Tuukka Talvitie

VERTAILUKONE

Tekniikan ja merenkulun koulutusohjelma
Ohjelmoinnin suuntautumisvaihtoehto
2012

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

VERTAILUKONE

Talvitie, Tuukka
Satakunnan ammattikorkeakoulu
Tekniikan ja merenkulun koulutusohjelma
Joulukuu 2012
Ohjaaja: Aarinen, Reino
Sivumäärä: 58
Liitteitä: 3

Asiasanat: Koulutus, koulutusohjelma, soveltuvuus, vertailu, vertailukone, Internet.

Työn tarkoituksena oli tehdä verkossa toimiva vertailukone kysymys- ja vastaustietokannalla. Idea on lähtöisin erään nettisivuston halusta saada vaihtoehto lisensioidulle vertailukoneelle. Kyseessä on verkkopohjaisilla syöttö- ja hakulomakkeilla käytettävä vertailukone, joka vertaa insinööriksi halukkaan henkilön vastauksia tietokannassa oleviin, esimerkiksi opettajien antamiin vastauksiin, ja laskee vastauksia vertailemalla sopivimman koulutusohjelman kyselyn tekijän ilmoittamien mieltymysten ja taitojen mukaisesti laskevassa järjestyksessä (prosenttiluvut).

VERTAILUKONE

Talvitie, Tuukka
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Faculty of Technology and Maritime Management
December 2012
Supervisor: Aarinen, Reino
Number of pages: 58
Appendices: 3

Keywords: Education, training programme, aptitude, comparison, comparison engine, Internet.

The purpose of this work was to create a comparison engine that works online with a questions and answers database. The idea came from the desire of a certain web site to get an alternative for licensed engine of the same purpose. This engine uses online input and search forms which are used to determine the most suitable training programmes for a user by comparing user input (traits and personal preferences) to training programmes representites answers and from these to calculates the most suitable training programmes in descending order (percentual figures).

SISÄLLYS

SYMBOLI- JA TERMILUETTELO

- Web (World Wide Web eli WWW). Internet-verkossa toimiva hajautettu hyperjärjestelmä.

- Hyperteksti sisältää linkkejä toisiin dokumentteihin. Vuorovaikutus web-palvelimen kanssa informaatiota toiseen suuntaan lähettämällä on sekin mahdollista. Hypertekstiä luetaan selaimella, joka hakee web-sivuiksi kutsuttuja dokumentteja web-palvelimilta ja esittää niitä käyttäjälle piirtämällä ne näytölle.

- Internet on maailmanlaajuinen tietoverkko, joka yhdistää paikallisia tietoverkkoja toisiinsa.

- Verkkosivu eli web-sivu, www-sivu tai arkikielessä nettisivu tarkoittaa maailmanlaajuisessa verkossa eli Internetissä julkaistua sivua.

- Tietoverkko on tietokoneiden ja niiden välisten tiedonsiirtoyhteyksien sekä näiden molempien avulla tarjottavien palvelujen yhdistelmä. Osan tietoverkon tarvitsemasta tiedonsiirtoyhteydestä voi muodostaa tietoliikenneverkko.

- Tietoliikenneverkko voi käsittää muitakin laitteita ja tiedonsiirtoprotokollia sisältäviä tietoverkkoja, kuten esimerkiksi puhelinverkon.

- WWW-selain (puhekielessä usein pelkkä selain) on tietokoneohjelma, joka antaa käyttäjänsä katsella ja lähettää tekstiä, kuvia ja muita WWW-sivuilta löytyviä tietoja.

- Palvelimella tarkoitetaan tietoliikenteen yhteydessä tietokoneessa suoritettavaa palvelinohjelmistoa, sekä tällaista ohjelmistoa suorittavaa tietokonetta. Palvelinohjelmistojen tehtävänä on tarjota erilaisia palveluja muille ohjelmille joko tietokoneverkon välityksellä tai paikallisesti samassa tietokoneessa. Palvelinta käyttävää sovellusta tai tietokonetta nimitetään asiakkaaksi.

- (Verkko)Sivusto: Verkkosivusto eli WWW-sivusto on tietyn organisaation tuottama tai tiettyä aihetta käsittelevä ja siten selkeän kokonaisuuden muodostava verkkosivujen joukko.

- URI (Uniform Resource Identifier). Merkkijono, jolla kerrotaan tietyn tiedon paikka (URL) tai yksikäsitteinen nimi (URN).
-
- URL: Erityisesti URI-merkkijonon erikoistapausta URL (Uniform Resource Locator) käytetään osoittamaan WWW-sivuja.

- Hash (#): Risuaita erottaa URL-merkkijonon lopusta vaihtoehtoisen osan, joka osoittaa pääresurssin alisteiseen resurssiin. Tämän työn tapauksessa se osoittaa pääliukusäätimen yhteen paneeliin, joita on neljä kappaletta.

- JVM (Java Virtual Machine). Toisin kuin tavanomaiset ohjelmointikielet, Java käännetään useimmin tavukoodiksi, joka suoritetaan virtuaalikoneessa (JVM).

- Java EE (Enterprise Edition) on spesifikaatio, jota sovelluspalvelimien ja ohjelmistokehitysympäristöjen tulee noudattaa Java-kielen kanssa. Erityisesti yrityskäyttöön räätälöity.

- Avoin lähdekoodi (Open Source) tarkoittaa tietokoneohjelmien tuottamis- ja kehitysmenetelmiä, jotka tarjoavat käyttäjälle mahdollisuuden tutustua ohjelman lähdekoodiin ja muokata sitä omien tarpeidensa mukaisesti. Avoimen lähdekoodin periaatteisiin kuuluu myös oikeus käyttää ohjelmaa mihin tahansa, kopioida ja levittää sekä alkuperäistä, että muokattua

versiota. Käytännössä avoin lähdekoodi tarkoittaa samaa asiaa kuin vapaat ohjelmistot, joista puhuttaessa halutaan usein painottaa ohjelmistokehityksen eettisiä ulottuvuuksia.

- MySQL Community Server. MySQL-tietokanta on hyvin suosittu web-palveluiden tietokantana. MySQL-tietokannan päälle rakennettava ohjelmalogiikka tehdään usein skriptiohjelmointikielellä (PHP, Python tai Perl), sivut julkaistaan Apache web-palvelimella, joka edelleen toimii Linux-käyttöjärjestelmän päällä. Tätä kutsutaan joskus LAMP-alustaksi. MySQL Community Server on muuhun kuin yrityskäyttöön tarkoitettu MySQL tietokantojen palvelinohjelmisto.

- Gibitavu [GiB] on yksikkö, joka on lähellä gigatavua. Yksi gibitavu on 230 tavua, kun gigatavu on 109 tavua. Yksi gibitavu on siis suunnilleen 1,073 gigatavua.

- Ajax (lyhennesana sanoista Asynchronous JavaScript And XML) on joukko web-sovelluskehityksen tekniikoita, joiden avulla web-sovelluksista voi tehdä vuorovaikutteisempia. Alkuperäisessä merkityksessään Ajax:lla on alun perin viitattu tekniikkaan, jossa verkkosivulla JavaScriptillä asynkronisesti tehtävistä HTTP-pyynnöistä palautetaan XML-merkkausta. Nykyisin Ajax-tekniikoilla viitataan yleisesti samankaltaiseen toimintatapaan.

- Moduuli tarkoittaa tietokoneohjelman itsenäistä osaa, jolla on syöttötiedot, tulostiedot ja oma toiminnallinen tehtävä (funktio).

- HTML5 (Hypertext Markup Language) on HTML-merkintäkielen viimeisin kehitysvaihe. Merkintäkielillä kuvataan tekstin rakennetta tai esitystapaa metainformaatiolla (tietoa dokumentin sisällöstä, muodosta, sijainnista ja ylläpidosta). Kielen viidenteen versioon liittyvät kiinteästi myös uudet CSS3- ja JavaScript-tekniikat. Sen olennaisena tavoitteena on vähentää webin riippuvuutta Flashista, kerätä tarvittavat tekniikat yhteen ja palauttaa internet tilaan, jossa sisältö perustuu kokonaan avoimiin standardeihin. [1]

- CSS3 (Cascading Style Sheets) ovat porrastettuja tyyliarkkeja (tekstitiedostoja), joilla ehdotetaan, miten niiden kuvaama dokumentti

voidaan esittää (muotoilu). CSS3 on viimeisin ja vielä kehitteillä oleva kolmosversio standardista.

- Skriptikielet ovat komentosarjakieliä, joilla voidaan automatisoida tehtäviä ilman, että tarvitsee käyttää varsinaisia ohjelmointikieliä. Skriptikieli ei aina käytä tulkkia, mutta käytäntö on yleinen.

- JavaScript on skriptikieli, joka on pääasiassa käytössä web-ympäristössä. Kieli on dynaamisesti tyypittyvä, tulkattava, olipohjainen skriptikieli.

- Olio-ohjelmointi (object-oriented programming) on ohjelmoinnin lähestymistapa, jossa ohjelmointiongelmien ratkaisut jäsennetään olioiden yhteistoimintana. Jokainen olio voidaan nähdä itsenäisenä pienenä koneena, jolla on tietty rooli tai vastuu.

- Dynaamisesti tyypittyvässä ohjelmointikielessä muuttujien tyyppi voi muuttua ajonaikaisesti, staattisessa vastaavasti eivät.

- Ohjelmointikielen tulkki on tietokoneohjelma, joka suorittaa ohjelmointikielisiä lauseita yksi kerrallaan. Tämä poikkeaa ohjelmointikielen kääntäjästä, joka kääntää koko lähdekoodin konekieliseksi ohjelmaksi sen myöhempää suorittamista varten.

- Konekieli on tietokoneen suorittimen ymmärtämä kieli. Konekieli koostuu sarjasta konekielisiä käskyjä, jotka esitetään biteillä 0 ja 1.

- jQuery on kaikille selaimille tarkoitettu ilmainen, avoimen lähdekoodin JavaScript-kirjasto.

- DOM (Document Object Model) on ohjelmointirajapinta ja tiedon esittämismalli, joka mahdollistaa HTML-dokumenttien sisällön muokkauksen. JavaScriptin kanssa sillä voidaan toteuttaa vuorovaikutteisia www-sivuja, jotka eivät vaadi jatkuvaa palvelinyhteyttä. DOM:ssa dokumentin tieto esitetään puumaisessa tietorakenteessa.

- MD5 on algoritmi, joka tuottaa tuloksenaan 128-bittisen tiivisteen, mikä tyypillisesti esitetään 32-merkkisenä heksakoodatussa muodossaan. Tällä tavalla tallennettua salasanaa ei voida käyttäjälle palauttaa siinä tapauksessa

että käyttäjä unohtaa salasanansa, sillä MD5-tiivisteestä ei pysty luomaan alkuperäistä merkkijonoa. On luotava uusi salasana.

- HTTP (Hypertext Transfer Protocol eli hypertekstin siirtoprotokolla) on protokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon. Protokolla perustuu siihen, että asiakasohjelma (esimerkiksi selain) avaa TCP-yhteyden palvelimelle ja lähettää pyynnön. Palvelin vastaa lähettämällä sopivan vastauksen, tavallisimmin HTML-sivun tai binääridataa kuten kuvia, ohjelmia tai ääntä.

- TCP/IP (Transmission Control Protocol / Internet Protocol) on usean Internet-liikennöinnissä käytettävän tietoverkkoprotokollan yhdistelmä.

- TCP (Transmission Control Protocol) on tietoliikenneprotokolla, jolla luodaan yhteyksiä tietokoneiden välille, joilla on pääsy Internetiin. TCP-yhteyksien avulla tietokoneet voivat lähettää toisilleen tavujonoja luotettavasti. TCP-protokolla pitää myös huolta, että paketit saapuvat perille oikeassa järjestyksessä. Tarvittaessa hävinnyt paketti voidaan lähettää uudestaan. Tätä tarkoitusta varten TCP-protokollaan on kehitetty erilaisia vuonvalvonta- ja ruuhkanhallintamekanismeja. Suurin osa Internetin liikenteestä perustuu TCP-protokollaan ja koko TCP/IP-protokollaperhe on saanut nimensä TCP-protokollan perusteella.

- IP (Internet Protocol). IP-protokolla on alemman tason protokolla, joka vastaa päätelaitteiden osoitteistamisesta ja datapakettien reitittämisestä verkossa. Sen päällä voidaan ajaa useita muita verkko- tai kuljetuskerroksen protokollia, joista TCP-protokolla on eräs yleisimmistä.

- Port (suomennos Portti) eli portit ovat TCP/IP-protokollia käyttävissä tietokoneissa olevia numeroituja palvelupisteitä. Palvelu voi kytkeytyä odottamaan yhteyttä johonkin hyvin tunnettuun porttiin. Käyttäjätkin joutuvat varaamaan portin siitä koneesta, josta he ottavat yhteyttä. Tämä porttinumero on satunnainen. Asiakaspään TCP-yhteydelle arvotaan satunnainen porttinumero, johon palvelin lähettää paluupaketit. Satunnaiset portit ovat väliltä 1024–65535.

- Socket (suomennos Pistoke) on UNIX-järjestelmissä keskeinen tietoliikenneyhteyttä kuvaava, käyttöjärjestelmän tarjoama kaksisuuntainen

ohjelmointirajapinta, joka mahdollistaa kahden mahdollisesti eri isäntäkoneella olevan prosessin välisen kommunikoinnin. Prosessille pistoke näkyy abstraktiona, jota voi lukea ja johon voi kirjoittaa tiedoston tapaan. Käyttäjän näkökulmasta jokaista UNIX-järjestelmässä kulkevaa tietoliikenneyhteyttä ja jokaista palvelinsovellusta vastaa yksi pistoke. Käyttöjärjestelmä ohjaa pistokkeeseen kuuluvan verkkoliikenteen pistokkeen omistavalle prosessille. Tunnistus tapahtuu porttinumeron perusteella. Pistokkeen ei tarvitse välttämättä kuljettaa TCP/IP-liikennettä, vaan sen alemman tason protokollaksi voidaan valita myös esimerkiksi IPX tai myös UNIX domain socket, jolloin ei ole kyse verkkoliikenteestä, vaan paikallisella koneella tapahtuvasta prosessienvälisestä kommunikoinnista.

- LAMP on kokoelma avoimen lähdekoodin ohjelmia, jotka yhdessä muodostavat WWW-palvelimen, jonka alla voidaan suorittaa dynaamisia web-sivuja. LAMP sisältää ohjelmat: Linux, avoimen lähdekoodin käyttöjärjestelmäydin. Apache, avoimen lähdekoodin web-palvelin. MySQL, avoimen lähdekoodin tietokantarajapinta. PHP, Perl ja/tai Python, avoimen lähdekoodin komentosarjakieli.

- Kirjasto (tietotekniikka)(muun muassa ohjelmakirjasto, luokkakirjasto) ovat kokoelmia, aliohjelmia, luokkia ja/tai ohjelmia, joita käytetään tietokoneohjelmien modulaarisessa kehittämisessä sekä ohjelmien suorittamisen aikana.

- Tagi (HTML) on ohjelmakomento, jolla kerrotaan miten tieto HTML-sivulla tulostuu.

- Validointi tarkoittaa tiedon oikeellisuuden varmistamisen prosessia.

- Pääavain (tietokantataulu) on kenttä tai kenttien yhdiste, jolla jokainen taulun tietue (record) voidaan tunnistaa yksikäsitteisesti.

- Istunto (engl. session) on tietotekniikassa käytäntö, jolla luodaan pysyvä yhteys protokollaan (kuten yhteys TCP/IP:n sovelluskerrokseen) tai yhteys käyttäjän (tai käyttäjän selaimen) ja palvelimen välillä. Yleisimmin istunto on sisällytetty protokollan sovelluskerrokseen, mutta myös ohjelmallinen lähestymistapa on yleisesti käytössä etenkin istuntoja tukemattomissa

protokollissa (kuten UDP) ja protokollissa joiden istunnon kesto on hyvin lyhytaikainen (kuten HTTP).

- Ohjelmallinen istunto tarkoittaa istuntoa joka luodaan päätteissä protokollan ulkopuolella.

- PHP sisältää sisäänrakennetun istuntojärjestelmän (PHP-istunto), joka yleisesti käyttää hyväkseen evästeitä, mutta osaa sijoittaa istuntotunnuksen myös jokaiseen PHP-kielen generoimaan hyperlinkkiin tukien näin periaatteessa myös selaimia, joissa ei ole (kytketty päälle) tukea JavaScriptille. PHP-istunto tallettaa erikoismuuttujaan $_SESSION kaikki koodin määrittelemät arvot, tukien näin samoja toimintoja kuin evästeet, sekä toimintoja jotka ovat arkaluontoisia kuten: salasanat, suojattujen sivujen suojatasot ja niin edelleen. PHP-istunto merkitään inaktiiviseksi selaimen viimeisen ikkunan sulkeuduttua, eikä istuntoa voi tallentaa ilman ulkopuolista ohjelmaa tai tietoa (kuten selaimen istuntotallennusmekanismia tai evästettä).

- Eväste (joskus myös keksi, engl. cookie) on dataa, jonka web-palvelin tallentaa käyttäjän tietokoneelle. Selain lähettää evästeet vain kyseiselle palvelimelle.

- Unicode on tietokonejärjestelmiä varten kehitetty merkistöstandardi ja käytännössä sama kuin yleismaailmallisen merkistön (engl. Universal Character Set, UCS) määrittävä kansainvälinen standardi ISO/IEC 10646. Unicode määrittää yksilöivän koodiarvon yli 100 000 kirjoitusmerkille.

- ISO 8859-1 tai epävirallisesti Latin 1 on tietokoneissa ja tietoliikenteessä käytettävä merkistö, joka on laadittu erityisesti länsieurooppalaisia kieliä silmällä pitäen.

1 JOHDANTO

Tällä hetkellä nettisivujen toteutus ja muotoilu ovat murroksessa (tosin niin voi kyllä tietotekniikassa sanoa aina). Toisaalta molemmilla osa-alueilla on sekä uusia toteutustapoja, että rasitteena moninaisten selaimien ja niiden eri versioiden yhteensopivuusongelmat. Selaimet ja etenkin niiden aiemmat versiot tukevat uudempia toteutustekniikoita varsin vaihtelevasti. Sivujen muotoilusta tulee helposti jossittelua ja varmistelua. Tee tällä selaimella näin, sen versio x:llä noin, toisella selaimella taas tällä tavalla ja vielä lopuksi varmistus, että sivu ylipäätään toimisi jotenkuten jokaisella. Sivujen suunnittelussa ja toteutuksessa tarvitseekin päättää, kuinka pitkälle menneeseen kyseisen sovelluksen/sivuston yhteensopivuudessa kannattaa ja on valmis menemään. Toteutuspuolen suuri mullistus on osittain jo käytössä oleva ja etenevä HTML5-teknologia ja muotoilupuolella viimeisempiä temppuja tarjoaa CSS3. JavaScript skriptikielen kirjastolla jQuery on taas yritetty helpottaa toimintoja, animaatioita, DOM-elementtien käsittelyä ja Ajax-sovelluksia eli dynaamisuutta nettisivustoilla. ”Html5 ja css3 tuovat sivujen visuaaliseen esittämiseen lukusia muitakin parannuksia, kuten ladattavat fontit, pyöristetyt reunat, varjostetut tekstit ja 3d-grafiikan. Hiljalleen nämä ominaisuudet täyttävät niitä aukkoja, joiden vuoksi Flashiin joudutaan nykyään turvautumaan visuaalisemmissa toteutuksissa. [1]”

Kuten jokainen isompi projekti, niin myös tämä aloitettiin suunnittelemalla. Huolellinen suunnittelu on erityisen tärkeää ohjelmistoalalla, koska monesti ohjelmisto jaetaan osiin (esimerkiksi moduulit, olio-ohjelmointi) työn jakamisen ja/tai arkkitehtuurin vuoksi. Hyvin tehdyistä suunnitteludokumenteista jokainen tiimin jäsen näkee koko ohjelman, mahdollisen oman osa-alueensa ja näiden rajapintojen määrittelyt (ellei tietenkin jotkin osat ole tarkoituksella salattu, esimerkiksi alihankkijalta).

Tässä työssä tätä periaatetta rikottiin useammastakin syystä, vaikka hyvin suunniteltu on yhden hengen ”tiimissäkin” suuri etu. Mitä tarkemmin sovellus on suunniteltu, sitä vähemmän tulee tehtyä turhaa työtä, noloja unohduksia ja työkin etenee nopeammin, kun pystyy kohta kohdalta etenemään kohti valmista sovellusta. Esitutkimuksen määrä on projektin lähtökohdasta ja tekotavasta johtuen vähäinen.

Määrittelyn ja suunnittelun hengessä ja kuvaliitteissä on kylläkin karkeasti määritelty ohjelman toimintaa, mutta yksi tärkeä osa-alue eli toteutustekniikka jätetään usein tarkemmin määrittelemättä. Tämä johtuu yksinkertaisesti siitä, että vaikka tietyt toteuttamistyökalut ja tekniikat on määritelty etukäteen (muun muassa ohjelmointikielet- ja kirjastot), niin työn tekijällä ei ole niin mittavaa kokemusta vastaavankaltaisesta sovelluksesta, että olisin voinut jo etukäteen tarkasti määrittää käytettävät tekniikat jokaisessa tilanteessa. Tässä projektissa sovellettiin opittua, opittiin myös paljon uutta ja hahmotettiin tällaisen ison projektin vaativuutta taas hieman paremmin. Yksi henkilö voi vielä vapaasti soveltaa sitä mukaa kuin etenee.

Idea opinnäytetyöhön tuli koulutusohjelmajohtajan (Ylikoski, Tietotekniikka) puolelta. Tarkoitus olisi ollut tehdä eräälle nettisivustolle vertailukone, jolla vierailijat pystyisivät kysymyssarjojen avulla määrittämään itselleen sopivamman koulutusohjelman. Sivuston toimijat päättivät kuitenkin samanaikaisesti jatkaa kaupallisen vaihtoehdon (joka oli siis jo käytössä) sopimusta, jonka seurauksena opinnäytetyö sovittiin niin, että teen kyseisen vertailukoneen itselleni. Olin kuitenkin jo saanut materiaalia siitä, millaista vertailukonetta sivusto olisi itselleen toivonut ja käytinkin tätä materiaalia suuntaa-antavasti hyväksi omassa työssäni. En ole kysynyt, saako tätä materiaalia liittää työhön. Tämä ei ole kuitenkaan kovin tärkeä yksityiskohta, koska lähinnä vain lainasin materiaalista vinkin jQuery-liukusäätimen käyttämisestä kysymyksiin vastaamisessa. Tästä pääsemmekin alkamaan/jatkamaan projektin niin sanottua esitutkimusvaihetta. Mikä on asiakkaan (allekirjoittanut) ongelma, miksi se on olemassa ja mitä vaihtoehtoja on sen ratkaisemiseksi ilman tai tietotekniikan avulla.

2 MÄÄRITTELY JA SUUNNITTELU

2.1 Esitutkimus

Yleensä ohjelmointiprojekti lähtee liikkeelle siitä, että on ongelma. Joskus on selvää, että ongelman ratkaisu voi olla vain tietotekninen. Kuitenkin, kun esimerkiksi

ohjelmointifirmalle annetaan tehtäväksi esitutkimus, tulee pohdinta aloittaa siitä, vaatiiko ongelman ratkaisu nimenomaan (uutta) tietotekniikkaa, vai voidaanko ongelma ratkaista ilman sitä tai muuntamalla vanhoja ratkaisuja/ohjelmistoja. Oikeastaan voisi sanoa, että ongelmani oli opinnäytetyöidean saaminen ja ongelman syynä vaatimus, että tutkinnon hyväksyttävä suorittaminen vaatii opinnäytetyön. Lisäksi koska olen lukenut tietotekniikkaa ja erityisesti ohjelmointia, niin ongelma tulee ratkaista tietotekniikalla ja todennäköisin suuntautumisvaihto työlle siis ohjelmoinnin puolelta. Työn kohteeksi valikoitui Johdannossa kuvatulla tavalla netissä toimiva vertailukone. Nyt kun nämä ongelmat oli edellä mainitulla tavalla ratkaistu, niin päästiin seuraavaan vaiheeseen.

2.1.1 Esitutkimusraportti

Tällainen dokumentointi aloitetaan yleensä lähtötilanteen kuvaamisella, jota edellä onkin jo hyvin pohjustettu. Selväksi on tullut ongelma, idea sen ratkaisemiseksi sekä se, että työ lähtee niin sanotusti puhtaalta pöydältä. En kuitenkaan työn tarkoituksen vuoksi (yhden hengen opinnäytetyö) kirjoittanut erillistä esitutkimusraporttia (vaatimusmäärittely tai tarvemäärittely), joka paljolti toistaisi nyt ja myöhemmin tulevia tietoja, vaan sulautan esitutkimuksen linjaan sovelluksen määrittelyn ja suunnittelun kanssa.

Tällöin pitääkin lähteä liikkeelle siitä, millaiset ovat toiminnalliset vaatimukset, vaadittavat ominaisuudet sekä näiden priorisointi. Heti määrittelyn aluksi päätinkin, että seuraavat ovat pakolliset ominaisuudet vertailukoneessani:

- Ensinnä tietenkin itse vertailu eli kyselyn tekijän vastauksien vertailu koulutusohjelmien edustajien vastauksiin. Tästä saadaan kyselyn tekijän soveltuvuusprosentti. 100% tarkoittaa, että henkilö on täydellinen koulutusohjelmaan ja 0% taas, että sopivuus on olematon.

- Järjestelmäkäyttäjän työn helpottamiseksi sivustolle ainakin taulujen luonti ja poisto (miksei välttämättä enempää, tästä lisää myöhemmin, MySQL Workbench ja kappale 2.2.2).

- Muille käyttäjille, eli koulutusohjelmien edustajille taulujen täyttäminen ja päivitys.

- Lisäksi sivustolle eri käyttäjäryhmille omat ohjeet sovelluksen yleiseen käyttöön liittyen.

Hyödyllisiä ominaisuuksia olisivat lisäksi:

- Kielen valinta. Esimerkiksi lisäyksenä englanninkieliset koulutusohjelmat ja englanninkielinen versio ohjelmasta.

- Lisää online-hallintaominaisuuksia järjestelmäkäyttäjälle.

- Salasanan uusinta myös koulutusohjelmien edustajille itselleen.

Hyödylliset ominaisuudet olisivat mukavia lisiä sovellukseen, mutta eivät toiminnan kannalta välttämättömiä. Nämä toiminnot voisivat kenties olla markkinataloudellisesti maksullisia päivityksiä/parannuksia.

Rajoituksina toteutukselle ei tässä tapauksessa ole budjetti, laitteisto eikä olemassa oleva aiempi toteutus, mutta internet-ympäristö eli sen standardit ja tekijän osaamat toteutustekniikat, jotka näitä standardeja noudattavat. Lähdin liikkeelle varsinaiseen sovelluksen määrittelyyn siltä pohjalta, että toteuttaisin vertailukoneeni käyttämällä hyväksi nettisivujen HTML-merkkausta, erillisiä tyylitiedostoja (CSS), tietokantoihin liittyen PHP komentosarjakieltä yhteistyössä MySQL-tietokantaohjelmiston kanssa, sekä dynaamisimpiin osioihin JavaScriptiä (ja sen jQuery-kirjastoa). HTML-merkkausta tarvitsee melkein sivulla kuin sivulla ja erilliset tyylitiedostot tekevät jo muutenkin useampaa ohjelmointi– ja merkkauskieltä sisältävistä sivutiedostoista paljon selkeämpiä lukea, helpompia ylläpitää (Elementtien tunnisteet ja luokat, ID ja Class), sekä huomattavasti vähentää työmäärää monimutkaisemmilla sivuilla, kun tehdään muotoiluun muutoksia HTML-elementtien luokkien avulla. Työssä on noudatettu sivujen tekemisessä seuraavaa ohjesääntöä:

- HTML/PHP sivulla oletusarvoisesti vain HTML- ja PHP-kieltä.
- JavaScript- ja jQuery koodi erillisiin tiedostoihin.

- Sivujen muotoilu kokonaan erillisissä tyylitiedostoissa (CSS).

