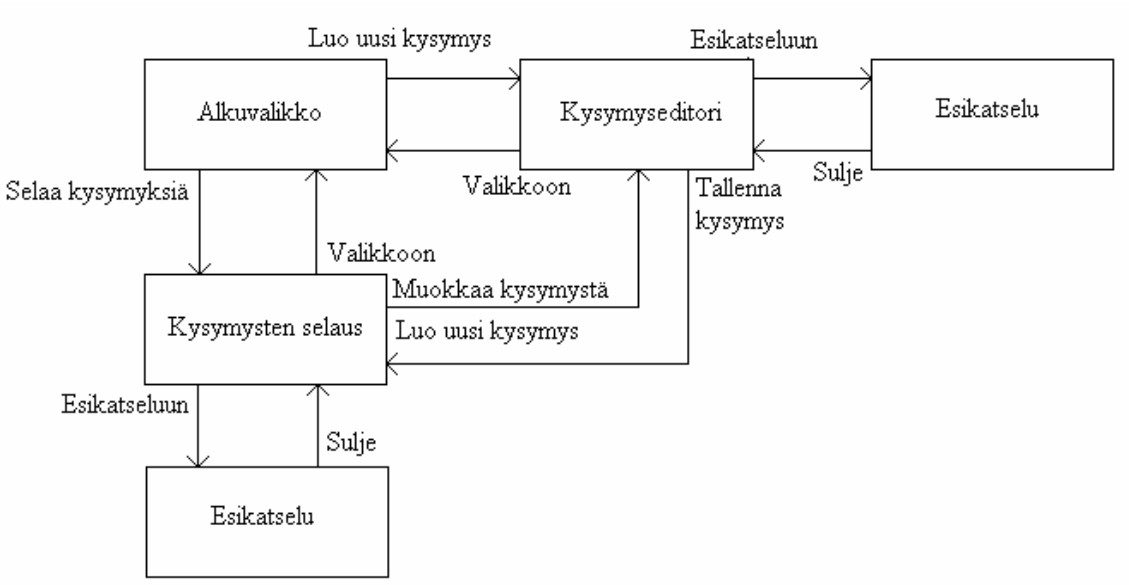
Kuva 7. Layout-sabluuna

Kuvassa 7 esitellään sabluuna, jonka näköisiksi kaikki opettajan käyttöliittymän lohkot pyrittiin rakentamaan. Tästä pohjasta huolimatta kysymyseditorin ja tenttieditorin välille syntyi tekovaiheessa pieniä käytännön eroja toimintalogiikassa ja painikkeiden

paikoissa. Nämä erot olivat kuitenkin niin pieniä, että näkymät saatiin suhteellisen helposti sovitettua samaan muottiin.

4 KYSYMYSEDITORI

Tässä luvussa selvitetään kysymyseditorin rakenne ja toiminnallisuus. Tämä tarkastelu pohjautuu tämän työn tekijän toteuttamaan kysymyseditorin kehitysversioon. Tällöin kysymyseditori ja tenttieditori olivat vielä kaksi erillistä sovellusta, jotka käyttivät samaa tietokantaa. Luonnollisesti kysymyseditoria kuten myös muuta tenttisovellusta on toukokuun 2005 jälkeen kehitetty eteenpäin. Jako kysymys- ja tenttieditorien välillä on osittain keinotekoinen ja se tehtiinkin vain, jotta kumpikin opettajan käyttöliittymän tekijöistä pystyisi aluksi etenemään itsenäisesti ja saataisiin nopeammin näkyvää aikaiseksi, kun ei tarvitsisi odotella toisen edistymistä joka välissä. Lopullisessa tenttisovelluksessa kysymyseditori ja tenttieditori yhdistetään yhdeksi kokonaisuudeksi siten, että käyttäjä ei huomaa mitään raja-aitoja tai eroja käyttöliittymän ulkoasussa siirtyessään esimerkiksi tenttieditorista selaamaan kysymyksiä ja palatessaan takaisin tenttieditoriin.



Kuva 8. Kysymyseditorin lohkokaavio

Kysymyseditorin rakenne oli kuvan 8 mukainen. Kuvasta on jätetty pois aiheiden ja kurssien lisäämiseen tarkoitetut lohkot, koska niiden prioriteetti oli matalampi ja nii-

hin ei ollut vielä toukokuussa 2005 toteutettu mitään toiminnallisuutta, vaan ne olivat pelkkiä näkymiä.

Kuvassa 8 näkyy myös lohkosta toiseen siirtymisen aiheuttava toiminta. Alkuvalikosta valitaan joko uuden kysymyksen luominen tai olemassa olevien kysymysten selaaminen ja siirrytään sen mukaisesti joko editointinäkymään tai selausnäkymään. Selausnäkymään esikatselumahdollisuus on sisällytetty sen takia, että itse selausnäkymässä näkyvät pelkät kysymykset, mutta esikatselunäkymässä kysymys näkyy kokonaisuudessaan vastauksineen. Samoin ulkoasu on samanlainen kuin tentissä. Selausnäkymästä on myös mahdollisuus valita kysymys ja siirtyä muokkaamaan sitä editointinäkymään. Uuden tai muokatun kysymyksen tallennus tapahtuu editorinäkymässä ja tallennuksen jälkeen siirrytään automaattisesti selausnäkymään jossa nyt näkyy myös juuri tallennettu kysymys.

Kysymyseditori on periytetty JApplet-luokasta ja sillä on lapsikomponenttina JPanel-luokasta periytetty PohjaPaneeli-olio, jonka sijoittelumanagerina on CardLayout. Tämän pohjapaneelin päällä ovat varsinaiset näkyvillä olevat paneelit:

