STADIA
HELSINGIN AMMATTIKORKEAKOULU

TEKNIIKAN JA LIIKENTEEN TOIMIALA

Tietotekniikka

Ohjelmistotekniikka

INSINÖÖRITYÖ

VERKKOLEHDEN JULKAISUJÄRJESTELMÄN KEHITTÄMINEN

Työn tekijä: Mikko Tuomela
Työn valvoja: Ilpo Kuivanen
Työn ohjaaja: Olli Stålström

Työ hyväksytty: __. __. 2008

Ilpo Kuivanen
lehtori

ALKULAUSE

Tämä insinöörityö tehtiin vapaaehtoisvoimin ylläpidettävälle verkkolehti FinnQueerille, jonka toimituskuntaan olen kuulunut vuodesta 1999. Kiitän vastaavaa toimittajaa, yht. tri Olli Stålströmiä tämän projektin aikana saamastani asiantuntevasta ja kannustavasta ohjauksesta.

Helsingin ammattikorkeakoulu Stadiassa työtä valvoi lehtori Ilpo Kuivanen, jolle osoitan kiitokseni asiallisista ja rakentavista ajatuksista työn sisällön suhteen.

Työpaikkani Itella Information Oy:n Kai-Jussi Ruuskaselle osoitan kiitokseni tämän projektin mahdollistamisesta. Fishpool Creations Oy:n ystävällistä henkilökuntaa kiitän kiinnostuksesta tätä projektia kohtaan sekä avusta projektin alkuvaiheessa.

Erityisesti haluan kertoa olevani kiitollinen kaikille niille lukuisille henkilöille, jotka ovat kannustaneet ja eri tavoin auttaneet minua tämän projektin ja koko opintojeni aikana.

Espoossa 21.4.2008

Mikko Tuomela

INSINÖÖRITYÖN TIIVISTELMÄ

Tekijä: Mikko Tuomela

Työn nimi: Verkkolehden julkaisujärjestelmän kehittäminen

Päivämäärä: 21.4.2008 Sivumäärä: 34 s. + 2 liitettä

Koulutusohjelma: Tietotekniikka Suuntautumisvaihtoehto: Ohjelmistotekniikka

Työn valvoja: lehtori Ilpo Kuivanen

Työn ohjaaja: Olli Stålström

Tämän insinöörityön puitteissa suunniteltiin ja toteutettiin yleistajuiselle sosiologian alan verkkolehdelle julkaisujärjestelmä. Tavoitteena oli löytää parhaat toimintatavat Catalyst-sovelluskehyksen käytössä sekä tutkia sitä miten toteutetaan kevyt ja selkeä web-käyttöliittymä.

Työn alkuosassa käytiin läpi dynaamisen webin toteutustekniikoita ja erilaisia MVC-arkkitehtuurimallia (malli-näkymä-ohjain) toteuttavia web-sovelluskehyksiä. Näihin kuuluvat työssä käytetyn Catalystin lisäksi muun muassa Django ja Ruby on Rails. Todettiin, että verkkosovellusten suunnittelussa ja toteutuksessa web-sovelluskehys on hyödyllinen sen hoitaessa useat sovelluksen rutiininomaiset tehtävät ja se eriyttää sovelluksen eri komponentit loogisella tavalla.

Verkkolehden julkaisujärjestelmälle asetettuja tavoitteita olivat muun muassa käyttämisen helppous ja käyttöliittymän ulkoasun keveys ja selkeys. Työssä käytiin läpi näihin liittyviä käsitteitä ja suosituksia ja analysoitiin soveltuvia toteutustapoja.

Järjestelmän MVC-mallin mukaisten komponenttien toiminta käytiin läpi ohjelmakoodiesimerkkejä apuna käyttäen. Huomattiin, että verkkosovelluksen käytännön toiminnasta johtuen puhtaan MVC-järjestelmän toteutus on haastavaa, sillä selainkäyttöliittymässä näytettyä sisältöä joudutaan jonkin verran käsittelemään ja räätälöimään ohjainkomponentissa. Tällöin ohjain- ja näkymäkomponentit eivät ole vapaasti vaihdettavia.