Ohjesäännöstä poikettiin vain, jos se on havaittu tarpeelliseksi sivun toiminnan ymmärtämiseksi. Itse olen tottunut tietokantatyöskentelyssä käyttämään ilmaista MySQL-tietokantaohjelmistoa yhteistyössä PHP-kielen kanssa ja siksi niitä käytetään myös tässä projektissa. Myöskin JavaScriptiä olen käyttänyt melko paljon muun muassa tuottamaan erilaisilla funktioilla dynaamisia ominaisuuksia nettisivuilla, mutta kielen jQuery-kirjasto oli itselleni uusi tuttavuus. Kuitenkin tutustuessani kirjastoon se kuulosti lupaavalta tekniikalta vastausliukusäätimen (Slider) koodaamiseen. Sillä olisi mahdollista sitoa käyttäjäliittymä ja liu'utustoiminta kätevästi yhteen. Vasta toteutusvaiheessa sain myös idean käyttäjäliittymän lopullisesta ulkoasusta liukusäätimillä toteutettuna. Siitä lisää myöhemmin sovelluksen liittymien kuvauksessa (kappale 3).

Kun resurssit eivät sovelluksen tyypistä johtuen aseta rajoitteita, niin lähdin liikkeelle siitä, että näillä teknologioilla työ on mahdollista (osaaminen, oppimiskyky työn aikarajoissa) ja järkevää (yleisiä ja hyväksi todettuja) toteuttaa. Ainoa kysymysmerkki oli jQuery, mutta tutustuessani kirjastoon näin, että sille löytyi hyvä dokumentaatio ja niin "virallisia", kuin innokkaiden harrastajienkin esimerkkejä, joten luotin, että ratkaisut myös jQuerya käytettäessä löydettäisiin.

2.2 Määrittely ja suunnittelu

Määrittely jää siinä mielessä tässä työssä vähemmälle, että määrittelydokumentin tavoite on tehdä asiakkaalle ja/tai mahdolliselle toimittajalle selväksi mitä ohjelma tekee, miten ja miten tehokkaasti, jotta asiakas voisi päättää onko sitä mitä haetaan, onko hintansa väärti ja mahdollinen erillinen toimittaja osaisi arvioida tekemisen aikataulut, kustannukset ja hinnan toteutukselle. Nyt toteuttaja on itse sekä asiakas, kuin myös tietystä näkökulmasta toimittaja. Olin jo päättänyt millainen sovelluksesta tulee (alustavasti) ja mitä se tekee. Tällöin ei ole järkevää tehdä niin sanottua tarjousmäärittelyä erikseen, vaan yhdistin määrittelyn ja suunnittelun kokonaisuudeksi. Joudun tiedon ja osaamisen puutteiden (en ole kokenut ammattilainen, enkä etenkään ryhmä sellaisia henkilöitä) takia ideoimaan joitakin

sovelluksen toteutuksia ja tekniikoita (jotka alan käytäntöjen mukaan pyritään projekteissa kuvaamaan jo määrittely– tai viimeistään suunnitteluvaiheessa) vasta toteutusvaiheessa.

2.2.1 Toimintaympäristö

Määrittelyvaiheessa ajattelin toteuttaa vertailukoneen siten, että se voisi toimia myös itsenäisesti. Liitettävyyden olemassa olevalle sivulle/sivustolle päätin toteutettavan sulauttamalla (kenties object HTML-tagi) tulevan sovelluksen etusivun halutulle sivuston sivulle. Näin sovellus olisi sovellettavissa mahdollisimman laajasti, kun palvelinkone täyttää seuraavat vaatimukset ohjelmistojen ja palveluiden suhteen:

- PHP

Tässä työssä käytetään dynaamisen sisällön luomiseen muun muassa PHP-komentosarjakieltä. Palvelimella (esimerkiksi perus HTTP-palvelin) tulee pystyä joko luonnostaan, lisäosan asentamalla tai erilliseen PHP-jakeluun osoittamalla (yleensä palvelimen konfigurointitiedostoa muuttamalla) saada sivuille käyttöön siis PHP-tulkki. Normaalisti HTTP-palvelimelta pyydetään staattinen pelkkää HTML-merkintäkieltä sisältävä sivu.

- MySQL

Työn tietokannat ovat MySQL-määrityksen mukaisia tietokantoja, joiden ohjelmalogiikka on siis toteutettu tässä työssä PHP-kielellä. MySQL on mahdollista käyttää kyllä muillakin ohjelmointikielillä (esimerkiksi Perl, Python), mutta tässä työssä on edellä mainituista syistä valittu kieleksi PHP. Tietokannat vaativat toimiakseen kuitenkin MySQL-palvelimen toimimaan HTTP-palvelimella tai nämä standardit täyttävät kokonaisuuden.

- JavaScript

JavaScript on asiakaspuolen (engl. client-side ja client eli yleensä nettiselain) oliopohjainen komentosarjakieli, jota on myös käytetty työssä sivujen dynaamisten

ominaisuuksien toteuttamiseen. Kuitenkin JavaScriptin oikeanlainen toiminta riippuu niin sanotusta asiakkaasta, eli tässä tapauksessa löytyykö selaimesta JavaScriptille tuki (Skriptikieli, eli siis taas tarvitaan kielen tulkki). Näin ollen tämä ei sinällään koske palvelinpuolta, mutta on kuitenkin selvennykseksi tuotu esiin.

- jQuery

JQuery taas on JavaScript-kielen kirjasto, joten sen käyttöä koskevat samat vaatimukset kuin JavaScriptin käyttöä. Käytännössä jQueryn käyttäminen sivulla ei vaadi muita lisätoimia, kuin kirjaston sisällyttäminen (.js tiedosto ja sen sijainti) sivuun, jolle löytyy HTML-kielestä oma täginsä (<SCRIPT src=”tiedostopolku”></SCRIPT>).

2.2.2 Käyttäjät

Sovelluksen käyttäjät ovat jaettavissa kolmeen käyttäjäryhmään, joskin henkilö voi kuulua useampaan ryhmään.

- Yleiset käyttäjät: Tämän käyttäjäryhmän jäsenet suorittavat pelkkiä vertailuja, eli vastaavat kysymyssarjaan ja saavat tuloksena soveltumisprosenttinsa tietokannan sisältämiin koulutusohjelmiin.

- Järjestelmäkäyttäjät: Tämän käyttäjäryhmän omia toimintoja ovat taulujen ja käyttäjien luonti, tuhoaminen ja ylläpito. Esimerkiksi jos tietokantaan lisätään uusia koulutusohjelmia. Näistä todettakoon, että määritelmän mukaisesti monet tietokantatehtävät hoidettaisiin MySQL-tietokantaohjelmiston omilla työkaluilla. Esimerkiksi MySQL Workbench nimisellä graafisella ohjelmiston käyttäjäliittymällä on helppoa muokata MySQL-tietokantoja ja tehdä muun muassa ylläpidollisia tehtäviä tietokantoihin ja muuttaa niiden asetuksia

- Muut käyttäjät: Ryhmän jäsenet ovat pääasiassa koulutusohjelmien edustajia, jotka vain käyvät päivittämässä vastauksensa järjestelmään. Heillä ei ole oikeuksia luoda tai tuhota tauluja (edes oman koulutusohjelmansa).

Systeemi on määritelty niin, että yleiset käyttäjät eivät saa, eivätkä tarvitse tunnuksia lainkaan itse sovellukseen, mutta tarvitsevat tietenkin tiedon noutamiseen suunnatut tunnukset tietokantaan (MySQL User Privileges). Tälle käyttäjäryhmälle on siis oma käyttäjäryhmä (vertailu) MySQL-tietokannoissa ja tällä käyttäjäryhmällä on vain kyselyoikeus (SELECT) tauluun, joka sisältää koulutusohjelmakohtaiset vastaukset ID-tunnuksineen (käytännössä tunnus kertoo vastauksen järjestys/rivi -numeron). Tätä oikeutta tarvitaan, jotta yleiset käyttäjät saavat tietokannasta ulos jotain, johon heidän vastauksiansa voidaan verrata.

Muut käyttäjät taas omaavat kysely- ja päivitysoikeudet (UPDATE) samaan tauluun. On huomioitava, että käyttäjän käyttäjätunnus rajoittaa päivityksen vain oman koulutusohjelman tauluun. Päivityksen kohteen määrittää siis sekä MySQL– että sovellustunnistus. Toki käyttäjäryhmä tarvitsee myös tiedon noutamisen (SELECT) oikeuden, mutta tämä on rajoitettu vain asianmukaisiin tauluihin.

Järjestelmäkäyttäjät huolehtivat omilla tunnuksillaan tietokannan rakenteesta. Järjestelmäkäyttäjillä on siis täydet oikeudet sovelluksessa, kuin myös tietokannassa. Olen ajatellut asian niin, että tietokantaa sotkemaan ei kaivata sivuston ulkopuolisia henkilöitä. Helposti tietämätön muu käyttäjä (käyttäjäryhmä) voisi luontioikeudella sekoittaa koulutusohjelman luomalla vaikka useampia tauluja, kun määrä on ohjeellisesti rajoitettu yhteen per koulutusohjelma.

Järjestelmäkäyttäjät ovat asia erikseen, mutta sovelluksen käyttö ei tule vaatimaan yleisiltä –tai muilta käyttäjiltä mitään erikoista IT-osaamista. Heidän osaltaan toiminta perustuu ohjeiden seuraamiseen (online-ohjeet, työkaluvihjeet) ja syöttölomakkeiden täyttämiseen. Järjestelmäkäyttäjien tulisi taas vähintään osata MySQL-tietokantaohjelmiston käyttöä UI-työkalulla (Workbench) tai sen puuttuessa komentotulkilla. Se on sitten sivuston ylläpitäjien oma asia odotetaanko järjestelmäkäyttäjiltä esimerkiksi palvelimien (Apache ja MySQL) ongelmatilanteiden korjaamistaitoa.

2.2.3 Yleistä järjestelmästä

Järjestelmän ei tarvitse pystyä ennakoimaan tai spekuloimaan, joten sovelluksella ei ole tarvetta oppimiskykyyn. Sillä on käytössään kysymykset (kaikille samat), vastausdata ja vertailun yhteydessä se saa käyttöönsä tekijän vastausdatan. Sovellus vertaa vastauksia (koulutusohjelmien edustajat vastaan vertailun tekijä) määrätyllä tavalla ja myöskin johtopäätökset, tuloste, on ennalta määrätty. Vain tulosteen tarkat arvot eli soveltuvuusprosentit eroavat käyttäjillä vastaustensa mukaisesti. Jos järjestelmään ei tarvitse luoda uusia tauluja (koulutusohjelmia) tai vaihtaa koulutusohjelmien edustajien tunnuksia (lähinnä salasana), niin ylläpito on todella vähäistä. Muiden käyttäjien (käyttäjäryhmä) ei tarvitse, kuin päivittää vastauksiaan mahdollisiin uusiin kysymyksiin tai vanhoihin, jos koulutusohjelma muuttuu (seuraus: vastaukset muuttuvat). Normaali vertailutoiminta ei vaadi tietokannan ylläpitoa tai järjestelmäkäyttäjien toimia.

2.2.4 Yleisiä rajoitteita

Kuten edellä tuli jo aihetta sivuttuakin, niin rajoitteita sovellukselle asettaa lähinnä vain tekijän (allekirjoittanut) osaaminen ja osaamisalueisiin liittyvät Internet-teknologioiden (esimerkiksi ohjelmointikielten) omat rajoitukset (kappaleet 2.4.1/2). Sinänsä tällainen vertailukone nettisovelluksena ja yksinkertaisen tyylikkäänä ei vaadi palvelinlaitteistolta paljonkaan (järjestelmävaatimukset [3] ja [4]). Jos palvelimella pyörii normaalikokoinen sivusto (10-20 sivua) ja muutama palvelu, niin kyllä sen tehot riittävät myös tähän. Ainoa pullonkaula voisi olla muistin määrä, eli jos palvelin on jo ympätty liian täyteen erilaisia samanaikaisesti toimivia palveluja ja sivuja vähäisillä muistimäärillä, mutta tuskinpa nykyisillä muistin hinnoilla tämä on kovinkaan todennäköinen skenaario.

2.2.5 Käyttö ja yksityisyys

Käytännössä tällainen sovellus lasketaan kenties yhtä paljon hupi –kuin hyötysovellukseksi ja ajattelinkin, että helppokäyttöisyys olisi yksi avainasia. Onhan toki myös hyödyllistä nähdä, miten vertailijan odotukset vastaavat koulutusohjelmien edustajien arvioita siitä, millainen koulutus on, mitä se vaatii opiskelijalta ja mitkä ovat tärkeimpiä taitoja/ominaisuuksia, joilla menestyä opiskelussa ja työelämässä. Toinen tärkeä tekijä on tietenkin myös kohtuulliset latausajat sovelluksessa. Tätä on yritetty varmistaa muun muassa toteuttamalla grafiikkaa mahdollisimman paljon kuvia yksinkertaisimmilla keinoilla (CSS). Tehdä sovelluksesta yksinkertaisen tyylikkään. Kenties sitten seuraavaan versioon voisi mahdollisten käyttökokemusten kautta lisätä graafisia elementtejä tai nostaa niiden tasoa. Nyt grafiikka on pitkälti toteutettu CSS-tyyleillä (Liite 1 -ja 3).

Liikkeelle on lähdetty siitä, että sovelluksen käyttäjästä (vertailun tekijä) tallennetaan mahdollisimman vähän tietoa tämän omalle koneelle, kuin myös palvelinkoneelle. Itse karsastan piilotettua tiedonkeräämistä palvelujen käyttäjistä jo ihan yksityisyydensuojasyistäkin. Päätin jo varhaisessa vaiheessa, että tietoa kysytään ja tallennetaan muun muassa tietokantaan vain edustajilta ja järjestelmäkäyttäjältä. Tämäkin jää oikeastaan vain pysyväisen taulutiedon (vastausdata) keräämiseen.

Asia on mietitty siten, että koulutusohjelman edustajan henkilökohtaisilla tiedoilla ei ole väliä. Hän saa tunnuksen ja salasanan, joilla pääsee vastaamaan koulutusohjelman puolesta kysymyksiin, mutta siinä kaikki. Henkilön vaihtuessa voidaan pyytää salasanan uusimista, mutta tunnus pysyy samana, sillä se on koulutusohjelmakohtainen (taulukohtainen). Tosiaan ainakaan henkilökohtaisen tiedon leviämisen pelossa salasanaa ei tarvitse vaihtaa, sillä mitään tällaisia tietoja ei kerätä. Järjestelmäkäyttäjälläkään ei tulisi lopullisessa sovelluksessa olla suurta merkitystä. Järjestelmään voidaan kyllä tehdä useampia järjestelmäkäyttäjiä, mutta määrittelyssä en nähnyt useammalle järjestelmäkäyttäjälle tarvetta, sillä tehtävät ovat yksinkertaiset, eikä järjestelmäkäyttäjästäkään tallenneta mitään niin yksityistä tietoa järjestelmään, etteikö samaa tunnusta voisi käyttää esimerkiksi useampi IT-tukihenkilö samanaikaisestikin.

2.2.6 Toiminnot

Yleensä tässä vaiheessa sovelluksen määrittelyä tulisi kuvata jo hyvinkin tarkasti sovelluksen toiminnot ja tarjoamat palvelut, syötteet, niiden käsittely ja sovelluksen tulosteet. Tämä tietenkin siksi, että asiakas/käyttäjä voi ottaa niihin kantaa ennen kuin niiden toteutusta lähdetään suunnittelemaan ja dokumentoimaan. Tämä jää itselläni pois jo ihan siitäkin johtuen, että en sovelluksen toimintaa miettiessäni ja määritellessäni mennyt näin tarkasti yksityiskohtiin. Itse määrittelin ja suunnittelin sovelluksen toimintaa niin sanotusti yleisellä tasolla, mutta jätin siis tarkoituksella toteutustavan, tarkat syötteet sekä tulosteet ratkaistaviksi vasta toteutusvaiheessa, kun pystyn käytännössä testaamaan erilaisia tekniikoita ja sovelluksen osia yhteen.

Määrittelydokumentoinnissa kuuluu käytännön mukaisesti kuvata myös ainakin toteutettavan sovelluksen:

- Turvallisuus ja suojaukset
- Joustavuus
- Laajennettavuus ja ylläpito
- Operointi
- Sovitettavuus erilaisiin ympäristöihin
- Testausnäkökohdat

Itse kuitenkin ohitin nämä kohdat niin määrittelyssä kuin suunnittelussakin yksinkertaisesti siitä syystä, etten voinut tarkemmin määrittää näitä kohtia. Kuten aiemmin mainitsin en ole niin kokenut ja osaava, että olisin pystynyt näitä määrittämään ennen kuin olen saanut merkitsevin kohdin valmista ja päässyt analysoimaan yllä mainittuja näkökohtia valmiista sovelluksesta tai sen osasta. Lisäksi osa näistä kohdista eivät edes ole tärkeitä tämäntyyppisessä projektissa. Osaa tullaan kuvaamaan myöhemmissä kappaleissa.

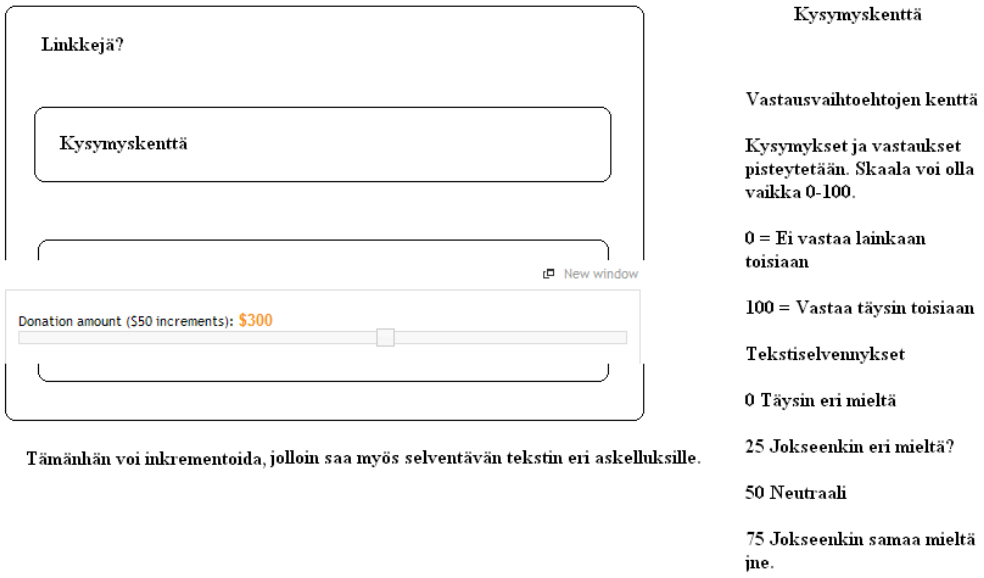
2.3 Sovelluksen liittymät

Sovelluksella on jo aiemmin mainitut liittymänsä tietokantaohjelmistoon (MySQL) ja palvelimeen (tässä työssä Apachen HTTP-palvelin), mutta nämä eivät luo sovelluksen toimintaan juurikaan rajoitteita. Tämä etenkin kun sovellus on suunniteltu siten, että se käyttäisi mahdollisimman vähän itsensä ulkopuolisia resursseja. Aiemmin mainitut minimivaatimukset olivat HTTP-palvelin, johon on saatavana PHP-tuki ja MySQL-palvelin, sekä JavaScript (eli samalla siis myös jQuery) tuen sisältävä selain. Työssä käytössä olevan MySQL-ohjelmiston ja Apache-palvelimen minimijärjestelmävaatimukset löytyvät tuotteiden nettimanuaaleista [3][4]. Kuten liitteestä voi havaita, eivät vaatimukset ole todellakaan vaativia.

Vaikka minulle olikin jo muodostunut kuva siitä millainen ohjelmasta tulisi, niin olin myös varma, että joutuisin muokkaamaan ideoitani ja sitä miten ajattelin ohjelman eri osioita toteutettavan. Huomattava on kuitenkin kuinka suunnitellessani määrittelyni mukaista käyttöliittymää sain vielä hyvän (ainakin omasta mielestäni) oivalluksen, joka sitten eteni ihan toteutukseen asti.

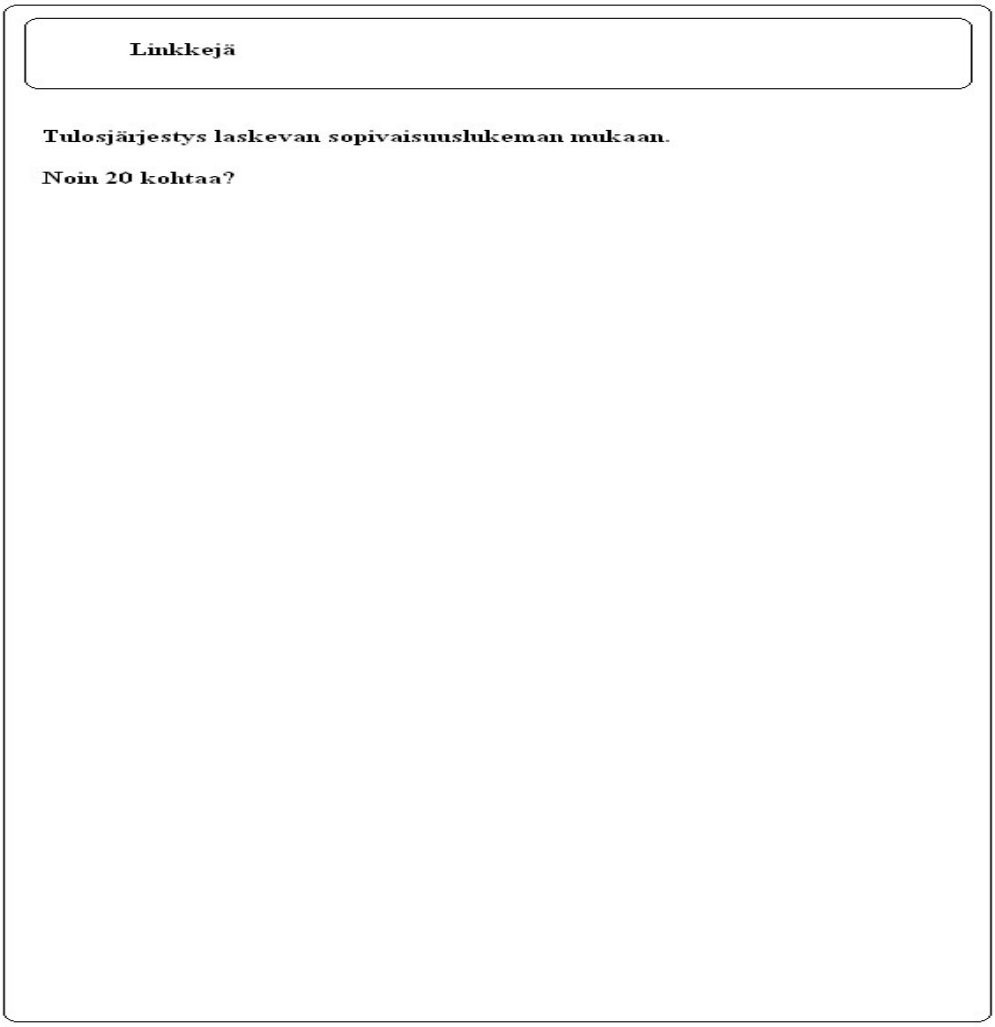
2.3.1 Käyttäjäliittymä

Käyttäjäliittymästä kirjoittamisen voisi aloittaa samaan tapaan kuin itse suunnittelunkin. Sain itse ajatuskulun ja ideoinnin alkuun siten, että lähdin ihan aluksi Paint piirto-ohjelmalla piirtelemään ja suunnittelemaan ohjelman ulkoasua. Minkä näköinen voisi määrittelemäni sovellus olla ja mikä olisi sekä käytännöllistä, että yksinkertaisen tyylikästä. Alla kuvat siitä millaisiin ruutuihin päädyin parin piirustelukerran jälkeen.



Kuva 1: Alustava luonnos vertailukoneen kysymys- ja vastaussivusta

Kuvassa yksi on luonnostelma siitä millaiselta suunnittelin niin sanotun vertailusivun näyttävän. Navigointi (sijoittelutapa vielä määrittelemätön) ylimpänä, kenttä johon tulee kysymykset seuraavaksi ja alimpana omaan kenttään jQuerylla toteutettu vastausliukusäädin. Informaation jaottelu on tärkeä suunnittelunäkökohta myös visuaalisella puolella. Kuvan oikeassa reunassa on kuvattu miten inkrementoidussa (askellettu) vastausliukusäätimessä voitaisiin selventää eri lukemien tarkoitusta. Hauskana yksityiskohtana on kuvaan liukusäätimen paikalle kuvankaappauksella ”varastettu” jQuery-kirjaston dokumentointisivulla demossa (incremented slider demo) esiintyvä liukusäädin [2].



Kuva 2: Luonnos vertailun tulossivusta

Kuvalla kaksi tahdon tuoda esille millaista tyyliä alustavasti ajattelin kaikilla sivuilla noudatettavan. Tämä siksi, että sitä voi verrata lopputulokseen (huomio: kuvaa typistetty korkeussuunnassa tilan vuoksi -> mittasuhdevirhe).

Näissä aivan ensimmäisissä luonnoksissa ei ole kuitenkaan esimerkiksi kuvattu linkkejä palkkia tarkemmin. En ole myöskään tässä vaiheessa kuvannut näppäimistä/funktionäppäimet toimintoja, jotka toteutusvaiheessa tarkentuivat lähinnä perinteisiin käyttöjärjestelmän ja selaimen tarjoamiin sekä mahdollisuuteen vaihtaa kysymyssivua nuolinäppäimillä.

Jätin myöskin sovelluksen väripaletin aivan loppuvaiheeseen, kun suurin osa sovelluksen toiminnoista ja navigoinnista toimi jo oikein sekä ruudut, napit ja muut syöttö- ja komentosäätimet oli sijoiteltu. Väripaletin kannalta ratkaisevaa oli sivuilla olevien elementtien ja niiden muodostaman kokonaisuuden pienehkö koko. Tahdoin myöskin käyttää mustaa ja harmaata käyttöliittymässä (henkilökohtainen mieltymys ja mielipide) ja pulmaksi muodostuikin iso valkoinen tyhjä tausta. Idean taustakuvasta hylkäsin nopsaan. Tarkka ja hieno taustakuva vie tilaa ja hidastaa (merkitsee lähinnä hitaimmilla yhteyksillä tai muutenkin raskailla sivustoilla) sivujen lataamista. Annoin alitajuntani työstää ongelmaa melkein parin viikon ajan työskennellessäni ja loppupuolella hoksasin heittää perinteillä vesilintua. Testasin käyttöliittymää ensimmäisellä oikein toimivalla sivulla (vertailu.php, silloin vielä s.php) valkoisella tekstillä, mustalla/harmaalla taustalla ja tykästyin siihen heti. Tämän ratkaisun kanssa kävi vielä yhteen, että varsinainen sovellus saa taustaväri kseen mustan. Eli siis jokaisen sivun taustaväri on musta, joka mielestäni jo tuo sivuun sisällön tuntua, mutta ilman teknistä raskautta (kuten taustakuva) tai ilman pelkästään visuaalisia elementtejä (kehyksiä ja palkkeja).

Myöskin vasta toteutusvaiheessa hoksasin, miten helpoitan Tooltip-ohjeilla ohjelman käyttöä. Ajattelin jo kyllä määrittelyvaiheessa, että teen joka käyttäjäryhmälle oman ohjesivun, mutta tämä ei ole riittävän kätevä, kun kaipaa apua esimerkiksi syötteen oikeellisuuden kanssa. Näin teinkin niin sanotut Tooltip-apulaiset (graafinen vihjetoiminto, työkaluvihje) tarpeellisiksi katsomilleni sivuille. Toki syötteen validointiskriptit (JavaScript varoitus, alert) viestittävät oikeasta syötemuodosta, mutta vihjeet saa näkyviin syötteen kirjoittamisen kanssa samanaikaisesti. Lisäksi vihjeet ovat oletusarvoisesti piilotettuna, mutta selkeästi näkyviin saatavissa. Piilottaminen on myös siitäkin kätevää, että sovelluksen sivut saadaan pienempään kokoon mahdollisia pienempiä (laitteita) näyttöresoluutiota varten.