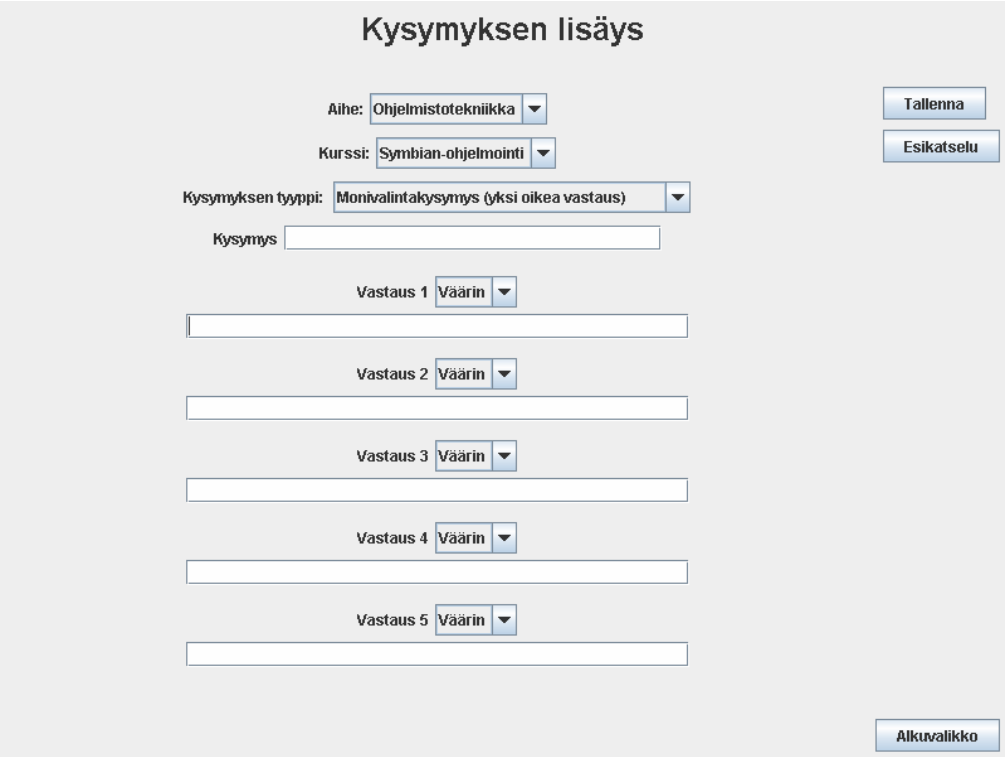
- EditoriPaneeli: sisältää kysymysten editoritoiminnot
- HakuPaneeli: kysymysten selaus
- AlkuPaneeli: sisältää päävalikon
- EsikatseluPaneeli: kysymysten esikatselu ja uusien kysymysten tallennus

Paneeleiden näkyvyyttä kontrolloidaan pohjapaneelin metodeilla, joita kutsutaan muiden paneelien tapahtumankäsittelijöistä. Tämä kuvataan tarkemmin jokaisen paneelin yhteydessä. Koska kysymyseditori ja tenttieditori ovat täysin erillisiä ja tenttieditorissa tarvitaan kuitenkin kysymyksen haku- ja muokkaustoimintoja, on hakupaneeliin ja editoripaneeliin toteutettu myös tenttieditorin vaatimia toimintoja ja kyseiset luokat ovat käytössä myös tenttieditorissa.

4.1 Editorinäkymä

Kuva 9 esittää editorinäkymän sellaisena kuin, se näkyy käyttäjän monitorin ruudulla. Käyttäjä valitsee pudotusvalikoista aiheen ja kurssin, jonka piiriin kysymys kuuluu. Sitten käyttäjä valitsee kysymyksen tyypin, joita olivat esseekysymys, monivalintaky-

symys yhdellä oikealla vastauksella ja monivalintakysymys useammalla mahdollisella oikealla vastauksella. Kysymys-kenttään kirjoitetaan varsinainen kysymys ja vastauskenttiin vastausvaihtoehdot. Jos kysymyksen tyypiksi on valittu esseekysymys, vastauskentät eivät ole näkyvissä. Jokaisen vastauskentän yläpuolella olevalla pudotusvalikolla vastaus asetetaan joko oikeaksi tai vääräksi. Mikäli käyttäjä valitsi kuvan mukaisesti monivalintakysymyksen yhdellä oikealla vastauksella, sovellus antaa asettaa kerrallaan vain yhden vaihtoehdoista oikeaksi. Esikatselu-painikkeella siirrytään tarkastelemaan kysymystä opiskelijan näkemässä ulkoasussa. Tallenna-painikkeella kysymys talletetaan tietokantaan. Alkuvalikko-painikkeen painaminen palauttaa luonnollisesti näkymän takaisin aloitusvalikkoon.



Kuva 9. Editorinäkymä

Näkymä on koottu viidestä eri paneelista, joista yksi on pohjimmainen ja muut neljä on sijoiteltu sen päälle BorderLayout-sijoittelumanagerin avulla. Yläreunan paneelissa ei ole otsikkoa lukuun ottamatta muita komponentteja. Keskipaneeliin on sijoitettu aiheen, kurssin ja kysymystyypin valintaan tarkoitetut pudotusvalikot ja niiden nimitekstit sekä kysymyksen syöttöön varattu tekstikenttä. Monivalintakysymyksien vastauksien syöttöön tarkoitetut tekstikentät sekä pudotusvalikot on sijoitettu erilliseen EditoriPaneeliRivi-luokkaan, joka sisältää yhden pudotusvalikon, ohjetekstin ja yhden tekstikentän. Näitä alustetaan silmukassa viisi kappaletta, jotta saadaan kuvan 9 mu-

kainen näkymä. Tallenna ja esikatselu-painikkeet ovat omassa paneelissaan oikealla ja alkuvalikkopainike on erillisessä paneelissa näytön alareunassa.

Toiminta

Kaikkien olioiden varsinaisen toiminnallisuuden luovat metodit, jotka vastaavat perinteisemmän ohjelmoinnin funktioita. Editoripaneeli sisältää seuraavat metodit:

- rakentaja: luo tyhjän editoripaneelin uuden kysymyksen luomiseksi
- parametrillinen rakentaja: saa parametreinaan valmiin kysymyksen kaikki tiedot ja kutsuu metodia, joka täyttää nämä tiedot näkyville
- init: huolehtii näkymän varsinaisesta alustuksesta, eli sijoittelee tekstikentät ja painikkeet paneeleihin ja asettaa paneelien sijoittelumanagerit. Kumpikin rakentaja-metodi kutsuu tätä init-metodia.
- alustaAiheet: hakee tietokannasta aiheet ja täyttää ne pudotusvalikkoon
- alustaKurssit: lukee ylimmästä pudotusvalikosta aiheen ja hakee tietokannasta kyseisen aiheen alla olevat kurssit sekä sijoittaa ne toiseen pudotusvalikkoon
- alustaEditoriMuokattavallaKysymyksella: saa parametreinaan aiheen, kurssin ja kysymyksen, ja täyttää ne näkymän kenttiin kysymyksen muokkausta varten. Tätä metodia kutsutaan aina kun jo olemassa olevaa kysymystä halutaan muokata.
- clearKentät: tyhjentää kysymyksen tallennuksen jälkeen kaikki editorin kentät uuden kysymyksen tekoa varten

Näiden metodien lisäksi editoripaneeliluokassa on sisäinen tapahtuman kuuntelijaluokka, joka reagoi painikkeiden painamiseen ja pudotusvalikkojen käsittelyyn. Kun käyttäjä tekee jonkun toiminnan tapahtumankäsittelijä tarkistaa ensin, mikä olio aiheutti tapahtuman ja valitsee jatkotoimenpiteet sen mukaisesti.

