Tallinna Ülikool
Matemaatika- Loodusteaduskond
Informaatika osakond

Bakalaureusetöö

Rain Viigipuu

Statistikamoodul elektroonilisele tagasisideküsitluse süsteemile

Juhendajad: Katrin Niglas
Jaagup Kippar

Autor: ..... "....." ..... 2007.a.
Juhendaja: ..... "....." ..... 2007.a.
Juhendaja: ..... "....." ..... 2007.a.
Osakonnajuhataja: ..... "....." ..... 2007.a.

Tallinn 2007

Sisukord

Lühendid

- **CSV** – Comma Separated Values. Failiformaat, mis võimaldab tekstifailis hoida andmeid tabeli kujul – määratakse ära väärtus, mis tabeli väljasid (tulpasid) eraldab (selleks võib olla näiteks koma, semikoolon või tabeldusmärk). Iga tabeli rida vastab reale tekstifailis.
- **XML** – Extensible Markup Language. W3C standard võimaldamaks luua märgenduskeeli kindla struktuuriga andmete hoidmiseks failis.
- **PHP** – PHP: Hypertext Preprocessor. Populaarne dünaamiliste veebilehtede loomiseks mõeldud skriptimiskeel.
- **TTF** – TrueType Font. Vektorkirjatüübi standard, mis loodi Apple Computers-i poolt 1980-ndate aastate lõpus (Wikipedia, „TrueType“).
- **HTML** – HyperText Markup Language. Märgenduskeel mis on loodud veebilehtede tegemiseks.
- **MVC** – Model-View-Controller. Kolmekihilise tarkvaraarhitektuuri üks nimetusi. Kirjeldab ära erinevate kihtide ülesanded ja seosed kihtide vahel.
- **DOS** – Disk Operating System. Operatsioonisüsteem, mis võimaldab kasutada väliseid mäluseadmeid (mäluseadmed, millele arvuti protsessoril ei ole otsest ligipääsu) ja andmeid sellistest seadmeilt (Wikipedia, „Disk Operating System“).

Sissejuhatus

2003. aasta kevadel kaitstud lõputöö (Ott, 2003) raames loodi elektrooniline tagasisideküsitluse süsteem, mille eesmärgiks on lihtsustada tudengite käest tagasiside saamist õpitud ainete ja õppejõudude kohta. Kui varemalt viidi küsitlusi läbi kasutades paberile trükitud ankeete ja hilisemaks elektrooniliseks andmete analüüsiks tuli tulemused käsitsi arvutisse sisestada, siis nüüd on küsitluste tulemused kohe elektroonilisel kujul olemas. Oti loodud süsteem võimaldab küll küsitlusi hallata ja läbi viia, kuid täielikult puudub küsitluste käigus kogutud andmete väljastamise, analüüsimise ja kokkuvõtete tegemise osa. Süsteemi kasutusele võtmiseks on see aga väga oluline.

2004/2005 aasta talvel võeti Tallinna Ülikoolis kasutusele tagasisideküsitluste süsteem, mille eesmärk on sama, mis Oti loodud süsteemil, kuid lisaks eelmisele on olemas võimalus vaadata küsitluse tulemustest kokkuvõtteid ja teha lihtsamat statistikat.

Käesoleva töö eesmärgiks oli esialgu luua Oti tagasisideküsitluse süsteemile puuduv statistika osa. Seoses Tallinna Ülikooli süsteemi loomisega tekkis aga mõte proovida teha statistika osa selliselt, et seda oleks võimalik kasutada ka mõne teise tagasisideküsitluse süsteemi abil kogutud andmete analüüsimiseks.

Laias laastus jaguneb töö kaheks suuremaks osaks:

- Elektrooniliste tagasisideküsitluste süsteemide uurimine ja analüüs – millistest elementidest (nt. küsimus, aine) küsitluste süsteemid koosnevad, kuidas ja milliseid andmeid nende elementide kohta meeles peetakse, ning hinnata, millisel määral erinevad tagasisideküsitluste süsteemid teineteisest erineda võivad. Materjalidena kasutan põhiliselt Martin Oti lõputöö raames loodud tagasisideküsitluste süsteemi, kuna see on vabalt kättesaadav ning seda on võimalik ise katsetada. Võrdluseks kasutan Tallinna Ülikoolis kasutusel olevat küsitluste süsteemi millest on olemas süsteemi loomisel aluseks olnud ja arenduse käigus loodud dokumendid. Ühtlasi on selles osas kogutud informatsioon rakenduse loomise aluseks, kuna selle põhjal tuleb välja millised andmed statistikamooduli arenduseks kasutada on.
- Rakenduse loomine, mille abil on võimalik küsitluste käigus kogutud andmetest teha kokkuvõtteid, arvutada lihtsamaid statistikuid, neid võrrelda ja nende põhjal graafikuid

koostada. Eesmärk on võimaldada võimalikult vähese vaevaga saada ülevaadet küsitluse käigus kogutud andmetest, teha nende andmete põhjal kokkuvõtteid ja lihtsamat statistikat. Töö teine osa sisaldab ka ülevaadet rakenduse loomisel kasutatud vahenditest ja meetotitest.

Lõputöö teoreetilise osa üheks eesmärgiks on kahe tagasisideküsitluste süsteemide andmebaaside analüüsimise teel teha selgeks, kas teoreetiliselt on võimalik luua rakendus, mille jaoks ei ole oluline, millise tagasisideküsitluste läbiviimise süsteemi abil analüüsitavad andmed kogutud on. Lisaks anda ülevaade ja valida välja tarkvaraarenduse metoodika ja vahendid, mille abil realiseerida töö praktiline osa.

Praktilise osa eesmärgiks ei ole luua täisfunktsionaalset statistikamooduli rakendust. Pigem keskendun sellele, et paika panna põhimõtted, kuidas rakendus on ülesehitatud ja kuidas peaks toimima, ning praktikas vastavalt neile põhimõtetele luua rakenduse raamistik ja vähemalt minimaalne osa funktsionaalsusest. See, et rakenduse põhiline funktsionaalsus on juba olemas, muudab hilisemad edasiarendused ja funktsionaalsuse lisamise/muutmise oluliselt lihtsamaks, kuna rakenduse üldine raamistik on juba olemas ja lisada tuleb ainult konkreetse võimalusega seotud kood õigesse kohta.

1. Elektroonilised tagasisideküsitluste süsteemid

Järgnevalt annan üldise ülevaate kahest küsitluste läbiviimise süsteemist: Martin Oti loodud tagasisideküsitluste süsteem (Ott, 2003) ja Tallinna Ülikoolis praegu kasutusel olevast tagasisideküsitluste süsteemist. Lisaks süsteemide üldisele kirjeldusele peatun põhjalikumalt andmebaaside kirjeldamisel ja analüüsimisel, kuna statistikamoodul peab oskama neid andmeid kasutada kokkuvõtete tegemiseks ja statistikute arvutamiseks. Lisaks sellele on vaja saada ülevaade sellest, milliseid andmeid erinevad süsteemid üldse meeles peavad ning kas nendest andmetest on võimalik välja korjata kokkuvõtete tegemiseks vajalikud andmed ja need vajadusel muuta sellisele kujule, et statistikamoodul oskaks neid kasutada. See mõjutab otseselt seda, mida statistikamoodul teha võimaldab ja mida mitte.

Materjalidena olen kasutanud Martin Oti 2003. aasta kevadel kaitstud bakalaureusetööd „Elektrooniline tagasisideküsitluste süsteem“, kus on muu hulgas antud ka küllaltki põhjalik ülevaade Oti küsitluste süsteemi poolt kasutatavast andmebaasist. Tallinna Ülikoolis kasutusel oleva süsteemi kohta ülevaate kirjutamisel kasutasin selle loomise aluseks olnud projektiplaani ning dokumenti nimega „Elektrooniline tagasisideküsitlus: õpetamise ja ainekursusete hindamine“, mis sisaldab tagasisideküsitluse süsteemi kirjeldust ja andmebaasi kirjeldust.

1.1. Oti tagasisideküsitluste süsteem

Süsteem vajab tööks Apache veebiserverit koos PHP toega ning kasutab andmete hoidmiseks MySQL andmebaasiserverit. Lähemalt saab mainitud vahendite kohta lugeda käesolava töö peatükkidest 2.1.1 (PHP), 2.1.2 (MySQL andmebaasiserver) ja 2.1.3 (Apache veebiserver).

Rakendus koosneb peamiselt kahest osast: küsitluste haldamise osast ja küsitlustele vastamise osast. Haldamise osa on mõeldud küsimuste, ainete ja õppejõudude lisamiseks, muutmiseks ja kustutamiseks ning nende andmete põhjal küsitluste koostamiseks.

Selleks, et haldamise osale ligi pääseda, peab kasutajal olema kasutajanimi ja parool. Süsteemis saab kasutaja olla kas administraator ja/või asjaajaja. Asjaajaja saab teha kõike, mis on seotud küsitlustega – lisada, muuta ja kustutada küsitlusi, aineid, õppejõude ja küsimusi. Administraator saab lisaks asjaajaja tegevustele ka kasutajaid lisada, kustutada või õigusi muuta.

Küsitluse lisamisel saab lisada järgmised küsitluse üldandmed:

- Küsitluse kood – unikaalne tunnussõna, määrab ära millisele küsitlusele vastama hakatakse
- Küsitluse parool – vajalik küsitlusele vastama pääsemiseks)
- Semester – kas küsitlus on sügis- või kevadsemestri kohta
- Õppeaasta – mis aastal küsiteldav semester toimus
- Õpperühm – millisele kursusele küsitlus on koostatud
- Alguskuupäev – mis ajast alates küsitlusele on võimalik vastuseid anda
- Lõppkuupäev – mis ajani on võimalik küsitlusele vastuseid anda
- Kommentaar

Peale küsitluse üldandmete salvestamist saab küsitlusele lisada eelnevalt süsteemi lisatud ained ja küsimused. Ainete lisamisel tuleb ära määrata ka õppejõud, kes seda ainet annab (valikus on ainult need õppejõud, kes on süsteemis märgitud seda ainet andma) ja vorm (loeng, praktikum, seminar). Küsitluse seadete all saab ära märkida, millisele küsimusele vastamine on kohustuslik ja millisele mitte.

Aine lisamisel süsteemi küsitakse aine koodi ja nimetust. Peale aine salvestamist saab valida süsteemi sisestatud õppejõudude hulgast need, kes seda ainet annavad ja seostada need selle ainega.

Õppejõu lisamisel tuleb sisestada õppejõu ees- ja perekonnanimi. Peale õppejõu salvestamist saab valida millist ainet (aineid) see õppejõud annab ning seostada need õppejõuga.

Uue küsimuse lisamisel süsteemi tuleb sisestada küsimuse tekst, selgitus ning mis tüüpi küsimus on. Küsimuse tüüpideks on tekst (vastuseks saab kirjutada vabas vormis teksti), hinnang (vastuseks on hinne ette antud vahemikus) ja valikvastustega (vastusevariandid on ette antud). Valikvastustega küsimuse juures saab määrata, kas valida saab ainult ühe vastusevariandi või mitu.

Küsitlustele vastama pääseb küsitlusele määratud unikaalse tunnussõna ja parooli abil (mille määrab küsitlusele selle kokkupanija). Küsitluse vastused salvestatakse ainult sel juhul, kui kõigile neile küsimustele, millele vastamine on kohustuslik, on vastatud. Kui mõnele sellisele

küsimusele on vastamata, siis antakse sellest ka vastava küsimuse juures veateatega teada ja sellele on võimalik vastus anda.

Järgnevalt kirjeldan põhjalikumalt millistest tabelitest andmebaas koosneb ja milliseid andmeid seal hoitakse. Kuna andmebaasis on tabelite nimed mõneti raskesti mõistetavad (inglise keeles ja lühendatud), siis järgnevas tabelite kirjelduses lisan tabeli nimele ka eestikeelse vaste või ka lühida kirjelduse, mida see tabel sisaldab. Tabelite vaheliste seoste kohta saab hea ülevaate Martin Oti töös leiduvalt andmebaasi struktuuri kujutavalt jooniselt (Ott, 2003, 4.2.2. Andmebaasi struktuur lk. 19, Joonis 2). Kompaktsem ja tehnilisem ülevaade andmebaasi tabelitest asub käesoleva töö lisas (Lisa 1).