Edellisessä kappaleessa mainittiinkin jo nuo JavaScript varoitukset (alert). Lisäksi muun muassa tietokantahakuihin (PHP) on tehty selventäviä varoituksia, jos ei saada esimerkiksi yhteyttä tietokantaan. Ne ovat tämän sovellusversion ainoat varoitukset, jos ei lasketa mukaan palvelimen (esimerkiksi MySQL) omia varoituksia. MySQL-palvelin voi esimerkiksi varoittaa, että taulua ei voida luoda, koska löytyy jo samanniminen. Aiheesta lisää kappaleessa 3, jossa kuvataan valmista käyttöliittymää.

2.3.2 Laitteistoliittymät

Sovelluksen laitteistoliittymistä ei ole paljon kerrottavaa. Kyseessä ei ole sovellus, joka tarvitsisi toimintaansa lisälaitteita tai vaatisi edes palvelinkoneeseen laitteistopäivityksiä.

2.3.3 Ohjelmistoliittymät

- MySQL-tietokantaohjelmisto

Ohjelmisto muun muassa pyörittää MySQL-palvelinta, joka mahdollistaa monen käyttäjän samanaikaisen yhteyden palvelimen tietokantoihin. Ohjelmistoon kuuluu lisäosana myös visuaalinen tietokannan suunnittelutyökalu (MySQL Workbench), joka sisältää SQL-kehityksen, hallinnoinnin, tietokannan suunnittelun, luonnin ja ylläpidon yhteen graafiseen kehitysympäristöön. CE-versio ei ole mitenkään rajoitettu, mutta kaupalliseen SE-versioon saa lisää moduuleita ja liitännäisiä. Standardoitu tietokanta-ajuri on ladattava erikseen. Ajuri toimii seuraavissa käyttöjärjestelmissä Windows, Linux, Mac OS X ja UNIX, samoin kuin myös palvelin ja graafinen käyttäjäliittymä (Workbench). Jonkin verran on kuitenkin eroa esimerkiksi UNIX-pohjaisilla jakeluilla. Yleensä ajuria tarvitsee MySQL-tietokantaan liittymistä varten tietokoneissa, joissa on Windows-käyttöjärjestelmä. MAC OS X- tai UNIX-käyttöjärjestelmissä voi yleensä käyttää keskusteluun tietokannan kanssa natiivia (sisäänrakennettu) MySQL-verkkoa tai nimettyä putkea (named pipe).

- MySQL Server 5.5
- MySQL Workbench 5.2 CE (Community Edition)
- MySQL Connector ODBC 5.1

Ohjelmisto siis pyörittää tietokantapalvelinta, hallinnoi tietokantoja ja tarjoaa niihin liittymän sovellukselle. Tietokantoihin on tallennettu käyttäjien tunnukset ja salasanat, kysymykset, käyttäjien vastaukset sekä kuvaus koulutusohjelmasta. Kuvaus on samassa taulussa, kuin missä on koulutusohjelman edustajan syöttämät vastaukset. Käytännön syistä koulutusohjelman kuvaus on taulun kommenttina.

Tärkeimpänä syynä on, että taulun nimikenttä ei ole rajoituksiensa vuoksi ollenkaan sopiva pitkiä, mahdollisesti skandinaavisia merkkejä sisältäviä paikkakuntien ja koulutusohjelmien nimiä varten. Salasanat on salattu MD5-algoritmillä (message-digest). Tietokantaliittymät on koodattu PHP-komentosarjakielellä ja tarjoavat käyttäjäryhmille oikeuksiensa mukaan mahdollisuuden hakea, luoda, päivittää ja tuhota tietoa tietokannoista. Tämän hetkisessä ohjelman versiossa netin kautta muokataan lähinnä vain vastaustietoja. Muuten muun muassa salasanoja muokkaavat järjestelmäkäyttäjät MySQL-ohjelmiston omalla graafisella työkalulla (Workbench) tai ilman käyttäjäliittymää komentotulkilla, jos jostain syystä ei tahdota asentaa graafista työkalua.

- Apache HTTP-palvelin

Apache on avoimeen lähdekoodiin perustuva HTTP-palvelinohjelma. Pelkkä Apache tukee ainoastaan staattisten tiedostojen jakamista HTTP-protokollan yli. Tästä syystä kehityskoneellekin on ladattu niin JavaScript, kuin myös PHP-ohjelmointikieli sivujen dynaamisia ominaisuuksia varten. Työssä käytetyt versiot:

- Apache, versionumero 2.2.21 (Win32)
- PHP, versionumero 5.2.17

Tässä työssä onkin noudatettu Windows-käyttöjärjestelmälle mukautettuna erästä kokoelmaa avoimen lähdekoodin ohjelmia (LAMP). WAMP on muunnos vastaavasta ohjelmistokokonaisuudesta (LAMP), jossa kokoelma ohjelmia muodostaa WWW-palvelimen, jonka alla voidaan ajaa dynaamisia websivuja.

WAMP sisältää ohjelmat:

- **W**indows-käyttöjärjestelmä
- **A**pache, avoimen lähdekoodin webpalvelin
- **M**ySQL, avoimen lähdekoodin tietokantarajapinta
- **P**HP, **P**erl ja/tai **P**ython, avoimen lähdekoodin komentosarjakieli

Web-palvelimen ei ole pakko olla juuri Apache. Tehtävään käy mikä tahansa palvelinohjelmisto, joka pystyy toimimaan yhteen PHP-kielen ja MySQL-ohjelmiston kanssa (ainakin siis itse MySQL-palvelimen kanssa).

2.3.4 Tietoliikenneliittymät

Kirjaan alla tietoliikenneprotokollat, joita käytetään tässä työssä hyväksi.

- Hypertext Transfer Protocol eli HTTP.
- Transmission Control Protocol / Internet Protocol eli TCP/IP.

Nämä web-teknologioiden niin sanotut perusprotokollat on laajasti kuvattu symboli– ja termiluettelossa. Täsmennyksenä voi kirjoittaa, että Apache-palvelin, MySQL-tietokanta (tietokantapalvelin), PHP ja sovelluksen käyttäjän selain (asiakas) keskustelevat kaikki näillä protokollilla keskenään. MySQL ja PHP käyttävät myös paikallisesti (localhost, ei vaadi nettiyhteyttä) viestintäänsä oletusarvoisesti Windows- ja UNIX-pohjaisessa käyttöjärjestelmässä portteja ja pistokkeita (englanniksi port ja socket samassa järjestyksessä).

2.4 Suunnittelurajoitteet ja pakotteet

Kuvaan tässä kappaleessa tietenkin sellaisia rajoitteita ohjelman toteuttamisella, joita ei ole jo aiemmin dokumentissa aihetta sivuttaessa kuvattu.

2.4.1 Standardit

Aikaisemmin mainitsinkin jo tuosta taulujen nimeämisestä. Taulujen nimet kuuluvat niin sanottuihin tunnisteihin (MySQL), jotka muutetaan sisäisesti Unicode muotoon ja niihin pätevät seuraavat säännöt [3]:

- Sallitut merkit lainausmerkittömissä tunnisteissa:

  - ASCII: [0-9,a-z,A-Z$_] (eli perus latinalaiset kirjaimet. Digitit/numerot välillä 0-9. Dollarimerkki. Alaviiva)
  - Jatketut (Extended): U+0080 .. U+FFFF
- Sallitut merkit lainausmerkityissä tunnisteissa, myös täysi Unicode BMP, poislukien U+0000:
  - ASCII: U+0001 .. U+007F
  - Jatketut (Extended): U+0080 .. U+FFFF
- ASCII null (U+0000) ja täydentävät merkit (U+10000 ja ylöspäin) eivät ole sallittuja lainausmerkittyinä tai ilmankaan tunnistimissa.
- Tunnistimet voivat alkaa numerolla, mutta vain lainausmerkeillä voivat kokonaan koostua numeroista.
- Tietokannan, taulun tai rivien nimet eivät voi päättyä välimerkkiin.
- Tietokanta- ja taulunimissä ei voi olla merkkejä “/”, “\”, “.”, tai merkkejä jotka eivät voi esiintyä tiedostojen nimissä.
- Tunnistimien lainausmerkki on (“`”) eli englanniksi backtick.
- Jos ANSI QUOTES SQL tila on päällä, myös tuplalainausmerkki (“) on sallittu.

Lisäksi tunnistimien kirjainkoolla on erilaisissa tilanteissa ja eri tunnisteilla merkitystä, mutta säännöt ja poikkeukset ovat niin monimutkaiset, että on paljon helpompaa, kun pitää itse kirjaa tunnisteidensa kirjainkoosta tai noudattaa selkeää linjaa. Esimerkiksi itse käytän MySQL-tunnisteissa poikkeuksetta pieniä kirjaimia, enkä pyri käyttämään skandinaavisia kirjaimia. Itselläni nämä MySQL-standardin mukaiset tunnistimien nimet merkitsevät lähinnä siinä, että miten itse nimesin skeemat, tietokannat, taulut ja miten sallin uusia tauluja nimettävän.

2.4.2 Ohjelmistorajoitteet

Tunnetuimmat rajoittajat, kuten käyttöjärjestelmä tai kääntäjä/tulkki, eivät ole asettaneet työlle mitään rajoitteita tai oikeastaan mitään rajoitteita, jotka eivät olisi allekirjoittaneelle selviöitä. Esimerkkinä: kun lähtee jonkun ohjelmointikielen koodaamista opiskelemaan, täytyy tietenkin oppia, miten muuttujia saa nimetä tai tyypittää, miten niitä kutsutaan ja käytetään. Tällaisia ominaisuuksia tarkoitan selviöillä. En ole kuitenkaan joutunut käyttöjärjestelmän tai tulkin vuoksi luopumaan jostain ideasta tai ominaisuudesta. Ja vaikka sovellus on kehitetty

Windows-käyttöjärjestelmässä (WAMP), perustuu se internetin teknologioihin ja se toimii millä tahansa käyttöjärjestelmällä, joka pystyy keskustelemaan minimivaatimukset täyttävän selaimen kanssa (kappale 2.2.3).

2.4.3 Muut rajoitteet tai pakotteet

Tämä kappale on oikeastaan otettu mukaan siitä syystä, että se tekee selväksi miten moninaisia näkökohtia voi joutua sovelluksen luomisessa ottamaan huomioon. Tekijöitä jotka voivat suureltakin osin rajoittaa sovelluksen toimintoja tai pakottaa tiettyyn toimintamalliin: Asiakkaan vaatimus tietystä ohjelmointikielestä, kehitysympäristöstä- tai työkaluista, viranomaismääräykset, työsuojelumääräykset, lait ja asetukset, ammattiyhdistysnäkökohdat, sovellusalueen niin sanotut yleisesti hyväksytyt (de facto) käytännöt. Käyttöliittymässä olenkin rikkonut erästä de facto ”standardia”. Teksti on yleensä mustaa valkoisella, käytäntö, jota rikoin mielestäni hyvin perustein. Kuitenkin olen esimerkiksi linkkien sijoittelussa, suoritusnäppäimissä ja yleensäkin elementtien sijoittelussa noudattanut ”sääntöjä”, jotka ovat tulleet tutuiksi tietotekniikan opiskelijoille muun muassa kurssissa (kursseissa), jossa käsitellään käyttöliittymäsuunnittelua.

Monesti sovelluksen käyttöliittymää sanelevat kuin huomaamatta inhimillisistä ominaisuuksista (biologia) ja tavoista johtuneet säännöt. Tekstiä haetaan sivulta vasemmalta alkaen lukusuunnan mukaisesti (länsimaat) ja ylhäältä alas, vakiintunut sommittelu, kuten sovelluksen tai ikkunan sulkeva ruksi oikeassa yläreunassa ja mielleyhtymät väreissä.

Ihminen myös lajittelee mielessään sivun elementit visuaalisen painon mukaisesti ja huomioi eri elementtejä tämän mukaisesti. Visuaaliseen painoon vaikuttavat muun muassa:

- Koko
- Korkeus/Syvyys
- Väri
- Tummuus
- Varjot
- Sijainti suhteessa sivun keskiakseliin

- Ympäröivä tila

Tutustumalla näihin ominaisuuksiin voidaan luoda visuaalisesti hyvin informoivia käyttöliittymiä, mutta on muistettava, että visuaalinen ilme ja käytettävyys monesti tasapainottelevat keskenään.

3 PÄÄLIUKUSÄÄTIMEN JA VASTAUSLIUKUSÄÄTIMIEN TOTEUTUS

Mainitsin aiemmin (luku 2.1.1) miten sain vasta toteutusvaiheessa idean lopullisesta käyttöliittymän ulkoasusta. Olin toteuttanut vastausliukusäätimen tarpeettoman isokokoisena, kun törmäsin eräässä esimerkissä kätevään toimintoon liu'uttaa sivuja liukusäätimessä. Tästä ideoin lopullisen muodon kysymysten vastaamiseen, jossa on erikseen pääliukusäädin, jolla vaihdetaan sen sisältämää sivua ja sen sisällä vastausliukusäätimet sivuilla (yksi per kysymys), joilla vastataan kysymyksiin. Alla kuva etusivusta (samalla sovelluksen pääsivu):

Kuva 3: Ruudunkaappaus pääsivusta (etusivu.php)

Käytännössä kysymyksiin vastaaminen ja vertailu toimii niin, että pääsivulla on kokonaan piilossa oleva lomake (hidden elements), johon kerätään talleta vastaus nappien painalluksella käyttäjän syötetiedot. Jokaisella pääliukusäätimen sivulla on

kysymys (sivunumero vastaa kysymyksen ID-tunnusta/järjestysnumeroa MySQL-tietokannassa, mutta ei ole sama muuttuja), vastausliukusäädin ja talleta vastaus nappi, jolla vastaukset (maksimissaan kolmenumeroinen lukuarvo (string)) siirtyvät lomakkeeseen niille varattuihin piilokenttiin. Kun käyttäjä tahtoo suorittaa vertailun, painaa hän alinta suorita vertailu nappia, jolloin tiedot lomakkeesta lähetetään (FORM ACTION POST) eteenpäin sivulle, joka suorittaa itse vertailun ja tulostaa tulokset näkyviin. Alla kuva tästä sivusta:

Kuva 4: Ruudunkaappaus sivusta, joka vertailee ja tulostaa (vertailu.php)

Kuvassa on havaittavissa pientä likistymisestä johtuvaa mittavirhettä, jotta se mahtuu ja sopii muuten hyvin dokumenttiin. Kuvasta kuitenkin näkee miten vertailun tulokset tulostuvat käyttäjälle näkyviin.

Palataanpa pikkuisen taaksepäin, liukusäätimiin. Pääliukusäätimestä on sivustolla itse asiassa alustettu useampi toteutus. Yksi etusivulla (etusivu.php - slider), yksi taulun luomisen vaiheessa kolme (do_s2.php - sliderdo) ja vielä yksi sivulla jossa muut käyttäjät ryhmän jäsenet täydentävät vastauksensa koulutusohjelman tauluihin (updatet.php - slideru). Näillä sivuilla toteutetaan aivan samalla tavalla taulujen

täyttö/päivitys. Lähes pelkästään tietojen tallentamisessa on liukusäätimiin liittyen käyttäjä-/sivukohtaisia eroja. Kaikkia pääliukusäätimiä koskevat samat tyylisäännöt ja jQuery-koodi. Samaten näillä sivuilla on aina kysymysten verran alustuksia vastausliukusäätimistä.

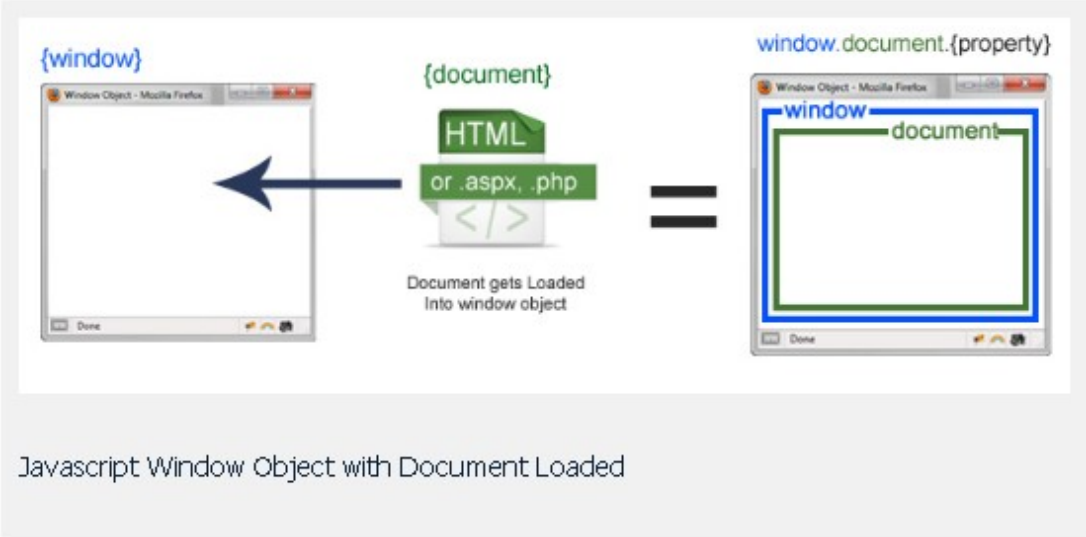
On huomioitava, että vastausliukusäätimen toteutus on tehty lähes suoraan sivuston http://jqueryui.com/home dokumentaatioiden pohjalta. Kyseessä on avoimen lähdekoodin käyttöliittymäkomponenttien kirjasto. Mallia on otettu sivulla http://jqueryui.com/demos/slider/#steps sijaitsevasta inkrementoidusta (askellettu) liukusäätimestä. Muutoksia on tehty lähinnä säätimen kokoon ja ulkonäköön eli muuttamalla alustusparametreja ja muokkaamalla ulkonäköä tyylitiedostolla. Säädin (kirjasto) tarvitsee vielä lisäksi seuraavat kirjastot, joista se on riippuvainen:

- jquery.ui.core.js
- jquery.ui.mouse.js
- jquery.ui.widget.js

Nämä kirjastot löytyvät kustomoidusta kirjastosta jquery-ui.custom2.js, joka on ladattu valinnat suorittaen samalta sivustolta. Näiden kirjastojen toiminta perustuu niin sanottuun peruskirjastoon, jota myöskin käytetään työssä hyväksi ja joka löytyy sivustolta http://jquery.com/ . Yleensä kun puhutaan jQurysta tarkoitetaan juuri tätä peruskirjastoa, joka itseltäni löytyy tiedostonimellä jquery2.js (jQuery JavaScript Library v1.6.2).

En näe syytä miksi kävisin lävitse vastausliukusäätimen toteutuksen erikseen, sillä jokainen voi tutustua käyttämääni säätimeen ja kirjastoihin, joilla se on toteutettu, edellä mainituilla sivustoilla.

Käyn kuitenkin liitteessä (Liite 1) lävitse pääliukusäätimen koodin, sillä vaikka siinäkin on käytetty hyväksi dokumentaatioita ja esimerkkitoteutuksia, niin se on kuitenkin oma kokonaisuutensa ja siitä pääsee hyvin jyvälle jQuery:sta. Vielä selvennyksenä kuva:



Kuva 5: Mikä on ero dokumentti- ja ikkunaobjekteilla [6]

Kuvan määritteitä koskevat tapahtumat $(document).ready (dokumentti valmis) ja $ (window).load (ikkuna ladattu). Dokumentti valmis tapahtuma suoritetaan, kun HTML-dokumentti on ladattu ja DOM on valmis. Ikkuna ladattu tapahtuma vähän myöhemmin, kun koko sivu on täysin ladattu (myös muun muassa kehykset, objektit ja kuvat). Niinpä sivun sisältöä muokkaavat funktiot tulee sisällyttää ikkunan ladattu tapahtumaan tai sisältötagiin itseensä.

Pääliukusäätimen toteuttava koodi löytyy liitteistä seuraavasti:

- Liite 1: Pääliukusäätimen alustusfunktio
- Liite 2: Pääliukusäätimen CSS

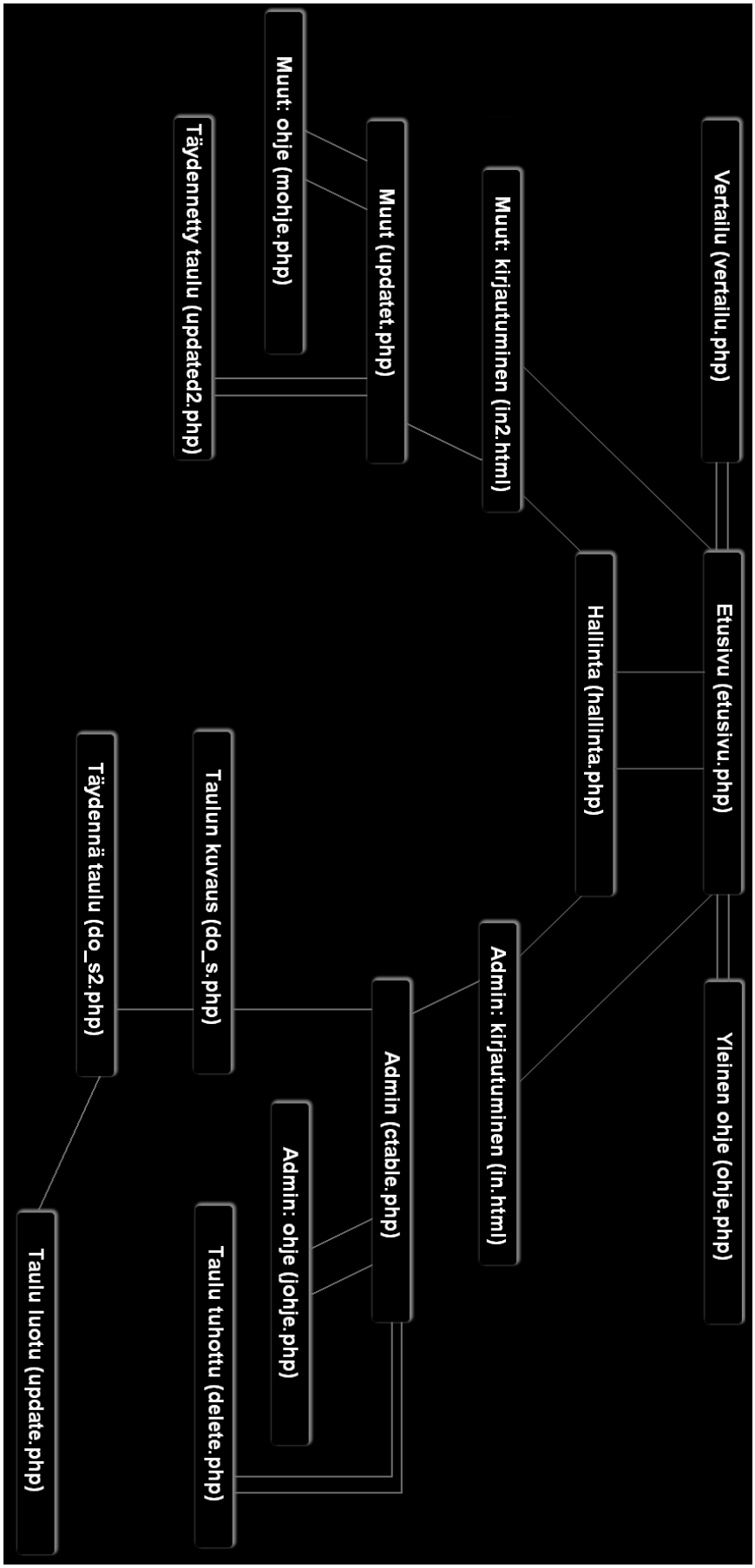
Pääliukusäätimen alustusfunktio on liitteessä kokonaisuudessaan, mutta CSS on vain siltä osin, mikä on merkittävää tai jota ei ole jo muualla (kuten seuraavassa kappaleessa) selvennetty.

4 HTML- JA PHP-SIVUJEN TOTEUTUS

Tässä kappaleessa käyn lyhyesti lävitse sivujen toiminnan. Se taas mikä on sivujen HTML, PHP –ja CSS-koodi käydään tarkemmin lävitse liitteessä (Liite 3). HTML ja

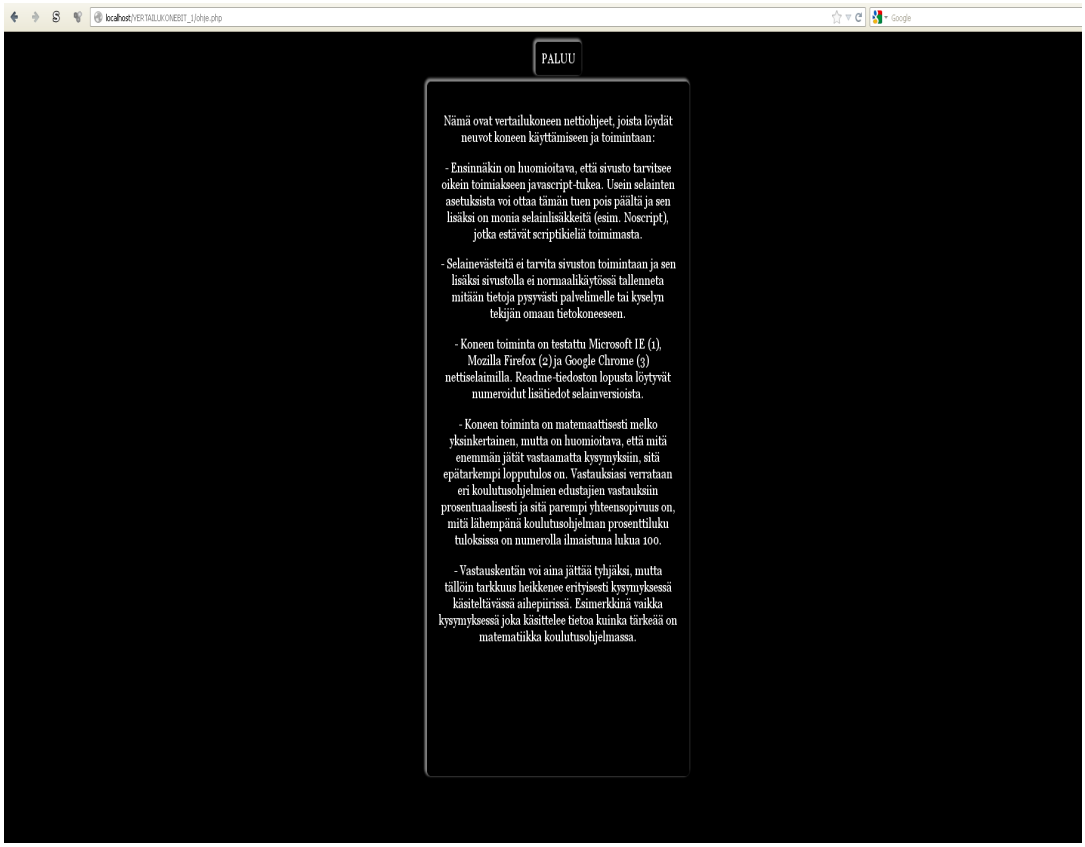
PHP ovat molemmat varsinaisissa sivutiedostoissa, joista pääosa on PHP-sivuja (*.php) ja muutama puhdas HTML-sivu (*.html). Sivut ja niiden elementtien tyylit ovat tosiaan erillisissä tyylitiedostoissa muutamaa poikkeusta lukuun ottamatta. Liitteessä nämä käsitellään kuitenkin yhdessä siten, että ensin on pätkä sivutiedostosta ja siihen osioon liittyvistä tyylisäännöistä ja sitten on vielä selvennetty koodia tekstiselvityksillä.