Seuraavassa luettelossa tapahtuman aiheuttava komponentti ja tapahtumaa seuraavat toimenpiteet.

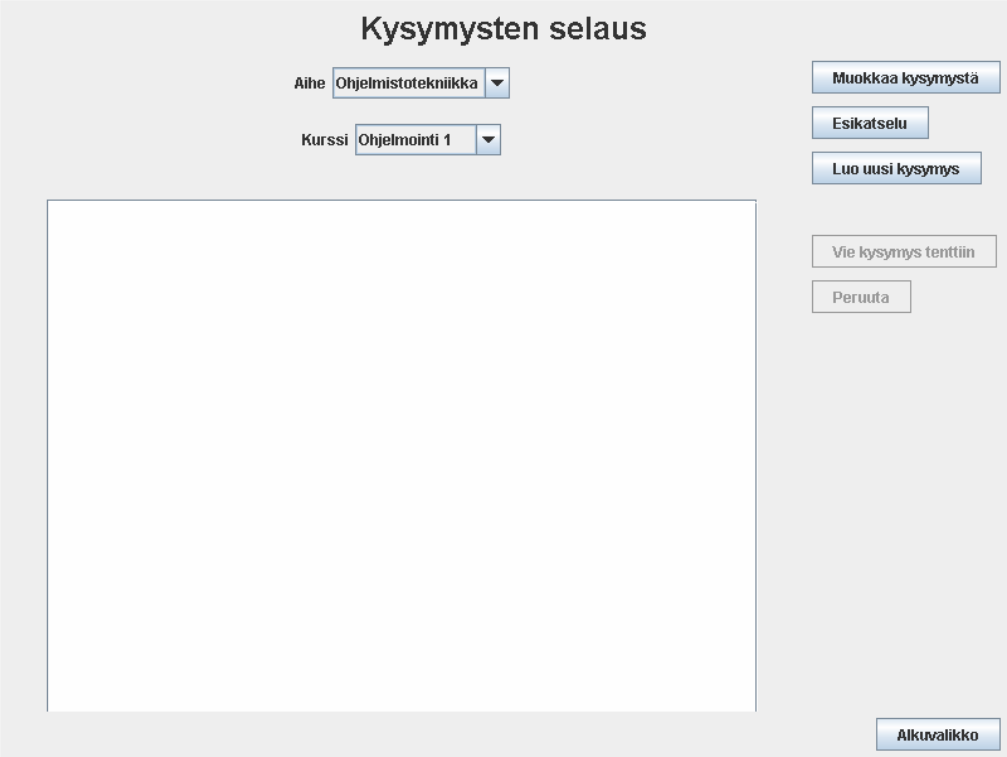
- aihepudotusvalikko: kutsutaan alustaKurssit-metodia.
- kysymyksen tyyppi-pudotusvalikko: luetaan valittu kysymystyyppi. Jos se on esseekysymys, piilotetaan vastauskentät näkyvistä.

- oikein-väärin pudotusvalikko: jos kysymyksen tyyppi on monivalinta yhdellä oikealla vastauksella, tarkistetaan, että vain yksi vastauksista on asetettu oikeaksi ja estetään useamman vastausvaihtoehdon asettaminen oikeaksi.
- esikatselupainike: luetaan kysymys- ja vastauskentistä tiedot muistiin ja kutsutaan esikatselupaneelin metodia, joka asettaa ne näkyville. Sen jälkeen kutsutaan editoripaneelin yläluokan metodia, joka laittaa editoripaneelin piiloon ja asettaa esikatselupaneelin käyttäjän näkymään.
- tallennuspainike: tallennetaan muistissa olevan kysymyksen tiedot tietokantaan, kutsutaan clearKentat-metodia ja yläluokan siirrytään hakupaneeliin, jossa talletettu kysymys näkyy.
- alkuvalikkopainike: kutsutaan yläluokan metodia, joka asettaa alkuvalikkopaneelin käyttäjän näkymään.

4.2 Selausnäkymä

Kuva 10 esittää selausnäkymän sellaisena, kuin se näkyy käyttäjän monitorin ruudulla. Käyttäjä valitsee pudotusvalikoista aiheen ja kurssin, jonka kysymyksiä hän haluaa tarkastella. Kysymykset tulostetaan keskellä olevaan JList-komponenttiin. Esikatselupainikkeella siirrytään tarkastelemaan kysymystä opiskelijan näkemässä ulkoasussa. Muokkaa kysymystä -painikkeella valittu kysymys kopioidaan editoriin ja siirrytään editorinäkymään. Luo uusi kysymys -painikkeella siirrytään myös editorinäkymään. Alkuvalikko-painikkeen painaminen siirtää näkymän alkuvalikkoon.

Selausnäkymässä pohjimmaisena on JPanel-luokasta periytetty HakuPaneeli-olio, jonka sijoittelumanagerina on BorderLayout. Hakupaneelin lapsikomponentteina on neljä JPanel-luokan oliota. Otsikko sijaitsee näytön ylälaidassa omassa paneelissaan. Keskelle olevaan paneeliin on sijoitettu aiheen ja kurssin valintaan tarkoitetut pudotusvalikot sekä JList-komponentti, johon kysymykset tulostetaan. Esikatselu-, muokkaus- ja luo uusi kysymys -painikkeet sekä himmeällä kuvatut, tenttieditorin käytössä olevat painikkeet ovat omassa paneelissaan oikealla. Todellisuudessa tenttieditorin painikkeet eivät ole näkyvissä muuten kuin tenttieditorista selausnäkymään tultaessa. Tähän kuvaan painikkeet on otettu puhtaasti esittelytarkoituksessa. Alkuvalikkopainike on erillisessä paneelissa näytön alareunassa.