Tabel „don“ ehk õppejõudude tabel

Õppejõudude tabelis (Lisa 1, Tabel 1) hoitakse andmeid õppejõudude kohta. Kasutaja poolt täidetakse ainult väljad *first_name* (õppejõu eesnimi) ja *last_name* (õppejõu perekonnanimi). Väljade *id* ja *lastupdate* väärtused genereeritakse süsteemi poolt ning sisaldavad vastavalt õppejõudu identifitseerivat koodi ja ajatemplit millal õppejõu andmeid viimati muudeti.

Tabel „subject“ ehk ainete tabel

Ainete tabelis (Lisa 1, Tabel 2) hoitakse andmeid ainete kohta. Kasutaja peab sisestama väljade *code* (õppeaine kood) ja *title* (õppeaine nimetus) väärtused. Välja *lastupdate* väärtus genereeritakse süsteemi poolt ja sisaldab ajatemplit aine andmete viimase muutmise ajaga.

Tabel „question“ ehk küsimuste tabel

Küsimuste tabelis (Lisa 1, Tabel 4) hoitakse andmeid küsimuste kohta. Kasutaja poolt tuleb määrata väljade *type* (küsimuse tüüp), *text* (küsimuse tekst) ja *explanation* (küsimust selgitav tekst) väärtused. Küsimuse tüübiks võib olla üks järgnevatest koodidest:

- 10 – tekst tüüpi, vastuseks saab kirjutada vabas vormis teksti
- 20 – hinnang tüüpi, vastuseks saab anda hinnangu ette antud skaalal
- 30 – valikvastusevariantidega küsimus, vastuseks saab ette antud variantidest valida ühe
- 40 – valikvastusevariantidega küsimus, vastuseks saab ette antud variantidest valida mitu

Väljade *id* (küsimust identifitseeriv kood), *created* (küsimuse lisamise kuupäev) ja *lastupdate* (ajatempel ajaga, millal küsimuse andmeid viimati muudeti) väärtused genereeritakse süsteemi poolt. Tabelis on ka väli *originusr,* kuid tabeli sisu vaadates tundub, et seda ei kasutata, kuna

kõikide küsimuste puhul on selle välja väärtuseks NULL (väärtus puudub).

Tabel „query“ ehk küsitluste tabel

Küsitluste tabelis (Lisa 1, Tabel 3) hoitakse andmeid koostatud küsitluste kohta. Kasutaja poolt tuleb sisestada järgmiste väljade väärtused: *year* (aasta, millal see semester toimus), *semester* (kas tegemist on sügis- või kevad semestriga), *course* (kursuse nimetus, kellele küsitlus koostati), *codeword* (küsitluse kood, mille abil pääseb küsitlusele vastama ja millega määratakse ära millisele küsitlusele vastama hakatakse), *password* (salasõna pääsemaks küsitlusele vastama), *startdate* (kuupäev koos kellaajaga, millal saab hakata küsitlusele vastama), *enddate* (kuupäev koos kellaajaga, mis ajast enam ei ole võimalik küsitlusele vastuseid anda) ja *explanation* (küsitluse juurde käiv lühike selgitus). Väljade *id* (küsitlust identifitseeriv kood), *creatdate* (kuupäev ja kellaaeg millal küsitlus loodi) ja *lastupdate* (ajatempel ajaga, millal küsitluse andmeid viimati muudeti) väärtused genereeritakse süsteemi poolt. Tabelis on ka väli *usr_group*, kuid tabeli sisu vaadates tundub, et seda ei kasutata, kuna kõikide küsitluste puhul on välja väärtuseks NULL (väärtus puudub).

Tabel „result_code“ ehk vastuste tabel

Vastuste tabelisse (Lisa 1, Tabel 6) salvestatakse küsitlustele antud vastused. Tegemist on tabeliga, kus ühegi välja väärtust kasutaja ise ei määra, vaid väärtused kas genereeritakse või määratakse süsteemi poolt. Tabeliväljad on järgmised:

- *query_id* – küsitluse id, mille kohta vastus käib
- *question_id* – küsimuse id, mille kohta vastus käib
- *subject_code* – aine kood, mille kohta vastus käib
- *don_id* – õppejõu id, kelle kohta vastus käib
- *sess_id* – sessiooni id binaarkujul
- *code* – vastuse kood. Vastavalt küsimuse tüübile, on selle välja väärtuseks kas hinne (hinnang tüüpi küsimus), valikvastustega küsimuse puhul variandi kood variantide tabelis (Lisa 1, Tabel 7) või vabatekst tüüpi küsimuse puhul vastuse id vabatekst-tüüpi küsimuste vastuste tabelis (Lisa 1, Tabel 5).

Tabel „result_text“ ehk vabatekst tüüpi küsimuste vastuste tabel

Tekst tüüpi vastuste tabel (Lisa 1, Tabel 5). Väli *code* genereeritakse süsteemi poolt. *Text* välja väärtuseks on vaba tekst tüüpi vastus. Juhul kui sisestatakse identne vastus, siis uut kirjet tabelisse ei tehta, vaid vastuste tabelisse (Lisa 1, Tabel 6) läheb kirja olemas oleva vastuse kood.

Tabel „qstdata“ ehk valikvastustega küsimuste vastuse variantide tabel

Valikvastustega küsimuste vastusevariantide tabel (Lisa 1, Tabel 7) sisaldab valikvastusevariantidega küsimuste vastusevariante. Kasutaja poolt täidetakse ainult väli *text* (vastuse variandi tekst). Väli *qst_id* määratakse süsteemi poolt vastavalt sellele, millisele küsimusele vastusevariante määratakse. Võti *nr* genereeritakse süsteemi poolt automaatselt ning sisaldab vastusevarianti identifitseerivat koodi ühe küsimuse piires.

Tabel „newresult“ ehk uus vastamiste salvestamise tabel

Vastamiste salvestamise tabelisse (Lisa 1, Tabel 8) lisatakse iga kord uus rida, kui mingile küsitlusele vastatakse. Välja *query_id* pannakse süsteemi poolt selle küsitluse id, millele vastati. Väljade *id* ja *creation_date* väärtused genereeritakse süsteemi poolt ja nende väljade sisuks läheb vastavalt vastamist identifitseeriv kood ja kuupäev koos kellaajaga millal vastamine andmebaasi salvestati. Välja *auth_str* antud hetkel ei kasutata, kuna andmebaasist vaadates on see välja väärtus alati NULL (väärtus puudub).

Tabel „newresult_code“ ehk uus vastuste salvestamise tabel

Uude vastuste salvestamise tabelisse (Lisa 1, Tabel 9) salvestatakse küsitlusele antud vastused. Ühe vastamisega vastamiste salvestamise tabelis (Lisa 1, Tabel 8) seob vastust väli *result_id.* Ülejäänud väljade väärtused on samad mis vanas vastuste tabelis (Lisa 1, Tabel 6).

Tabelid *newresult* (Lisa 1, Tabel 8) ja *newresult_code (*Lisa 1, Tabel 8) asendavad *result_code* (Lisa 1, Tabel 6) tabeli, salvestades ühe vastamise ja vastamise käigus tekkivate vastuste andmed eraldi tabelitesse vähendades sedasi korduva info salvestamist vastuste tabelisse. Varasema lahenduse puhul läks vastuste tabelisse iga vastuse juurde ka küsitluse kood ja aeg millal vastamine baasi salvestati. Uue lahenduse puhul pannakse küsitluse kood ja vastuste salvestamise aeg ainult ühe korra kirja vastamiste tabelisse (*newresult,* Lisa 1, Tabel 8) ja vastuste tabelis (*newresult_code*, Lisa 1, Tabel 9) salvestatakse vastuse juurde ainult vastamist identifitseeriv kood.

Tabel „lecture“ ehk õppejõude ja aineid siduv tabel

Seosetabel, mis määrab ära seosed ainete ja õppejõudude vahel (Lisa 1, Tabel 12). Koosneb kahest väljast: *don_id* (õppejõudu identifitseeriv kood) ja *subject_code* (aine kood).

Tabel „qryqst“ ehk küsitlusi ja küsimusi siduv tabel.

Seosetabel, mis määrab ära seosed küsitluste ja küsimuste vahel, ehk millistest küsimustest küsitlused koosnevad (Lisa 1, Tabel 10). Seose määravad ära väljad *query_id* (küsitlust identifitseeriv kood) ja *qst_id* (küsimust identifitseeriv kood). Lisaks seosele on tabelis kirjas kas küsimusele vastamine vastavas küsitluses on kohustuslik või vabatahtlik (väljas *optional*) ning küsimuste järjekord küsitluses (väljas *nr*).

Tabel „qrylecture“ ehk küsitlusi, aineid ja õppejõude siduv tabel

Seosetabel, mis määrab ära küsitluses esinevad ained ja õppejõud (Lisa 1, Tabel 11). Lisaks seostele pannakse sellesse tabelisse kirja see, mis aine vormiga tegemist on – kas loengu, praktikumi või seminariga. Aine vorm on võimalik ka märkimata jätta.

Tabel „user“ ehk kasutajate tabel

Kasutajate tabelis (Lisa 1, Tabel 13) on kirjas info nende kohta, kes pääsevad ligi küsitluste administreerimiskeskkonda. Kasutajat lisades saab määrata väljade *usrname* (kasutajanimi), *passwd* (kasutaja parool), *role* (kasutaja roll – võib olla kas administraator, asjaajaja või mõlemad), *first_name* (kasutaja eesnimi) ja *last_name* (kasutaja perekonnanimi). Süsteemi poolt genereeritakse sisu järgmistesse tabeli lahtritesse: *created* (kuupäev, millal kasutaja lisati), *last_login* (kuupäev ja kellaaeg, millal kasutaja viimati süsteemi logis), *lastupdate* (ajatempel ajaga, millal kasutaja andmeid viimati muudeti). Tabelis on ka väli *creat_date*, kuhu on varem kasutaja loomise kuupäeva salvestatud, kuid enam seda ei kasutata, vaid selle asemel kasutatakse välja *created*.

Tabel „session“ ehk sisseloginud kasutajate info tabel

Tabel kasutajatest, kes on administreerimiskeskkonda sisse loginud (Lisa 1, Tabel 14). Koosneb väljadest *sess_id* (PHP sessiooni id), *usrname* (kasutajanimi, kes on sisse loginud), *sess_data* (PHP sessioonis olevad andmed, ei kasutata), *auth_str* (arvuti võrguaadress, kust kasutaja on sisse loginud) ja *last_update* (ajatempel ajaga, millal antud sessiooniga seotud andmeid viimati muudetud on).

1.2. Tallinna Ülikoolis kasutusel olev tagasisideküsitluste süsteem

Sarnaselt Oti süsteemiga koosneb tagasisideküsitluste süsteem nii küsitluste haldamise kui

küsitlustele vastamise osast. Lisaks aga on olemas ka küsitluste käigus kogutud andmete analüüsi osa, mis Oti süsteemis puudus. Erinevalt Oti süsteemist tegeleb küsitluste haldamisega ainult administraator ja kuna TLÜ tagasisideküsitluste süsteem on tihedalt seotud ülikooli õppeinfosüsteemiga, siis küsitlustele vastamine on lahendatud mitte kursuse kaupa (nagu Oti süsteemis), vaid iga tudeng saab hinnata neid aineid, millele ta on registreerunud. Ülevaate sellest, millised on erinevate kasutajate võimalused süsteemis annab järgnev nimekiri:

- **Administraator** – määrab kasutajatele õigused ning tagasiside küsimustike täitmise aktiivperioodid, sisestab ainekursuste hindamise küsimustiku sisu.
- **Tudeng** – saab elektroonliselt täita tagasiside küsimustiku ainete kohta, millele on eelnevalt registreerunud, näeb kokkuvõtet teaduskonna ainekursuste hindamise tulemustest.
- **Õppejõud** – näeb enda poolt õpetatavate ainekursuste tagasiside küsitluste tulemusi.
- **Õppetooli juhataja** – näeb antud õppetooli õppejõudude ainekursuste tagasiside küsitluste tulemusi.
- **Osakonna juhataja** – näeb antud akadeemilise osakonna õppejõudude ainekursuste tagasiside küsitluste tulemusi. Kui õppejõud töötab mitmes osakonnas korraga, näeb osakonna juhataja vaid tulemusi, mis käivad tema osakonnas õpetatavate ainete kohta.
- **Dekanaat** – dekaanid ja prodekaanid näevad kõigi antud teaduskonna õppejõudude tagasiside küsitluste tulemusi. Kui õppejõud töötab mitmes teaduskonnas korraga, näevad dekaan ja prodekaan vaid tulemusi, mis käivad tema teaduskonnas õpetatavate ainete kohta.
- **Juhtkond** – täies mahus on tagasiside küsitluste tulemused kättesaadavad rektorile, prorektoritele ja õppeosakonna kvaliteedispetsialistile (Tallinna Ülikool, 2004).