Catalystin kaltainen MVC-mallia toteuttava web-sovelluskehys on käytännöllinen ja tehokas verkkosovelluksen toteutuksessa. Erityisen tärkeä etu tällaisen sovelluskehyksen käytössä on se, että tällöin ohjelmointiprosessin painopiste siirtyy aikaisempaa enemmän järjestelmäsuunnittelun suuntaan, jolloin tietoturva-asioihin ja järjestelmän toiminnan sekä tietokannan suunniteluun voidaan keskittyä paremmin.

Avainsanat: MVC, Catalyst, sovelluskehykset, WWW, verkkojulkaisu, julkaisujärjestelmät

ABSTRACT

Name: Mikko Tuomela

Title: Development of a web magazine's managing system

Date: 2008-04-21 Number of pages: 34 + 2 attachments

Department: Information technology Study Programme: Software engineering

Instructor: Ilpo Kuivanen

Supervisor: Olli Stålström

Within this thesis project a managing system was developed for a web magazine of the sociology field. The aim was to find the best practices in the use of the Catalyst web framework and to research the implementation of a light-weight and clear web user interface.

In the first part of the work some technologies of dynamic web and different web frameworks implementing the model-view-controller architecture model were described. Among these are, in addition to Catalyst which was used in this project, Django and Ruby on Rails. It was concluded that in the design and development of web applications a web framework is useful as it takes care of most of the everyday routines of the application and it separates the application's different components in a logical way.

Some aims laid down for the system were, among others, the ease of use and lightweightness and clearness of the user interface. Related terms and recommendations were discussed and applicable implementation methods were analysed.

The functionality of the system's MVC components was described with the aid of code samples. It was observed that due to the reasons of practical functionality developing a pure MVC system is challenging because in a browser user interface the presented content has to be somewhat processed and tailored in the controller component. Thus the controller and view components are not freely interchangeable.

A web framework implementing the MVC model, such as Catalyst, is both convenient and efficient in the development of a web application. An especially important advantage in the use of a framework like this is that the focus of the software development process shifts to the direction of system design, enabling the developers to concentrate more on issues related to security, system functionality and the design of the database.

Keywords: MVC, Catalyst, frameworks, WWW, web publishing, content management systems

SISÄLLYS

1 JOHDANTO

World Wide Webistä, joka monille muodostaa näkyvän osan Internetistä, on muodostunut kansainvälisen aktivismin ja yhteistoiminnan kenttä. Avoimet standardit, ilmaiset ohjelmistot ja edullisemmaksi käyvä tekniikka ovat tuoneet Internetin laajalle yleisölle. Tämä on luonnollinen kasvualusta verkkolehdille ja muille verkkopalveluille.

Tämä työ on tehty FinnQueerille, joka on vapaaehtoisvoimin ylläpidettävä suomalainen, monikielinen verkkolehti. Se perustettiin syksyllä 1999, kun homoseksuaalisuuden sairausleiman lopusta väitellyt Olli Stålström haastettiin oikeuteen kunnianloukkauksesta epäiltynä ja hänen väitöskirjaansa vaadittiin tuhottavaksi. Langettavan tuomion tapauksessa olisi FinnQueer toiminut opinnäytetyön sekä asiaan liittyvien artikkelien levityskanavana.

Vapauttavan tuomion jälkeen FinnQueer kehittyi yleistajuiseksi sosiologian alan verkkolehdeksi, jossa on artikkelien ja uutisten lisäksi julkaistu PDF-muodossa alaan koskevia verkkokirjoja ja opinnäytetöitä. Verkkolehdellä oli oma osansa syksyn 2001 parisuhdelakia koskevassa keskustelussa, ja sen nopea raportointi toimitti muun muassa kansanedustajille asiaa koskevia lausuntoja. [1]

FinnQueerin tekniset lähtökohdat ovat olleet, että uusia teknisiä ratkaisuja käytetään vain sisällön tukemiseksi eivätkä ne saa olla esteenä yhteensopivuudelle eri selainten kanssa – eri käyttäjäryhmien tulee voida selata sivustoa vaikeuksitta. Tämä asettaa eräitä erityisvaatimuksia verkkolehden käyttöliittymälle sekä teknisille ratkaisuille, joista enemmän luvuissa 4 ja 5.