Kuva 10. Kysymysten selausnäkymä

Toiminta

Hakupaneeli sisältää seuraavat metodit:

- rakentaja luo hakupaneelin normaalitilaan
- parametrillinen rakentaja saa parametreinaan aiheen ja kurssin ja tällöin voidaan tarkastella ainoastaan kyseisen kurssin kysymyksiä. Tätä rakentajaa käytetään tenttieditorin puolella.
- init huolehtii näkymän varsinaisesta alustamisesta eli sijoittelee pudotusvalikot, listan ja painikkeet paneeleihin ja asettaa paneelien sijoittelumanagerit. Kumpikin rakentaja-metodi kutsuu tätä init-metodia.
- alustaAiheet hakee tietokannasta aiheet ja täyttää ne pudotusvalikkoon
- alustaKurssit lukee ylimmästä pudotusvalikosta aiheen ja hakee tietokannasta kyseisen aiheen alla olevat kurssit sekä sijoittaa ne toiseen pudotusvalikkoon
- haeKysymykset lukee alemmasta pudotusvalikosta kurssin ja hakee tietokannasta kyseisen kurssin kysymykset sekä tulostaa ne JList-komponenttiin.

Näiden metodien lisäksi hakupaneeliluokassa on sisäinen tapahtuman kuuntelija -luokka, joka reagoi painikkeiden painamiseen ja pudotusvalikkojen käsittelyyn. Seuraavassa esitellään tapahtuman aiheuttanut komponentti ja siitä seuraavat toimenpiteet.

- aihepudotusvalikko: kutsutaan alustaKurssit-metodia.
- kurssipudotusvalikko: kutsutaan haeKysymykset-metodia.
- esikatselupainike: luetaan JList-komponentista valittu kysymys ja kutsutaan esikatselupaneelin metodia, joka asettaa sen näkyville. Sen jälkeen kutsutaan hakupaneelin yläluokan metodia, joka laittaa hakupaneelin piiloon ja asettaa esikatselupaneelin käyttäjän näkymään.
- muokkaa kysymystä -painike: luetaan JList-komponentista valittu kysymys ja kutsutaan editoripaneelin metodia, joka täyttää kysymyksen editorin kenttiin. Sen jälkeen kutsutaan hakupaneelin yläluokan metodia, joka laittaa hakupaneelin piiloon ja asettaa editoripaneelin käyttäjän näkymään.
- luo uusi kysymys -painike: kutsuu yläluokan metodia, joka asettaa editoripaneelin käyttäjän näkymään.
- alkuvalikkopainike: kutsutaan yläluokan metodia, joka asettaa alkuvalikkopaneelin käyttäjän näkymään.

Seuraavat tapahtuman käsittelijät liittyvät tenttieditorin painikkeisiin.

- Vie kysymys tenttiin: lukee listakomponentista valitun kysymyksen ja vie sen tenttieditoriin sekä asettaa näkymän tenttieditoriin
- Peruuta painike: Palauttaa näkymän takaisin tenttieditoriin ilman muita toimenpiteitä.

4.3 Esikatselunäkymä

Kuva 11 esittää kysymyksen esikatselun sellaisena, kuin se näkyy käyttäjän monitorin ruudulla. Sulje painiketta painamalla palataan takaisin joko editoriin tai selausnäkymään sen mukaan, kummasta esikatseluun tultiin. Esikatselunäkymä on jaettu kahteen eri paneeliin. Yläreunan paneelissa on pelkkä sulje painike, jolla palataan takaisin edelliseen näkymään. Keskellä olevassa paneelissa on otsikko ja JList-komponentti, johon kysymys vastauksineen tulostetaan.

Toiminta

Esikatselupaneeli sisältää seuraavat metodit:

- rakentaja alustaa esikatselupaneelin kysymystä lukuun ottamatta.
- alustaEsikatselu lisää esikatselupaneeliin kysymyksen vastauksineen sekä tallettaa tiedon mistä paneelista kysymys tuotiin.

Sulje nappia painettaessa esikatselupaneelin tapahtumankäsittelijä tarkistaa, mistä näkymästä esikatseluun tultiin ja kutsuu yläluokan metodia, joka vaihtaa kyseisen näkymän päällimmäiseksi.

Kuva 11. Esikatselunäkymä

4.4 Alkuvalikko

Kuva 12 esittää kysymyseditorin päävalikkoa. Kuvasta on jätetty pois näytön alareunaan jäävä tyhjä tila. Käyttäjä valitsee radio-napeista joko uuden kysymyksen teon tai tietokannan kysymyksien selauksen ja jatka napin painalluksella näkymä vaihtuu valituksi.

Kuva 12. Kysymyseditorin päävalikko

Valikkonäkymä koostuu esikatselunäkymän lailla kahdesta eri paneelista. Ylemmässä on otsikko ja alemmassa radionapit ja jatka-painike.

Toiminta

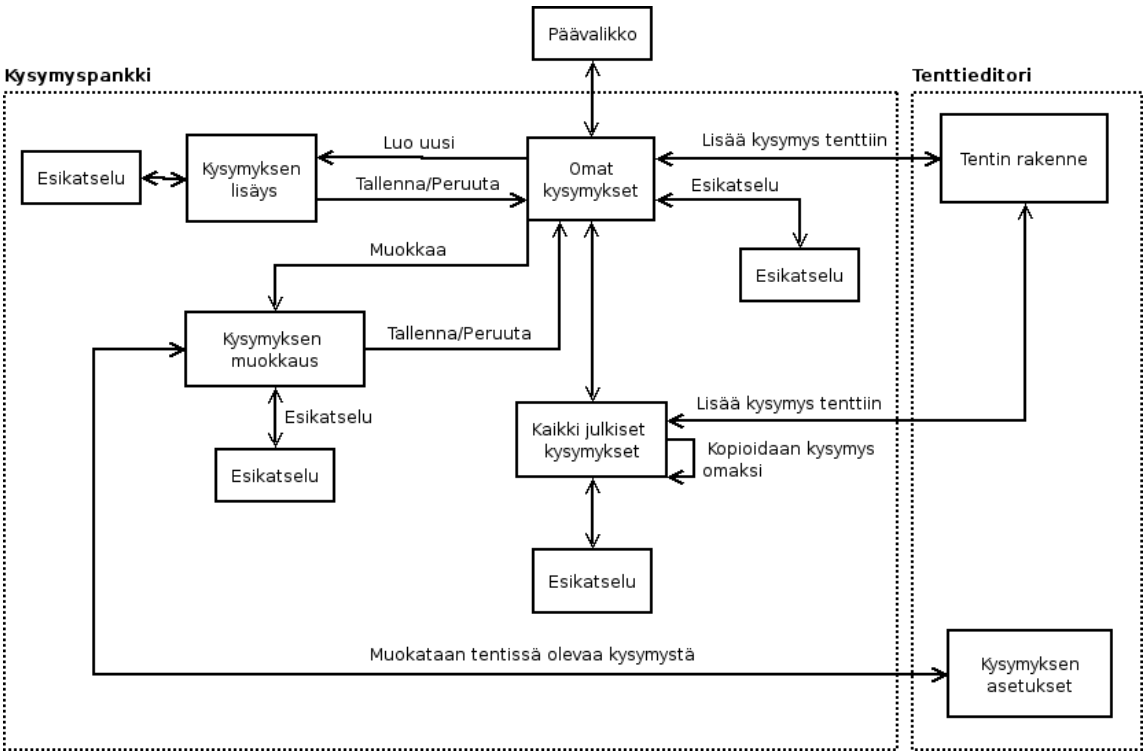
Valikkopaneeli sisältää ainoastaan rakentaja-metodin, joka alustaa valikkopaneelin, sekä tapahtumankäsittelijä-luokan. Kun jatka-painiketta painetaan, tapahtumankäsittelijä tarkistaa, kumpi radionapeista oli valittuna ja kutsuu vastaavaa yläluokan metodia.