Sisselogimine elektroonilisse ainekursuste registreerimise ja tagasiside süsteemi toimub TLÜ veebilehel asuva õppeainete kataloogi kaudu.

Süsteem võimaldab ankeete (ehk küsitlusi) koostada ja neid hiljem muuta. Ankeedi juures saab sisestada järgmised ankeedi üldandmed:

- Tagasiside ankeedi tüüp (0-õpetamise ja ainekursuste hindamine; 1-õppekorralduse hindamine; 2-vilistlaste hindamine; ...)
- Ankeedi pealkiri
- Ankeedi päis

- Ankeedi jalus
- Kommentaar

Ankeedile saab määrata aktiivperioodi, ehk ajavahemiku, millal ankeedi täitmine võimalik on. Selleks määratakse ankeedi juurde alguskuupäev ja lõppkuupäev.

Küsimuste sisestamisel tuleb määrata järgmised andmed:

- Järjekorra number
- Küsimuse sisu
- Küsimuse tüüp (hinnatakse õppekorraldust, õppejõudu, ...)
- Vastuse võimalikud tüübid (0-radiobutton; 1-drop-down; 2-text)
- Vastuse võimalikud väärtused olenevalt vastuse tüübist

Tagasisideküsitlusele vastamisel tuleb vastajal valida kursus või õppejõud, kelle kohta ta ankeeti täitma hakkab. Peale seda kui vastaja on täitnud poolte (50%) ainete ankeedid, millele ta on registreerunud, ilmub üliõpilaste andmebaasi vastavasisuline märge.

Statistika osa võimaldab küsitluste käigus kogutud andmetest kokkuvõtteid teha. Järgnev nimekiri andmete analüüsi osa võimalustest on pärit „Elektrooniline tagasisideküsitlus: õpetamise ja ainekursusete hindamine“ dokumendist:

- Ühe ja sama ainekursuse küsimustiku kõikide punktide keskmised eraldi
- Kõigi hinnatud ainekursuste keskmised eraldi punktide kaupa
- Õppekorralduse eeskirjast kinnipidamise määra selgitamine ühe aine lõikes (vastava küsimuste tüübiga küsimuste keskmine)
- Õppekorralduse eeskirjast kinnipidamise määra selgitamiseks kogu ülikoolis arvutatakse kõigi ainete kohta (vastava küsimuste tüübiga küsimuste keskmine)
- Õppejõu metoodiliste pädevuste hindamiseks ühe aine kohta (vastava küsimuste tüübiga küsimuste keskmine)
- Õppejõu metoodiliste pädevuste hindamiseks kogu ülikoolis arvutatakse keskmine kõigi ainete kohta (vastava küsimuste tüübiga küsimuste keskmine)
- Küsimustele vastanute arv/protsent ainete kaupa
- Küsimustele vastanute arv/protsent õppejõudude kaupa
- Iga ainekursuse kohta koondatud kommentaarid, märkused ja ettepanekud
- Vastusevalikute eelistamine küsimuste kaupa ainete järgi

- Pingerida ainete kaupa (pingeridade puhul on oluline, et keskmiste hinnete kõrvale tuleks ka vastanud üliõpilaste arv)
- Õppejõudude pingerida (pingeridade puhul on oluline, et keskmiste hinnete kõrvale tuleks ka vastanud üliõpilaste arv)
- Õppejõudude pingerida valitud küsimuse korral (pingeridade puhul on oluline, et keskmiste hinnete kõrvale tuleks ka vastanud üliõpilaste arv)
- Programm peaks võimaldama valida arvandmeid, et vajadusel mõnes andmetöötlusprogrammis graafikuid koostada
- Kogu programm peaks võimaldama printimist ja andmete kopeerimist (Tallinna Ülikool, 2004)

Kuna TLÜ tagasisideküsitluste süsteem on tihedalt seotud õppeinfosüsteemi ja TLÜ veebikeskkonnaga, siis järgnevalt kirjeldan, kuidas süsteemid omavahel andmeid vahetavad ja suhtlevad.

TLÜ tagasisideküsitluse andmebaas kasutab Õppeinfosüsteemi andmetabeleid. Kuna aga Õppeinfosüsteem kasutab Oracle andmebaasi ja veebipõhine rakendus (ehk siis õppeainete kataloog ja koos sellega ka tagasisideküsitluste süsteem) MySQL andmebaasi, siis andmeid uuendatakse suunal Oracle->MySQL. Andmete uuendamise ja ekspordi/impordiga tegeleb õppeinfosüsteemi poolne osa. Enne uute andmete lisamist kustutatakse MySQL tabelitest kõik olemasolevad andmed ja seejärel lisatakse uued andmed tabelitesse. Kasutajate andmeid hoitakse omakorda eraldi andmebaasis (kasutatakse FoxPro andmebaasi). Andmevahetus erinevate süsteemide vahel toimub järgmiselt:

- Kasutajate süsteemist (FoxPro andmebaas) imporditakse õppeinfosüsteemi (Oracle andmebaas) kasutajate andmed
- Oraclest eksporditakse andmed veebipõhisesse süsteemi (MySQL andmebaasi)
- Igal öösel impordib õppeinfosüsteem registreeritud kursuste nimekirja MySQL andmebaasist Oracle andmebaasi tagasi. Nimekirja juures on ära märgitud, kas kursus on lisatud veebikeskkonna vahendusel või õppeosakonnas.
- Pärast andmete läbivaatust ning kursuse kuulajaks aktsepteerimistunnuse sisestamist osakonna poolt eksporditakse andmed veebipõhisesse süsteemi tagasi.
- Tudeng saab registreerimisperioodil (antud juhul on mõeldud veebipõhise registreerumise perioodi, mis ei pea kattuma akadeemilises kalendris märgitud kursustele registreerumisperioodiga) veebi kaudu kustutada ainult neid kursuseid,

millele ta on registreerunud veebi kaudu. Õppeosakonnast lisatud kursuseid veebi vahendusel kustutada ei saa.

- Tudeng võib täita nende ainekursuste kohta, millele ta on registreerunud, küsitluslehe.

Andmebaasi skeemi sellest, kuidas tagasisideküsitluse süsteemi andmebaasi tabelid omavahel seotud on, leiab dokumendist „Elektrooniline tagasisideküsitlus: õpetamise ja ainekursusete hindamine“ leheküljelt kaheksa.

Järgnevalt annan lühidalt ülevaate sellest, millised tabelid on tagasisideküsitluse süsteemi andmebaasis ja millist infot tabelites hoitakse.

Tabel „kasutaja“ ehk tabel sidumaks kasutajanimed isikutega

Tabelis hoitakse unikaalset kasutaja koodi, unikaalset isiku koodi ja logimiseks vajalikku nime. Tabel täidetakse õppeinfosüsteemi poolt.

Tabel „isik“ ehk isikuandmeid sisaldav tabel

Tabelis hoitakse meeles isiku ees- ja perekonnanime, isikukoodi, sünniaega ja sugu. Isikut identifitseerib unikaalne isiku kood. Sisu tuleb õppeinfosüsteemist.

Tabel „ankeet“ ehk ankeedi (küsitluse) üldandmeid sisaldav tabel.

Sellesse tabelisse salvestatakse ankeedi pealkiri, päise ja jaluse tekstid, aktiivsusperioodi algus- ja lõppkuupäev ning ankeedi kohtakäiv kommentaar. Lisaks salvestatakse sellesse tabelisse ankeedi tüüp (0-õpetamise ja ainekursuste hindamine; 1-õppekorralduse hindamine; 2-vilistlaste hindamine; ...). Ankeeti identifitseerib unikaalne ankeedi kood.

Tabel „stud_dekl“ ehk tabel, kus on info selle kohta, millistele ainetele tudeng on registreerunud.

Lisaks andmetele aine ja sellele registreeruva tudengi kohta peetakse selles tabelis meeles ka seda, kas tudeng on selle aine kohta tagasisidet andnud või mitte.

Tabel „kysimus“ ehk küsimustega seotud infot sisaldav tabel

Küsimuse tabelisse salvestatakse unikaalne küsimuse id, ankeeti identifitseeriv id, küsimuse järjekorra number, küsimuse sisu, vastuse tüüp (0-radiobutton; 1-dropdown; 2-text) ning vastuse võimalikud väärtused.

Tabel „kystyyp“ ehk tabel, kus hoitakse küsimuse tüüpe
Tabelis hoitakse unikaalset küsimuse tüübi koodi ja tüübi nimetust nii eesti kui inglise keeles.

Tabel „vastus“ ehk vastuste hoidmise tabel.
Sellesse tabelisse salvestatakse küsitlustele antud vastused. Tabelisse salvestatakse aine, õppejõu, semestri ja küsimuse koodid, ankeedi täitmise kuupäev ning vastus.

Tabel „oppeaine“ ehk õppeaine kohta infot sisaldav tabel
Tabeli sisu tuleb õppeinfosüsteemist ning sisaldab aine kohta käivat infot.

1.3. Kokkuvõte

Mõlemad küsitluste süsteemid on mõeldud õppeainete ja õppejõudude kohta tagasiside saamiseks ja seetõttu ka üpris sarnased. Kõige olulisem sarnasus on see, et mõlemas süsteemis on sisuliselt ühte tüüpi küsimused. Nad on küll erinevalt kirjeldatud – Oti süsteemis sisuliselt, ehk kas küsimusele saab vastata valikvastustega või vaba tekstina, TLÜ süsteemis vastavalt sellele millise HTML elemendi abil küsimusele vastata saab (märkeruudud, tekstikast), kuid statistikamooduli juures ei ole see oluline. Statistikamooduli seisukohalt on oluline küsimus, et kas süsteemide abil kogutud andmetest on võimalik välja korjata kokkuvõtete tegemiseks vajalikud andmed ja see muuta (vajadusel) sellisele kujule, et statistikamoodul oskaks nende põhjal kokkuvõtteid teha ja neid kokkuvõtteid kuvada?

Toetudes eelnevatele ülevaadetele on vastus sellele küsimusele jah, on küll. See paistab välja tagasisideküsitluste süsteemide andmebaaside ülevaadetest. Mõlema süsteemi puhul on võimalik teada saada millised ained, õppejõud ja küsimused küsitlustes osalevad. Hoolimata sellest, et neid andmeid peetakse meeles natuke erineva struktuuriga andmebaasides ja ka erineval kujul, on mõlemas süsteemis olemas piisavalt andmeid, et nende põhjal on võimalik leida sarnaseid statistikuid ja neid kuvada ühtse süsteemi järgi.

Tallinna Ülikooli tagasisideküsitluste süsteemi andmebaasitabelite kirjeldused ei ole nii detailsed kui Oti süsteemi puhul, kuna mul ei olnud võimalik andmebaasitabelitest täpselt järgi vaadata, mis väljad seal on ja mis andmeid nad sisaldavad. Ma ei pea seda ka oluliseks puuduseks, kuna

praktiliselt ei ole töö eesmärgiks luua statistikamoodulile võimalust kasutada TLÜ küsitluste süsteemi poolt kogutud andmeid, vaid rakendus nii üles ehitada, et selle võimaluse lisamine hiljem võimalik oleks. Pigem on TLÜ küsitluste süsteemi analüüs lisatud selle eesmärgiga, et oleks võimalik ära näha võimalikud sarnasused Oti süsteemiga ja kas nende süsteemide andmebaasides hoitavad andmed on piisavad, et neid saaks kasutada üks statistikamoodul kokkuvõtete tegemiseks.