Selvyyden vuoksi on hyvä aloittaa etusivusta, josta voi sitten toiminnallisuuden mukaan haaroittua eri suuntiin. Etusivua tuli jo kuvan muodossa ja liukusäätimen toiminnallisuuden yhteydessä kuvattua jonkin verran (kappale 3). On huomattava, että sivuilla toistuvia toimintoja en selitä tietenkään kahteen kertaan. Jos näyttää siltä, että jokin asia jää selvittämättä, niin kyseessä on joko joku aivan perusasia tai sitten jo aiemmin selvennetty toiminto. Pyrin kuitenkin osoittamaan sivun toiminnallisuuden havainnollisesti, vaikka tässä dokumentissa ei varsinaista koodia esitelläkään. Koodi löytyi siis tarkemmin liitteestä (Liite 3). Alla kuitenkin ensinnä sivuston sivukartta kuvana:



Kuva 6: Sivuston sivukartta

Etusivu ja vertailusivu käytiin lyhyesti läpi edellisessä kappaleessa ja yleisellä tasolla ohjesivujen (ohje- , mohje- ja johje.php) toiminnassa ei ole mitään erikoista. Ohjesivuilla on samalla tyylillä esitelty kullekin käyttäjäryhmälle näitä koskevat toiminnalliset seikat ja selvennetty käyttäjäryhmäkohtaisia operaatioita. Kuva Yleisestä ohjeesta, joka liittyy etusivuun:

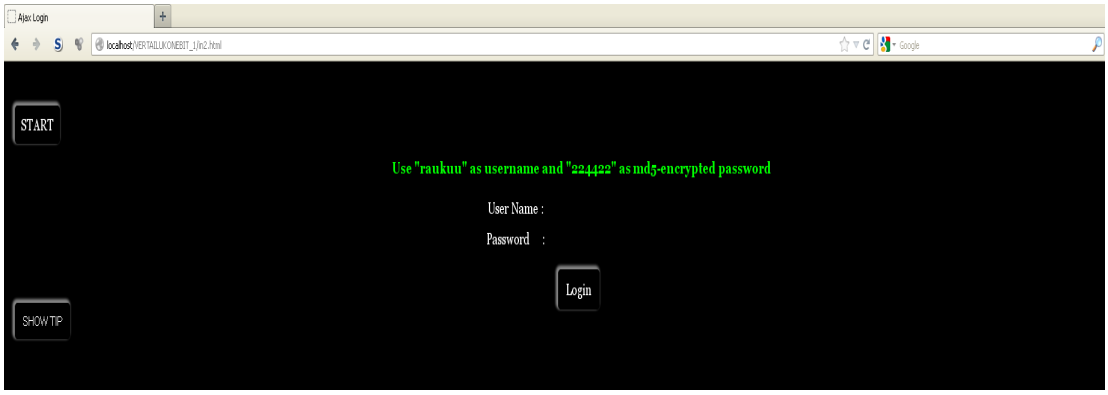


Kuva 7: Yleinen ohje (ohje.php)

Hallintasivun todettiin jo aikaisemminkin olevan vain niin sanottu navigaatioristeys, jolla erotetaan toisistaan käyttäjäryhmät. Yleiset käyttäjät pääsivulle ja muut- sekä järjestelmäkäyttäjät omille pääsivuilleen (updatet.php ja ctable.php samassa järjestyksessä), jotka vaativat sisäänkirjautumisen. Kuva hallinnasta:

Kuva 8: Hallinnan navigaatiopalkki

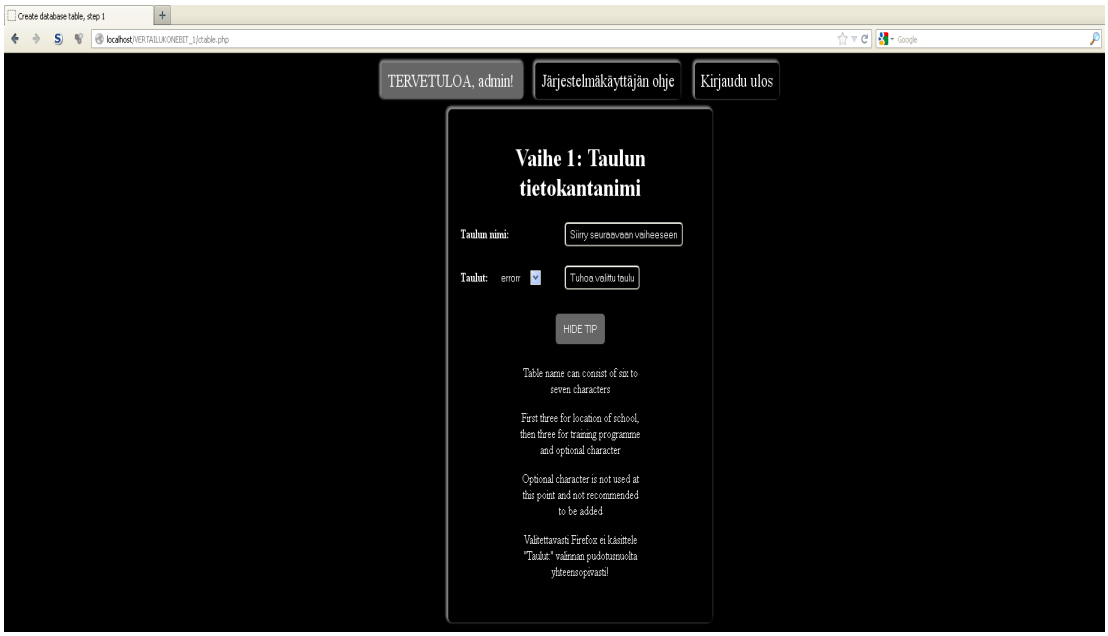
Molemmille (Muut, Järjestelmäkäyttäjät) on oma kirjautumissivunsa (in2.html ja in.html, sama järjestys). HTML-kirjautumissivut ovat lähes pelkästään ulkokuori, varsinaiset sisäänkirjautumiset toteuttavat tiedostot ajax_login2.php ja ajax_login.php, joiden toiminta on käyty lävitse liitteessä (Liite 3). Kuva muiden käyttäjien kirjautumissivusta (in2.html):



Kuva 9: Kirjautumissivu

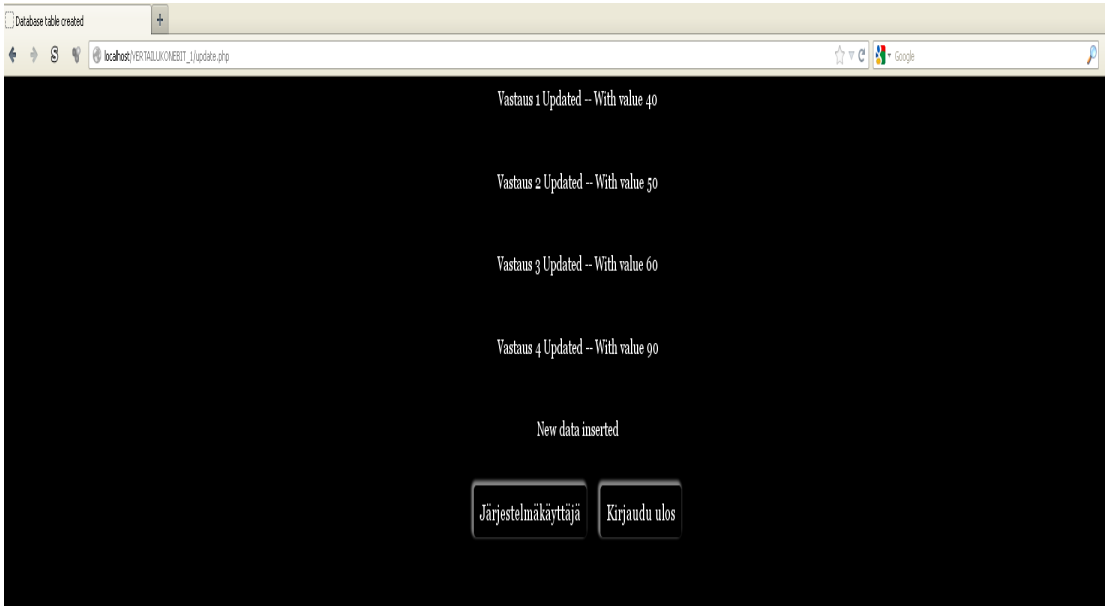
Muut käyttäjät pystyivät päivittämään vain käyttäjätunnuksensa nimistä taulua ja päivitys tapahtui visuaalisesti aivan samalla tavalla, kuin vertailuun vastaaminenkin eli liukusäädinsivulla. Muiden käyttäjien kirjautumissivulla on myös toinen sivuston työkaluvihjeistä. Tässä kuvassa se on piiloitettuna (auki oleva näkyy kuvassa 10). Sivu, joka näyttää päivitysoperaatiosta yhteenvedon (updated2.php), tekee myöskin varsinaisen tietokantatallennusoperaation.

Järjestelmäkäyttäjät pystyvät pääsivullaan (ctable.php, kuva 10) tuhoamaan taulun valitsemalla sen pudotuslaatikosta ja painamalla nappia. En nähnyt tarpeelliseksi toimintoa, jolla voisi tuhota useamman kerralla, sillä tällainen tyhjennys voidaan suorittaa MySQL-palvelimen omilla työkaluilla jopa helpommin. Sivu, joka hoitaa itse operaation on delete.php ja merkitsevä php-funktio löytyy liitteestä (Liite 3). Kuva järjestelmäkäyttäjien pääsivusta:



Kuva 10: Järjestelmäkäyttäjien ctable.php

Järjestelmäkäyttäjät pystyvät myös luomaan tauluja ja operaatio on jaettu useampaan vaiheeseen. Taulun luonti aloitetaan pääsivulla (ctable.php) antamalla taululle nimi ja painamalla nappia (Siirry seuraavaan vaiheeseen). Tästä siirrytään seuraavaan vaiheeseen (ja sivulle do_s.php), jolla annetaan taululle kuvaus. Kuvaus oli varsinainen koulutusohjelman nimi MySQL-taulun kommenttikentässä. Kuvaus hyväksytään napin painalluksella ja samalla siirrytään viimeiseen käyttäjän syötettä vaativaan vaiheeseen (do_s2.php), jossa on siis taas pääliukusäädin ja vastausliukusäätimet. Syötteitä ei kuitenkaan vaadita, vaan taulu voidaan luoda heti oletusarvoilla (00). Tässä vaiheessa siis syötetään enää vastausarvot, jonka jälkeen viimeisellä sivulla (update.php) on yhteenveto, josta alla kuva:



Kuva 11: Syötteiden yhteenveto

Ilman syötteitä kaikki vastauskentät ilmoitetaan tietenkin sisältävän nollan. Sivulta voi kirjautua suoraan ulos tai palata pääsivulle (ctable.php).

Siinä oli lyhyesti kuvattuna koko sivusto. Teknisempään puoleen voi halutessaan tutustua siis liitteissä (Liite 1, 2 ja 3).

5 TIETOKANNAT

Työssä käytetään hyväksi kolmea eri tietokantaa (MySQL SCHEMA). Sisältöjen esittely on hyvä aloittaa niin sanotusti ylimääräisestä tietokannasta o2011. Tästä testitietokannasta käytetään edelleen yhtä taulua itse työssä ja se taulu on testikysymykset. Tällä taululla testattiin useamman kysymyksen dynaamista käsittelyä ja pääliukusäätimen sivujen muodostamista. Taulu jäi käyttöön myös tähän työn lopulliseen versioon, koska työssä ei ole määritetty ehdotonta kysymysdataa.

5.1 Tietokanta o2011

testikysymykset

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
id	INT(3)	✓	✓						
kysymys	VARCHAR(120)								NULL

Kuva 12: Taulun testikysymykset tiedot

Kuvasta näkyykin jo taulun tiedot (rakenne), mutta laitetaan nyt vielä selvemmin: Ensimmäisellä rivillä on ID, joka on siis käytännössä kysymykselle rivinumero. ID on myös taulun pääavain, sen arvo ei saa olla nolla, sallittu merkkimäärä on kolme (3) ja tyyppi kokonaisluku (int). Toisella rivillä on kysymys, joka on tietotyypiltään merkkidataa ja jolle ei ole asetettua pituutta. Maksimipituus on kuitenkin määritetty 120 merkkiin. Alla on kuva taulun sisällöstä:

id	kysymys
1	Kuinka kiinnostunut olet soveltamaan työssäsi matematiikkaa?
2	Kuinka kiinnostunut olet soveltamaan työssäsi fysiikkaa?
3	Kuinka kiinnostunut olet soveltamaan työssäsi kemiaa?
4	Kuinka kiinnostunut olet työskentelemään ryhmässä?
NULL	NULL

Kuva 13: Taulun testikysymykset kysymykset

5.2 Tietokanta o2011_vastaukset

Seuraava tietokanta on o2011_vastaukset, jossa on taulu per koulutusohjelma. Malliesimerkkinä tietokannasta otetaan taulu mallipohja, jota kloonaamalla luodaan tietokannan muut taulut (selitetty kappaleessa 5.2).

mallipohja

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
id	INT(3)	✓	✓					✓	
vastaus	INT(2)								NULL

Kuva 14: Taulun mallipohja rakenne

Tästäkin taulusta (onhan se vastine kysymyksille) löytyy ID, joka on siis rivinumero (ja pääliukusäätimen sivunumero). Tällä kohdistetaan kysymykset ja vastaukset toisiinsa. Taulusta löytyy myös vastaus, joka on siis vastine esitettyyn kysymykseen. Ainoa ero rakenteessa edelliseen löytyy siitä, että rivinumero inkrementoidaan (askellus) automaattisesti eli numeroidaan järjestyksessä. Sen lisäksi erona on, että vastaus on maksimissaan kaksi merkkiä käsittävä kokonaisluku. Tämä siksi että vertailu sovelluksessa tehdään vertailemalla vastausten lukuarvoja (kappale 4, vertailu.php). Alla kuva taulun tiedoista:

id	vastaus
1	10
2	20
3	30
4	40
NULL	NULL

Kuva 15: Taulun mallipohja tiedot

Muistutuksena voi täsmentää tässäkin, että taulun lukuarvoilla ei ole merkitystä. Taulua siis käytetään muiden taulujen mallina, mutta kloonauksessa ei siirry muuta kuin taulun rakenne. Tiedot täytetään ensin alustusarvoina ja sitten käyttäjän syötteenä. On myös huomattava, että taulun kommenttina on aina koulutusohjelman täysi kuvaus, esimerkiksi: ”Rauma - Tietotekniikan koulutusohjelma”.

5.3 Tietokanta o2011_clauditis

Viimeisenä tietokantana on o2011_clauditis, jossa säilytetään käyttäjätunnuksia ja salasanoja. Tietokannassa on kaksi taulua, portum ja sepem. Portum taulussa on pelkästään järjestelmäkäyttäjiä ja sepem taulussa on taas käyttäjäryhmä Muut käyttäjät. Taulut ovat rakenteeltaan täysin identtiset, joten otan tässä esimerkiksi Muut käyttäjät taulun.

sepem

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Defau
user_id	INT(4)	✓	✓					✓	
user_name	VARCHAR(10)		✓	✓					
password	VARCHAR(70)		✓						

Kuva 16: Muiden käyttäjien tunnustaulu sepem

Käyttäjät (user_id) on numeroitu kokonaisluvulla, joka rajoittaa käyttäjien (koulutusohjelmat) lukumäärän nelinumeroiseksi. Tämä kenttä on taulun pääavain ja autoinkrementoitu, se ei voi myöskään olla tyhjä arvoltaan. Seuraavaksi on käyttäjänimi (user_name), joka on taulussa asetettu maksimissaan kymmenen merkin pituiseksi (validointi itse asiassa hyväksyy vain arvon, joka on välillä 6-7 merkkiä) merkkijonoksi (ei saa olla tyhjä, pitää olla myös uniikki) ja vielä viimeiseksi salasana, joka saa olla 70 merkkiä pitkä, mutta ei arvoltaan tyhjä. Alla kuva esimerkkitaulun sisällöstä:

user_id	user_name	password
1	raukuu	495c9b0d045fcf0d41443849f501c2e6
2	ikadat	495c9b0d045fcf0d41443849f501c2e6
3	xermer	495c9b0d045fcf0d41443849f501c2e6
4	pormer	495c9b0d045fcf0d41443849f501c2e6
5	admin	495c9b0d045fcf0d41443849f501c2e6
6	errorr	495c9b0d045fcf0d41443849f501c2e6
NULL	NULL	NULL

Kuva 17: Sepem tunnustaulun sisältö

Kuvasta nähdäänkin hienosti kuinka sovelluksessa kaikilla käytetty sama salasana (224422) muodostuu tietokantaan algoritmilla MD5. Kuvassa näkyy myös niin sanotut testi- käyttäjät/koulutusohjelmat, joiden tunnus on siis ainutkertainen. Samannimistä taulua ei hyväksytä tietokantaan. Koulutusohjelman täysi kuvaus taas löytyi vastaustaulusta.

6 MUIDEN FUNKTIOIDEN TOTEUTUS

Sivuilla toimivat muut funktiot ovat kaikille sivuille yhteisessä yhdistetty.js nimisessä JavaScript-kielisessä tiedostossa. Funktioita on käytännössä kaksi erilaista, mutta selvyyden vuoksi jokaiselle sivulle on omansa näistä kahdesta. Projektin alussa jokaisella HTML-tiedostolla oli kaikki koodi ympättynä sivuille (kehittäjän ajatuskulun vuoksi) ja vasta viimeistelyvaiheessa koottiin muun muassa JavaScript-funktiot erilliseen ulkoiseen tiedostoon. Myöskin johtuen funktioiden vähäisestä lukumäärästä ei nähty tarpeelliseksi yhdistää samankaltaisia funktioita. Alla kaksi erilaista funktiota, joita hyödynnetään sivuilla.

```
function laske(){
document.getElementById("sum1").value=
parseInt(document.getElementById("val1").value)+
parseInt(document.getElementById("val2").value)+
parseInt(document.getElementById("val3").value)+
parseInt(document.getElementById("val4").value);
}
```

Yllä olevaa laske() funktioita käytetään etusivulla (etusivu.php) laskemaan vastauskenttien arvot (huomaa kokonaislukuarvot -> parseInt) yhteen. Funktiota kutsutaan, kun lomaketiedot lähetetään (HTML POST-metodi) ja samalla siirrytään vertailusivulle (vertaile.php). ParseInt karsii muun muassa vastauksista pois %-merkin ja muuttaa lomakkeen tekstikenttien luovuttamat merkkijonot (string) lukuarvoiksi (int). Funktiota kutsutaan myös joka kerta, kun käyttäjä muuttaa yhtäkään vastauksistaan (painaa Talleta vastaus nappia, kutsuu yhtä copysv-funktioista).

```
function copysv(){
document.getElementById("val1").value=
parseInt(document.getElementById("amount1").value);
laske();
}
```

Yllä olevassa copysv()-funktiossa otetaan tekstikenttään (HTML id="amount1") siirretty ensimmäisen vastausliukusäätimen lukuarvo. Vastausliukusäätimet on koodattu siten, että niiden arvo näkyy suoraan tekstikentässä aina, kun kutsutaan säätimen liu'u (slide, event, ui) tapahtumaa. Lisäksi saman tapahtuman käsittelevään funktioon kuuluu selitteen lisääminen lukuarvoon. Copysv- funktion lopuksi kutsutaan vielä laske-funktiota. Copysv-funktiota (ja sen sisarfunktioita) kutsuttiin aina, kun painetaan vastausliukusäätimeen yhdistettyä "Talleta vastaus" nappia. Summa siis päivitetään aina, kun jotain vastauksista muokataan. Seliteteksteistä: Esimerkiksi siis 100% merkitsee selitettä "Täysin samaa mieltä". Tämä löytyy koodin muodossa kappaleesta 3. Näitä funktioita oli omansa jokaiselle säätimelle ja vastaukselle. Eli siis esimerkiksi amount2 -> copysv2().

7 VALIDOINTISKRIPTIT

Sovelluksessa on suurimmalle osalle syötetietoja tiukat tiedon oikeellisuuden varmistukset. Ainoastaan ne tekstikentät, jotka saavat arvonsa syötteenä vastausliukusäätimiltä (merkkijono, esimerkiksi tekstikenttä tunnisteella ID, "amount1"), ovat ilman validointiskriptejä. Alla on JavaScript-kielellä toteutetut validointifunktiot erillisen tiedoston (validate.js) mukaisessa järjestyksessä. Ensin on do_s.php nimisen sivun funktiot kokonaan (koodikommentit vihreällä).

7.1 Sivu do_s.php:

```
/* Validation for do_s.php*/

function OBclickdos(){
//Lisää tietue
var retvalue;
```

```
//Paluuarvon avulla tarkastellaan erilliset validointiskriptit lävitse, onko väärää syötettä -> retvalue false
//Muuten tiedot lähetetään eteenpäin oikealle sivulle
        retvalue=cnull();
        if(retvalue == false) { return retvalue; }

        retvalue=validateminmax();
        if(retvalue == false) { return retvalue; }

        retvalue=alphanumeric();
        if(retvalue == false) { return retvalue; }

        document.Formdos.action = "do_s2.php"
        //document.Formdos.target = "_blank";        -- Open in a new window
        document.Formdos.submit();
        return true;
}
```

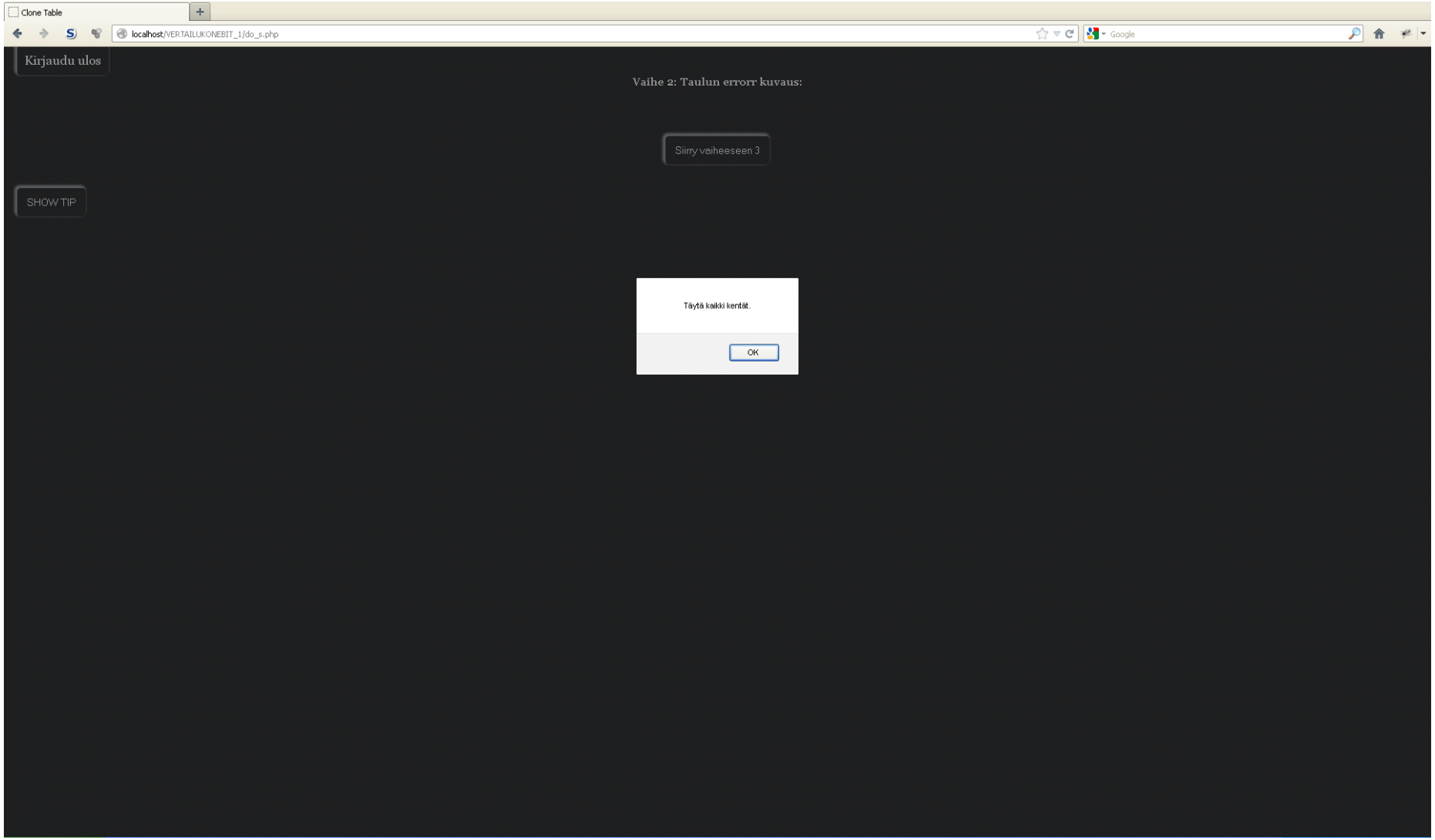
Ensimmäinen funktioista (yllä) on niin sanottu pääfunktio, joka kutsuu kaikki validointifunktiot lävitse. Jos yksikään näistä ei palauta totuusarvoa false (virhe syötteessä), niin lomakkeen, josta syötteet saadaan, tietojen lähettämisen kohteeksi määritetään sivu do_s2.php, kutsutaan lähetä funktiota (submit) ja palautetaan arvona tosi (true). Pääfunktion toiminta siis perustuu arvon palauttamiseen. Funktiota kutsutaan, kun käyttäjä yrittää vahvistaa (nappi -> ”Siirry vaiheeseen 3”) taulun luomisessa vaiheen, jossa kysytään taulun kuvausta. Jos syötteessä ei ole virheitä, funktio palauttaa arvon tosi ja annettu syöte lähtee eteenpäin (järjestelmäkäyttäjien taulun luontiprosessi, kuvattu kappaleessa 4). Lomake jota tarkastellaan on nimeltään (name=”Formdos”) Formdos.

```
function cnull(){

 if (document.forms["Formdos"]["table_d"].value==""){
 alert("Täytä kaikki kentät.");
 return false;
```

```
 }else{
 return true;
 }
}
```

Funktio cnull() tarkistaa syöttökentän, johon sijoitetaan taulun kuvaus ja joka ei saa siis olla tyhjä. Kentän ollessa tyhjä käyttäjä (järjestelmäkäyttäjä) saa JavaScript-varoituksen, joka kehoittaa täyttämään kaikki kentät (käytännössä siis palauttaa arvon epätosi, false). Muussa tapauksessa funktio palauttaa arvon tosi. Alla kuva varoituksesta:


Clone Table
localhost/VERTAILUKONEBIT_1/do_s.php
Google
Kirjaudu ulos
Vaihe 2: Taulun errorr kuvaus:
Siirry vaiheeseen 3
SHOW TIP
Täytä kaikki kentät.
OK

Kuva 18: Kuva validointiskriptin (JavaScript) antamasta varoituksesta (alert).

```
function validateminmax(){
 if (document.getElementById('tdesc').value.length > 40) {
 alert ("Taulun kuvaus saa olla vain 40 merkkiä pitkä");
 return false;
 }else{
 return true;
 }
}
```

Funktio validateminmax() taas nimensä mukaisesti varmistaa, että syöte on sille asetetuissa merkkirajoissa eli maksimissaan 40 merkkiä pitkä. Pitempi syöte palauttaa arvon epätosi, muuten funktio palauttaa arvon tosi. Edellinen funktio varmisti, että kentässä on jotain tietoa.

```
function alphanumeric(){
// Function to check letters, numbers, space and minus sign
 var letterNumber = /^[0-9a-zA-ZåäöÅÄÖ\s\-]+$/;

 if(document.forms["Formdos"]["table_d"].value.match(letterNumber)){
   return true;
  }else{
        alert("Taulun kuvaus saa sisältää kirjaimia, numeroita, välilyöntejä ja
miinusmerkin.");
   return false;
  }
}
```

Viimeinen funktio (alphanumeric()) taas varmistaa vertaamalla syötettä säännölliseen lausekkeeseen (regular expression, regex muuttujassa letterNumber), että syöte sisältää vain sallittuja merkkejä. Taulun kuvausta (table_d) verrataan lausekkeeseen, ja jos syöte on jollain tapaa väärin, annetaan käyttäjälle varoitus ja palautetaan arvo epätosi. Sallitut merkit: /^[0-9a-zA-ZåäöÅÄÖ\s\-]+$/ eli numerot,

kirjaimet (isot, pienet sekä ääkköset), välilyönnit sekä miinusmerkki. JavaScript-varoitus toimii siten, että se tekee alla olevan sivun inaktiiviseksi ja antaa viestilaatikossa koodissa kirjoitetun varoitusviestin (kuvan tapauksessa ”Täytä kaikki kentät.”). Varoitus kuitataan Ok-painikkeella, jonka jälkeen sivu palautuu sellaisena käyttöön, kuin mitä se oli ennen virheellisen syötteen lähettämisyritystä (säilyttäen siis tiedot, eli esimerkiksi virheellisen syötteen). Kuvaus tuli taulun kommenttiin (MySQL), mutta taulun nimessä ei validoinnissa sallittu välejä, jotka eivät siis saa esiintyä MySQL-tunnisteen lopussa --

7.2 Sivu ctable.php

```
/* Validation for ctable.php*/

function CTclickctable(){
//Lisää tietue
	var retvalue;

	retvalue=cnull2();
	if(retvalue == false) { return retvalue; }

	retvalue=validateminmax2();
	if(retvalue == false) { return retvalue; }

	retvalue=alpha2();
	if(retvalue == false) { return retvalue; }

	document.ctform.action = "do_s.php";
	document.ctform.submit();
	return true;
}
```

Tällä sivulla on aivan samanlainen pääfunktio kuin edelliselläkin. Funktiotkin ovat pääosin samat. Cnull2() tarkistaa syötetiedon olemisen, validateminmax2() taas tarkistaa syötetiedon pituuden rajat ja alpha2() sallitut merkit. Toimintaperiaate on täysin sama.

```
function cnull2(){

  if (document.forms["ctform"]["table_name"].value==""){
  alert("Täytä kaikki kentät.");
  return false;
  }else{
  return true;
 }
}

function validateminmax2(){

        var x2=document.getElementById('table_name').value.length;

 if ( x2 > 7 || x2 < 6) {
 alert ("Taulun nimi saa/pitää olla vain 6-7 merkkiä pitkä");
 return false;
 }else{
 return true;
 }
}
```

Funktiossa hyvin yksinkertainen vertailu syötteen merkkien lukumäärästä.

```
function alpha2(){
// Function to check letters, numbers and minus sign
 var alpha2 = /^[a-zA-Z]+$/;

 if(document.forms["ctform"]["table_name"].value.match(alpha2)){
```

```
  return true;
 }else{
  alert("Taulun nimi saa sisältää vain kirjaimia (ei edes välejä, esim. välilyönti)");
  return false;
 }
}
```

Funktio tarkastaa sallitut merkit: /^[a-zA-Z]+$/; (kirjaimet, isot sekä pienet).