5 TENTTISOVELLUKSEN NYKYINEN VERSIO

Tässä luvussa tarkastellaan tenttisovelluksen syksyn 2005 versiota. Alusta asti sovelluksen toimintaa suunniteltaessa päänvaivaa aiheutti kysymysten muokkausmahdollisuus sen jälkeen, kun kysymys on jo käytössä jonkun toisen käyttäjän tentissä. Ensin tähän suunniteltiin erilaisia ajastimia, jotka asettavat kysymyksen muokattavaksi sen jälkeen kun tentti on pidetty, mutta nyt ongelma on ratkaistu yksinkertaisemmin. Kaikki kysymykset ovat tallessa kysymyspankissa ja ne luokitellaan julkisiksi, eli ne ovat kenen tahansa käyttäjän saatavilla. Kun käyttäjä valitsee tenttiinsä pankista kysymyksen, siitä tehdään hänelle kopio ja talletetaan käyttäjän omiin kysymyksiin. Nämä omat kysymykset ovat ainoastaan käyttäjän itsensä näkyvissä ja muokattavissa. Samanlainen järjestely on tehty tenttien kanssa. Tenttipankki sisältää julkiset tentit, joita käyttäjät voivat kopioida omiin tentteihinsä. Omia tenttejä voi muokata ja ne ovat käyttäjän yksityisessä käytössä. Kun käyttäjä tekee itselleen uuden kysymyksen tai tentin, hän voi halutessaan julkaista sen, jolloin siitä tehdään kopio kysymys- tai tenttipankkiin.

5.1 Kysymyspankki

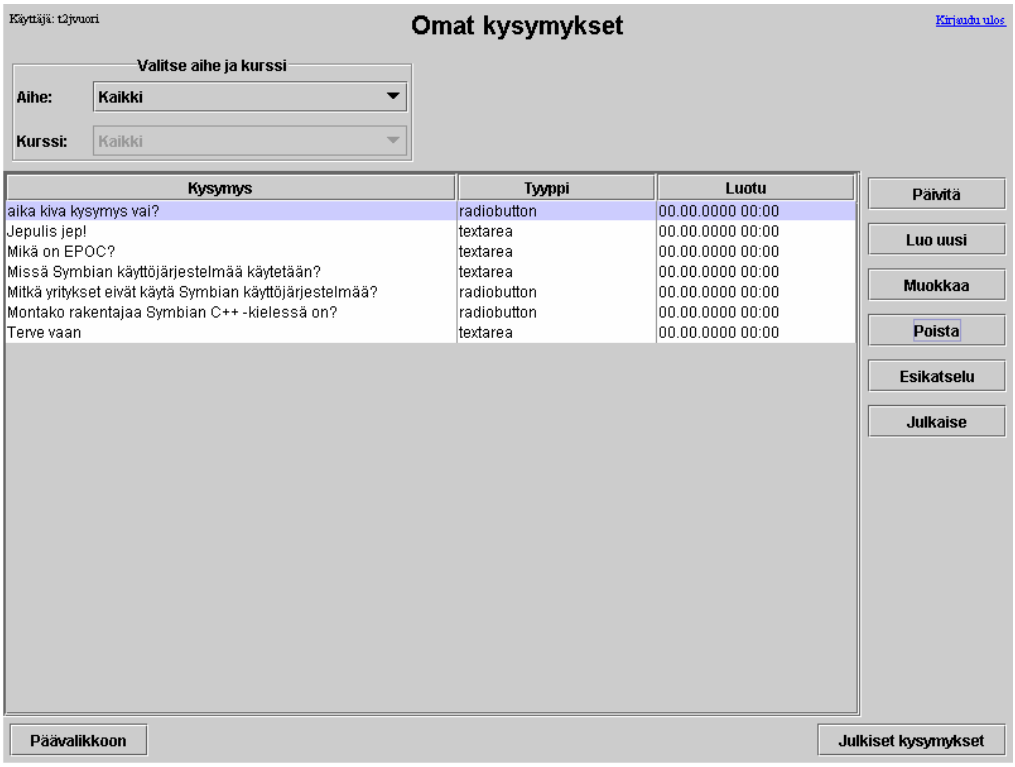
Kuvassa 13 näkyy kysymyspankin lohkokaavio. Päävalikosta tullaan näyttöön, jossa näkyvät käyttäjän omat kysymykset. Käyttäjä voi luoda uusia kysymyksiä, muokata jo olemassa olevia kysymyksiä tai siirtyä tarkastelemaan julkisia kysymyksiä. Julkisia kysymyksiä voi kopioida, jolloin niistä tulee omia kysymyksiä. Jos omia kysymyksiä on tultu selaamaan tenttieditorista, niin valitun kysymyksen voi viedä takaisin tenttieditoriin.