2. Statistikamoodul

Statistikamoodul on rakendus, mis võimaldab erinavate tagasisideküsitluse süsteemide abil kogutud andmetest kokkuvõtteid ja statistikat teha. Algselt oli see mõeldud täiendama ainult Martin Oti poolt loodud tagasisideküsitluste läbiviimise süsteemi, kuna sealt oli see osa täielikult puudus. Sel juhul oleks olnud see mõistlik integreerida Oti küsitluste süsteemiga. Seoses sellega, et Tallinna Ülikoolis võeti kasutusele teine tagasisideküsitluste süsteem, tekkis mõte, et loodava rakenduse abil võiks olla võimalik analüüsida ka neid andmeid, mis on kogutud teiste tagasisideküsitlussüsteemide abil. Seetõttu sai loobutud plaanist integreerida moodul Oti süsteemiga ja sai võetud suund luua eraldiseisev rakendus, mis võimaldaks analüüsiks vajalikke andmeid võtta sealt, kuhu küsitluste süsteemid neid salvestavad ning oskaks neid andmeid vajadusel muuta sobivasse formaati.

Kuna statistikamoodul on eraldiseisev rakendus (see tähendab, et see ei ole sõltuv ühestki küsitluste läbiviimise süsteemist), siis on rakenduse loomiseks vajalike vahendite ja meetodite valik on üpris vaba. Sellest hoolimata on vahendite valikul mõistlik lähtuda sellest mis on juba küsitluste süsteemi töötamiseks paigaldatud, et statistikamooduli paigaldamine oleks võimalikult lihtne.

Kindlasti ei ole eesmärgiks luua väga laialdaste võimalustega veebipõhist andmeanalüüsitarkvara – pigem peaks rakenduse abil saama ülevaate küsitluste käigus kogutud andmetest. Teisejärgulisena võiks rakendus võimaldada paindlikumat tulemuste kuvamist: näiteks kokkuvõtte kuvamist tulp- või sektordiagrammina, määrata milliseid andmeid näidatakse tulemuste tabeli tulpades ja milliseid ridades. Lisavõimalusena võiks saada lisada elementidele (tabelitele, diagrammidele) peal- ja allkirju, et mõne lihtsama trükise jaoks saaks tulemusi süsteemist otse välja trükkida.

2.1. Tarkvara arhitektuur

Tarkvara arhitektuuriks nimetatakse üldist rakenduse struktuuri kirjeldust kus pannakse paika, millistest osadest (komponentidest) rakendus koosneb ja kuidas need osad omavahel seotud on (millised seosed komponentide vahel on). Samuti määratakse ära millist funktsionaalsust rakenduse osad ja nende vahelised seosed täitma peavad.

Esimest korda mainiti tarkvara arhitektuuri 1960-ndatel aastatel Edsger Dijkstra poolt. Laiemalt hakati sellest rääkima siiski alles 1990-ndate aastate alguses suuresti tänu Rational Software Corporation-i ja Microsoft-i tegevusele.

Aegade jooksul on välja kujunenud hulk tarkvara arhitektuuri lahendusi, mis on paljude erinevate tarkvaraprojektide teostamisel kasutust leidnud. Mõned näited enam levinud arhitektuuridest on:

- Klient-server – võrgutarkvara arhitektuur, mis eraldab kliendi (tavaliselt graafiline kasutajaliides) serverist. Näitena sellise arhitektuuriga süsteemist võiks tuua mõne andmebaasiserveri graafilise kasutajaliidese – andmebaasiserver tegeleb andmete säilitamisega, kasutajaliides võimaldab serverile päringuid saata ja näitab serveri poolt tagastatud andmeid (Wikipedia, „Client-server“).
- Hajusarvutus (Distributed computing) – süsteem, mille komponendid töötavad erinevate arvutite peal samaaegselt ja iseseisvalt tegeledes ühe ja sama probleemi või ülesande lahendamisega. Selliselt on üles ehitatud rakendused, mis vajavad ülesande lahendamiseks väga suurt arvutusvõimsust. Põhimõtteliselt lahendatakse probleem jagades see väiksemateks probleemideks ja jagatakse need osad erinevatele arvutitele lahendamiseks, hiljem kombineeritakse probleemi lahendus väiksemate probleemide lahendustest kokku. Arvutid kellele väiksemaid probleeme lahendada antakse, leitakse interneti vahendusel vabatahtlike näol, kes paigaldavad oma arvutisse kliendi, mis kasutab ainult tavapärasest tööst ülejäävat arvutusvõimsust ülesande lahendamiseks. Hajusarvutust kasutatakse näiteks mõtestatud maaväliste raadiosignaalide otsimiseks, suurte algarvude otsimiseks (rohkem kui 10 millioni kohalised arvud) ja efektiivsemate vähi ja AIDS-i vastaste ravimite leidmiseks. Need projektid on nii suured ja vajavad lahenduse leidmiseks nii suurt arvutustvõimsust, et ükski arvuti ega inimene ei suuda üksi mõistliku aja jooksul lahendust leida (http://distributedcomputing.info/, „What is Distributed Computing?“).
- *Three-tier* (*three-layer*) mudel – Kolmekihiline klient-server arhitektuur, kus kasutajaliides, äriloogika ja andmete kiht on iseseisvad komponendid. Komponente on võimalik eraldi uuendada ja välja vahetada, kui vajadused või tehnoloogia muutub. Selleks, et ühte komponenti välja vahetada, ei pea tegema muudatusi teistesse komponentidesse. Samasugused põhimõtted on kirjeldatud ka MVC arhitektuuri poolt, mida vahel nimetatakse MVC disainimustriks[1] (Wikipedia, „Three-tier (computing)“).

[1] Disainimuster tarkvaraarenduse kontekstis on üldine, korratav lahendus sageli esinevale probleemile tarkvara

- Monoliitne süsteem – Süsteem, kus kasutajaliides, äriloogika ja andmete salvestamine asuvad ühes osas/komponendis koos. Monoliitne süsteem on näiteks DOS operatsiooni süsteem (Wikipedia, „Software architecture“).

Statistikamooduli juures on oluline, et andmetega tegelev programmi osa oleks ülejäänud rakendusest võimalikult sõltumatu ja eraldiseisev. Kuna analüüsitavad andmed võivad asuda erinevates kohtades ja erineval kujul – olenevalt sellest, kuidas ja kuhu küsitluste läbiviimise süsteem neid salvestab, siis see võimaldab koondada erinevate küsitlussüsteemide eripäradest tulenevad muudatused ühte programmi ossa. Suhtlus andmetega tegeleva osa ja ülejäänud programmi vahel peab aga olema kindlalt ja üheselt määratletud, et andmete kihi muutumisel ei peaks ülejäänud programmikoodis muudatusi tegema.

Sellise rakenduse aluseks sobib hästi MVC tarkvaraarhitektuur, mille põhjal eraldatakse üksteisest programmi visuaalne pool (kasutajaliides) ja programmi loogika (andmed ja andmetega tehtavad operatsioonid).

Järgnevalt kirjeldan veidi põhjalikumalt mida MVC arhitektuuri erinevad kihid endast kujutavad, mis on nende ülesanneteks ja kuidas nad omavahel seotud on. Tegemist on üldiste põhimõtetega ja seega võivad nende realisatsioonid erinevate rakenduste puhul suuresti erineda. Enamasti on see tingitud rakenduse loomiseks kasutatavate vahendite omapäradest ja kindlasti ka sellest, kuidas arendaja(d) neist põhimõtetest aru saavad ja neid tõlgendavad.

Mudeli (*Model*) ülesanneteks on andmete hoidmine, salvestamine ja hoolitsemine selle eest, et need oleksid ülejäänud rakendusele kättesaadavad. Lisaks on mudeli ülesandeks andmete töötlemine – näiteks kui on vaja leida ettevõtte töötajate keskmist palka, siis selle arvutamine on mudeli ülesanne.

Mudel koosneb kohast, kus andmeid hoida (milleks on harilikult andmebaas), programmi funktsioonidest, mis andmeid hoidlast võtavad ja sinna salvestavad ning liidesest mille kaudu ülejäänud programm mudeliga suhtleb. Mudel ise teiste kihtidega ei suhtle.

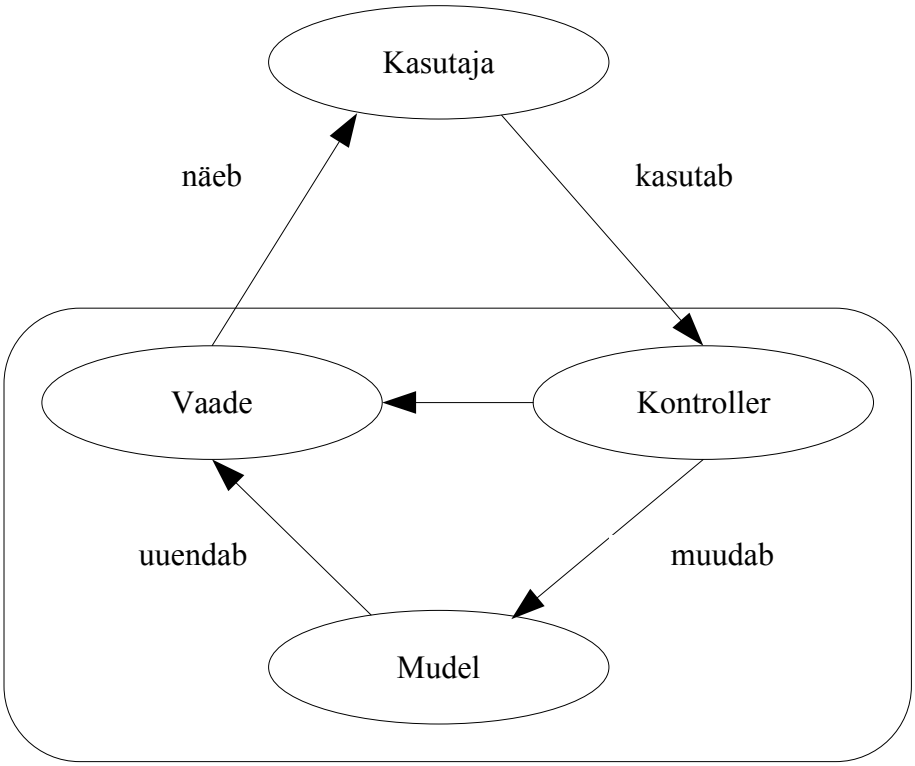
Vaade (*View*) hoolitseb selle eest, kuidas andmeid kasutajale näidatakse. Veebirakenduste puhul on kasutajaliideseks harilikult HTML-is koostatud veebileht. Samas võib see olla ka XML fail,

loomisel (Wikipedia, „Design Pattern (computer science)“).

mida teised rakendused kasutada saavad (näiteks Postimees Online uudistevoog). Vaade võib näitamiseks vajalikke andmeid küsida nii otse andmete kihilt kui ka läbi kontrollkihi.

Kontrolleri (*Controller*) ülesanneteks on tõlkida kasutajapoolsed tegevused programmile arusaadavale kujule ja koordineerida vastavalt sellele infole vaate ja mudeli tegevusi. Selles kihis toimub ka esmane kasutajalt programmile edastatavate parameetrite valideerimine.

Järgneval joonisel (Joonis 1) on MVC arhitektuuri kirjeldav diagramm millel on noolte abil ära näidatud ka andmete liikumise suunad erinevate kihtide vahel.



Joonis 1: MVC arhitektuuri kujutav skeem

Kui kasutaja näiteks vajutab nuppu, siis süsteemi tulev päring võetakse vastu kontrolleri poolt. Kontroller valideerib päringuga kaasa tulnud andmed (nupu vajutamise peale saadetakse näiteks vormis olevad andmed) ning muudab vajadusel mudelit (salvestab vormist tulnud andmed baasi). Seejärel saadab kontroller käskluse vaatele, et vaade ennast uuendaks. Vaade küsib mudelilt andmed ja näitab neid kasutajale.