8 MUUT OMINAISUUDET

Tässä kappaleessa käsitellään vielä dokumentin lopuksi hieman sovelluksen muita ominaisuuksia, jotka mainittiin kappaleessa 2.2.7.

8.1 Turvallisuus ja suojaukset

Sovelluksen tietokantaa ja rakennetta muuttavat toiminnot on suojattu tekemällä toimintoihin liittyvät sivut salasanalla suojatuiksi (kirjautuminen). Tämä tietenkin tarkoittaa sitä, että suojatut sivut ohjaavat tuntemattomat käyttäjät kirjautumiseen. Tämän lisäksi syötekentät on kaikki määritetty siten, että on säädetty sallitut pituudet ja syötteen muoto. Syötekenttien tiedot valitoidaan ennen niiden lähettämistä eteenpäin. On kuitenkin selvää, että tähän osa-alueeseen voisi perehtyä aina paremminkin.

8.2 Joustavuus, laajennettavuus ja ylläpito

Sovellus ei ole juuri lainkaan joustava, sillä sen palveluita ei voi muuttaa ilman uutta ohjelmakoodia. Ainoa joustava piirre tulee sen toimintaympäristöstä (Internet), jonka

standardien mukaisesti sovellus toimii lähes selaimessa kuin selaimessa käyttöjärjestelmästä välittämättä.

Sovellus on siten laajennettavissa, että kysymysten lisääminen on melko helppoa. Muutamaa ohjelmakoodin pätkää muokkaamalla (seuraava versio) vielä dynaamisemmaksi kokonaisuudeksi, jäisi uuden ohjelmakoodin muokkaaminen vallan pois uusien kysymysten lisäämisen prosessista.

Aikaisemmin tuli jo mainittua, kuinka ylläpidon tarve on hyvin minimaalinen sovelluksessa. Työtä aiheuttaa oikeastaan vain uusien koulutusohjelmien lisääminen (järjestelmäkäyttäjän tehtäviä tietenkin). Ei ole järkevää mitenkään pienestä syystä muokata kysymyssarjoja, jolloin kaikilta koulutusohjelmilta (edustaja) pitää pyytää vastausten päivitystä.

8.3 Operointi ja Sovitettavuus erilaisiin ympäristöihin

Sovelluksessa ei ole omaa tietojen varmistamista, vaan esimerkiksi tietokantojen varmuuskopiointi tulee tehdä MySQL-palvelinohjelmiston omilla työkaluilla. Workbench on tähän hyvä ja varmuuskopiointi suositeltava tapa. Ainakaan tässä versiossa sovellusta ei ole lokitoimintoa, joka kuvaisi esimerkiksi tietokantojen päivitykset.

Sovellus on tosiaan toimintaympäristönsä johdosta helposti siirrettävissä eri käyttöympäristöihin, mutta sovelluksen sovittaminen eri kielialueeseen ei tässä versiossa ole oikeastaan vaihtoehto. Jos sovellukseen jossain vaiheessa tulee määrittelyssä hyödylliseksi todettu mahdollinen lisäominaisuus eli englannin kielen vaihtoehto, niin tällöin ei olisi suurempia esteitä sovittaa sovellusta muuallekin.

8.4 Testausnäkökohdat

Sovellusta ei ole erikseen testattu, vaikka sellainen hyvin onnistuisi koulutuksen puolesta ja harjoituksen perusteella. Sovelluksen testaus on tapahtunut enemmänkin tarkistelemalla pala kerrallaan koodin toimintaa tulosteiden muodossa. Jo nyt projektin kokonaissivumäärä liitteiden kanssa on ylitse 100 sivua.

LÄHDELUETTELO

1. Kenneth Falck, 6/2010, Tietokone: Nettitekniikat sodassa, http://www.tietokone.fi/lehti/tietokone_6_2010/nettitekniikat_sodassa_8287.

2. http://jqueryui.com/demos/slider/#steps

3. http://dev.mysql.com/doc/refman/5.0/en/identifiers.html

4. http://dev.mysql.com/doc/refman/4.1/en/windows-installation.html

5. http://httpd.apache.org/docs/2.2/platform/windows.html

6. http://eligeske.com/jquery/what-is-the-difference-between-document-and-window-objects-2/

LIITTEET

Erilliset liitetiedostot: LIITE 1, LIITE 2 ja LIITE 3. Theseus-julkaisuarkistoa varten liitteet on kuitenkin tähän opinnäytetyödokumentin nettiversioon lisätty.

Ulkoinen Liite 1

Pääliukusäätimen alustusfunktio

Koodirivejä on jouduttu muokkaamaan dokumenttiin paremmin sopivaksi, sillä itselläni on tapana etenkin kommenttirivien kohdalla tehdä niistä todella pitkiä. Rivikommentit saattavat siis mennä useamman rivin mittaiseksi ja muutenkin saattaa rivien muodossa olla omituisuuksia. Kommenttirivejä on myös poistettu ja suomeksi on selvitetty poistettujen kommenttien asiaa! (jquery.aslider.js):

// Liukusäädin liitännäisen (plugin) alku – Puolipiste ennen funktiokutsua on turvatoimenpide siltä varalta, että funktio joskus liitetään huonosti suljettuihin muihin funktioihin.

```
;(function($, window, document, undefined) {
```

//undefined on (ECMAScript 3) määrittelemätön globaali muuttuja, mutta sitä ei oikeasti välitetä eteenpäin, joten se on varmasti määrittelemätön. ECMAScript 5: Sitä ei voida enää muokata.

//window- ja document osiot laitetaan sisään globaalin sijasta paikallisina muuttujina, koska se pikkuisen selkeyttää toimintaa.

```
//Konstruktori -> jQuery.aSlider -- el -> element ja opt -> options
        $.aSlider = function(el, opt) {
```

// Jotta vältetään muuttujien laajuudesta johtuvia ongelmia, käytetään 'base' sanaa 'this' sanan sijaan, kun viitataan tähän luokkaan sisäisistä funktioista. --this-- on jQuery objekti.

```
                var base = this, o;
```

/* Globaalien muuttujien suuri lukumäärä on huono idea, koska se lisää riskiä aiheuttaa ongelmia muiden ohjelmien kanssa. Ideaalisti globaaleja muuttujia ohjelmakomponentissa olisi vain yksi.*/

// Käärii ul-elementin tarvittaviin div-osioihin ja antaa pääsyn jQuery-elementille.

base.el = el;

// $el - jQuery objekti elementistä el.

base.$el= $(el).addClass('aBase').wrap(' '); //aSlider ja aWindow nimetyt div-luokat “kääritään” uuden elementtiluokan el-aBase ympärille

//Lisätään käänteinen viittaus DOM-objektiin

base.$el.data("aSlider", base);

// Sisäänpääsy elementin jQuery ja DOM versioihin (base.$el and base.el)

//Muuttuja alustamiselle, nimetön funktio sisällölle

base.init = function(){

// "o" kirjainta käytetään koodissa "base.opt(options)" sijaan

base.opt = o = $.extend({}, $.aSlider.defaults, opt);

//Jottei muuteta mahdollisten tulevien liitännäisen toteutuksien oletusasetuksia, välitetään tyhjä objekti kohteena

-- Extend funktio siis yleensä yhdistää (ominaisuudet) kaksi objektia ja tallettaa tuloksen ensimmäiseen (tyhjä kohde) --

//.options.option_name – Mallilla pääsee käsiksi mihinkä tahansa asetukseen.

// Sitten ruvetaan antamaan alustusdataa

base.initialized = false;

// onBeforeInitialize -> takaisinkutsu kohtaan, jonka jälkeen liitännäinen alustetaan - laukaistaan heti aSlider-objektin luomisen jälkeen.

```
if ($.isFunction(o.onBeforeInitialize)) { base.$el.bind('before_initialize', o.onBeforeInitialize); }
//Liitetään tapahtumankäsittelijä suoraan elementtiin, tapahtuma -> before_initialize.

base.$el.trigger('before_initialize', base);   //manuaalisesti laukaistaan käsittelijä

// Säilötään olemassa olevat DOM-elementit myöhempää käyttöä varten. Base.$el = original ul
//closest-funktio nappaa lähimmän valitsijaa vastaavan elementin aloittaen nykyisestä ja edeten ylöspäin (div.aSlider ja div.aWindow) DOM-puussa
// wrap -> otetaan sopivien elementtien (base.$el) vanhemmat (ul) ja niistä lähin joka vastaa valitsijaa div.aSlider ja lisätään luokka 'aSlider-'  (– ei ole kirjoitusvirhe).
base.$wrapper = base.$el.parent().closest('div.aSlider').addClass('aSlider-');
//parent() -> siis yksi taso ylöspäin DOM-puuta (Dokumenttiobjektimalli – Document Object Model).

base.$window = base.$el.closest('div.aWindow');
base.win = window; // Alustetaan muuttujat oikeilla arvoilla
base.$win = $(base.win);

//Sijoitetaan olemassa olevat div-osiot toisien osioiden sisälle käyttäen hyväksi jQuery-objekteja ja sijoitetaan samalla oikeaan objektiin
base.$controls = $('<div class="aControls"></div>').appendTo(base.$wrapper);
base.$nav = $('<ul class="thumbNav" />').appendTo(base.$controls);

// Asetetaan muutamia oletusarvoja ja haetaan arvoja
base.flag = false; // tapahtumalippu estämään useat kutsut (click/focusin)
//hovered – on tosi, jos liukusäädin juuri parhaillaan leijuu (hover).
base.hovered = false;
base.panelSize = [];  // sisältää mitat ja vasemman sijainnin jokaiselle paneelille
//currentPage – Juuri näkyvä sivu, sivunumero. Päivitetään, kun animaatio on valmis.
base.currentPage = o.startPanel = parseInt(o.startPanel,10) || 1; // varmistetaan ettei ole merkkijono
o.changeBy = parseInt(o.changeBy,10) || 1; // "o" käytettiin "base.options" sijasta
```

```
base.adj = (o.infiniteSlides) ? 0 : 1; // sivumäärän rajoitus, rajattu vai rajaton
base.width = base.$el.width(); //diviin kääritytn ul-elementin leveys
base.height = base.$el.height(); //diviin kääritytn ul-elemntin korkeus

base.outerPad = [ base.$wrapper.innerWidth() - base.$wrapper.width(), base.$wrapper.innerHeight() - base.$wrapper.height() ];
//innerWidth -> Nykyinen leveys ensimmäisellä elementillä kohdistetuissa elementeissä (mukaan lukien pehmuste muttei reunusta).
//width -> Nykyinen leveys ensimmäisellä elementillä kohdistetuissa elementeissä (vain elementti, ei yksikköä pikseleillä).
// Laajennetaan liukusäädin sopimaan vanhempaan
if (o.expand) {
                base.$outer = base.$wrapper.parent();
base.$window.css({ width: '100%', height: '100%' });
// ilmeisesti tarvitaan Opera-selainta varten (ei testattu). Asetetaan kaksi CSS-ominaisuutta määritellyille elementeille (.$window).
                base.checkResize(); // Säädetään liukusäätimen mitat vanhemman (elementti) uudelleenmitoituksessa (resize)
                }

// rakennetaan eteenpäin/taaksepäin napit
if (o.buildArrows) { base.buildNextBackButtons(); }
                        base.updateSlider(); //Päivittää liukusäätimen
//$lastPage - jQuery objekti aloitussivusta – sama sivu kuin "slider.currentPage" kunnes "slide_complete" kutsutaan. -- $currentPage - jQuery objekti Kohde-sivusta
                        base.$lastPage = base.$currentPage;
// Varmistetaan että easing (animaatio, swing) funktion löytyy.
                        if (!$.isFunction($.easing[o.easing])) { o.easing = "swing"; }
// Otetaan indeksi (ajokerrat) liukusäätimestä tällä sivulla, montako on sivulla
                        base.runTimes = $('div.aSlider').index(base.$wrapper) + 1;
// base.regex -> parametrikuvio (3 osaa) joihin verrataan ja määrite i -> case-sensitive (kirjainkoko merkitsee)
base.regex = new RegExp('panel' + base.runTimes + '-(\\d+)', 'i'); // hash-tagi
```

```
if (base.runTimes === 1) { base.makeActive(); } // ensimmäinen liukusäädin sivulla aktiiviseksi

// Jos hash ei voida käyttää liitännäisen starttiin, mennään startti paneeliin
base.setCurrentPage(base.gotoHash() || o.startPage, false);

// Lisätään näppäimistönavigaatio
//Jos tarvitsee napata näppäinpainalluksia, on hyödyllistä liittää keyup-funktio dokumentiin (document).
                $(document).keyup(function(e){
//enableKeyboard : true,  jos epätosi, näppäimistön nuolinäppäimet eivät toimi.
// Estä nuolinäppäinten toiminta, kun kohdistus on lomake-elementeissä
if (o.enableKeyboard && base.$wrapper.is('.activeSlider') && !e.target.tagName.match('TEXTAREA|INPUT|SELECT')) {

        if (!o.vertical && (e.which === 38 || e.which === 40)) { return; }
//vertical : false, Jos tosi, kaikki paneelit liukuvat pystysuuntaan, muuten vaaksasuuntaan. Ei pystysuuntaan liukumista -> ei ylös ja alas nuolinäppäimiä
                switch (e.which) {
                        case 39: case 40: // oikea & alas nuolinäppäin
                        base.goForward();
                                break;
                        case 37: case 38: // vasen & ylös nuolinäppäin
                        base.goBack();
                                break;
                                }
                        }
                });

            };

// kutsutaan alustuksen aikana päivittämään liukusäädin, jos siihen lisätään tai poistetaan paneeleja
        base.updateSlider = function(){
```

// Tarvitaan säätimen päivitykseen. Poistaa elementin ja kaiken sisällön (poistaa elementit DOM-puusta)

base.$el.children('.cloned').remove();

//children -> yksi taso alaspäin DOM-puussa, luokalla ”cloned” varustetut lapsielementit poistetaan.

base.$nav.empty(); //Tämä funktio poistaa kaikki lapsielementit (muut perilliset myös), mutta myös kaiken sisältötekstin.

// asetetaan currentPage ->1, jottei se voi ole nolla – tämä tapahtuu, kun kaikki paneelit poistetaan ja sitten lisätään uusia

base.currentPage = base.currentPage || 1;

base.$items = base.$el.children();

//pages - Sivujen lukumäärä liukusäätimessä.

base.pages = base.$items.length; //length -> elementtien lukumäärä jQuery-objektissa --- base.$items = base.$el.children();

base.dir = (o.vertical) ? 'top' : 'left'; // left -> vaakasuunta, vertical -> epätosi

-- Tämä vertailumuoto toimii näin: Ennen kysymysmerkkiä olevaa testataan ja jos testi on tosi -> suoritetaan se mitä on kysymysmerkin jälkeen, muuten se mitä on kaksoispisteen jälkeen. Tässä siis vertailu palauttaa epätoden --

//lisätään ohjaimet, kuten esimerkiksi navigaatioelementit

base.$controls

.add(base.$nav)

.add(base.$forward)

.add(base.$back)[(base.pages <= 1) ? 'hide' : 'show']();

// piilotetaan navigaatio, jos sivuja on vain yksi

if (base.pages > 1) {

// rakennetaan/päivitetään navigaatiovälilehdet

base.buildNavigation();

}

// Huippu (top) ja häntä (tail). Listassa näkyvät nimikkeiden (item) numerointi, huipulla on viimeinen osio ja hännällä ensimmäinen

// Tämä tukee ns. rajatonta selausta (kuin sivut pyörähtäisivät ympäri), varmistaa myös ettei kloonatut elementit saa kaksoiskappaleita elementtitunnuksista (ID)

if (o.infiniteSlides && base.pages > 1) {

// Joten poistetaan ID-tunnukset ja lisätään kloonatuille elementeille sen sijaan luokka -cloned-

base.$el.prepend(base.$items.filter(':last').clone().removeAttr('id').addClass('cloned')); //huipulla on loppuosio (prepend, last)

// Lisätään tuki useamman liukusäätimen yhtäaikaiselle käytölle

if (o.showMultiple > 1) { //showMultiple : false, jos numeroarvo, näyttää niin monta paneelia kerralla.

// Joten poistetaan ID-tunnukset ja lisätään kloonatuille elementeille sen sijaan luokka -cloned- ja luokka -multiple- jotta eroavat edellisistä

base.$el.append(base.
$items.filter(':lt('+o.showMultiple+')').clone().removeAttr('id').addClass('cloned').add
Class('multiple')); //pre -> huippu ja ap -> häntä

} else { //Ei useampia paneeleja, joten poistetaan ID-tunnukset ja lisätään kloonatuille elementeille sen sijaan luokka -cloned-

base.$el.append(base.$items.filter(':first').clone().removeAttr('id').addClass('cloned')); //hännällä on ensimmäinen osio

}

base.$el.find('.cloned').each(function(){

// disable, kohdistettavat elementit paneeleissa, jotta estetään paneelien siirtyminen tabulaattorilla

$
(this).find('a,input,textarea,select,button,area').attr('disabled', 'disabled');

$(this).find('[id]').removeAttr('id');

});

}

// Lisättiin kaksi uutta nimikettä (item), lista säilötään uudestaan (cache), sitten asetetaan kaikkien paneelien mitat

base.$items = base.$el.children().addClass('panel' + (o.vertical ? ' vertical' : ''));

base.setDimensions();

```
// Asetetaan jokaisen paneelin mitat
                    if (o.resizeContents) {
                    base.$items.css('width', base.width);
                    base.$wrapper.css('width',
base.getDim(base.currentPage)[0]); //otetaan mitat funktio
          base.$wrapper.add(base.$items).css('height', base.height);
                    } else {
          base.$win.load(function(){   base.setDimensions();   });   //   asetetaan
mitat, kun kaikki sivulla on ladattu
                    }

                    if (base.currentPage > base.pages) {
                    base.currentPage = base.pages;
                    }
                    base.setCurrentPage(base.currentPage, false);
          base.$nav.find('a').eq(base.currentPage - 1).addClass('cur');
// päivitetään, että mikä on tämän hetkinen valinta
          };

// Luo numeroidut välilehdet ja lisää ne navigaatioon
     base.buildNavigation = function() {
     if (o.buildNavigation && (base.pages > 1)) { //yhtä enemmän sivuja
                    var t, $a;
//:not -> valitsee ne elementit, jotka eivät vastaa valitsijaa. filter -> taas ottaa ne
elementit, jotka vastaavat valitsijaa -> joten ei kloonattuja nimekkeitä (items)
          base.$items.filter(':not(.cloned)').each(function(i) {
                              var index = i + 1;
     t = ((index === 1) ? 'first' : '') + ((index === base.pages) ? 'last' : ''); //jotta
erotetaan li-elementit
$a = $('<a href="#"></a>').addClass('panel' + index).wrap('<li class="' + t + '" />');

     base.$nav.append($a.parent()); // käytetään ilmaisua $a.parent(), jotta lisätään
<li> elementti (eikä vain <a>), <ul> elementtiin
```

```
$a.html('<span>' + index + '</span>'); //sivunumero, <span> tagi tarjoaa tavan liittää “koukku” tekstin tai dokumentin osaan.

$a.bind(o.clickControls, function(e) {
        if (!base.flag && o.enableNavigation) {
// estetään funktioiden tupla-ajo (kerran toiminnolle click, toinen focusin toiminnolle)
base.flag = true; setTimeout(function(){ base.flag = false; }, 100);
                base.gotoPage(index);
//if (o.hashTags) { base.setHash(index); } –hashTags : true, pitäisikö linkkien vaihtaa hashia (#) URL issa
                                }
                //Estetään oletustoiminto
                e.preventDefault();
});
});

// Lisätään navigointivälilehtien selaus – käytetään, jos kooksi asetetaan nolla
if (!!o.navigationSize && o.navigationSize < base.pages) {

        if (!base.$controls.find('.aNavWindow').length){
base.$nav
.before('<ul><li class="prev"><a href="#"><span>' + o.backText + '</span></a></li></ul>')
/*backText : "&laquo;", // Linkkiteksti, jolla säädin siirtyy taaksepäin (piilotettu CSS llä ja valitulla kuvalla, nuolikuva)*/
.after('<ul><li class="next"><a href="#"><span>' + o.forwardText + '</span></a></li></ul>') /*forwardText : "&raquo;", // Linkkiteksti, jolla säädin siirtyy eteenpäin, piilotettu CSS llä ja valitulla kuvalla, nuolikuva)
.wrap('<div class="aNavWindow"></div>');
                        }
                }

        }
```

```
                    };

            // Luo eteenpäin/taaksepäin napit (funktio)
            base.buildNextBackButtons = function() {
base.$forward = $('<span class="arrow forward"><a href="#"><span>' + o.forwardText + '</span></a></span>');

base.$back = $('<span class="arrow back"><a href="#"><span>' + o.backText + '</span></a></span>');

// Nappien sitominen (bind). Lisätään siis elementeille käsittelijä (funktio) tapahtumaan.
//clickBackArrow      : "click",        // tapahtuma, jolla aktivoidaan taaksepäinnuolen toiminto
base.$back.bind(o.clickBackArrow, function(e) {
                                    estetään funktioiden tupla-ajo (kerran toiminnolle click, toinen swipe ille)
if (o.enableArrows && !base.flag) {
                    base.flag = true; setTimeout(function(){ base.flag = false; }, 100);
                                    base.goBack();
                                    }
                                    e.preventDefault();
                            });

//clickForwardArrow   : "click",        //tapahtuma, jolla aktivoidaan eteenpäinnuolen toiminto
                    base.$forward.bind(o.clickForwardArrow, function(e) {
                            if (o.enableArrows && !base.flag) {
                    base.flag = true; setTimeout(function(){ base.flag = false; }, 100);
                                    base.goForward();
                                    }
                                    e.preventDefault();
                            });
```