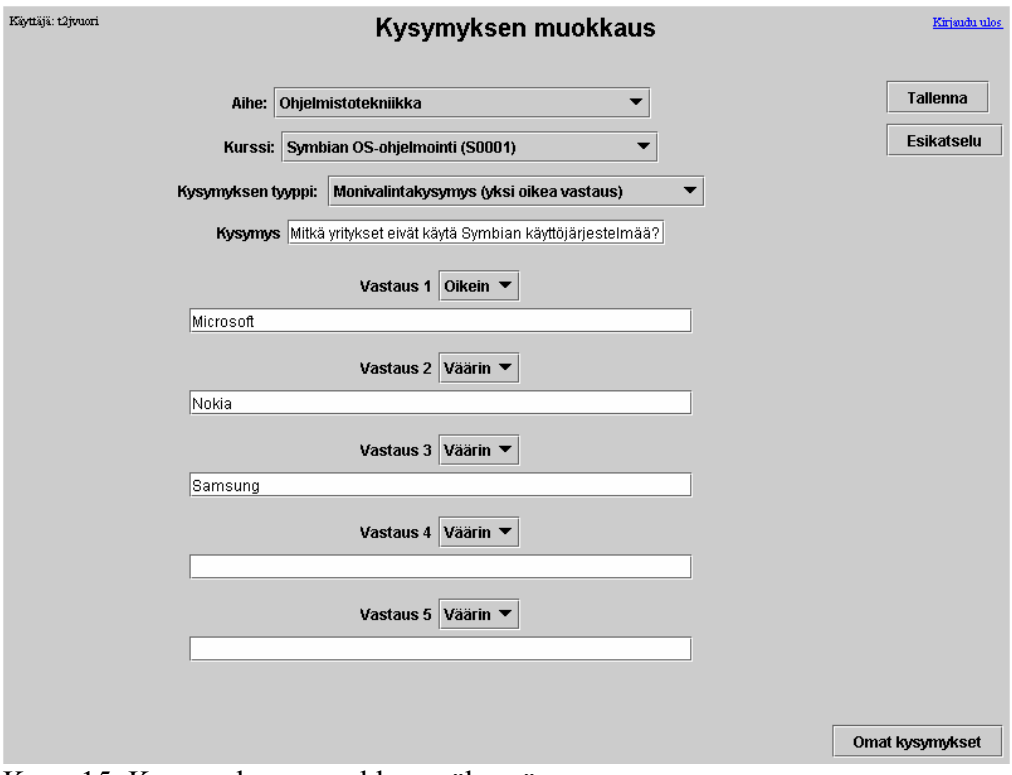
Kuva 13. Kysymyspankki /3/

Kuvassa 14 näkyy omien kysymysten selausnäkymä. Esitettävän tiedon määrä on selkeästi kasvanut vanhaan versioon nähden. Kysymyksistä näkyvät selausnäkymässä suoraan myös tyyppi ja luontihetki. Kysymyksiä voi tarkastella joko tietyn aiheen ja kurssin alta tai hakea näyttöön kerralla kaikki omat kysymykset. Uloskirjautumismahdollisuus on tuotu kaikkiin näkymiin, samoin oma käyttäjätunnus on näkyvillä näytön vasemmassa yläkulmassa. Toimintopainikkeet ovat edelleen oikeassa laidassa ja siirtymispainikkeet näytön alareunassa. Kuvasta se ei tietenkään näy, mutta kaksoisklikkaamalla kysymystä se siirretään esikatseluun, mikä nopeuttaa kysymysten selausta.

Kuvassa 15 näkyy kysymysten muokkausnäkymä, joka on kokenut selvästi vähemmän muutoksia verrattuna vanhempaan versioon. Ainoastaan uloskirjautumismahdollisuus ja oman käyttäjätunnuksen näkyminen on lisätty näkymään.