2.2. Vahendid

Nii TLÜ kui Oti tagasisideküsitluste läbiviimise süsteem kasutavad töötamiseks Apache veebiserverit koos PHP toega. Andmete hoidmiseks kasutatakse MySQL andmebaasiserverit. Kuna statistikamooduli näol on tegemist põhimõtteliselt sarnase veebirakendusega nagu tagasisideküsitluse süsteemid, siis sobivad eelpool mainitud vahendid suurepäraselt ka selle rakenduse loomiseks. Mõned lisatingimused vahenditele siiski on: PHP versioon peaks olema vähemalt 4.3.x, kuna statistikamooduli puhul kasutan selliseid PHP võimalusi, mida varasemates versioonides saadaval ei ole. Lisaks on alates sellest versioonist vaikimisi lisatud GD teek, mida kasutatakse graafikute genereerimiseks. Statistikamooduli töötamiseks ei ole andmebaasiserveri olemas olu hädavajalik – seda on vaja ainult juhul kui andmed, mille põhjal statistikat tehakse, asuvad andmebaasis. Statistikamoodul ise mingeid andmeid andmebaasis ei hoia – vajalike konfiguratsiooni seadete või teiste salvestamist vajavate andmete hoidmiseks kasutan tekstifaile. See muudab küll vajalike andmete salvestamise ja lugemise natukene keerulisemaks, kuid samal ajal on statistikamoodulit lihtsam paigaldada – puudub vajadus luua ligipääs andmebaasile ja vajalikud tabelid. Juhul kui analüüsitavad andmed ei asu andmebaasis vaid selle asemel näiteks CSV failis (või failides), puudub vajadus andmebaasiserveri järele täielikult.

2.2.1. PHP

PHP on avatud lähtekoodiga serveripoolne skriptimiskeel võimaldamaks luua dünaamilisi veebirakendusi. Dünaamiliste veebilehtede puhul ei hoita serveris staatilisi, valmis tehtud HTML lehti, vaid leht pannakse kokku vastavalt kasutajalt tulnud parameetritele serveri poolt. Andmeid, mille põhjal veebilehti kokku pannakse, hoitakse sageli andmebaasides või neid saadakse veebiteenuste abil mõnelt teiselt veebilehelt (näiteks Postimehe uudised).

2.2.2. MySQL

MySQL on vabavaraline, avatud lähtekoodiga relatsiooniline andmebaasiserver, mis on algusest peale loodud töötama ka odavama riistvara peal. Eesmärgiks on võetud pakkuda kiiret ja lihtsalt kasutatavat andmebaasiserverit ja ilmselt seetõttu on ta ka väga levinud – eriti PHP-s kirjutatud rakenduste puhul. MySQL ei ole küll päris platvormist sõltumatu, kuid see on saadaval rohkem kui kahekümnele erinevale platvormile, sh. suuremad Linuxi distributsioonid, Mac OS X, UNIX

ja Microsoft Windows (MySQL AB 2004, http://www.mysql.com/products/).

2.2.3. Apache veebiserver

Apache veebiserver on avatud lähtekoodiga vabatahtlike poolt arendatav võimekas veebiserver, mille arendust koordineerib Apache tarkvara sihtasutus.

Apache veebiservar, MySQL andmebaasiserver koos PHP-ga moodustavad populaarse arendusplatvormi, mille põhjuseks on ilmselt asjaolu, et tarkvara paigaldamine ja seadistamine on märkimisväärselt lihtne nii Microsoft Windowsi kui ka Linux (UNIX) operatsioonisüsteemiga arvutitele.

2.2.4. Templeidid

Templeidid on vahend eraldamaks sisu ja vormi (struktuuri) teineteisest. Ühelehekülgelise uudiskirja templeit võib näiteks sisaldada paari veergu, kohta pildile ja lünki pealkirjale ning uudiskirja nimele (Raul Viigipuu, 2005). Uudiskirja templeidi paneb kokku ja kujundab disainer, hiljem lisab toimetaja selleks ette nähtud kohtadesse sisu.

Veebirakenduste puhul kasutatakse templeite selleks, et hoida lahus rakenduse visuaalne pool ja äriloogika. Niimoodi ei pea disainer teadma midagi programmeerimisest selleks et rakendusele kujundust teha ja programmeerija ei pea ennast kujunduse nüanssidega vaevama ning kumbki saab keskenduda oma tööle.

Statistikamooduli templeidid on HTML failid, kus vastavad märgendid asendatakse andmetega. Templeitide parsimiseks kasutan veebilehel http://www.php.ee avaldatud Anti Veeranna artiklis „Templeidid“ (http://www.php.ee/3379) kirjeldatud meetodit. Põhimõtteliselt seisneb meetod selles, et templeidi fail loetakse programmi poolt sisse, asendatakse eelnevalt kokku lepitud templeidimuutujad ja väljastatakse tulemus või säilitatakse tulemus muutujas, et asendada sellega templeidimuutuja mõnes teises templeidis. Selleks, et templeitide kirjutamine lihtsam oleks (ja ka nende parsimine), lisasin templeitide parserile alamtempleitide kasutamise võimaluse. Praktikas tähendab see lihtsalt seda, et ühte faili on võimalik vastavate märgenditega eraldatult kirjutada mitu templeiti. Templeitide kirjutamise poole pealt vähendab see tunduvalt

erinevate templeidifailide hulka ja seega ei tule neid ka koodi poolelt nii palju sisse lugeda.

Näiteks järgnev tabeli näitamiseks tehtud templeit kasutades alamtempleite näeks välja selline:

```
<table>
-     <!-- SUB: rida -->
-     <tr>
-           <!-- SUB: lahter -->
-           <td>{VAR:sisu}</td>
-           <!-- END SUB: lahter -->
-     </tr>
-     <!-- END SUB: rida -->
</table>
```

Ilma alamtempleite kasutamata tuleks iga SUB ja END SUB märgendi vahel olev osa eraldi templeidifaili kirjutada (antud näite põhjal tekiks kolm erinevat faili ühe asemel) ja muudaks templeitide kasutamise tülikaks nii koodi poole pealt kui ka templeitide koostaja seisukohast. Alamtempleitide puhul leitakse aga faili sisselugemisel vastavalt SUB märgenditele ühest failist kõik vajalikud templeidid ja seejärel asendatakse sisseloetud templeitides muutujad ja tulemus väljastatakse.

2.2.5. GD teek

GD teek on avatud lähtekoodiga programmeerimiskeeles C kirjutatud teek dünaamiliseks piltide genereerimiseks programmeerijatele. Seda kasutatakse peamiselt dünaamiliselt graafikute genereerimiseks või suurte piltide konverteerimiseks pisipiltideks (*thumbnails*). GD teek on pakendatud[2] näiteks sellistesse keeltesse nagu Perl, PHP jt. Pildiformaatidest on toetatud PNG, JPEG ja GIF formaat, millest veebirakenduse jaoks täiesti piisab.

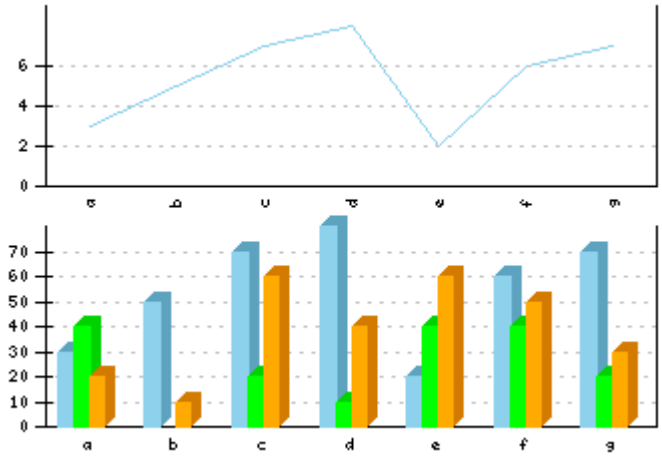
Graafikute joonistamise koha pealt võib GD teeki võrrelda siiski kui pliiatsi ja joonlauaga – selle abil on võimalik pildile lisada jooni, kaste ja teksti – nende abil graafikute moodustamine on siiski programmeerija ülesanne. Siiski on internetist leida ka valmis lahendusi, mis GD teegi abil erinevat tüüpi graafikute genereerimise lihtsamaks teevad. Lähemaks tutvustamiseks olen neist välja valinud kaks – põhilisteks kriteeriumiteks valiku tegemisel oli dokumentatsiooni olemasolu

[2] Pakend on siin andmestruktuur või programm, mis sisaldab (pakendab) teisi andmeid või programme, nii et „sissepakitud“ elemendid võivad eksisteerida uuemas süsteemis. (e-teatmik, „wrapper“)

ja kasutamise lihtsus. Samas on oluline ka see, kui paindlikult on võimalik graafikuid enda maitse järgi kohandada ja milliseid erinevaid graafikute tüüpe toetatakse. Rakendust leiavad neist esialgu küll lihtsamad (näiteks tulpdiagramm, sektordiagramm), kuid jätkusuutlikuse huvides on oluline, et ainult nendega ei piirdutaks.

2.2.6. PHPlot

PHPlot on PHP-s kirjutatud klass[3] mis lihtsustab GD teegi abil graafikute genereerimist lahendades graafikute joonistamise tehnilise poole (näiteks tulpdiagrammi puhul tulpade ja teljestiku joonistamise). Ette tuleb anda ainult andmed, mille põhjal graafikuid genereeritakse (tulpdiagrammi näidet jätkates, tulpade kõrgused, värv ja milline teljestik välja näeb). Töötab PHP versioonidega kolm ja neli, graafiku tüüpidest on teiste seas toetatud joongraafikud, tulpdiagrammid ja sektordiagrammid. Tekstide kuvamiseks graafikul on võimalik kasutada TTF kirjatüüpi.



Illustratsioon 1: PHPlot võimaldab ka ühe pildi peale mitu graafikut joonistada.

PHPlot võimaldab paindlikult konfigureerida milliseid pealkirju ja silte kuvatakse, kui suur on teljestiku skaala ja kui mitmeks osaks see on jagatud. Ühtlasi on võimalik määrata joonte, tulpade ja sektorite värve ja sildi tekstide kirjatüüpe.

PHPloti suurimaks miinuseks on, et selle arendus on viimaste aastate jooksul küllaltki visalt edenenud. Veebilehe andmetel on viimane stabiilne versioon välja lastud 2001 aasta kevadel

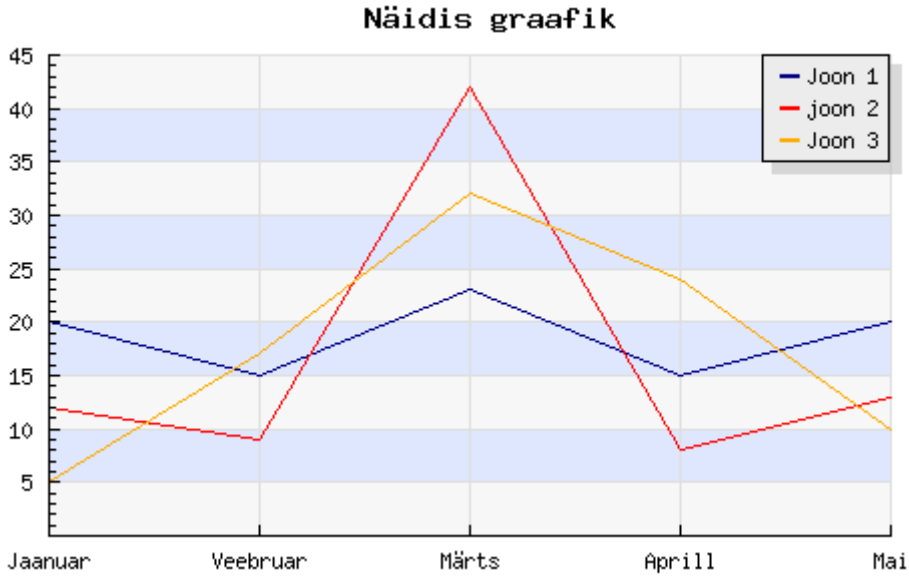
[3] Klass mõistet on siinkohal kasutatud objekt-orienteeritud programmeerimise kontekstis ja tähendab mingi hulga üksteisega seotud muutujate ja funktsioonide kogumit.