// tabulaattorilla nuoliin siirtyminen näyttää, että niillä on focus/hover (ulkoraja poistettu tyylitiedostossa)

```
base.$back.add(base.$forward).find('a').bind('focusin focusout',function(){
                $(this).toggleClass('hover');
                });

// Lisätään elementit sivulle
base.$back.appendTo((o.appendBackTo !== null && $(o.appendBackTo).length) ? $(o.appendBackTo) : base.$wrapper );

base.$forward.appendTo((o.appendForwardTo !== null && $(o.appendForwardTo).length) ? $(o.appendForwardTo) : base.$wrapper );

base.$arrowWidth = base.$forward.width(); // oletetaan että vasen/oikea nuolinäppäimet ovat saman levyisiä (ne ovat) – käytetään vaihdolle (toggle)
        };

// Muutetaan liukusäätimen mittoja, jos vanhempi (parent, elementti) muuttaa kokoaan (funktio)
                base.checkResize = function(stopTimer){
                clearTimeout(base.resizeTimer);
                base.resizeTimer = setTimeout(function(){
                var w = base.$outer.width() - base.outerPad[0],
                        h=(base.$outer[0].tagName === "BODY" ? base.$win.height() : base.$outer.height()) - base.outerPad[1];
// base.width = yhden paneelin leveys, joten kerrotaan paneeleiden lukumäärällä; outerPad on nuolien pehmuste (padding).
        if (base.width * o.showMultiple !== w || base.height !== h) {
                        base.setDimensions(); // muutetaan paneelikoot
// varmistetaan että sivu on linjassa (arvo -1 animaatioaikana, jotta se erottuu kun animationTime = 0)
        base.gotoPage(base.currentPage, null, -1);
                }
if (typeof(stopTimer) === 'undefined'){ base.checkResize(); }
```

```
                }, 500);
            };

// Joko muutetaan sisällön kokoa tai paneeli muutetaan sisällön mukaan
                base.setDimensions = function(){
                var w, h, c, edge = 0,
                        fullsize = { width: '100%', height: '100%' },
                        // määritetään paneelin leveys
pw = (o.showMultiple > 1) ? base.width || base.$window.width()/o.showMultiple :
base.$window.width(),
                        winw = base.$win.width();

                if (o.expand){
                        w = base.$outer.width() - base.outerPad[0];
        base.height = h = base.$outer.height() - base.outerPad[1];
base.$wrapper.add(base.$window).add(base.$items).css({ width: w, height: h });
        base.width = pw = (o.showMultiple > 1) ? w/o.showMultiple : w;
                }

        base.$items.each(function(i){
                        c = $(this).children();
                if (o.resizeContents){
                                // muutetaan paneelin koko
                                w = base.width;
                                h = base.height;
                                $(this).css({ width: w, height: h });
            if (c.length) {
if (c[0].tagName === "EMBED") { c.attr(fullsize); } // needed for IE7; also c.length
> 1 in IE7
if (c[0].tagName === "OBJECT") { c.find('embed').attr(fullsize); }
                // muutetaan sisällön koko
                if (c.length === 1){ c.css(fullsize); }
                }
                        } else {
```

```
// selvitetään leveys & korkeus -> talteen
w = $(this).width() || base.width; // jos sisältö ei ole vielä latautunut, leveys on nollassa, joten asetetaan sitten base.width tässä tilanteessa
if (c.length === 1 && w >= winw){
w = (c.width() >= winw) ? pw : c.width(); // yksinäisen lapsielementin leveys
c.css('max-width', w); // asetetaan maksimileveys kaikille lapsielementeille
}
$(this).css('width', w); // asetetaan paneelin leveys
h = (c.length === 1 ? c.outerHeight(true) : $(this).height()); // haetaan korkeus leveyden asettamisen jälkeen
if (h <= base.outerPad[1]) { h = base.height; } // jos korkeus on vähemmän kuin ulompi pehmuste (outer padding), asetetaan se nykyiseen korkeuteen
$(this).css('height', h);
}
base.panelSize[i] = [w,h,edge]; // base.panelSize = []; sisältää jokaisen paneelin vasemman sijainnin ja mitat
edge += (o.vertical) ? h : w;
});
// asetetaan liukusäätimen kokonaisleveys, rajoitettu -> 32766 (Opera)
base.$el.css((o.vertical ? 'height' : 'width'), edge);
};

// haetaan useamman paneelin mitat tarpeen mukaan
base.getDim = function(page){
if (base.pages < 1 || isNaN(page)) { return [ base.width, base.height ]; } // estää virheet, kun base.panelSize on tyhjä
page = (o.infiniteSlides && base.pages > 1) ? page : page - 1;
var i,
w = base.panelSize[page][0],
h = base.panelSize[page][1];
if (o.showMultiple > 1) {
```

```
                for (i=1; i < o.showMultiple; i++) {
                w += base.panelSize[(page + i)%o.showMultiple][0];
                h = Math.max(h, base.panelSize[page + i][1]);
                        }
                }
                return [w,h];
        };

        /*Rtl -> Right to left (oikealta vasemmalle). changeBy -> montako paneelia liikutaan. Asetuksissa on tällä hetkellä yksi(1)*/
        base.goForward = function() {
        base.gotoPage(base.currentPage + o.changeBy * (o.playRtl ? -1 : 1));
        };

        base.goBack = function() {
    base.gotoPage(base.currentPage + o.changeBy * (o.playRtl ? 1 : -1)); //Arvot testataan eri järjestyksessä
        };

//Kun liikutaan useampi paneeli kerralla (numbered navigation ja ”click”)
        base.gotoPage = function(page, callback, time) {
                // tarkista onko sivu, id tai luokka
                if (/^[#|.]/.test(page) && $(page).length) {
                        page = $(page).closest('.panel').index() + base.adj;
                }
                // rewind ("faster" animation) tapahtuu kun changeBy > 1 (takaisinkelaus, “nopeampi” animaatio)
                if (o.changeBy !== 1){
                        if (page < 0) { page += base.pages; }
                        if (page > base.pages) { page -= base.pages; }
                }
                if (base.pages <= 1) { return; } // estää animaation
                base.$lastPage = base.$currentPage;
```

```
                    if (typeof(page) !== "number") { //tarkista muuttujan tyyppi ja
vertaa, että ei yhtäsuuri kuin ”number” (numero, tyyppi)
                        page = o.startPanel;
                        base.setCurrentPage(page);
                    }

/* base.pages = base.$items.length; length -> elementtien lukumäärä jQuery-
objektissa --- base.$items = base.$el.children();
base.adj = (o.infiniteSlides) ? 0 : 1; --- sivumäärän rajoitus, rajattu vai rajaton
infiniteSlides  : true,    jos epätosi, liukusäädin ei kääri eikä kloonaa paneeleja */

// muuttujia ajan tasalle ja sen mukaan, että mikä on infiniteSlides-asetus
if (page > base.pages + 1 - base.adj) { page = (!o.infiniteSlides) ? 1 : base.pages; }
        if (page < base.adj ) { page = (!o.infiniteSlides) ? base.pages : 1; }

base.currentPage = ( page > base.pages ) ? base.pages : ( page < 1 ) ? 1 :
base.currentPage;
base.$currentPage = base.$items.eq(base.currentPage - base.adj);
                    base.exactPage = page;

base.targetPage = (page === 0) ? base.pages - base.adj : (page > base.pages) ? 1 -
base.adj : page - base.adj;
                    base.$targetPage = base.$items.eq( base.targetPage );
                    time = time || o.animationTime;
                    if (time >= 0) { base.$el.trigger('slide_init', base); }
//laukaistaan kun paneelin liu’uttaminen alustetaan (ennen kuin ohjaimet, controls,
animoituvat).
                    base.slideControls(true, false);
                    //viivästytetään animaation alkamista
                    setTimeout(function(d){
                    // määritä liukusäätimen koko uudestaan, jos sisällön koko
vaihtelee
                if (!o.resizeContents) {
```

```
// animoidaan wrapperin koonmuutos ennen ikkunaa (window), estää välkehdintää (Firefox)
                d = base.getDim(page); //hae mitat funktio

/* The .animate()-funktio sallii toteuttaa animaatioita mille tahansa numeeriselle CSS-ominaisuudelle. Parametriksi vaaditaan kartta (map) CSS-ominaisuuksista. */

/*.animate( properties, options )  --- http://api.jquery.com/animate/ */
base.$wrapper.filter(':not(:animated)').animate(
                // estää animoimasta mittoja nolliin
                { width: d[0] || base.width, height: d[1] || base.height },
                { queue: false, duration: (time < 0 ? 0 : time), easing: o.easing }
//easing : "swing",    -- Muut optiot kuin "linear" tai "swing" vaativat easing-liitännäistä tai jQuery käyttäjäliittymän (UI)
                                        );
                            }
d = {}; // base.panelSize = [];  sisältää koon ja vasen sijainnin joka paneelille

d[base.dir] = -base.panelSize[(o.infiniteSlides && base.pages > 1) ? page : page - 1][2];
//var dir sisältää liu'un suunnan – left (vasen) -> vaakataso, pystytaso epätosi

// Animoidaan Slideri
                base.$el.filter(':not(:animated)').animate(
d, { queue: false, duration: time, easing: o.easing, complete: function() { base.endAnimation(page, callback, time); } }
//complete: Funktio, jota kutsutaan kun animaatio on valmis.
                            );
                }, parseInt(o.delayBeforeAnimate, 10) || 0); //parseInt()-funktio jäsentää merkkijonon ja palauttaa kokonaisluvun.
                };

// Funktio, joka kutsutaan kun jQuery animaatio on valmis (easing)
                base.endAnimation = function(page, callback, time){
```

```
if (page === 0) { //var dir sisältää liu'un suunnan – left (vasen) -> vaakataso, pystytaso epätosi

base.$el.css( base.dir, -base.panelSize[base.pages][2]); // .css()-funktio on kätevä tapa hakea tyyliominaisuus määritetystä elementistä
page = base.pages;
} else if (page > base.pages) { //sivunumero suurempi kuin maksimi sivujen lukumäärä!
// paluu takaisin alkuasemaan
base.$el.css( base.dir, -base.panelSize[1][2]);
page = 1;
}
base.exactPage = page;
base.setCurrentPage(page, false);
// lisää aktiivi paneeliluokka
base.$items.removeClass('activePage').eq(page - base.adj).addClass('activePage');
if (!base.hovered) { base.slideControls(false); } //

if (o.hashTags) { base.setHash(page); } //Hash-tagit toimimaan myös nuolille, ei vain klikkaamalla numeroituja navigointilinkkejä

if (time >= 0) { base.$el.trigger('slide_complete', base); }
// takaisinkutsu ulkoisesta liu'utuksen ohjauksesta: $('#slider').aSlider(4, function(slider){ })
if (typeof callback === 'function') { callback(base); }
};

base.setCurrentPage = function(page, move) {
page = parseInt(page, 10); // 1. parametri -> kohdemerkkijono 2. parametri -> Numero ( 2 - 36) joka edustaa käytettävää numerojärjestelmää
if (base.pages < 1 || page === 0 || isNaN(page)) { return; }
if (page > base.pages + 1 - base.adj) { page = base.pages - base.adj; } //base.adj = (o.infiniteSlides) ? 0 : 1;
if (page < base.adj ) { page = 1; }
```

-- Useimmat jQueryn DOM:a käsittelevät metodit työskentelevät jQuery oliototeutuksella ja tuottavat uuden kun kohdistetaan uuteen joukkoon elementtejä. Tällainen uusi elementtijoukko ”siirtyy” pinoon, jota hallitaan olion sisällä. Näitä tulee aina lisää, kun filteröidään ja jos tarvitaan pinosta vanhempi elementti, niin voidaan käyttää end()- funktiota noutamaan niitä pinosta --

```
// Asetetaan näytettävää.
if (o.buildNavigation){// vaihtaa nykyisen (current) nav-elementin, linkit -> numeroidut linkit
                base.$nav
                        .find('.cur').removeClass('cur').end()
                        .find('a').eq(page - 1).addClass('cur');   // Kun eq-metodille annetaan jQuery-olio, joka edustaa joukkoa DOM-elementtejä, metodi muodostaa uuden olion yhdestä joukon elementistä. Indeksinumero kertoo monennestako (indeksointi alkaa tutusti nollasta)
        }

        // piilotetaan/näytetään nuolet infinite scroll moodin mukaan
        if (!o.infiniteSlides){
                base.$wrapper
.find('span.forward')[ page === base.pages ? 'addClass' : 'removeClass']('disabled').end()

.find('span.back')[ page === 1 ? 'addClass' : 'removeClass']('disabled');
        }

        // vaihto vain vasemmalle, jos move ei ole yhtäsuuri kuin false
        if (!move) {
                var d = base.getDim(page); //haetaan paneelin mitat
                base.$wrapper
                        .css({ width: d[0], height: d[1] })
                        .add(base.$window).scrollLeft(0);   // resetti, arvo nollaksi varmuuden vuoksi.
```

```
base.$el.css( base.dir, -base.panelSize[(o.infiniteSlides && base.pages > 1) ? page : page - 1][2] ); //Paneeleissa HTML-sisältöä (voi olla mitä tahansa).
                        }// css( ominaisuuden_nimi , arvo  )
                        // päivitetään paikallinen muuttuja
                        base.currentPage = page;
                        cpaget= page; // muuttuja liukusäädin/sivunumerolle

base.$currentPage = base.$items.removeClass('activePage').eq(page - base.adj).addClass('activePage');
                };

// Asetetaan nykyinen liukusäädin aktiiviseksi, jotta näppäimistöpohjainen navigaatio toimisi
                        base.makeActive = function(){
                        if (!base.$wrapper.is('.activeSlider')){
                                $('.activeSlider').removeClass('activeSlider');
                                base.$wrapper.addClass('activeSlider');
                        }
                        };

//Metodi yrittää löytää hash-tiedon, joka vastaa ID-tunnusta ja/tai oikeaa paneelia paneeli-x (x on indeksinumero), jos jompi kumpi löytyy, niin yritetään löytää vastaava nimeke (item), jos sekin löytyy -> palautetaan sivunumero

                //Tämä mahdollistaa linkeillä suoraan tiettyyn paneeliin osoittamisen
                base.gotoHash = function(){
                        var h = base.win.location.hash,
                                i = h.indexOf('&'),
n = h.match(base.regex); //base.regex = new RegExp('panel' + base.runTimes + '-(\\d+)', 'i');
                        if (n === null && !/^#&/.test(h)) {
                                // #quote2&panel1-3&panel3-3
                                h = h.substring(0, (i >= 0 ? i : h.length));
                        // varmistetaan että elementti on samassa liukusäätimessä
```

```
                n = ($(h).closest('.aBase')[0] === base.el) ? $(h).closest('.panel').index() : null;
            } else if (n !== null) {
                // #&panel1-3&panel3-3
                n = (o.hashTags) ? parseInt(n[1],10) : null; //hashTags : true, --vaihtavatko linkit hash-tagia URL-osoitteessa?
            }// onko hash-tag asetettu
            return n;
        };

        // Funktio, joka määrittää hash-tagien muodon
        base.setHash = function(n){
            var s = 'panel' + base.runTimes + '-',
                h = base.win.location.hash;
            if ( typeof h !== 'undefined' ) { //indexOf()-metodi palauttaa tietyn arvon ensimmäisen esiintymisen merkkijonossa (kirjainkoko merkitsee).
            base.win.location.hash = (h.indexOf(s) > 0) ? h.replace(base.regex, s + n) : h + "&" + s + n;
            }// Regex was earlier. string.replace(searchvalue,newvalue)
        };

//Liu'utuksen ohjaimet (controls, nav-näppäin ylös tai alas ) Kutsutaan, jos asetusten mukaisesti tahdotaan (toggleControls -> true , toggleArrows -> true) tuoda ja piilottaa ohjaimia ja navigointinuolet

        base.slideControls = function(toggle){ //parametri toggle -> true or false. tosi -> näytä ohjaimet/nuolet (jos sallittu asetuksista)
            var dir = (toggle) ? 'slideDown' : 'slideUp',
                t1 = (toggle) ? 0 : o.animationTime, /*animationtime -> aika paneelien vaihtumisen välissä*/
                t2 = (toggle) ? o.animationTime: 0,
                op = (toggle) ? 1 : 0,
                sign = (toggle) ? 0 : 1; // 0 = visible, 1 = hidden
            if (o.toggleControls) {
```

```
                                base.$controls.stop(true,true).delay(t1)[dir]
(o.animationTime/2).delay(t2);
                        }
                        if (o.buildArrows && o.toggleArrows) { //true, rakentaa
eespäin ja taaksepäin näppäimet ja true, tulevat esiin (leijumaan, hover) ja muuten
pysyvät piilossa

base.$forward.stop(true,true).delay(t1).animate({ right: sign * base.$arrowWidth,
opacity: op }, o.animationTime/2);

base.$back.stop(true,true).delay(t1).animate({ left: sign * base.$arrowWidth,
opacity: op }, o.animationTime/2);
                        }

/*.stop( [clearQueue] [, jumpToEnd] ) , http://api.jquery.com/stop/ */
                };

                // Ajetaan alustaja
                base.init();
        };

// Oletusasetusten luonti. Jokaisella paneelilla on hash-tagi. Paneeli = jQuery kääritty
(wrapped) LI kohta (item)

        $.aSlider.defaults = {
expand        : false,        // true, liukusäädin kasvaa täyttämään vanhempi (parent)
elementin
resizeContents      : true,        // true, yksinäiset objektit paneelissa laajentuvat koko
näkymään
vertical              : false,       // true, kaikki paneelit liukuvat pystysuuntaan, muuten
vaakasuuntaan
showMultiple        : false,     // numeroarvo, näyttää niin monta paneelia kerralla
easing              : "swing",    // muut efektit kuin "linear" tai "swing" vaativat easing-
liitännäisen tai jQuery käyttäjäliittymän (UI)
```

buildArrows : true, // true, rakentaa nuoliohjaimet (eteen- ja taaksepäin)

buildNavigation : true, // true, luo listan ankkurilinkkejä liitettäväksi joka paneeliin

appendForwardTo : null, // Liitetään eteenpäinnuoli HTML-elementtiin (jQuery- olio, valitsija(selector) tai HTML-noodi (HTMLNode)), jos ei nolla

appendBackTo : null, // Liitetään taaksepäinnuoli HTML-elementtiin (jQuery- olio, valitsija(selector) tai HTML-noodi (HTMLNode)), jos ei nolla

appendControlsTo : null, // Liitetään ohjaimet (navigaatio) HTML-elementtiin (jQuery- olio, valitsija(selector) tai HTML-noodi (HTMLNode)), jos ei nolla

appendNavigationTo : null, // Liitetään navigointinäppäimet HTML-elementtiin (jQuery- olio, valitsija(selector) tai HTML-noodi (HTMLNode)), jos ei nolla

toggleArrows : false, // true, sivussa olevat ohjainuolet liukuvat ulos ”leijumaan” ja muuten piilossa

toggleControls : false, // true, sama ohjaimilla (navigaatio) kuin sivunuolillakin

forwardText : "»", // Linkkiteksti, jolla liukusäädin liikkuu eteenpäin (piilossa kuvan ja CSS-muotoilun avulla, korvataan kuvalla, default.png:

Kuva 1: Kuva nuolista, joita käytetään pääliukusäätimessä ohjauselementteinä

backText : "«", // Linkkiteksti, jolla liukusäädin liikkuu taaksepäin (piilossa kuvan ja CSS-muotoilun avulla, korvataan kuvalla, default.png)

// Funktio
enableArrows : true, // false, nuolet näkyvät, mutta “click” ei toimi.
enableNavigation : true, // false, näkyvissä, mutta ei toimi.
enableKeyboard : true, // false, näppäimistön nuolinäppäimet eivät toimi.

// Navigation
startPanel : 1, // Aloituspaneeli
changeBy : 1, // Sivumäärä, joka kerralla siirrytään.
hashTags : true, // Muuttavatko linkit hash-tagia URL-osoitteessa?
infiniteSlides : true, // false, liukusäädin ei kääri eikä kloonaa paneeleja
navigationSize : false, // Maksimiluku navigaatiolehdyköiden lukumäärälle; false -> pois päältä (ei toimenpiteitä)

// Takaisinkutsut

// Interaktiivisuus

```
clickForwardArrow    : "click",          // Tapahtuma, joka aktivoi eteenpäinnuolen toiminnan
clickBackArrow       : "click",          // Tapahtuma, joka aktivoi taaksepäinnuolen toiminnan
clickControls        : "click focusin", // Tapahtumat, jotka aktivoivat navigaation ohjaustoiminnan

        };

//jQuery.fn ($.fn) on prototyyppi jQuery-oliosta
        $.fn.aSlider = function(opt, callback) {

/* Usein liitännäisen tarkoitus on vain muokata elementtejä jollain tapaa ja laittaa eteenpäin seuraavalle metodille. Jotta joten liitännäinen pitää yllä liitettävyyttä (chainability) varmistetaan, että liitännäinen palauttaa --this-- avainsanan. */

                return this.each(function(){

// Mitä liitännäinen tekee:

// --this-- on nyt DOM-elementti joka on jokaisen kierroksen (iteration, looppi) kohde
                        var page, anySlide = $(this).data('aSlider');

// Alustetaan liukusäädin, mutta estetään useampi alustus. Ei yritetä asettaa samaa dataa kahta kertaa elementteihin. Jos löytyy jo alustus, niin päivitetään data.
                        if ((typeof(opt)).match('object|undefined')){
                if (!anySlide) {
                                (new $.aSlider(this, opt));
                                } else {
                        anySlide.updateSlider(); // kutsutaan alustuksen aikana, jos paneeli lisätään tai poistetaan
                                }
// jos asetus on numero, käsitellään ulkoinen linkki sivulle #: $(element).aSlider(#)
```

```
                        } else if (/\d/.test(opt) && !isNaN(opt) && anySlide) {

page = (typeof(opt) === "number") ? opt : parseInt($.trim(opt),10); // accepts "  2  "
                                    // ei huomioida (out of bound) vääriä sivuja
                if ( page >= 1 && page <= anySlide.pages ) {
                anySlide.gotoPage(page, callback); // sivu (page) #, yksi takaisinkutsu
                                    }
// gotoPage( page, callback, time )
// page = 4
// callback alerts page, time is -1 to instantly switch pages without triggering any
events
 // Hyväksytään ID tai luokkanimi
} else if (/^[#|.]/.test(opt) && $(opt).length) {
                                    anySlide.gotoPage(opt, callback);
                        }
                });
        };

})(jQuery, window, document);
```

Ulkoinen Liite 2

Pääliukusäätimen CSS

Sitten vielä hieman tuosta pääliukusäätimien muotoilusta eli tyyleistä (aslider_temp.css):

```
/*etusivu.php*/
#slider{
	width: 700px;
	height: 196px;
	list-style: none;
	overflow-y: auto;
	overflow-x: hidden;

}
```

-- Ensinnä asetetaan (pääliukusäätimen tapauksessa tyylitiedostossa) mitat, leveys ja korkeus. Seuraavaksi määritetään, ettei käytetä (pääliukusäädinhän oli ul-elementti) listamerkkiä. Lopuksi asetetaan molemmille ulottuvuuksille ylivuotoasetus. Jos pystysuunnassa sisältö ulottuu elementin ulkopuolelle -> muodostetaan vierityspalkki ja vaakasuunnassa ylimenevä osa piiloitetaan --

```
/* Opera width restriction */
.aBase { max-width: 32766px; } -- Opera-selaimen leveysrajoitus --

/*etusivu.php*/
#sliderj{ width: 480px; height: 14px; margin-left:auto; margin-right:auto;}
#sliderj2{ width: 480px; height: 14px; margin-left:auto; margin-right:auto;}
#sliderj3{ width: 480px; height: 14px; margin-left:auto; margin-right:auto;}
#sliderj4{ width: 480px; height: 14px; margin-left:auto; margin-right:auto;}
```

-- Vastausliukusäädinten muotoilu (alustusarvot HTML-sivulla). Reuna-asetukset (margin) aiheuttavat sen, että selain laskee reuna-alueet ja liukusäädinten ollessa pääliukusäätimen sisällä käytännössä keskittää li-elementit (vastausliukusäätimet) --

-- Välissä paljon väriasetuksia eri pääliukusäätimen osille ja niiden eri tiloissa (esimerkiksi aktiivinen jne.) --

```
/**********************
 COMMON SLIDER STYLING
 **********************/
/* Overall Wrapper: 45px right & left padding for the arrows, 28px at bottom for navigation */
div.aSlider {
	display: block;
	margin: 0 auto;
	overflow: visible !important; /* needed for Opera and Safari */
	position: relative;
	padding: 0 45px 28px 45px;
}
```

-- Uutta on suhteellinen sijoittelu (position: relative), joka tarkoittaa että kohde sijoitetaan suhteessa alkuperäiseen sijaintiin. Tässä tapauksessa ei ole määritetty lukuarvoa, koska tavoite on että kohde voi tulla muiden elementtien päälle. Toinen on ylivuotoasetus, jossa tahdotaan ylimenevä osakin näkyväksi. Jos määrittelylohko

halutaan merkitä tärkeäksi (se saa suuremman painoarvon kuin normaalisti Cascade-prosessissa), ominaisuuden arvon jälkeen tulee sijoittaa avainsana "!important" --

```
/* aSlider viewport window */
div.aSlider .aWindow {
	overflow: hidden;
	position: relative;
	width: 100%;
	height: 100%;
	border-top-left-radius: 10px 5px;
	border-top-right-radius: 10px 5px;
	border-bottom-right-radius: 10px 5px;
	/* outer shadows  */
	-moz-box-shadow: -2px -2px 2px 2px #888;
	-webkit-box-shadow: -2px -2px 2px 2px#888;
	box-shadow: -2px -2px 2px 2px #888;
}
```

-- 100% leveys ja korkeus tarkoittavat, että kohde täyttää koko tilansa tai sivun --

```
/* aSlider base (original element) */
.aBase {
	background: transparent;
	list-style: none;
	position: absolute;
	overflow: visible !important;
	top: 0;
	left: 0;
	margin: 0;
	padding: 0;
}
```

-- Tausta (background, transparent) läpinäkyvä tarkoittaa, että se on läpinäkyvä. Kaikki takana näkyy. Absoluuttisesti sijoiteltu elementti sijoitetaan suhteessa

ensimmäiseen vanhempi-elementtiin (parent), jonka sijoittelu on muu kuin staattinen (oletusasetus kaikilla elementeillä). Jos sellaista ei löydy, niin sisältöosio on <html>. Huippu- ja vasen-asetukset kertovat, kuinka monta pikseliä elementti on irti sen elementin reunasta (ja samasta reunasta), jonka sisällä on --

```
/* all panels inside the slider */
.aBase .panel {
	background: transparent;
	display: block;
	overflow: hidden;
	float: left;
	padding: 0;
	margin: 0;
}
```

-- Float-asetus (kellua) kertoo mihin suuntaan elementti hakeutuu suhteessa muihin --

```
.aBase .panel.vertical {
	float: none;
}
```

-- Elementti esiintyy juuri siinä kohtaa, missä se on tekstissä --

```
/* Navigation Arrows */
div.aSlider .arrow {
	top: 50%;
	position: absolute;
	display: block;
}

div.aSlider .arrow a {
	display: block;
	height: 140px;
	margin: -70px 0 0 0; /* half height of image */
```

```
        width: 45px;
        text-align: center;
        outline: 0;
        background: url(../images/default.png) no-repeat;
}
```

-- Outline (ulkoviiva) on linja, joka piirretään elementin ympärille. Se näyttää tabulaattorin käytön yhteydessä elementin rajat, jotka nuolista on siis poistettu. Nuolilinkin taustakuva on default.png, jota ei saa toistaa. Samassa kuvassahan on molemman puolen nuolet sekä oletuksena, että aktiivisina. Kuvaa siis pilkotaan käyttöön – Default.png:


Kuva 1: Pilkottava kuva nuolilinkeistä

```
/* back arrow */
div.aSlider .back { left: 0; }
div.aSlider .back a { background-position: left top; }
div.aSlider .back a:hover, div.aSlider .back a.hover { background-position: left -140px; }
div.aSlider .back.disabled { display: none; } /* disabled arrows, hide or reduce opacity: opacity: .5; filter: alpha(opacity=50); */
```

-- Nuolen sijoittelu (vasemman puolen nuoli) oikeaan kohtaan ja sen muutos kun siihen kohdistetaan hiiriosoitin (hover) --

```
/* forward arrow */
div.aSlider .forward { right: 0; }
div.aSlider .forward a { background-position: right top; }
div.aSlider .forward a:hover, div.aSlider .forward a.hover { background-position: right -140px; }
div.aSlider .forward.disabled { display: none; } /* disabled arrows, hide or reduce opacity: opacity: .5; filter: alpha(opacity=50); */
```

-- Sama oikean puolen (eteenpäin) nuolelle --

```
/* Navigation Links */
div.aSlider .aControls { outline: 0; display: none; }
div.aSlider .aControls ul { margin: 0; padding: 0; float: left; }
div.aSlider .aControls ul li { display: inline; }
```

-- Inline-asetus: Inline-elementtien välille ei muodostu etäisyyttä. Ei rivinvaihtoa edessä tai perässä, eikä ”hylji” viereisiä elementtejä --

```
div.aSlider .aControls ul a {
        font: 11px/18px Georgia, Serif;
        display: inline-block;
        text-decoration: none;
        padding: 2px 8px;
        height: 18px;
        margin: 0 5px 0 0;
        background-image: url(../images/default.png);
        background-position: center -288px ;
        background-repeat: repeat-x;
        text-align: center;
        outline: 0;
        border-radius: 0 0 5px 5px;
```

```
        -moz-border-radius: 0 0 5px 5px;
        -webkit-border-radius: 0 0 5px 5px;
}

-- Taustakuva (default.png) sijoitetaan taustan keskelle ja sitä toistetaan vain vaakasuunnassa (oletus on molemmat). Pikseliarvo kertoo kuvan kohdan x-akselilla --

div.aSlider .aControls ul a:hover {
        background-image: none;
}

/* Navigation size window */
div.aSlider .aControls .aNavWindow { overflow: hidden; float: left; }
```

Ulkoinen Liite 3

HTML- ja PHP-sivujen toteutus

Etusivu.php:

```
<?php session_start(); ?>
```

-- Kahdella vaakaviivalla merkitsen koodisivulle kuulumattomat tekstiselvennykset kuten tässä sen, että ensimmäisenä etusivulla aloitetaan käyttäjälle PHP-istunto (ohjelmallinen). Käyttäjähän saa henkilökohtaisen istuntotunnuksen (asiaa selvennetty termiluettelossa) --

```
<HTML>

<HEAD>

<META http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
```

-- Metadatana ilmoitetaan sivulle merkistö jota käytetään -> iso-8859-1 --

```
<TITLE>SLIDERS</TITLE>

<!-- Favorite page browser icon for PC-computer -->
```

```
<LINK rel="shortcut icon" href="favicon.ico">
```

-- Yllä on määritetty niin sanottu pikakuvakeikoni sivulle eli esimerkiksi välilehdessä näkyvä sivun logo. Tämä logokuva on suoraan projektikansion juuressa --

```
<!-- jQuery (required) -->
<!-- <SCRIPT src="http://ajax.googleapis.com/ajax/libs/jquery/1.6/jquery.min.js">
</SCRIPT> -->

<SCRIPT src="js/jquery/jquery2.js"></SCRIPT>
<SCRIPT src="js/jquery/jquery-ui.custom2.js"></SCRIPT>
<SCRIPT src="js/jquery/ui.slider2.js"></SCRIPT>
<SCRIPT src="js/jquery/temp.js"></SCRIPT>
```

-- Tässä on määritelty sivuilla tarvittavien jQuery-kirjastojen sijainnit. Ensimmäinen oli itse asiassa alun perin verkkosijainti. Googlen online ohjelmointikirjastoja --

```
<!-- External JavaScript -->
<SCRIPT language="JavaScript" SRC="js/yhdistetty.js"></SCRIPT>
<SCRIPT language="JavaScript" SRC="js/validate.js"></SCRIPT>
```

-- Niin sanotut ulkoiset JavaScript-tiedostot ja niiden sijainti. Tiedostojen sisältö käsitelty muissa kappaleissa --

```
<!-- Styles for the page -->
<LINK rel="stylesheet" href="css/yhdistetty.css">
```

-- Sivun tarvitsemat CSS-tyylitiedostot ja sijainti --

```
<!-- SLIDER STYLES -->
<LINK rel="stylesheet" href="css/aslider_temp.css">
```

-- Liukusäätimen oma tyylitiedosto --

```
<SCRIPT src="js/jquery/jquery.aslider.js"></SCRIPT>
```

-- Ja vielä liukusäätimen oma jQuery-kooditiedosto --

```
<!-- MAIN SLIDER INITIALIZATION -->
<SCRIPT>
        // DOM Ready
        $(function(){
                $('#slider').aSlider();
        });
</SCRIPT>
<!-- MAIN SLIDER INITIALIZATION --END -->
```

-- Yllä on kutsuttu pääliukusäätimen alustusfunktiota. $-merkki tarkoittaa jQuerya, suluissa on liukusäätimen uniikki ID (slider) ja funktionimi, johon konstruktori (alustaja) kohdistetaan. On muodostettu Slider-olio --

```
<!-- ANSWER SLIDERS INITIALIZATION-START -->
<SCRIPT type="text/javascript">
 $(function() {

    var seli = {
            0: " - Täysin eri mieltä",
                                    10: " - Lähes täysin eri mieltä",
            20: " - Huomattavan eri mieltä",
                                    30: " - Jonkin verran eri mieltä",
            40: " - Hyvin vähän eri mieltä",
                                    50: " - Neutraali",
            60: " - Hyvin vähän samaa mieltä",
                                    70: " - Jonkin verran eri mieltä",
            80: " - Huomattavan samaa mieltä",
                                    90: " - Lähes täysin samaa mieltä",
           100: " - Täysin samaa mieltä"
        };
```

-- Vastausliukusäätimen tekstiselvennykset on tässä syötetty muuttujaan talteen --

```
  $("#sliderj").slider({
    value:50,
    min: 0,
    max: 100,
    step: 10,
          slide: function(event, ui) {
                $("#amount1").val(ui.value + ' %' + seli[ui.value]);
    }
  });
  $("#amount1").val(seli[$("#sliderj").slider("value")]+ ' %');
 });
</SCRIPT>
```

-- Yllä on sitten ensimmäisen (tässä työssä näitä on lähes identtisinä neljä kappaletta) vastausliukusäätimen alustus. Oikeastaan ainoa ero on, että mihin tekstikenttään säätimen toiminta kohdistuu. Tähän on alustuksessa pääliukusäätimestä poiketen otettu näkyviin muutama oletusarvo. Järjestyksessä: Missä arvossa liukusäädin on alussa (50), minimiarvo (0) ja maksimiarvo (100) sekä askellusarvo (10). Asteikkohan on välillä 0-100.