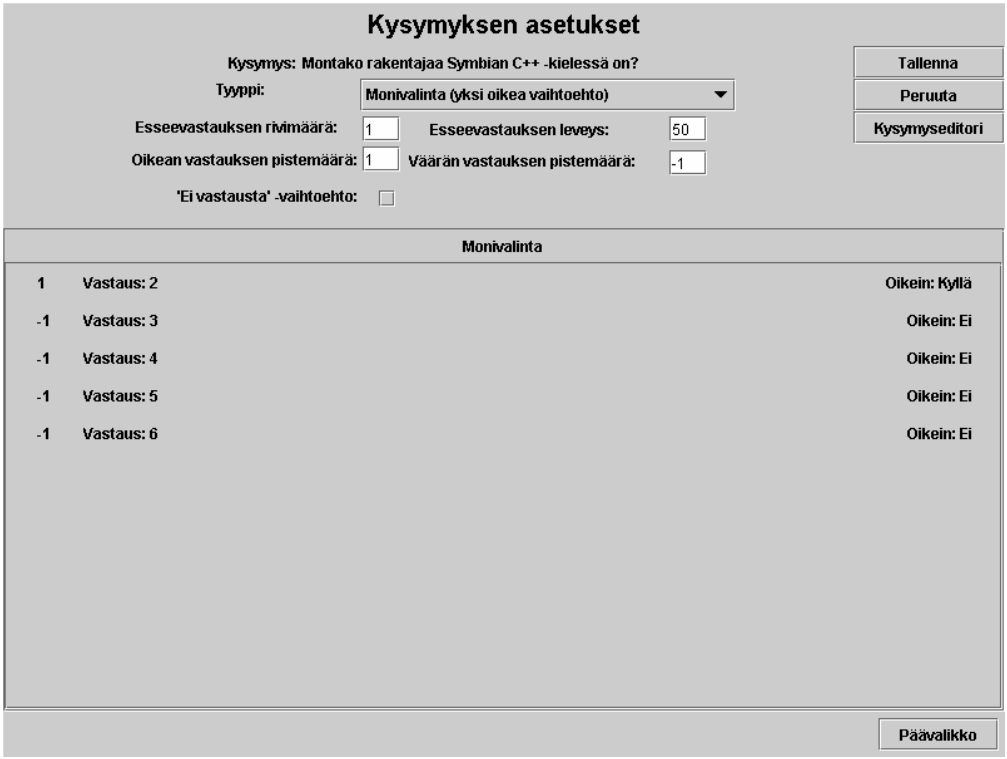


Kuva 14. Omien kysymysten selausnäkymä



Kuva 15. Kysymyksen muokkausnäkymä

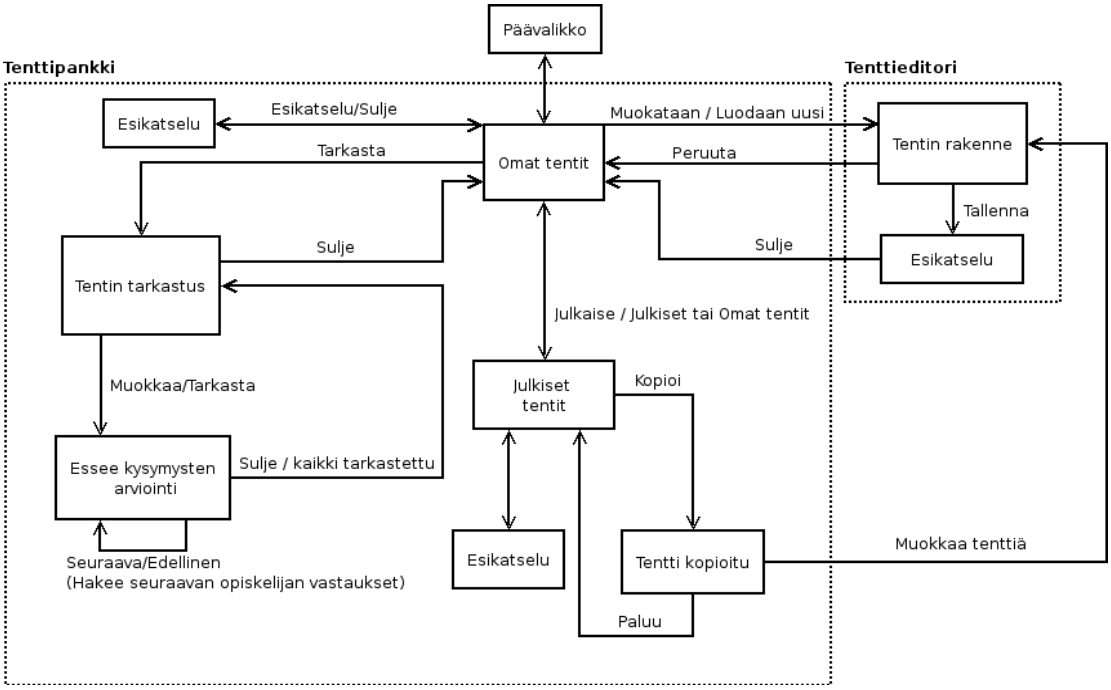
Kuvassa 16 näkyy uusi näkymä, kysymyksen asetukset. Nämä asetukset koskevat kysymystä, joka on jo osa jotakin tenttiä. Oikeat ja väärät vastaukset voi pisteyttää erikseen. Esseekysymyksen vastauskentän koon voi määritellä ja lisätä kysymykseen en tiedä/ei vastausta vastausvaihtoehdon, josta ei saa miinuspisteitä. Näkymästä on siirtymismahdollisuus editoriin, jos haluaa vielä viime hetkellä muokata itse kysymystä tai vastausvaihtoehtoja.



Kuva 16. Kysymyksen asetukset

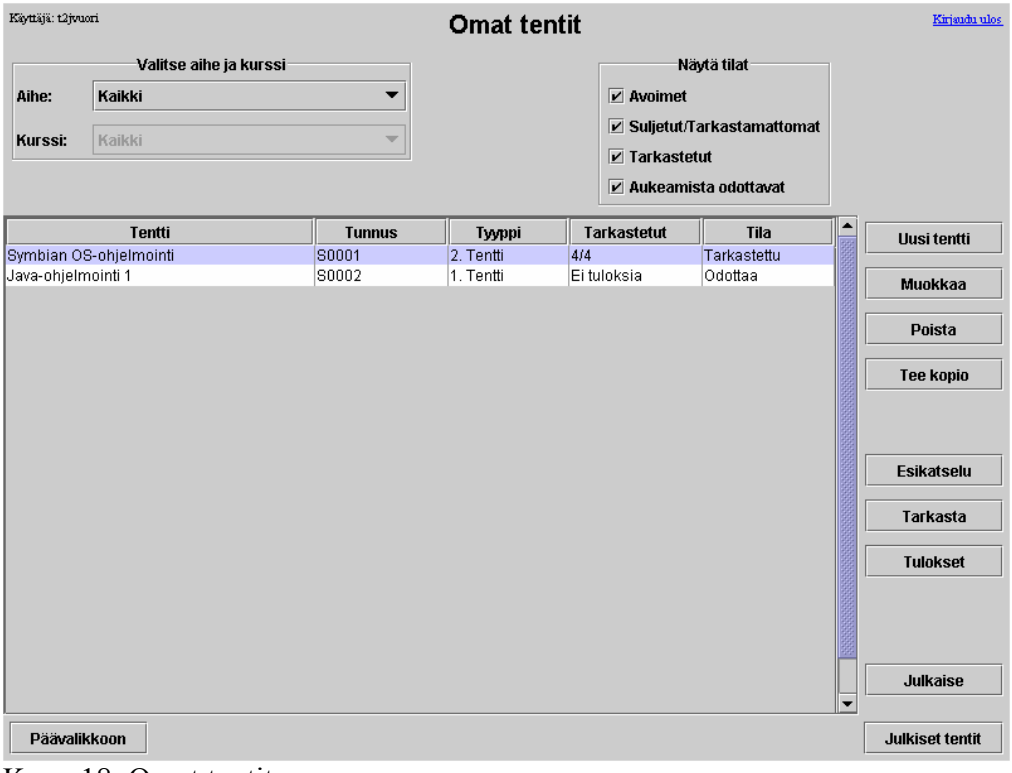
5.2 Tenttipankki

Kuvassa 17 näkyy tenttipankin lohkokaavio. Päävalikosta tullaan näyttöön jossa näkyvät käyttäjän omat tentit. Tenttipankissa käyttäjällä on mahdollisuus tehdä tenteille kaikki samat toimenpiteet mitä kysymyspankissa kysymyksille. Ainoa merkittävä ero on tenttien tarkastus. Kun tentti on pidetty ja opiskelijoiden vastaukset on talletettu tietokantaan, opettaja voi tarkastaa tentin. Monivalintakysymykset järjestelmä tarkistaa automaattisesti ja opettajan tehtäväksi jää esseekysymyksien pisteytys. Tentit voi tarkistaa yksi kerrallaan, järjestelmällä on tallessa tieto, kuinka monen opiskelijan tentit on tarkistettu ja kuinka monta tenttiä on tarkastamatta.