(versioon 4.4.6). Arenduses on ka versioon 5, kuid sellest on viimase kolme aasta jooksul välja tulnud kolm arendusversiooni (viimane 2006 aasta lõpus).

2.2.7. JpGraph

JpGraph on sarnaselt PHPlotile PHP-s kirjutatud abivahend graafikute koostamiseks. Erinevalt PHPlotist ei ole aga tegemist ühe klassiga, mis vajalikku funktsionaalsust pakub, vaid JpGraphi teek koosneb paljudest klassidest vastavalt funktsionaalsusele, mis muudab selle kasutamise palju paindlikumaks ja efektiivsemaks. See tähendab, et kui ma oma rakenduses kasutan ainult joon- ja tulpdiagramme, siis on võimalik teha programmi jaoks kättesaadavaks ainult nende diagrammide koostamiseks vajalik osa JpGraphi teegist. PHPloti klassi puhul see võimalik ei olnud, kuna seal oli kogu graafikute genereerimiseks vajalik funktsionaalsus koondatud ühte klassi.

JpGraph teegist on kaks versiooni – üks, mis töötab PHP versiooniga 4 ja teine, mis töötab PHP versiooniga 5. Graafiku tüüpidest on toetatud teiste hulgas nii joon-, tulp-, kui ka sektordiagrammid ja nende väljanägemine koos teljestikuga on väga suurel määral kohandatav.



Illustratsioon 2: JpGraph teegi abil loodud joongraafik koos legendiga

Statistikamooduli juures otsustasin kasutada JpGraph teeki, kuna hoolimata sellest, et seda on natukene keerukam kasutada kui PHPlot-i, on see väga hästi ja põhjalikult dokumenteeritud. Samuti on oluliseks eeliseks see, et JpGraph-i teeki arendatakse pidevalt ja aktiivselt edasi

(viimane versioon on välja lastud 2007 aasta kevadel) ja ilmselt seetõttu on see ka palju viimistletum kui PHPlot teek.

2.3. Rakendus

Rakendus võimaldab küsitluste käigus kogutud andmetest ülevaadet saada nii konkreetse küsitluse, õppejõu, aine või küsimuse lõikes. Selleks, et saada ülevaade mõne konkreetse küsitluse raames kogutud andmetest, tuleb peamenüüst valida „Küsitlused“ ja avanenud küsitluste nimekirjast soovitud küsitlus. Küsitlusel klikkides avaneb vaade, kus näidatakse milliste ainete/õppejõudude kohta küsitlus oli koostatud ja milliseid küsimusi küsiti. Siin saab valida, millise aine/õppejõu tulemuste vastu huvi tuntakse ja millise küsimuse tulemusi näha tahetakse. Küsimuste nimekirjas on ära märgitud ka küsimuse tüüp, millest oleneb see, mismoodi tulemusi on võimalik kuvada ja mida nendega teha. Näiteks ei ole võimalik tekst tüüpi vastustega küsimuste tulemustest arvutada keskmisi ega kuvada neid tulpdiagrammina. Kui ained ja küsimused on välja valitud, tuleb vajutada nuppu „Lisa“ või „Lisa ja vaata“. Nende vahe seisneb selles, et esimene lisab vastavalt valikutele koostatud tabeli või graafiku statistika vaatesse ja ei muuda statistika vaadet kohe aktiivseks – see võimaldab mugavalt mitu erinevat versiooni koostada mingi küsimise (küsimuste) tulemustest ja hiljem minna vaatama, milline variant paremini tulemusi edasi annab.

Selleks, et saada ülevaadet mõne õppejõu, aine või küsimuse lõikes, tuleb peamenüüst teha vastav valik ja ülejäänud funktsionaalsus on sarnane sellega, mis Küsitluste puhul juba kirjeldatud sai. Erinevus seisneb ainult selles, et nende valikute puhul ei ole piiravaks elemendiks küsitlused, vaid tulemusi piiratakse vastavalt sellele, mis on peamenüüst valitud. Näiteks õppejõu puhul otsitakse välja ainult need küsitlused, kus see õppejõud on osalenud ja statistika vaates näidatavas kokkuvõttes kasutatakse ainult nendele küsitlustele antud vastuseid.

Selline rakenduse ülesehitus võimaldab küllaltki lihtsalt ja samas paindlikult saada ülevaade küsitluste käigus kogutud andmetest. Kui võtta näiteks küsimus, et „Kuidas olid õppematerjalid ette valmistatud?“ ja kui seda küsimust on küsitud mitme küsitluse puhul, siis küsitluste valiku all saab näha, kuidas olid materjalid vastanute arvates ette valmistatud mingi konkreetse perioodi jooksul mille kohta küsitlus tehti. Samal ajal on nende samade andmete põhjal võimalik ka teada saada seda, mida vastanud arvavad mõne kindla õppejõu ette valmistatud materjalidest

(peamenüüst valik „Õppejõud“), või mõne kindla aine jaoks ette valmistatud materjalidest (peamenüüst valik „Ained“) või kuidas materjalide ette valmistamise tase üldiselt on (peamenüüst valik „Küsimused“).

Järgnevates peatükkides keskendun peamiselt sellele, kuidas rakendus on koodi poolelt kokku pandud, mismoodi erinevad osad omavahel seotud on ja kuidas varem mainitud vahendeid ja meetodeid on praktikas rakendatud.

2.3.1. Koodi struktuur

Rakenduse kood on jagatud vastavalt funktsionaalsusele klassidesse. Iga klass on enda nimelises failis ja asub kaustas *classes*. Visuaalne pool on defineeritud templeidifailides, mis asuvad kaustas *templates*. Lisaks sellele on rakenduse juurkaustas failid *index.php* ja *config.php*, millest esimese ülesandeks on teha rakenduse jaoks kättesaadavaks *config.php* failis asuvad rakenduse seaded, rakenduse tööks vajalik minimaalne funktsionaalsus ja anda tööjärg üle kontrollerile, mis vastavalt kasutajalt tulnud päringule juhib programmi tööd edasi.

Koodi jagamisel klassidesse lähtusin põhiliselt sellest, kuidas rakendus jagunes loogilisteks osadeks kasutajaliidese poole pealt (edaspidi võib neid nimetada komponentideks) ning kuna eesmärgiks oli eraldada rakenduse andmetega tegelev koodi osa ülejäänud rakendusest, siis sai arvesse võetud ka seda, kuidas funktsionaalsus paigutub MVC arhitektuuri diagrammile. Lisaks sellele jaguneb rakenduse kood selliselt, kus ühte osa kasutab kogu rakendus, hoolimata sellest, mida kasutaja rakendusega teeb, ning teist osa on vaja ainult mõne kindla ülesande täitmiseks. Seega võib klassid jagada laias laastus kaheks:

- baasklassid – sisaldab sellist funktsionaalsust, mida on vaja ja mis peaks olema kasutatav terve programmi ulatuses
- komponendi klassid – need klassid sisaldavad ainult mingi komponendi tööks vajalikku funktsionaalsust. Kui komponenti programmis hetkel vaja ei ole, siis ei ole ka selle koodi osa kättesaadavus oluline.

Vastavalt MVC arhitektuurile jaotuvad nii baasklassid kui ka komponendi klassid oma korda andmetega tegelevateks klassideks ja vaate ehk kasutajale nähtava osa genereerimise eest hoolitsevateks klassideks (vastavalt siis baasklassid üldisteks andmete ja näitamisega tegelevateks klassideks, ning komponentide klassid komponentide spetsiifilisteks andmete ja

näitamise klassideks). Kui andmete ja näitamisega tegelevad klassid on olemas nii baasklasside kui ka komponentide klasside hulgas, siis kontrolleri funktsionaalsust sisaldava klass kuulub ainult baasklasside hulka – komponentidel oma kontrolleri klassi ei ole.

Kuna rakenduse puhul on kõige olulisem see, et oleks võimalikult lihtsalt aru saada, millistes klassides sisaldub andmetega tegelev funktsionaalsus, siis on seda silmas peetud klasside nimede valikul (mis omakorda kajastuvad ka koodifailide nimedes): andmetega tegelevate klasside nimes leidub inglise keelne sõna „*data*“. Näiteks küsitluste komponendi andmete klassi nimeks on „*query_data*“. Sarnaselt on nimed valitud ka vaatega tegelevatele klassidele sisaldades inglise keelset vastet sõnale „vaade“: „*view*“. Nagu juba eelnevast võib aimata, on klasside nimed inglise keelsed. Valitud on nad sellised, et nad võimalikult täpselt kirjeldaks seda, millist osa rakenduse funktsionaalsusest klass sisaldab.

Lisaks eelpool kirjeldatud rakenduse koodi jagamisele võimalikult loogilisteks tükkideks olen koodi enda loetavuse hõlbustamiseks püüdnud kinni pidada ka järgnevatest reeglitest:

- Muutujate ja funktsioonide nimed peavad võimalikult täpselt edasi andma seda, milliseid andmeid selles muutujas hoitaks või mida funktsioon/meetod teeb.
- Muutujate ja funktsioonide/meetodite nimed peavad olema täissõnad ja mitme sõna puhul olema eraldatud alakriipsuga.
- Iga klass on eraldi failis, faili nimi on sama mis klassi nimi.

Selline rakenduse struktureerimine ja võimalikult korrektse ja loetava koodi kirjutamisel on mitmeid häid kaasmõjusid, mis :

- rakenduse kood on paremini loetav – see on hea mitte ainult selleks kui mõni teine programmeerija tahab koodi muudatusi teha, vaid ka rakenduse loojale endale, et paari kuu vanusest koodi lõigust oleks võimalikult kiiresti ja kergelt aru saada.
- võimaldab paremini vältida koodi dubleerimist – kuna olemasolevast koodibaasist ja selle funktsionaalsusest on lihtsam terviklikku ülevaadet hoida, siis väheneb oluliselt võimalus, et ühte ja sama probleemi lahendav koodilõik esineb mitmes kohas.
- võimaldab kiiremini aru saada millisest failist mingit funktsionaalsust leida võib ja seega lihtsustab rakenduse muutmist ja/või funktsionaalsuse lisamist.
- lihtsustab vigade otsimist, kuna kindla struktuuriga rakenduse puhul võimalik vea asukoht kiiremini üles leida ning loetava koodi puhul oluliselt lihtsam vea tekkimise põhjusest aru saada.

Sellisel tarkvaraarendusel on siiski ka mõned varjuküljed:

- võib muuta kiirete paranduste sisseviimise keerukamaks - kui tahta struktuurist ja loogikast kinni pidada, siis iga lisatud võimaluse puhul tuleb läbi mõelda kuhu see süsteemi kui terviku suhtes kõige paremini sobiks.
- muudab arendusprotsessi aeglasemaks, kuna lihtsalt programmikoodi kirjutamisele lisandub veel rakenduse planeerimine – milline saab koodi struktuur olema ja milliseid tehnoloogiaid programmi loomisel kasutada.

Kuivõrd see reaalselt probleem on, sõltub juba konkreetsest ülesandest mida rakendus lahendama peab. Mõne väiksema programmi puhul ei ole mõtet kulutada aega ja energiat selle planeerimise peale või korraliku koodi kirjutamise peale. Samal ajal kui mõne suurema rakenduse puhul, kus on kaasatud mitu arendajat, on kindla struktuuri ja koodi kirjutamise reeglite paika panemine ainult kasuks, kuna siis saavad kõik asjast ühte moodi aru ja tekib vähem probleeme erinevate arendajate tehtud töö ühilduvuses.

2.3.2. Kontroller

Kontrolleri funktsionaalsus on koondatud klassi *controller* ja see on rakenduse alguspunktiks – ehk see on esimene asi mis rakendusest käima pannakse.