Seuraavana on liu'utusefektin (slide) funktio. Tässä on, kuten aiemmassa kappaleessa tulikin jo esille, yhdistetty liu'uttamiseen säätimen arvon siirtäminen tekstikenttään (ID amount1). Sen lisäksi että siirretään säätimen arvo, niin siihen lisätään %-merkki ja tekstiselvennys (ui.value: Integer - nykyinen kahvan arvo). Koska edellä oleva on tehty liu'utuksen yhteyteen, on se myös toistettava, jotta säädin saa oletusarvonsa sivun latautuessa -> arvoa ei ole vielä liu'utettu (slide) --

```
</HEAD>

<!--BODY-START -->
<BODY>
```

```
<?php
session_start();
function fetch_question($index, $con){
```

-- Etusivun ensimmäinen PHP-funktio. Kyseessä on funktio, joka annetulla indeksinumerolla (1.parametri) hakee oikean kysymyksen vastausliukusäätimen yhteyteen. 2. parametri on turha, sillä sitä piti alkujaan käyttää tunnistamaan, että mihin paikkaan tietokannassa vastaus tulisi tallentaa. Tähän löytyi kuitenkin kätevämmäksi keinoksi JavaScript-muuttuja, johon tallennetaan pääliukusäätimen sivunumero --

```
$con = mysql_connect("localhost","vertailu","lakenet");
if (!$con){die('Could not connect: ' . mysql_error());}
mysql_select_db("o2011", $con);
```

-- Perusmallin yhteyden muodostaminen oikeaan tietokantaan ja epäonnistuneeseen liittymiseen liittyvä virheilmoitus. Lisäksi vielä oikean tietokannan valinta --

```
// ID is numbering for questions in mysql database, index is given in function call parameters
$result1 = mysql_query("SELECT kysymys FROM testikysymykset WHERE id='$index' ");
```

-- Muuttujaan (result1) tallennetaan kyselyn tulos, joka sisältää oikean kysymyksen, oikeasta taulusta. Lauseessa where-ehdolla tunnistetaan oikea kysymys funktion parametrina saamasta indeksinumerosta (käytännössä taulun rivinumero).

```
$result2 = mysql_query("SELECT id FROM testikysymykset WHERE id='$index' ");
```

-- Muuttujaan (result2) tallennetaan taas samalta riviltä rivinumero --

```
if (!$result1 || !$result2) {exit("Error in SQL");}
```

-- Jos muuttujien alustaminen ei onnistu, niin funktion suorittaminen keskeytetään ja näytetään varoitusteksti --

```
// printing table rows
while($row = mysql_fetch_row($result1)){
// $row is array - foreach() puts every element of $row to $cell variable
   foreach($row as $cell)
      echo "<FONT COLOR=White><B>$cell</B></FONT>";
  }
```

-- Yllä olevassa while-silmukassa haetaan muuttujasta result1 tulostettavia rivejä niin kauan kuin niitä löytyy. Jokainen rivi sijoitetaan muuttujaan (cell), joka tulostetaan sivulle pienen muotoilun (fontin väri valkoinen ja muoto lihavoitu) kanssa. Käytän yllä olevaa muotoa yleensä tulostamaan tauluja (useampi rivi), mutta tässä olen käyttänyt sitä vain yhden rivin tulostamiseen --

```
while($row = mysql_fetch_array( $result2 )){
                $_SESSION['pagei']= htmlspecialchars($row['0']);
        //echo "<FONT COLOR=White><B> $_SESSION[pagei] \n tarkistus=> \n";
                //echo ( htmlspecialchars($row[0]) );
                //echo "</B></FONT>";
                ?>
```

-- Muuttuja result2 sisälsi siis rivinumeron, joka pistetään istuntomuuttujaan talteen. Tätä muuttujaa käytettiin alun perin tarkastamaan, että liukusäädin antaa oikean sivunumeron (saman kuin yllä oleva funktio) --

```
      <FORM>
                        <INPUT id="paget" type="hidden" value="<?php
echo( htmlspecialchars($row['0']) ); ?>" />
      </FORM>
```

-- Välissä HTML-koodia, jossa tarkistusmuuttuja tallennetaan lomakkeen tekstikenttään. Nyt kun sivunumeron toimintaa ei enää tarkasteta, niin kenttä on piiloitettu. Voitaisiin kommentoida kokonaan pois! --

```
      <?php
   }
 mysql_close($con);
}
?>
```

-- Yllä vielä lopuksi suljetaan nyt tarpeettomaksi muuttunut tietokantaliittymä --

```
        <!-- LINKS to other pages-START -->
        <DIV id="navet">
        <A class="nolink"> TERVETULOA VERTAILUKONEESEEN! </A>
                <A href="etusivu.php">START</A>
                <A href="ohje.php">Yleinen ohje</A>
                <A href="hallinta.php">Hallinta</A>
        </DIV>
        <!-- LINKS-END -->
```

-- Yllä on uniikilla tunnistemella (ID, navet) varustettu linkkiosio. Osiossa on ihan perusmallilla sivulinkkejä, joita on sitten muotoilulla muutettu enemmän linkkipalkin muotoon. Alla linkkejä koskeva muotoilu (yhdistty.css):

```
a{
font-size:140%;
border: 0;
text-decoration: none;
}
```

-- Ensinnä on linkeille (tagi a) määritelty: fonttikoko (140%), linkin rajaus nolla (0) ja tekstin muotoilu normaali (none).

```
#nav, #navet{
display: block;
width: auto;
margin: 10px auto;
text-align: center;
white-space: nowrap;
line-height: 3em;
}
```

-- Sitten on määritelty osiolle navet seuraavaa: näyttö on block eli siis osiota koskee aika lailla samanlainen muotoilu kuin kappaleita* (<p> eli paragraph), leveys on automaattinen (selain laskee), marginaali 10 pikseliä automaattisesti (selain laskee), tekstin kohdistus keskelle (edellä olevat varmistavat linkkipalkin keskittämisen sivulla), estetään tekstin wrappaus (tiivistäminen) ja asetetaan rivin korkeudeksi 3em (em on w3c-konsortion suosittelema mitta). 1em vastaa nykyistä fontin kokoa, joten silloin esimerkiksi 2em tarkoittaa kaksi kertaa nykyisen fontin koko --

* Selaimet lisäävät automaattisesti jonkin verran tilaa (margin) ennen ja jälkeen kappale-elementin (<p>).

```
#nav a.nolink, #navet a.nolink {
/* rounded corners */
border-radius: 5px;
-moz-border-radius: 5px;  /* Firefox 4 */
-webkit-border-radius: 5px;  /* Safari and Chrome */
padding: 10px;
background: #666;
}
```

-- Tässä määritetään navet-osion niiden linkkien joiden luokka on nolink arvoja: Kulman pyöristys (yleinen ja muutamalle eri selaimelle oma määritys) yhdellä arvolla. Yksi arvo tarkoittaa, että sekä vaaka- että pystytason säteet (neljännes ellipsi) käyttävät samaa arvoa ja pyöristys on tasainen. Suomeksi siis määrittää kulman molempien päiden pyöristyksen. Lisäksi 10 pikselin ”pehmustus” (väli elementin

reunasta sen sisältöön) ja linkin tausta(väri) #666 -> harmaan sävy. Varsinaisissa linkeissähän väri muuttuu harmaaksi, kun niitä painetaan --

```
#nav a , #navet a{

/* outer shadows  */
-moz-box-shadow: -2px -2px 2px 2px #888;
-webkit-box-shadow: -2px -2px 2px 2px#888;
box-shadow: -2px -2px 2px 2px #888;

text-align: center;
background: black;
color: white;
}
```

-- Yllä on seuraavaksi määritelty navet-osiossa oleville luokattomille linkeille muun muassa tuo kulman pyöristys. Sen lisäksi on määritelty myös tekstin keskittäminen (linkin teksti), taustan väriksi musta ja tekstin väriksi valkoinen. Sen lisäksi näille varsinaisille linkeille on määritelty myös varjostus. Neljä arvoa järjestyksessä: vaakatason tasoitus (positiivinen arvo -> varjon tasoitus oikealle puolelle, negatiivinen -> vasemmalle), pystytason tasoitus (positiivinen arvo -> varjon tasoitus alapuolelle, negatiivinen -> yläpuolelle, vaihtoehtoiset sumennuksen etäisyys (mitä suurempi arvo > reunan varjostuksen sumennus) ja varjon levittämisen etäisyys. Lisäksi perässä on vielä vaihtoehtoinen väri (#888 -> harmaan sävy). Asiasanaa ei ole käytetty (inset), joten varjostus tulee oletuksena elementin ulkopuolelle, eikä sisäpuolelle --

```
#nav a:hover, #nav a.nolink:hover , #vbutton1:hover, a.nolink:hover, #vbutton2:hover, #vbutton3:hover, #navet a:hover, #navet a.nolink:hover {
background: #666;
}
```

-- Tässä onkin nyt hyvä huomauttaa, kun esimerkkejä on aimo liuta, että tätä tyylitiedostoa käytetään monella sivulla ja useammalle uniikille elementille. Tässä

onkin lueteltu aika liuta elementtejä, joille tehdään sama hover-efekti (leijumis). Eli kun hiiri kohdistuu elementtiin, sen taustaksi muutetaan väri mustasta harmaaksi --

```
<!-- SLIDER-START -->

<UL id="slider">
```

-- Tällä järjestelettömän listan aloitustagilla itse asiassa sijoitetaan pääliukusäädin sivulle --

```
<LI><?php fetch_question(1, "con1"); ?>

<P>
<DIV class="inputdiv4"><LABEL class="inputl" for="amount1">Asteikko (10% increments):</LABEL> <INPUT class="inputi" type="text" id="amount1" /></DIV>
</P>

<DIV id="sliderj"></DIV>

<BR />
<BUTTON class="talleta" onclick="javascript:copysv()">Talleta vastaus</BUTTON>
</LI>
</UL>
<!-- SLIDER-END -->
```

-- Olen ottanut dokumenttiin vain ensimmäisen listan jäsenen eli ensimmäisen vastausliukusäätimen, sillä seuraavien liittäminen dokumenttiin ei palvelisi mitään järkevää tarkoitusta. Erot koodissa ovat yksityiskohdissa, mutta ei toimintaperiaatteessa. Koodissa on määritelty listan jäseneksi (li-tagi) kysymys (tässä tapauksessa: ”**Kuinka kiinnostunut olet soveltamaan työssäsi matematiikkaa?**”), etiketti ja tekstikenttä elementille amount1, sijoitettu vastausliukusäädin osioelementillä (<DIV id="sliderj"></DIV>) sekä nappi ”Talleta vastaus”, joka

kutsuu funktiota copysv() (esitellään kappaleessa 6). Tämän jälkeen on suljettu listan jäsen ja itse lista. Tästä näemmekin hyvin, että listan jäsenet (li) ovat itse asiassa pääliukusäätimen (ul) sivuja --

```
<!-- ANSWERFORM-START -- this is where the slider values are moved as values for form fields -->

<DIV ID="answerdiv"> <FORM ID="answerf" NAME="answerf" METHOD="post" action="vertailu.php" >

<INPUT TYPE="hidden" ID="val1" NAME="val1" SIZE="3" VALUE="00">
<INPUT TYPE="hidden" ID="val2" NAME="val2" SIZE="3" VALUE="00">
<INPUT TYPE="hidden" ID="val3" NAME="val3" SIZE="3" VALUE="00">
<INPUT TYPE="hidden" ID="val4" NAME="val4" SIZE="3" VALUE="00">
<INPUT TYPE="hidden" ID="sum1" NAME="sum1" SIZE="3" VALUE="00">

<INPUT type="submit" ID="vbutton1" class="v"  VALUE="Suorita vertailu"/>
</FORM></DIV>

<!-- ANSWERFORM-END -->

<!-- BODY-END -->
</BODY>
</HTML>
```

-- Tämän sivun (etusivu.php) viimeisessä koodiosiossa on esillä piiloitettu lomake, johon säilötään käyttäjän antamat vastaukset. Lomakkeesta löytyy piilokenttä jokaiselle vastaukselle ja vastauksien summalle. Tuota summaahan päivitettiin joka kerta, kun päivitettiin jotain vastauksista (talletetaan yksittäinen vastaus). Lomakkeesta on hyvä huomata mihin tiedot ohjautuvat (vertailu.php) ja eri elementtien tunnistimet (ID), sillä seuraavaksi on taas tyylitiedostosta lomakkeeseen liittyvät koodinpätkät --

```
#answerdiv{
text-align: center;
}

#vbutton1, #vbutton2, #vbutton3{

text-align:center;
width: 400px;
font: 12px Georgia, Serif;
font-size: 140%;

/* outer shadows  */
-moz-box-shadow: -2px -2px 2px 2px #888;
-webkit-box-shadow: -2px -2px 2px 2px#888;
box-shadow: -2px -2px 2px 2px #888;

border: 0;
text-decoration: none;
background: black;
color: white;

/* rounded corners */
border-radius: 5px;
-moz-border-radius: 5px;
-webkit-border-radius: 5px;
padding: 10px;
margin: 10px;
}
```

-- Uutta selvitettävää ei näissä tyyleissä olekaan. Etusivulta pääsee linkin kautta sivulle hallinta.php, joka on oikeastaan vain navigointihaara, sillä sivulta pääsee vain takaisin (linkki START), järjestelmäkäyttäjien sivulle (linkki) ja muiden käyttäjien sivulle (linkki). Nämä linkit noudattavat samaa rakennetta kuin etusivunkin, eikä

niiden ulkonäössäkään (CSS-tyyli) ole eroa. Näin ollen en sivua käy lävitse, mutta sivusta on kyllä kuva (Opinnäytetyö).

Tämän lisäksi etusivulta pääsee vielä lisäksi vertailukoneen yleiseen ohjeeseen (ohjesivuja on kaiken kaikkiaan kolme kappaletta), jossa selvitetään koneen vertailun käyttö sekä muuta yleistä. Ohjeiden sivumalli on sama, vain ohjetekstit eroavat, sillä omat ohjeet löytyvät (kirjautumisen jälkeen) niin muille käyttäjille kuin järjestelmäkäyttäjillekin. Käsittelen ohjesivujen ulkonäön pelkästään kuvalla yhdestä ohjesivusta (ohje.php), koska HTML- tai CSS-koodi ei tarjoa mitään uutta --

Kuva 1: Yleinen ohjesivu (ohje.php)

-- Tästä on luontevaa jatkaa vertailusivulle (vertailu.php). Alla sivun koodi käsiteltynä tuttuun tapaan:

```
<?php session_start(); ?>
<?php
$val1= $_POST[val1];
$val2= $_POST[val2];
$val3= $_POST[val3];
```

```
$val4= $_POST[val4];
$sum1= $_POST[sum1];
$sum2= $sum1/4;
?>
```

-- Alkuun huomioidaan session aloituskomento, joka tarvitaan aina kun aiotaan käsitellä istuntoa. Seuraavaksi otetaan talteen etusivun lomakkeen lähettämät muuttujat ja sijoitetaan yhteenlasketun arvon muuttujalla lasku vertailua ajatellen. Muuttujathan validoitiin ennen lähetystä! --

```
<HTML>
<HEAD>
<META http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<LINK rel="stylesheet" href="css/navigointi.css">

<TITLE>Vertailu</TITLE></HEAD>
<BODY>

<!-- LINKS to other pages-START -->
        <DIV id="nav1">
                <A href="etusivu.php">START</A>
        </DIV>
<!-- LINKS-END -->

<!--IF VASTAAJAN KESKIARVO > VERTAILTAVA KESKIARVO =>
SUMUP2/VASTAUSKESKIARVOLLA -->

<?php fetch_answers($val1, $val2, $val3, $val4, $sum1, $sum2); ?>
```

-- Funktio (kutsutaan), joka noutaa kaikkien koulutusohjelmien vastaukset (tarkemmin koulutusohjelmien vastausten yhteenlasketun summan) ja saa parametreina vertailun tekijän vastaukset. Kaikkia parametreja ei lopullisessa versiossa tarvita --

```
<?php
//Function gets the summed up value of answers(%) from all tables in database
function fetch_answers($value1, $value2, $value3, $value4, $sumup1, $sumup2){

$databases= o2011_vastaukset;
$connection= @mysql_connect("localhost", "vertailu", "lakenet") or die(mysql_error());
if (!mysql_select_db($databases)) die("Can't select database");

$tables= @mysql_list_tables($databases) or die(mysql_error());

-- Yllä on funktion toteutuksen alku. Hakee oikeasta tietokannasta kaikki taulut (lista) --

$tablenumber= 0;
while($tablenumber< mysql_num_rows($tables)){

//Table names for tablelist  (1)
$tablenames[$tablenumber]= mysql_tablename($tables, $tablenumber);
//Retrieving table comments for every table for tablelist  (2)
$tabledesc[$tablenumber]= mysql_query("SHOW TABLE STATUS LIKE '$tablenames[$tablenumber]' ");
//Gets the summed up value of answers(%) from all tables in database for tablelist (3)
$tableresults[$tablenumber]= mysql_query("SELECT SUM(vastaus) FROM ($tablenames[$tablenumber])");

-- Silmukassa tallettaa joka taulusta vektorimuuttujiin taulun nimen (1), taulun kommentin (2 eli koulutusohjelman kuvaus) ja taulun vastauksien summan (3) --

//From index 0 every table sum
while ($row = mysql_fetch_array ($tableresults[$tablenumber], MYSQL_BOTH)) {
$results[$tablenumber]= $row[0];
}
}
```

-- While-silmukassa varsinainen summien tallennus muuttujaan MySQL-kyselytiedosta (query) --

```
//From index 17 (array-row) every table comment
while ($row1 = mysql_fetch_array ($tabledesc[$tablenumber], MYSQL_BOTH)) {
$results1[$tablenumber]= $row1[17];
```

-- Sama operaatio taulujen kommenteille (koulutusohjelman kuvaus) --

```
}

$tablenumber++;
}

//Calculating result (compability %) and formatting it for printing
for ($i=0; $i<$tablenumber; $i++){

//Setting the percent values to be from 0 to 100
if($results[$i]/4>$sumup2){
$resultti[$i]= round($sumup2/($results[$i]/4)*100);
$resultti[$i] .= " &#37;  ";
$resultti[$i] .= $results1[$i];
}else{
//Setting the percent values to be from 0 to 100
$resultti[$i]= round($results[$i]/4/$sumup2*100);
$resultti[$i] .= " &#37;  ";
$resultti[$i] .= $results1[$i];
}
}
```

-- Yllä olevassa for-silmukassa suoritetaan jokaisen koulutusohjelman summan ($results[$i]) vertailu kyselydataan ($sumup2). Suurempi summista on aina jakajana (if ja else). Loppu on tulostuksen formatointia, kuten summan pyöristys (round),

prosenttimerkin sekä taulun kommentin ($results1[$i]) lisäys -> yhteensopivuus + prosentti + koulutusohjelma muuttujassa $resultti[$i], huomaa eri muuttuja. $results, $results1 ja $resultti --

```
//Sorting (from 100 to 0 -- from up to down) and printing
rsort($resultti, SORT_NUMERIC);
echo "<table><tr><th>Tuloksesi (soveltuvuusprosentti tietokannan koulutusohjelmiin)</th></tr>";
for ($i=0; $i<$tablenumber; $i++){
echo "<tr><td>$resultti[$i]</td></tr>";
}
echo "</table>";
}
?>

<!-- LINKS to other pages-START -->
	<DIV id="nav2">
		<A href="etusivu.php">START</A>
	</DIV>
<!-- LINKS-END -->

</BODY>
</HTML>
```

-- Tämän lisäksi vielä tulokset lajitellaan (rsort, numeerinen) suurimmasta pienempään ja tulostetaan tauluun tässä järjestyksessä. Tulostus tauluun on tehty tyyliseikkojen (CSS) vuoksi. Koska loppu on tuttua (listauksen perään myös linkki etusivulle), niin voidaankin käsitellä CSS-puoli. Otan tiedostosta (navigointi.css) otteen uudesta eli taulun muotoilusta:

```
table{
border: 1;
border-spacing: 40px;
background-color:black;
```

```
width:100%;
}
```

-- Taulussa on yhden pikselin paksuinen reunus, 40 pikseliä väliä solujen reunuksien välillä, taulun taustaväri on musta ja leveys koko ruudun mittainen --

```
th{
height:200px;
font-size:40px;
background-color:black;
color:white;
}
```

-- Taulun otsikkosolut ovat korkeudeltaan 200 pikseliä, fontin koko on 40 pikseliä, taustaväri musta ja tekstin väri valkoinen --

```
td{
font-size:28px;
background-color:black;
color:white;
text-align: center;
}
```

-- Taulun sisältösoluissa ei ole sen kummempaa selvennettävää, kuin että teksti keskitetään solussa --

Seuraavaksi tarvitseekin sitten edetä hallintasivusta eteenpäin. Aloitetaan valitsemalla käyttäjäksi (vaihtoehdot: paluu, muut käyttäjät ja järjestelmäkäyttäjät eikä muita) muu käyttäjä. Tästä valinnasta siirrytäänkin sitten kirjautumissivulle, joka muilla käyttäjillä on in2.html (sivu ei sisällä lainkaan PHP-koodia). Sivu on muodostettu siten, että HTML-puoli tarjoaa käyttäjälle kirjautumislomaketta, johon täytetään aivan normaalisti käyttäjätunnus (sama kuin koulutusohjelman taulun nimi) ja salasana. Lomakkeella ei kuitenkaan ole lähetystoiminnolle määritetty kohdetta (action=""), vaan jQuery-funktio nappaa lomakkeen lähetyksestä (submit) kopin -> $

("#login_form2").submit(function(){ …ja niin edelleen. Alla on taas koodi selvitettynä:

```
<SCRIPT LANGUAGE="JavaScript">
// jQuery -- Everything inside function will load as soon as the DOM is loaded but before the page contents are loaded.
$(document).ready(function(){
//When the form is submitted, code in function is run. This happens prior to the actual submission, so we can cancel the submit action if needed.
	$("#login_form2").submit(function(){
		//Remove all the classes, add the messagebox classes and start fading login update text
		$
("#msgbox2").removeClass().addClass('messagebox2').text('Validating....').fadeIn(10
00);
```

-- Funktion sisältö latautuu, kun dokumentin objektimalli on latautunut, mutta ennen sivun sisällön latautumista. Kun lomake lähetetään, funktio ajetaan (ennen varsinaista tietojen lähettämistä, joten lähetys voidaan vielä perua). Lomakkeen lähetysfunktiossa aivan ensimmäiseksi tartutaan elementtiin (tässä tapauksessa lomakkeen perään lisättyyn span-osioon) msgbox2 kiinni ja varmistetaan, ettei se sisällä luokitusta, jonka jälkeen lisätään sille luokka messagebox2, lisätään sille tekstiä (”Validating....”) ja käytämme valmista funktiota fadeIn tuomaan toimintaan animaatiota. Funktio tekee 1000 millisekuntissa vahvasti läpikuultavasta tulostekstistä läpikuultamattoman. Käyttäjälle siis animoituu lomakkeen hyväksymisen yhteydessä teksti ”Validating....” --

```
		//Request success handler -- Check the username existance and password matching from database -- use in function
$.post("ajax_login2.php",{user_name:$('#username2').val(),password:$
('#password2').val(),rand:Math.random() } ,function(data){
```

-- Eli kuten tuossa jo englanniksikin sanotaan, niin jQueryn post-funktiolla lähetetään käyttäjätunnus (user_name ja password) ja salasana (sekä satunnainen numero,

nimettynä rand) tarkistettavaksi toiseen tiedostoon (ajax_login2.php), joka hoitaa tietokantaosion. Satunnaisnumeroa (nimetty rand) nollan ja yhden välillä käytetään, jotta jokainen pyyntö olisi aina erilainen -> ei saada palvelimelta vastaukseksi välimuistiin tallennettua vastausta. Takaisin saatavaa tietoa käytetään funktiossa --

```
                if(data=='yes') //If correct login detail (username and password)
                {
        $("#msgbox2").fadeTo(200,0.1,function()  //start fading the messagebox
                        {
         //Add login message, change the class of the box (login ok) and start fading

 $(this).html('Logging in').addClass('messageboxok').fadeTo(900,1,
          function()
                        {
                                //Redirect to secure page
                                document.location='updatet.php';
                        });

                        });
                }
                else //Wrong username, password or combination of these two{
$("#msgbox2").fadeTo(200,0.1,function() //start fading the messagebox
                        {
 //Add message, change the class of the box (login not ok) and start fading

$(this).html('Not right login data').addClass('messageboxerror').fadeTo(900,1);
                        });
        }

      });
                return false; //Not to post the form physically
        });
        //Now call ajax, also focus move from
        $("#password2").blur(function()
```

```
        {
                $("#login_form2").trigger('submit2');
        });
});
</SCRIPT>
```

-- Yllä oleva if-else rakennelma perustuu siihen, että datan tarkistus (ajax_login2.php) palauttaa joko arvon ”yes” -> oikea tunnus ja salasana tai arvon ”no”, jolloin molemmat tai jompikumpi tiedoista on väärin. Jos tiedot ovat oikein tehdään taas viestin häivytys (animaatio, fade) ja lisätään luokka ”messageboxok” HTML-elementtiin. Luokka lisätään tähän elementtiin, koska sen kautta se lisätään sivuun. Häivytykseen käytetään fadeTo-funktiota, jotta saadaan kätevästi ajettua heti funktion perään, joka ohjaa onnistuneen kirjautumisen mukaiselle sivulle --

-- Else vaihtoehto on siis kuten sanottua väärien tietojen vaihtoehto. Siellä tehdään muuten samanlaiset viestinnät, mutta käyttäjää ei ohjata uudelle sivulle. Ilmoitetaan (häivytys), että tiedoissa oli vikaa (”Not right login data”), sivulle lisätään muotoilua varten tällä kertaa luokka ”messageboxerror”. Lisäksi palautetaan vielä false (epätosi), joka estää että lomaketietoja yritettäisiin lähettää osoitteeseen, jota ei ole missään vaiheessa kerrottu eikä tarvittu --

Välissä kuva kirjautumisen viestilaatikoista:

Kuva 2: Kolme eri viestilaatikkoa kesken animoinnin

Sitten on tietenkin vielä hyvä käydä tarkemmin lävitse, että miten tapahtuu kirjautumistietojen tarkistus (ajax_login2.php):

```
<?php session_start();

//Connect to database from here
$link = mysql_connect('localhost', 'muutothers', 'tenekal');
```

```
if (!$link) {
   die('Could not connect: ' . mysql_error());
}
//select the database
mysql_select_db('o2011_clauditis');
```

-- Ensinnä on ihan perusjutut. Luodaan yhteys tietokantaan ja valitaan oikea tietokanta --

```
//get/set the posted values
$user_name=htmlspecialchars($_POST['user_name'],ENT_QUOTES);
$pass=md5($_POST['password']);
```

-- Seuraavaksi otetaan talteen lähetetyt arvot. Htmlspecialchars-funktio muuttaa erikoismerkit HTML-vastineikseen ja salasanasta otetaan MD5-muunnos, jotta se voisi ylipäätään vastata tietokanta-arvoa, joka on jo MD5-algoritmilla käsitelty --

```
//Fetch matching user by comparing user names
$sql="SELECT user_name, password FROM sepem WHERE (user_name)='". $user_name."'";
$result=mysql_query($sql);
$row=mysql_fetch_array($result);
```

-- Tehdään haku tietokannasta vertaamalla saatua ja tietokannassa olevia käyttäjänimiä, tulos palauttaa (jos käyttäjä löytyy) sekä tunnuksen että salasanan muuttujaan $sql, josta tehdään MySQL-kysely -> $result ja tästä vielä haetaan rivi -> $row --

```
//if username exists
if(mysql_num_rows($result)>0){
        //compare the password
        if(strcmp($row['password'],$pass)==0)
        {
                echo "yes";
```

```
		//now set the session from here if needed
		$_SESSION['u_name2']=$user_name;
	}
	else
		echo "no";
}
else
	echo "no"; //Invalid Login

mysql_close($link);
?>
```

-- Jos on saatu yhtään tuloksia (niitä saadaan joko nolla tai yksi), verrataan salasanatietoja toisiinsa (strcmp()). Funktio palauttaa nollan, jos vertailtavat merkkijonot vastaavat toisiaan ($row['password'] ja $pass). Jos vastaavuus löytyy, niin palautetaan aiemmin jo mainittu vastaus ”yes” ja asetetaan käyttäjänimi istuntomuuttujaan. Jos vastaavuutta vertailussa ei löydy, palautetaan vastaus ”no”. Vastaus ”no” palautetaan myös jos MySQL-kysely ei palauttanut mitään -> ei löytynyt käyttäjätunnusta tietokannasta. Suljetaan yhteys --

Lopuksi on vielä hyvä ottaa viestilaatikoiden tyylikoodi esille (in.css):

```
#msgbox1, #msgbox2{
display:none;
}
```

-- Viestilaatikot (msgbox1 on järjestelmäkäyttäjien puolelta) ovat aluksi poissa sivulta (näyttö -> ei) --

```
.messagebox{
	position:absolute;
	width:100px;
	margin-left:30px;
	border:1px solid #c93;
```

```
        background:#ffc;
        padding:3px;
}

.messageboxok{
        position:absolute;
        width:auto;
        margin-left:30px;
        border:1px solid #349534;
        background:#C9FFCA;
        padding:3px;
        font-weight:bold;
        color:#008000;

}

.messageboxerror{
        position:absolute;
        width:auto;
        margin-left:30px;
        border:1px solid #CC0000;
        background:#F7CBCA;
        padding:3px;
        font-weight:bold;
        color:#CC0000;
}
```

Elementit, joiden muotoilussa sijainti on absoluuttinen eivät huomioi muita elementtejä, vaan voivat jopa tulla muiden elementtien päälle. Automaattinen leveys jättää elementin leveyden laskemisen selaimen vastuulle. Viestilaatikoiden näkyvyyttä ja tyyliähän muokataan jQuery-funktioiden kautta.