Kuva 17. Tenttipankin lohkokaavio /3/

Kuvassa 18 näkyy omien tenttien selausnäkymä. Selkein ero kysymyksiin verrattuna on tenttien tilojen näyttö. Tenttiä luotaessa sille on määritelty aukioloaika, jolloin opiskelijat voivat käydä tekemässä kyseisen tentin. Ennen tätä aukioloaikaa tentti odotustilassa. Aukioloajan jälkeen tentti on suljettu ja, kun opettaja on tarkastanut kaikki kyseisen tentin tehneiden opiskelijoiden suoritukset, tentti siirtyy tarkastettu tilaan.



Kuva 18. Omat tentit

Kuvassa 19 näkyy tenttien muokkausnäkymä. Otsikon alla näkyy aihe ja kurssi, jolle kyseinen tentti on tehty. Samoin siinä näkyy tentin järjestysnumero ja aukioloaika, jotka esimerkkikuvassa ovat melko pitkät. Keskellä on listattu tentin kysymykset samassa järjestyksessä, kuin ne ovat itse tentissäkin. Järjestystä voi helposti vaihtaa kysymyksen edessä olevilla nuolipainikkeilla. Kysymyksen perässä oleva asetukset-painike vie käyttäjän näkymään, jossa säädetään kysymyskohtaisia asetuksia kuten oikeiden ja väärien vastauksien pisteytys. Tämä näkymä esiteltiin kuvassa 16. Poista-painikkeilla voidaan poistaa kyseisellä rivillä oleva kysymys. Kysymyksien alla olevalla lisää kysymys -painikkeella lisätään tenttiin uusi kysymys. Vasemmassa alakulmassa olevilla omat tentit ja esikatselupainikkeilla siirrytään nimien mukaisesti tarkastelemaan joko omia valmiita tenttejä tai esikatselemaan kyseistä työn alla olevaa tenttiä. Tentin asetukset-painikkeen takana on muun muassa tentin aukioloaika, tentin aihe, kurssi ja nimitiedot. Kysymysasetukset -painikkeen takaa voidaan asettaa kaikkien kysymysten asetukset johonkin oletusarvoon, ja sen jälkeen asetella yksittäisten kysymysten asetuksia oletusarvoista poikkeaviksi. Tallenna-painikkeella tentti talletetaan tietokantaan käyttäjän omiin tentteihin. Jos käyttäjä yrittää poistua näkymästä tallentamatta tenttiä, tästä kysytään käyttäjältä varmistus.

Kuva 19. Tenttien muokkausnäkymä

5.3 Kehityskohteet

Kuvien liittäminen osaksi kysymystä on merkittävä tenttisovelluksen kehityskohde. Jotta sovelluksella voi tehdä esimerkiksi mielekkäitä matematiikan tenttejä, täytyy kysymyksiin voida joko liittää kuvia tai vaihtoehtoisesti lisätä sovellukseen kaavaeditori. Kuvien lisäysmahdollisuus tuo mahdollisuuksia myös ei-matemaattisten aineiden tenttien tekemiseen.

Toinen merkittävä kehityskohde liittyy sovelluksen helppokäyttöisyyteen. Kaikkiin sovelluksen näkymiin tulisi lisätä käyttäjää opastavaa ohjeistusta. Vaikka sovelluksen ulkonäkö onkin melko yksinkertainen verrattuna moniin muihin ohjelmiin, ei toimintojen hahmottaminen ole ensimmäisiä käyttökertoja kokeilevalle mikään itsestäänselvyys.

Myös tenttisovelluksen visuaalinen ilme tarvitsee kehitystä. Luvussa 3.1 kerrottiin värien käytöstä graafisen käyttöliittymän osana. Vihreä ja punainen osana järjestelmän signaalia välittävät sen käyttäjälle tehokkaammin kuin pelkkä harmaa. Reunuksilla saadaan koottua samankaltaiset toiminnot selkeämmin yhteen. Joissakin tenttisovelluksen näkymissä näin on jo tehty, esimerkiksi kuvan 18 näkymässä tenttien tilat on ympäröity reunuksella ja erotettu näin muista toiminnoista. Sama käytäntö tulisi ulottaa kaikkiin tenttisovelluksen näkymiin.

6 YHTEENVETO

Tämän työn tekijän vastuualueena tenttisovelluksessa oli kysymyseditori-osion koodaaminen. Tavoitteena oli luoda huhtikuun 2005 loppuun mennessä Java-apletti, jolla pystyy luomaan kysymyksiä ja tallettamaan ne tietokantaan sekä selamaan tietokannassa olevia kysymyksiä. Selausnäkymästä piti myös voida valita kysymys ja siirtyä editorinäkymään muokkaamaan siitä uusi kysymys. Nämä toiminnot olivat tärkeitä kysymyseditorin esittelyn kannalta ja välttämättömiä tenttieditorin esittelyn kannalta. Nämä tavoitteet saavutettiin.

Suurimmat ongelmat muodostuivat tietokantayhteyden kanssa. Kysymyseditori käytti tenttieditorin kanssa yhteisiä ohjelmamoduuleita tietokantayhteyden toteutukseen. Tietokantaan jouduttiin tekemään muutoksia vielä ohjelmalohkojen tekovaiheessa. Koska koodaus tehtiin aluksi lähinnä etätyönä, tuli tilanteita, jossa kysymyseditori ei ollut yhteensopiva tietokantayhteysmoduulien kanssa. Ongelma ratkaistiin ohjelmiston viimeistelyvaiheessa tekemällä kaikki koodaustyö rakennusosaston projektia varten varatuissa tiloissa, yhdessä sovittuina ajankohtina, jolloin kysymys- ja tenttieditori saatiin pidettyä yhteensopivina muutoksista huolimatta. Laajemmissa projekteissa jonkinlainen versionhallintaohjelmisto on kuitenkin välttämätön, jotta teon alla olevasta ohjelmistosta on aina saatavilla myös toimiva versio.

Kysymyseditorin jatkokehitystarpeet eivät eroa koko tenttisovelluksen kehitystarpeista. Kysymykseen pitäisi voida lisätä kuva, joko osaksi kysymystä tai vastausvaihtoehdoksi. Käytön aikaisia ohjeita pitäisi myös lisätä.

LÄHDELUETTELO

Painetut lähteet

1 Peltomäki, Juha - Silander, Simo, Java 2 ohjelmoinnin peruskirja. Docendo Finland Oy, Porvoo 2003.

2 Metsämäki, Markku: Graafisen käyttöliittymän suunnittelu. Oy Edita Ab (Painatuskeskus Oy), Helsinki 1998.

Painamattomat lähteet

3 Vuorinen, Jyri, työn kuvat 1, 6, 13 ja 17.