Rakenduse kontroller hoolitseb selle eest, et valideerida kasutajalt tulevad andmed, ning vastavalt nendele andmetele juhtida rakenduse tööd. See, kui suurt rolli kontrolleri kiht tegelikult rakenduse töös mängib, on iga arendaja enda otsustada. Rakenduse võib üles ehitada nii, et kõik rakenduses toimuv käib läbi kontrolleri. See tähendab, et kui näiteks vaate kiht tahab saada andmete kihilt andmeid näitamiseks, siis võib see toimuda nii, et vaate kiht pöördub kontrolleri poole vastava päringuga, kontroller küsib andmete kihilt andmed ja annab need vaate kihile. Samas võib teha ka nii, et vaate kiht küsib andmed otse andmete kihilt, ilma et vahepeal kontrolleri poole pöörduks.

Statistika moodulis on kontrolleri ülesandeks kasutajalt tulnud andmete valideerimine ja vastavalt päringule vajalike rakenduse komponentide laadimine ja käima panemine. Juhul kui kontroller saab andmed, mida on vaja salvestada, siis pöördub kontroller otse vastava andmete klassi poole. Kui aga kontroller saab päringu, mille peale peab midagi näitama, siis pöördub

kontroller otse vastava näitamise klassi poole. Näitamise klass pöördub vajadusel ise vastava komponendi andmete klassi poole ja küsib sealt vajalikud andmed.

Osaliselt on kontrolleri ülesandeks ka õiguste kontrollimine – näiteks kui kasutaja sisselogimine ebaõnnestub, siis ta reaalselt kontrollerist kaugemale ei jõua. Kontrolleri tasandil on mõistlik lahendada ka erinevate komponentide nägemise/kasutamise õigused – kui kasutajal ei ole õigusi mõnda komponenti kasutada, siis selle komponendi koodi selle kasutaja jaoks ei laeta.

2.3.3. Vaade

Vaate kihi ülesanneteks on rakenduse kasutajaliidese genereerimine ja andmete kihilt näidatavate andmete küsimine ning nende näitamine kasutajale. See hõlmab ka erinevaid andmete ekspordi meetodeid. Näiteks kui on vaja andmeid eksportida CSV faili formaati, siis selle faili genereerimine on vaate klassi ülesanne vastavalt CSV faili templeidile. Vaate klassid kasutavad andmete kasutajale näidatavale kujule panemiseks templeite ja graafikute genereerimiseks JpGraph-i teeki.

Iga vaate klass teab, milliselt klassilt ta näitamiseks vajalikke andmeid küsib. Samuti on kindlaks määratud andmete klassi poolsed meetodid ja andmete struktuur mis kujul vaate klass andmeid vastu võtab. Kuna vaate klassil peaaegu alati vaja andmete klassi poole pöörduda, siis luuakse vaate klassi isendi loomisel automaatselt ka vastav andmete klassi isend. Niimoodi on võimalik hilisem vaate klassi kood hoida lihtsam, kuna ei pea alati kontrollima, kas andmete klassist on isend olemas või peab selle alles tegema.

Vaate klassid sisaldavad klassi nimes sufiksit *_view* ja laiendavad templeitide klassi, nii et igas vaate klassis on olemas ka templeitide kasutamiseks vajalik funktsionaalsus.

2.3.4. Mudel ehk andmete kiht

Rakenduse mudeli ehk andmete kihi ülesandeks on abstraheerida rakenduse suhtlus reaalsete andmetega. Tänu andmete kihile ei pea ülejäänud rakendus midagi teadma sellest, kuidas andmeid tegelikult hoitakse – kas nad paigutatud andmebaasi, või loetakse neid lihtsalt otse tekstifailidest. Andmete klassil on kindlaks määratud meetodid, mida vaate klassid kasutavad

andmete küsimiseks. Andmete klassi sisene probleem on vajalikud andmed kokku korjata ja sobilikule kujule panna.

Lisaks on andmete klassi ülesandeks on andmete põhjal erinevate statistikute arvutamine. See funktsionaalsus on andmete kihis sellepärast, et näiteks MySQL andmebaas võimaldab juba arvutada lihtsamaid statistikuid ja antud juhul on lihtsam seda ära kasutada kui see kõik oma rakenduses uuesti teha. Samas, kui andmed ei asu andmebaasis, mis lihtsamaid arvutusi teha võimaldab, siis saab selle funktsionaalsuse andmete klassi lisada ja ülejäänud rakendus ei pea sellest jällegi midagi teadma – näitamise klassile lähevad ikka samad andmed, ainult andmete kokku panemise viis on erinev.

Rakenduse andmete kiht koosneb üldisest andmete klassist (*data.php*) ja komponentide andmete klassidest (klassi nimi lõpeb *_data* sufiksiga). Komponentide andmete klassid laiendavad üldist andmete klassi, nii et nad saavad kasutada seal olevaid meetodeid, mis lihtsustavad andmete pärimist andmebaasist. Ülejäänud rakenduse olemasolust ei tea andmete kiht midagi – ülejäänud rakendus on teadlik andmete kihi olemasolust. Sellise seotuse suunaga olen saavutanud selle, et andmete kiht on ülejäänud rakendusest nii eraldiseisev kui see vähegi võimalik on, samal ajal jättes selle siiski rakenduse osaks.

Andmete kihis on mõistlik lahendada ka see õiguste kontroll, milliseid andmeid kasutajale näidatakse ja milliseid mitte. Kui kasutajal ei ole õigusi mingit osa andmetest näha, siis selles kihis on see võimalik lahendada selliselt, et neid andmeid ei päritagi andmebaasist. Selline lahendus suurendab nii rakenduse efektiivsust kui turvalisust – andmebaasist küsitakse ainult need andmed, mida edaspidi vaja läheb ning väheneb oht, et kuskil koodis olev viga võimaldab näha ka neid andmeid, mis ei peaks olema kättesaadavad.

Kokkuvõte

Käesoleva töö esimeses osas analüüsin nii Oti kui ka TLÜ uue tagasisideküsitluse süsteeme, et saada vajalik info statistikaosa tegemiseks ning selgitamaks, kas see statistikamoodul võiks ka ühine olla. Selle juures võtsin arvesse andmebaaside kui ka süsteemide üldist ülesehitust.

Töö teine osa keskendub peamiselt rakenduse loomisele ja sellega seonduvate tehnoloogiate kirjeldamisele. Lisaks sellele on käsitletud ka erinevaid võimalusi lahenduse tarkvaraarhitektuuri osas ning töö sisaldab ka ülevaadet erinevatest tarkvara arendusvahenditest, mille abil sellelaadset süsteemi realiseerida.

Lõputöö käigus koostatud tagasisideküsitluste süsteemide analüüsi ja rakenduse loomiseks vajalike vahendite ülevaateid saab kasutada ka siis, kui tekib vajadus luua Oti tagasisideküsitluste süsteemi integreeritud statistika osa. Rakenduse loomise ja selleks kasutatavate meetodite ja vahendite kirjeldus annab aga ülevaate ühest võimalusest, kuidas reaalselt lahendada MVC arhitektuuril baseeruv veebirakendus.

Töö käigus loodud programm ei ole päriselt valmis. Vastavalt ülesande püstitusele on valmis rakenduse raamistik. Funktsionaalsusest on kõige rohkem valmis andmete analüüsimise osa küsitluste lõikes. Õppejõudude, ainete ja küsimuste lõikes andmete analüüsimiseks vajalik funktsionaalsus on puudulik. Hoolimata sellest annab olemas olevad võimalused pildi sellest, mida statistikamooduli abil teha saab, kuna kasutusloogika jääb põhimõtteliselt samaks terves rakenduses. Vastavalt küsimuste tüüpidele on võimalik tulemustest arvutada keskmisi, sagedusi ning andmeid ka lihtsalt vaatamiseks väljastada. Erinevaid statistikuid on võimalik kuvada kas tabelisse või ka tulp- või sektordiagrammil.

Kõige olulisemaks puuduseks olemasoleva rakenduse juures on korralik kasutajate ja õiguste süsteem. Korraliku all pean ma silmas konkreetsemalt seda, et oleks võimalik aine, õppejõu, küsitluste ja isegi küsimuste tasemel ära määrata, milliseid andmeid sisse loginud kasutaja näeb ja milliseid mitte. Hetkel on kasutajate süsteem lahendatud nii, et kasutatakse Oti tagasisideküsitluste süsteemi kasutajaid – ligi pääsevad kõik kellel on ligipääs küsitluste läbiviimise süsteemi administreerimiskeskkonnale.

Rakenduse edasiarenduse seisukohalt on kindlasti lisada puudujääv funktsionaalsus (sarnane võimalused teha ka õppejõudude, ainete ja küsimuste järgi andmete analüüsimiseks) ning edaspidi vastavalt vajadusele ja kasutajate soovidele lisada erinevaid võimalusi andmetest ülevaate saamiseks ja võrdlemiseks.

Statistikamooduli rakendus: http://frgp.pri.ee/rain/statistika.
Oti tagasisideküsitluste süsteem, mille kogutud andmeid statistikamoodul kasutab:

- küsitlustele vastamise pool: http://frgp.pri.ee/rain/quiz2
- administreerimise pool: http://frgp.pri.ee/rain/quiz2/admin

Summary

The main topic of the thesis is about developing statistic module for electronic query system. For that purpose I analyzed two different electronic query system: one developed by Martin Ott two years ago as his bachelor thesis and the other created by Tallinn University during the winter 2004/2005. During the analysis I mainly focus on the systems database design, but I also give an overview of the main functionality of those systems. At the end of the analysis I figure, is it possible to write an application which can be used to analyze data collected by both electronic query systems – not simultaneously, but by changing only the data layer of the application.

In this part of the thesis where I cover the methods which I use to develop the statistic module, I give overview about several software architecture patterns and explain why I chose three-layer architecture. I also describe the tools I am using for the module. Mainly I am using the same tools as electronic query systems, except statistic module does not use database to store the data. For that I use plain textfiles. Also I need some additional libraries for drawing graphs (GD library to make generating graphics on the fly and a class called JpGraph which helps to generate graphs. After that I give a more specific overview about the three layers of statistic module and describe what they are, what they do and why it is good to have them there.

Kasutatud infoallikad

- Ott, Martin. 2003. Elektrooniline tagasisideküsitluste süsteem. Tallinn.
- Tallinna Ülikool. 2004. Elektrooniline tagasisideküsitlus: õpetamise ja ainekursuste hindamine. Tallinn. Küsida IT Osakonnast.
- MySQL AB veebilehekülg. http://www.mysql.com (viimati vaadatud: 04.05.2007)
- PHP veebilehekülg. http://www.php.net (viimati vaadatud: 04.05.2007)
- Resources for Software Architects. http://www.bredemeyer.com/ (viimati vaadatud 04.05.2007)
- Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Main_Page (viimati vaadatud: 04.05.2007)
- Distributed Computing Info. http://distributedcomputing.info/ (viimati vaadatud: 04.05.2007)
- e-teatmik. http://www.vallaste.ee/index.htm (viimati vaadatud: 04.05.2007)
- PHPlot graafikute joonistamise klass. http://sourceforge.net/projects/phplot (viimati vaadatud: 04.05.2007)
- Viigipuu, Raul. 2005. Ülevaade AutomatWeb platvormist. HTML kujunduse ühendamise juhend AutomatWebiga.
- Eesti PHP portaal. http://www.php.ee. (viimati vaadatud: 04.05.2007)
- JpGraph veebilehekülg. http://www.aditus.nu/jpgraph/index.php (viimati vaadatud 04.05.2007)

Lisad

1. Andmebaasitabelite kirjeldused

Martin Oti loodud tagasisideküsitluste süsteemi andmebaasitabelite kirjeldused tabelitena. Sellisel kujul olevatest tabelite kirjeldustest saab ülevaate sellest, millisel kujul andmeid andmebaasis hoitakse. See aga lihtsustab oluliselt vajalike päringute koostamist statistikamooduli jaoks. Kuna Oti lõputöös (Ott, 2003) tabelite kirjeldusi sellisel kujul ei ole, siis koostasin need ise.