Työn tavoitteena on ollut kehittää toimiva tekninen ratkaisu verkkolehdelle niin, että käyttäjän kannalta tehokas käyttöliittymä saadaan aikaan säilyttäen mahdollisimman suuri yhteensopivuus selainten kanssa, ja samalla verkkolehden toimittaminen on helppoa. Ratkaisun tulee olla kohtuullisella vaivalla laajennettava, helposti ylläpidettävä ja sitä tulee olla mahdollista hyödyntää muissakin verkkojulkaisuissa. Tässä yhteydessä tutkitaan, tarkastellaan ja arvioidaan erilaisia dynaamisen webin teknologioita.

Koko verkkolehden uudistaminen on laaja projekti ja julkaisujärjestelmä on vain osa sitä. Tämän insinöörityön puitteissa toteutettu ohjelmisto ei ole vielä

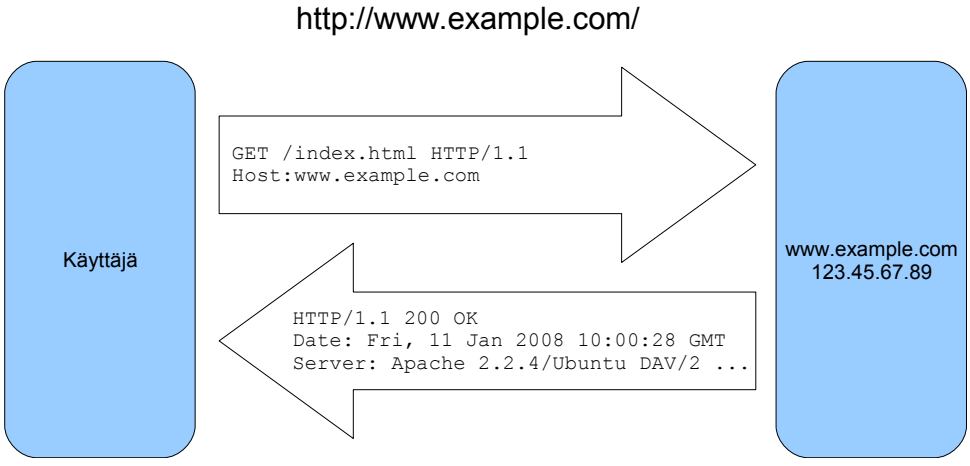
tuotantokäytössä - se on kuitenkin pohja modulaariselle julkaisujärjestelmälle, johon vielä toteutetaan useita lisäominaisuuksia ja joka käy läpi useita testauksia. Tässä työssä keskitytäänkin sen kehittämiseen, teknisiin ratkaisuihin ja Catalyst-sovelluskehyksen soveltamiseen julkaisujärjestelmän kehittämisessä.

2 WEB-TEKNOLOGIAT

Kuten mitkä tahansa tekniset järjestelmät, Internet ja sen tunnetuin osa, World Wide Web eli WWW, koostuvat erilaisista standardeista ja alijärjestelmistä. Ajan myötä selaimille asetetut vaatimukset kasvavat ja samalla palvelujen tekijöiden odotukset asiakasohjelmistojen (selainten) ominaisuuksista kasvavat.

2.1 World Wide Web

Internet kehittyi alun perin yhdysvaltalaisesta sotilasverkko ARPAnetistä. Verkko pysyi pitkään lähinnä tutkijoiden käytössä, mutta sen avulla voitiin käyttää monia verkkoa hyödyntäviä sovelluksia ja palveluita, kuten sähköposti ja erilaisia tietopalveluita. Vasta 1990-luvulla uusien helppokäyttöisten palveluiden, erityisesti WWW:n, myötä Internet levisi suuren yleisön tietoisuuteen.



Kuva 1: Dokumentin hakeminen WWW-palvelimelta

Vuonna 1990 Tim Berners-Lee julkaisi ehdotuksensa maailmanlaajuisesta hypertekstidokumenttien verkosta, World Wide Webistä sekä sen protokollasta (HyperText Transfer Protocol, HTTP). Myös URL-osoitteet, joiden avul-

la verkon eri palvelut voidaan paikallistaa, juontavat juurensa tästä ehdotuksesta.

Palvelinohjelmisto kuuntelee tiettyä porttia (HTTP:n oletusportti on 80) ja vastaa sille lähetettyihin sivupyyntöihin (kuva 1). Staattisten sivujen tapauksessa se siirtää palvelimen tiedostojärjestelmässä olevia valmiita HTML-sivuja edelleen selaimelle.