Koska kyseessä on ensimmäinen sivu, jossa tulee vastaan työkaluvihje, niin sen HTML- ja CSS-koodi esitellään vielä tältä sivulta:

```
<!-- Script for tooltip -- Show/Hide tooltip on button click and change button text -->
<SCRIPT type="text/javascript">
$(document).ready(function(){
  $("button").click(function(){
        if ($(this).text() == "SHOW TIP") { $(this).text("HIDE TIP"); } else { $
(this).text("SHOW TIP"); };
    $("p.hs").toggle();
  });
});
</SCRIPT>
</HEAD>
```

-- Yllä on sivun HEAD-osiosta napattu työkaluvihjeen skripti eli varsinainen toimintakoodi. Siinä kohdistetaan funktion toimet napin painallukseen (sivulla vain yksi nappi eli button). Jos sivulla on useampia nappeja (kuten minulla onkin toisella sivulla), tarvitsee oikeaan nappiin osoittaa HTML-tunnuksella/luokalla (ID tai class). Riippuen siitä kumpi on napin tekstisisältö (lukee napissa sivulla), niin napin teksti vaihdetaan vastakkaiseksi ja kappaleisiin, joiden luokka on ”hs” kohdistetaan toggle-funktio. Ilman parametreja tämä jQuery-funktio vain vaihtaa elementin näkyvyysasetuksen vastakkaiseksi (visibility: visible tai hidden) --

```
… …
<!-- The text for tooltip -->
<DIV id="intool"><BUTTON class="tip" id="toggle2">SHOW
TIP</BUTTON></DIV>
<DIV id="tooltip2">
<P class="hs">Olet kirjautumassa sisään syöttämään tiedot järjestelmään joko
ensimmäistä kertaa tai päivittämään tietoja.</P>
<P class="hs">Molemmat toimivat samalla tavalla. Voit täyttää/päivittää vain
käyttäjänimesi mukaista taulua.</P>
</DIV>
</BODY></HTML>
```

-- Tässä onkin sitten itse työkaluvihjeosio (napin paljastama/piiloittama teksti) sivulta. Sen anti jää lähinnä havainnollistaa mihinkä skriptin toimet kohdistuvat --

Työssä on tarkalleen kaksi erillistä kirjautumisjärjestelmää, muille käyttäjille ja järjestelmäkäyttäjille omansa. Järjestelmäkäyttäjillä on oma salasanataulu sekä omat sivut ja tiedostot, jotka hoitavat kirjautumisen. Etenkään kirjautumisessa ei ole mitään rakenteellista eroa, mutta itsestäni vain tuntui hyvältä erottaa järjestelmäkäyttäjät muista erilleen. Mitään erityistä tietoturvatietoa päätöksen takana ei tässä tapauksessa ole.

Väärin kirjautumalla käyttäjä saa pelkästään ilmoituksen (messageboxerror), että ”not right login data”, mutta mitä sitten kun syöttää oikeat tiedot? Käyttäjä ohjataan muiden käyttäjien pääsivulle (document.location='updatet.php';). Sivu on pohjaltaan (oikeastaan malli, template) hyvin samanlainen kuin etusivu, koska muut käyttäjät voivat vain päivittää vastauksiaan tai suorittaa vastaamisen ensimmäisen kerran. Nämä molemmat onnistuvat vielä täysin samalla sivulla (ero hoidetaan PHP-koodilla), kokemus on käyttäjälle aivan sama. Sivulla on samanlainen linkkipalkki, pääliukusäädin, vastausliukusäätimet ja samat kysymyksetkin kuin etusivulla. Eroa löytyy lähinnä vastausdatan käsittelystä.

Sivulla on linkkipalkissa muiden käyttäjien oma ohje, jonka ainoa ero yleiseen ohjeeseen, on ohjetekstikappaleiden sisältö. Toinen ero on tietenkin ainoa linkki, jolla pääsee sivulta pois, ”Kirjaudu ulos”. Tämä linkki ohjaa käyttäjän sivulle logout2.php, joka ei näy käyttäjälle. Käydään koodin avulla toiminta lävitse:

```
<?PHP session_start();
session_destroy();
header("Location:in2.html");
//Destroys session for user logging out and redirects to start page
?>
```

-- Istunto tarvitsee aina niin sanotusti startata, kun sen tietoja muokataan. Niin myös siinä tapauksessa, että seuraavaksi istunto tuhotaan (destroy). Tuhoaminen hävittää kaiken istuntoon liitetyn datan. Tämä on tarpeellista muun muassa jo senkin vuoksi,

että käyttäjä voi kirjautua ulos. Jos istunto muuttujaa ei uloskirjautumisessa tuhottaisi, niin käyttäjä voisi ”uloskirjautuneena” palata istuntoonsa, koska käyttäjän istuntoa ei oikeasti olekaan tuhottu. Lopuksi käyttäjä ohjataan muiden käyttäjien kirjautumissivulle. Tein uloskirjautumisen ensinnä pääsivulla (etusivu.php), mutta havaitsin että selaimet eivät toimi tällaisen kanssa yhteen, vaan muistavat kumminkin istuntomuuttujia.

Käyn vielä lävitse sen miten muiden käyttäjien eli koulutusohjelman vastaukset tallennetaan, sillä se on ainoa merkittävä ero toimintaan verrattaessa sovelluksen etusivuun. Alla relevantti koodi :

```
<?php session_start();
// if session is not set redirect the user
if(empty($_SESSION['u_name2']))
        header("Location:in2.html");
?>
```

-- Tietenkin, jos käyttäjälle ei ole kirjautumisen yhteydessä saatavaa istuntomuuttujaa, kaikki sivupyynnöt ohjataan sisäänkirjautumiseen (in2.html) --

```
<?php    -- Edelleen siis updatet.php --
session_start();

$db_host = 'localhost';
$db_schema = 'o2011_vastaukset';
$db_user = 'muutothers';
$db_pwd = 'tenekal';
```

-- Muilla käyttäjillä oma käyttäjäryhmä ja salasana! Nämä on tietenkin myös määritetty MySQL-komentotulkilla palvelimelle, sillä eiväthän ne muuten toimisikaan oikein --

```
$connection= @mysql_connect($db_host, $db_user, $db_pwd) or die
(mysql_error());
```

```
if (!mysql_select_db($db_schema)) die("Can't select database");

// Select 1 from table_name will return false if the table does not exist.
$val = mysql_query("SELECT 1 FROM $_SESSION[u_name2]");

if($val !== FALSE){

// run the query and put the results in an array variable called $result
$result = mysql_query("SELECT * FROM $_SESSION[u_name2] ORDER BY id");
```

-- Tässä haetaan kaikki tieto taulusta, jonka nimi on siis sama kuin käyttäjänimi (koulutusohjelman taulun nimi = $_SESSION[u_name2]). Näinkin varmistetaan, että käyttäjä saa nähtyä ja päivitettyä vain itselleen tarkoitettua dataa. Lisäksi SELECT 1 lauseella varmistetaan, että taulu on käyttäjällä olemassa. Jos ei ole annetaan varoitus (ELSE-osiossa) --

```
// set a counter in order to number the input fields for each record
$i = 0;

// open a form
print "<form name='namestoupdate2' method='post' action='updated2.php'>\n";
```

-- Tässä PHP-kielen kautta tulostetaan HTML-dataa. Aloitetaan lomake, johon käyttäjän vastaukset kerätään (Talleta vastaus nappi + copysv-funktiot, yhdistetty.js). Mikä käyttäjälle ei näy (piilokentät) on se, että aikaisemmat tiedot tulostetaan lomakkeeseen eli vastausten rivinumerot (ID) ja vastaukset (vastaus) --

```
// start a loop to print all
// the mysql_fetch_array function puts each record into an array. Each time it is called, it moves the array counter up until there are no more records left
while ($answers = mysql_fetch_array($result)) {
```

-- Eli while silmukassa jokainen tietue tallennetaan vektorimuuttujaan. Silmukka jatkuu kunnes vektorimuuttuja ei enää kasva, tietueet ovat loppuneet --

```
// assuming you have two important columns the index (id), vastaukset (two digit number)
// start displaying the info; the most important part is to make the name for array as they are POSTed onwards
//print "$answers[id]";
   print "<p><input class='inputi' type='hidden' size='3' ID='id[$i]' name='id[$i]' value='{$answers[id]}' />";
        print "<input class='inputi' type='hidden' size='3' ID='vastaus[$i]' name='vastaus[$i]' value='{$answers['vastaus']}' /></p>\n";
```

-- Eli yllä lomakkeeseen tulostetaan jokainen vastaus järjestysnumerolla (pääliukusäätimen sivunumeroa käytetään tallentamisessa), josta tiedetään sitten laittaa vastukset myöhemmin oikeaan kohtaan taulussa --

```
// add 1 to the count, close the loop, close the form, and the mysql connection
++$i;
}
print "<DIV ID='vbdiv'><input class='inputi' type='submit' ID='vbutton3' value='Täydennä taulu -- $_SESSION[u_name2]' />";
print "</form>";
mysql_close();

}else{
echo "<script type=\"text/javascript\"> alert('Käyttäjälle ei ole luotu taulua!'); </script>";
}

?>
<!-- BODY-END -->
</BODY></HTML>
```

-- Viimeisenä tulee vielä nappi, jolla hyväksytään tietojen lähettäminen eteenpäin. Lomakkeeseen tiedot hyväksytään järjestyskohtaisesti kuten etusivullakin eli kysymyskohtaisella tallenna vastaus napilla, joka on vastausliukusäätimen alapuolella --

-- Vastaamissivulla (updatet.php) ei siis tehdä muuta kuin kerätään syötedata dynaamiseen lomakkeeseen ja lähetetään eteenpäin (Täydennä taulu -- $_SESSION[u_name2]) sivulle updated2.php (ainoa uusi asia on funktio, joka hoitaa tietojen tallennuksen):

```
<BODY>
<?php
$db_host = 'localhost';
$db_schema = 'o2011_vastaukset';
$db_user = 'muutothers';
$db_pwd = 'tenekal';

$connection= @mysql_connect($db_host, $db_user, $db_pwd) or die (mysql_error());

if (!mysql_select_db($db_schema)) die("Can't select database");

// find out how many records there are to update (ID indexing number)
$size = count($_POST['id']);
```

-- Lasketaan count-funktiolla miten monta ”riviä” on (ID oli oikeastaan tietokannassa rivinumero) --

```
// start a loop in order to update each record
$i = 0;
while ($i < $size) {
// define each variable
$id= $_POST['id'][$i];
$vastaus = $_POST['vastaus'][$i];
```

```
// do the update and print out some info just to provide some visual feedback
$query = "UPDATE $_SESSION[u_name2] SET vastaus = '$vastaus' WHERE id =
'$id' LIMIT 1";
mysql_query($query) or die ("Error in query: $query");
print "<p class='o'>Vastaus $id Updated -- With value $vastaus</p><BR />";
++$i;
}
mysql_close();
?>
```

-- While silmukassa tehdään niin monta kierrosta, kunnes indeksinumero muuttujassa i saa saman arvon kuin mitä count-funktio antoi rivien määräksi. Silmukassa sitten napataan vektorimuuttujiin vastaukset järjestyksessä talteen. Sen jälkeen tehdään taulun päivitys ja tulostetaan vastaukset myös käyttäjälle näkyviin, jotta tämä tietää operaation onnistuneen ja näkee vielä antamansa vastaukset (Vastaus $id Updated -- With value $vastaus). Validointihan tulee sitä kautta, että tekstikenttiin data syötetään liukusäätimen arvona (int) --

```
<P class="o">New data updated</P>
<DIV class="nbdiv"><A class="nolink" href="updatet.php">Muut</A>
<A class="nolink" href="logout2.php">Kirjaudu ulos</A></DIV>

</BODY></HTML>
```

Nyt jäljellä sivujen läpikäynnistä on vielä sitten järjestelmäkäyttäjien sivut. Järjestelmäkäyttäjien pääsivulla (ctable.php) on tuttu rakenne ja tyyli. Ylimpänä on linkkipalkki, jossa on käyttäjien oma ohje ja uloskirjautuminen, joissa ei niissäkään rakenteellisia eroja ole. Suunnitteluvaiheessahan totesin, että ainakin tässä sovelluksen versiossa tarjotaan järjestelmäkäyttäjälle toiminnot taulun tuhoaminen ja uuden taulun luominen. Edellä mainitun lisäksi sivulla on myös työkaluvihje, jossa on avut taulun luomiseen ja nimeämiseen. Työkaluvihjehän oli näkymätön, kunnes painetaan työkaluvihjeen nappia, jolloin ohjeteksti tulee näkyviin ja napin teksti

muuttuu (Show tip <-> Hide tip). Aloitetaan yksinkertaisemmalla eli tuhoamisella (ctable.php ja delete.php):

-- Pääsivun relevantti koodi --

```
<?php
$db_host = 'localhost';
$db_schema = 'o2011_vastaukset';
$db_user = 'root';
$db_pwd = 'haikuko';

-- Huomaa järjestelmäkäyttäjien eri tunnukset --

$connection= @mysql_connect($db_host, $db_user, $db_pwd) or die (mysql_error());
$db = mysql_select_db($db_schema) or die ("Database not responding");

$sql = "SHOW TABLES";     -- kyselylauseen tallennus muuttujaan --
$result = mysql_query($sql);  -- yksinkertainen näytä kaikki taulut kysely --

//Echo submit form for deleting tables -- dropdown box lists tables -- submit button forwards to delete.php
echo '<form method="post" id="try" action="delete.php">';
echo '<B id="dbtaulu">Taulut:</B>';
echo '<select class="styled-select" name="batch" id="batch">';
        while($r = mysql_fetch_assoc($result)) {
                $tables = $r['Tables_in_o2011_vastaukset'];
                echo '<option>'.$tables.'</option>';
        }
echo '</select>';
echo '<input class="submit2" type="submit" value="Tuhoa valittu taulu"/>';
echo '</form>';
?>
```

-- Yllä on siis tulostettu taas lomake, mutta tähän lomakkeeseen luodaankin pudotusvalikko (luokalla styled-select, tunnuksella batch), johon napataan silmukassa funktiolla mysql_fetch_assoc assositiivinen vektori (sisältää avain/arvo pareja) eli taulut siten listattuna, että niiden sijainti tietokannassa tunnetaan. Pudotusvalikossa näkyvät siis tietokannan taulut eli tässä tapauksessa koulutusohjelmien taulunimet. Näistä voidaan valita aktiivinen ja napin (luokalla submit2, tyyppi submit-nappi) painalluksella lähetetään tieto valitusta taulusta eteenpäin sivulle delete.php:

```
<?php
session_start();
$table= $_POST['batch'];  -- Huomaa tiedon taltuunotto muuttujaan $table! --
?>

<HTML><HEAD>
<META http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<LINK rel="stylesheet" href="css/hallinta.css">
<TITLE>Dropping MySQL Tables</TITLE></HEAD>
<BODY>

<?php
$db_host = 'localhost';
$db_schema = 'o2011_vastaukset';
$db_user = 'root';
$db_pwd = 'haikuko';
$con= @mysql_connect($db_host, $db_user, $db_pwd) or die (mysql_error());
if (!mysql_select_db($db_schema)) die("Can't select database");

$sql = "DROP TABLE $table";  -- Kyselylause, jolla taulu (muuttujassa $table) tuhotaan --

//retval for checking if delete was succesful
$retval = mysql_query( $sql, $con );
```

-- Tarkistusmuuttujaan ($retval) tallentuu kyselyn tulos. Jos mitään ei tallennu, tiedetään siitä ettei taulun tuhoaminen jostain syystä onnistunut --

```
if(! $retval ){
  die('Could not delete table: ' . mysql_error());
}

echo "<DIV id='pdelete'><P class='o'>$table -- Deleted successfully\n</P></DIV>";
mysql_close($con);
?>
```

-- Sitten vielä lopuksi ilmoitetaan käyttäjälle, jos tuhoaminen onnistui ja mikä taulu tuhottiin ($table -- Deleted successfully) --

```
<!-- Links to other pages -->
<DIV id="navde"><A href="ctable.php">PALUU</A>
<A href="logout.php">Kirjaudu ulos</A></DIV>

</BODY>
</HTML>
```

Sitten palataan takaisin järjestelmäkäyttäjien pääsivulle, jotta käydään vielä lävitse taulun luomisen prosessi. Ensimmäinen vaihe taulun luomisessa:

```
<H1>Vaihe 1: Taulun tietokantanimi</H1>

<!-- POST user input -- table name -->
<FORM method="post" name="ctform" action="">

<P class="visible">Taulun nimi: <INPUT CLASS="box" TYPE="text" id="table_name" NAME="table_name" VALUE="" SIZE=7>

  <INPUT class="submit1" TYPE="submit" NAME="submit" onclick="return CTclickctable();" VALUE=" Siirry seuraavaan vaiheeseen "></P>
```

```
</FORM>
```

-- Tässä käytetään samaa menetelmää kuin aiemmin. Lomaketieto lähetetäänkin itse asiassa validointifunktioon (löytyvät kappaleesta 7), josta jos ja kun se pitää paikkansa se sallitaan eteenpäin oikealle sivulle (do_s.php). Tällä kertaa katsotaan ettei syöte ole tyhjä, se on sallituissa pituusrajoissa ja sitten vielä yhdessä regex-vertailussa (säännöllinen lauseke), että se sisältää vain kirjaimia (ei saa olla edes välejä). Tietenkin epäonnistumisista annetaan havainnollinen JavaScript varoitus --

Syöte oikein -> tiedostolle (do_s.php), jonka merkitsevä (uusi) koodi:

```
<?php session_start();
// if session is not set redirect the user
if(empty($_SESSION['u_name']))
        header("Location:in.html");

//Validation of submitted data
if ( (!$_POST[table_name]) ){
        header("Location: ctable.php");
exit;
}else{$table_name= $_POST[table_name];
$_SESSION[table_name]= $_POST[table_name];
}
?><HTML>  --Havainnollistamisen vuoksi yllä oleva--
… …
<BODY>

<?php
$db_host = 'localhost';
$db_schema = 'o2011_vastaukset';
$db_user = 'root';
$db_pwd = 'haikuko';

//Create table by cloning table o2011_vastaukset.mallipohja
```

```
$sql= "CREATE TABLE $table_name LIKE mallipohja";
//Insert default values to the ready table
$sql2= "INSERT INTO $table_name VALUES (1, 00) ";
$sql3= "INSERT INTO $table_name VALUES (2, 00) ";
$sql4= "INSERT INTO $table_name VALUES (3, 00) ";
$sql5= "INSERT INTO $table_name VALUES (4, 00) ";

-- Yllä uuden taulun rakenteen luominen kloonaamalla (kappale 5.2) Sen lisäksi (kun
kysymysten määrä tunnetaan) taulun alustus oikealla määrällä default-arvoja --

$con_c= @mysql_connect($db_host, $db_user, $db_pwd) or die (mysql_error());
if ($con_c){ $msg= "<P class='o'>Connection to database - YES</P>"; }
if (!mysql_select_db($db_schema)) die("Can't select database");

//If in future more questions/answers, do a loop!
$result= @mysql_query($sql, $con_c) or die (mysql_error());
$result2= @mysql_query($sql2, $con_c) or die (mysql_error());
$result3= @mysql_query($sql3, $con_c) or die (mysql_error());
$result4= @mysql_query($sql4, $con_c) or die (mysql_error());
$result5= @mysql_query($sql5, $con_c) or die (mysql_error());

-- Default-arvojen syöttö --

if ($result){ $msg2= "<P class='o'>Table named - $table_name</P>"; }?>

-- Jos taulun kloonaaminen onnistui -> sopiva tuloste (Table named - $table_name) --

</DIV id="dosl"><A href="logout.php">Kirjaudu ulos</A></DIV>

<?php //echo "$msg"; ?>  -- Connection to database – YES --
<?php //echo "$msg2"; ?> -- Table named - $table_name --

<H1>Vaihe 2: Taulun <?php print "$_SESSION[table_name] "; ?> kuvaus:</H1>
<BR />
```

```
<!-- POST table name again with table description -->
<FORM Name="Formdos" method="post" >

<DIV id="submitbd"><INPUT type="text" id="tdesc" name="table_d" size=40 value=""></P>
<P><INPUT class="buttons" id="submitb" type="submit" name="submitb" onclick="return OBclickdos();" value="Siirry vaiheeseen 3"></P></DIV></FORM>
```

-- Tulostetaan luodun taulun nimi ja pyydetään taululle kuvaus (”Vaihe 2: Taulun <? php print "$_SESSION[table_name] "; ?> kuvaus:”). Työkaluvihje kertoo, että tähän toivotaan siis varsinainen koulutusohjelman ”nimi”. Esimerkki: Rauma - Tietotekniikan koulutusohjelma. Vanhan kaavan mukaisesti uusi syöte (taulun kuvaus) lähetetään validointifunktiolle, joka päättää jatkosta --

Tarkistetaan tutut tyhjä syöte, väärän mittainen syöte ja että tämä syöte saa sisältää kirjaimia (myös isoja ja ääkkösia), numeroita, miinusmerkkejä ja välejä (regex -> /^[0-9a-zA-ZåäöÅÄÖ\s\-]+$/). Syöte on ok, joten se ohjataan seuraavaan vaiheeseen (do_s2.php). Nyt on taas sitten kyseessä sivu, joka on hyvin etusivun kaltainen. Löytyy pää- ja vastausliukusäätimet, tutut kysymykset, käyttäjän tarkistus ja tiedon vastaanotto sekä tallentaminen istuntoon. Alla on funktio, jolla oikeaan tauluun ($_SESSION[table_name]) syötetään oikea kuvaus ($_SESSION[table_d2]=$_POST[table_d]):

```
<?php
$db_host = 'localhost';
$db_schema = 'o2011_vastaukset';
$db_user = 'root';
$db_pwd = 'haikuko';

//Adding comment to table -- comment is the actual data for table, tablename only used as username and operations parameter
$sql= "ALTER TABLE $_SESSION[table_name] COMMENT = '$_SESSION[table_d2]' ";
```

-- Kyselylause, jolla kommentin lisääminen (taulun muuttaminen, ALTER) tehdään. Taulun oikea nimi on jo käyttäjällä istuntomuuttujissa ja sivun alussa istuntomuuttujaan tallennettiin myös vastaanotettu taulun kuvaus --

```
$connection= @mysql_connect($db_host, $db_user, $db_pwd) or die (mysql_error());

if ($connection){ $msg= "connection to database - YES"; }

if (!mysql_select_db($db_schema)) die("Can't select database");

$result= @mysql_query($sql, $connection) or die (mysql_error());

if ($result){ $msg2= "<P>Taulu kommentoitu</P>"; }

?> --Lopussa ei olekaan mitään uutta tai ihmeellistä –
```

Sivulla tehdään pääperiaatteessa sama kuin muiden käyttäjien kohdalla, eli vastausliukusäätimillä valitaan ja napin painalluksella tallennetaan vastaukset. Tämähän on järjestelmäkäyttäjän kohdalla täysin vapaaehtoista. Jos vastausdataa ei ole etukäteen saatu koulutusohjelmalta, niin suoraan napauttamalla Täydennä taulu nappia jää alustusarvot voimaan ja taulun lopullinen luonti tapahtuu ketjun viimeisessä vaiheessa (update.php). Funktio on samanlainen kuin muilla käyttäjillä, tulostetaan dynaaminen lomake, josta tiedot sitten napin painalluksella siirtyvät eteenpäin. Muutos on vain muuttujanimissä ja kohdetiedostossa.

Näytän vielä update.php tiedostosta funktion, jolla tallennus tapahtuu. Onhan taas itse sivu kuitenkin vastaava kuin updated2.php eli muiden käyttäjien vastaava ja jo selostettu sivu. Tiedot tallentuvat ja sen jälkeen annetaan vielä tuloste käyttäjälle, että data on syötetty (Esimerkki: Vastaus 1 Updated -- With value 40). Alla funktion merkitsevä kysely:

```
// do the update and print out some info just to provide some visual feedback
```

```
$query = "UPDATE $_SESSION[table_name] SET vastaus = '$vastaus' WHERE id = '$id' LIMIT 1";
mysql_query($query) or die ("Error in query: $query");
```

-- Kyselylauseessa varmuuden vuoksi rajoitetaan lause koskemaan korkeintaan yhtä riviä. Jos tietokannassa on jotain vikaa, se ei ainakaan sotkeennu vielä mahdollisesti paljon pahemmin --

Siinä oli käsiteltynä kaikki sivut. Useimmista sivuista poistettiin toistuvia tai samalla lailla tehtyjä osioita ja pyrittiin tekstillä selventämään sivun toiminta kokonaisuudessaan. Tällaisia poistettuja kohtia ovat esimerkiksi ulkoisten funktiotiedostojen ja kirjastojen linkitys, yleiset istuntokäskyt alussa, perus sivumuotoilu jne.

EOF