Nimi	***Andmetüüp***	***Semantika***
id	Kuni 5 kohaline positiivne lühike täisarv. Väärtus nõutav.	Süsteemi poolt genereeritav õppejõudu identifitseeriv kood. Moodustab primaarvõtme.
first_name	String pikkusega kuni 30 tähemärki.	Õppejõu eesnimi.
last_name	String pikkusega kuni 30 tähemärki.	Õppejõu perenimi.
lastupdate	Kuupäev.	Aeg, millal õppejõu andmeid viimati muudeti.

Tabel 1Õppejõudude tabel (don). Sisaldab õppejõududega seotud andmeid.

Nimi	***Andmetüüp***	***Semantika***
code	String pikkusega 7 tähemärki. Väärtus nõutav.	Aine kood. Moodustab primaarvõtme.
title	String pikkusega kuni 70 tähemärki.	Aine nimetus.
lastupdate	Kuupäev.	Aeg, millal aine andmeid viimati muudeti.

Tabel 2Ainete tabel (subject). Sisaldab ainete nimetusi.

Nimi	*Andmetüüp*	*Semantika*
id	Kuni 8 kohaline positiivne täisarv. Väärtus nõutav.	Süsteemi poolt genereeritav küsitlust identifitseeriv kood. Moodustab primaarvõtme.
year	Kuni 5 kohaline positiivne lühike täisarv.	Aasta, millal see semester toimus, mille kohta antud küsitlus on koostatud.
semester	Valikvariant – sügis või kevad.	Kas tegemist sügis- või kevadsemestriga.
course	String pikkusega kuni 10 tähemärki.	Kursuse nimetus, kellele küsitlus koostati.
codeword	String pikkusega kuni 25 tähemärki.	Küsitlusele pandud kood, mille abil pääseb küsitlusele vastama.
password	String pikkusega kuni 25 tähemärki.	Salasõna. Et vastata saaksid ainult need, kes seda teavad.
startdate	Kuupäev.	Aeg, millal küsitlusele saab vastama hakata.
enddate	Kuupäev.	Aeg, millal küsitlusele ei saa enam vastata.
explanation	String pikkusega kuni 255 tähemärki.	Küsitluse juurde käiv selgitus.
creatdate	Kuupäev.	Aeg, millal küsitlus kokku pandi.
usr_group	Kuni 8 kohaline positiivne täisarv.	Ei kasutata.
lastupdate	Kuupäev.	Aeg, millal küsitluse andmeid viimati muudeti.

Tabel 3Küsitluste tabel (query). Sisaldab küsitlustega seotud andmeid.

Nimi	*Andmetüüp*	*Semantik*
id	Kuni 8 kohaline positiivne täisarv. Väärtus nõutav.	Süsteemi poolt genereeritav küsimust identifitseeriv kood. Moodustab primaarvõtme.
type	Kuni 3 kohaline lühike täisarv. Väärtus nõutav.	Määrab ära küsimuse tüübi. Süsteemis on nelja tüüpi küsimusi: hinnang (kood 20), tekst (kood 10), üksik-valik (kood 30) ja mitmik-valik (kood 40). (Ott 2003, 23)
text	String pikkusega kuni 120 tähemärki. Väärtus nõutav.	Küsimuse tekst.
explanation	String pikkusega kuni 255 tähemärki.	Küsimuse juurde käiv selgitav tekst.
created	Kuupäev.	Aeg, millal küsimus lisati.
originusr	String pikkusega kuni 16 tähemärki.	
lastupdate	Kuupäev.	Aeg, millal küsimuse andmeid viimati muudeti.

Tabel 4: Küsimuste tabel (question). Sisaldab küsimustega seotud andmeid.

Nimi	*Andmetüüp*	*Semantika*
code	Kuni 8 kohaline positiivne täisarv. Väärtus nõutav.	Süsteemi poolt genereeritav vastuse teksti identifitseeriv kood. Moodustab primaarvõtme.
text	Väärtus nõutav.	Vastuse tekst.

Tabel 5Tekst-tüüpi küsimuste vastuste tabel (result_text). Salvestatakse tekst tüüpi küsimustele antud vastused. Juhul kui vastuse tekst on juba tabelis olemas, siis seda vastust enam topelt ei salvestata, vaid vastuste tabelis (Tabel 5) hakkab vastuse kood (väli code) viitama olemas olevale tekstile.

Nimi	*Andmetüüp*	*Semantika*
query_id	Kuni 8 kohaline positiivne täisarv. Väärtus nõutav.	Küsitluse id. Moodustab primaarvõtme.
question_id	Kuni 8 kohaline positiivne täisarv. Väärtus nõutav.	Küsimuse id. Moodustab primaarvõtme.
subject_code	String pikkusega kuni 7 tähemärki. Väärtus nõutav.	Aine kood. Moodustab primaarvõtme.
don_id	Kuni 5 kohaline positiivne lühike täisarv. Väärtus nõutav.	Õppejõu id. Moodustab primaarvõtme.
form	Valik, kas loeng, praktikum, seminar või ei panda midagi. Väärtus nõutav.	Aine vorm. Moodustab primaarvõtme.
sess_id	Kuni 16 kohaline binaarne andmetüüp.	Sessiooni id. Moodustab primaarvõtme.
code	Kuni 8 kohaline positiivne täisarv. Väärtus nõutav.	Vastuse kood. Vastavalt küsimuse tüübile, on selle välja väärtuseks kas hinne (hinnang tüüp küsimus), valikvastustega küsimuse variandi kood variantide tabelis või vabatekst tüüpi küsimuse puhul vastuse id vabatekst-tüüpi küsimuste vastuste tabelis (Tabel 5).Moodustab primaarvõtme.

Tabel 6Vastuste tabel (result_code). Tabel kuhu salvestatakse küsitlusele vastaja poolt sisestatud vastused.

Nimi	*Andmetüüp*	*Semantika*
qst_id	Kuni 8 kohaline positiivne täisarv. Väärtus nõutav.	Küsimuse id, mille vastuse variandiga on tegemist. Moodustab primaarvõtme.
nr	Kuni 3 kohaline positiivne lühike täisarv. Väärtus nõutav.	Süsteemi poolt genereeritav ühe küsimuse variante eraldav kood. Moodustab primaarvõtme.
text	String pikkusega kuni 120 tähemärki. Väärtus nõutav.	Küsimuse variandi tekst.

Tabel 7Valikvastustega küsimuste vastusevariantide tabel (qstdata).

Nimi	*Andmetüüp*	*Semantika*
id	Kuni 8 kohaline positiivne täisarv. Väärtus nõutav.	Süsteemi poolt genereeritav vastuse teksti identifitseeriv kood. Moodustab primaarvõtme.
query_id	Kuni 8 kohaline positiivne täisarv.	Küsitluse kood.
creation_date	Kuupäev.	Vastuse salvestamise aeg.
auth_str	String pikkusega kuni 32 tähemärki	

Tabel 8Uus vastamiste salvestamise tabel (newresult). Sisaldab infot küsitlusele vastamiste kohta.

Nimi	*Andmetüüp*	*Semantika*
result_id	Kuni 8 kohaline positiivne täisarv. Väärtus nõutav.	Vastamise id.
subject_code	String pikkusega kuni 7 tähemärki. Väärtus nõutav.	Aine kood.
don_id	Kuni 5 kohaline positiivne lühike täisarv. Väärtus nõutav.	Õppejõu kood.
form	Valik, kas loeng, praktikum, seminar või ei panda midagi. Väärtus nõutav.	Aine vorm.
question_id	Kuni 8 kohaline positiivne täisarv. Väärtus nõutav.	Küsimuse id.
code	Kuni 8 kohaline positiivne täisarv. Väärtus nõutav.	Vastuse kood. Vastavalt küsimuse tüübile, on selle välja väärtuseks kas hinne (hinnang tüüpi küsimus), valikvastustega küsimuse variandi kood variantide tabelis või vabatekst tüüpi küsimuse puhul vastuse id vabatekst-tüüpi küsimuste vastuste tabelis (Tabel 5).

Tabel 9Uus vastuste salvestamise tabel (newresult_code). Sisaldab küsitlusele antud vastuseid ja viidet kirjele vastamiste tabelis (Tabel 8).

Nimi	*Andmetüüp*	*Semantika*
query_id	Kuni 8 kohaline positiivne täisarv. Väärtus nõutav.	Küsitluse id. Moodustab primaarvõtme.
qst_id	Kuni 8 kohaline positiivne täisarv. Väärtus nõutav.	Küsimuse id. Moodustab primaarvõtme.
optional	Valik, kas „no“ (ei) või „yes“ (jah).	Kas küsimusele vastamine on kohustuslik või vabatahtlik. Vaikimisi on väärtus „no“ (ei).
nr	Kuni 5 kohaline positiivne täisarv. Väärtus nõutav.	Küsimuse järjekorra number küsitluse piires.

Tabel 10: Küsitlusi ja küsimusi siduv tabel (qry_qst). Määrab ära millised küsimused küsitluses esinevad, kas küsimusele vastamine on kohustuslik või vabatahtlik ja mis järjekorras küsimused küsitluses paiknevad.

Nimi	*Andmetüüp*	*Semantika*
query_id	Kuni 8 kohaline positiivne täisarv.	Küsitluse kood. Moodustab primaarvõtme.
subject_code	String pikkusega kuni 7 tähemärki. Väärtus nõutav.	Aine kood. Moodustab primaarvõtme.
don_id	Kuni 5 kohaline positiivne lühike täisarv. Väärtus nõutav.	Õppejõu kood. Moodustab primaarvõtme.
form	Valik, kas loeng, praktikum, seminar või ei panda midagi. Väärtus nõutav.	Aine vorm. Moodustab primaarvõtme.

Tabel 11: Seosetabel, mis määrab ära millised ained ja õppejõud küsitluses esinevad (qrylecture).

Nimi	*Andmetüüp*	*Semantika*
don_id	Kuni 5 kohaline positiivne lühike täisarv. Väärtus nõutav.	Õppejõu kood. Moodustab primaarvõtme.
subject_code	String pikkusega kuni 7 tähemärki. Väärtus nõutav.	Aine kood. Moodustab primaarvõtme.

Tabel 12: Seosetabel, mis määrab ära seosed ainete ja õppejõudude vahel (lecture).

Nimi	*Andmetüüp*	*Semantika*
usrname	String pikkusega kuni 16 tähemärki. Väärtus nõutav.	Kasutajanimi, mille abil saab administreerimiskeskkonda logida. Moodustab primaarvõtme.
passwd	String pikkusega kuni 255 tähemärki. Väärtus nõutav.	Parool.
role	Valik – admin, asjaajaja või mõlemad. Väärtus nõutav.	Kasutaja roll administreerimiskeskkonnas.
first_name	String pikkusega kuni 30 tähemärki.	Kasutaja eesnimi.
last_name	String pikkusega kuni 30 tähemärki.	Kasutaja perekonnanimi.
creat_date	Kuupäev	Kasutaja loomise aeg.
last_login	Kuupäev	Kasutaja viimase logimise aeg.
lastupdate	Kuupäev	Aeg millal kasutaja andmeid viimati uuendati.
created	Kuupäev	Kasutaja loomise aeg.

Tabel 13: Kasutajate tabel (user) kus on kirjas info nende kasutajate kohta, kes omavad ligipääsu administreerimisliidesesse ja millist rolli kasutaja seal omab.

Nimi	*Andmetüüp*	*Semantika*
sess_id	String pikkusega kuni 16 tähemärki. Väärtus nõutav.	PHP sessiooni id. Moodustab primaarvõtme.
usrname	String pikkusega kuni 16 tähemärki.	Kasutajanimi, kes on sisse loginud.
sess_data	Andmed binaarkujul.	PHP sessioonis olevad andmed. Väli ei ole kasutuses.
auth_str	String pikkusega kuni 32 tähemärki.	Arvuti võrguaadress, kust kasutaja on sisse loginud.
last_update	Kuupäev	Aeg millal sessiooni andmeid viimati uuendati.

Tabel 14: Sessioonide tabel (session), kus on kirjas, millised kesutajad on parasjagu sisse loginud.

2. CD plaat

- Lõputöö teksti elektrooniline versioon
- Lõputöö raames loodud statistikamooduli rakenduse lähtekood
- Oti loodud tagasisideküsitluste süsteemi lähtekood koos andmebaasiga (Oti poolt parandatud ja muudetud versioon)