Vuonna 1993 esitelty CGI-standardi mahdollisti parametrien välittämisen palvelimelle, jolloin dynaamiset WWW-sivut tulivat mahdollisiksi. Sittemmin on kehitetty useita eri teknologioita dynaamisuuden toteuttamiseksi. Näistä osa on palvelimella toimivia ohjelmamoduuleita ja jotkut taas selaimessa toimivia.

Termillä *Web 2.0* tarkoitetaan WWW:n uutta aaltoa, jossa käyttäjät osallistuvat aktiivisesti palvelun sisällöntuottamiseen ja kehittämiseen. Sillä voidaan myös tarkoittaa tällaisten palvelujen toteuttamisessa käytettävää tekniikkaa, johon kuuluvat muun muassa JavaScriptin ja XML:n avulla toteutetut käyttöliittymät, RSS ja vuorovaikutteiset ominaisuudet kuten blogien kommentointi ja sivujen muokkaus wiki-tyyliin.

Internetin saatavuus ja suuri käyttäjäkunta tekevät siitä houkuttelevan ympäristön verkkolehdille ja muille perinteisten viestimien sähköisille muodoille. Samalla sen palveluiden sovittaminen erilaisille päätelaitteille ja selainohjelmistoille nousee merkittäväksi haasteeksi.

2.2 Selainohjelmistot

World Wide Webiä käytetään selaimeksi kutsutun tietokoneohjelmiston avulla. Alkuaikojen suosituimman graafisen selaimen, Mosaicin, seuraajiksi ilmaantuivat pian Netscape Navigator sekä Microsoftin Internet Explorer.

Netscapen tukema Mozilla-projekti sai hitaan alun jälkeen paljon suosiota ja sihen pohjautuva Firefox on nykyään selaimista toiseksi käytetyin. KDE-projektin Konqueror ja Applen MacOS X:n mukana toimitettava Safari pohjautuvat samaan ohjelmakoodiin, KDE:n KHTML-komponenttikirjastoon.[1] Avoimen lähdekoodin WWW-komponenttien pohjalta on rakennettu suuri määrä eri asioihin painottuvia selaimia, joiden käyttäjäkunta on kuitenkin pysynyt

1 Tarkkaan ottaen sekä Konqueror että Safari pohjautuvat WebKitiin, joka on kehitetty KHTML:stä. Nämä selaimet ovat tätä kirjoittaessa parhaat ACID3-testin suorittavat selaimet.

verrattain pienenä. Tällaisten ”vaihtoehtoselainten” joukkoon kuuluvat esimerkiksi Epiphany, Mozilla Suite/Seamonkey ja Camino.

Taulukko 1: Selainten markkinaosuudet maailmalla helmikuussa 2008. Osuudet on mitattu Network Applicationsin yhteistyökumppanien sivustojen kävijöistä. ja ovat tarkkuudestaan huolimatta vain suuntaa-antavia. [2]

	Selain	Osuus	Saatavilla	Lähdekoodi
1.	Microsoft Internet Explorer	74,88 %	Windows	Suljettu
2.	Mozilla Firefox	17,27 %	Windows, Linux, Mac, UNIX ym.	Avoin
3.	Apple Safari	5,70 %	Mac, Windows	Suljettu/Avoin[2]
4.	Opera	0,69 %	Windows, Linux, Mac ym.	Suljettu
5.	Netscape Navigator	0,68 %	Windows (ym.)[3]	Suljettu/Avoin

Vaikka periaatteessa WWW:n tekniikkaa määrittävät erilaiset standardit, pitää usein selainten ominaisuudet ja viat ottaa huomioon web-suunnittelussa [8, s. 3]. Tällöin on myös olennaista tietää eri selainten markkinaosuudet, sillä optimoitaessa sivustoa näkymään tietyllä käyttäjäkunnalla voi käydä niin, että sivuston saattaminen näkymään yhdellä selaimella rikkoo sen toisella. Taulukossa 1 on listattu viiden suosituimman selaimen markkinaosuudet. Näitä lukuja tarkasteltaessa on huomioitavaa, että osuudet vaihtelevat maittain suurestikin ja erityyppiset käyttäjät käyttävät eri selaimia ja käyvät eri sivustoilla.

Selainten Cascading Style Sheets (CSS) -tukea testaamaan on kehitetty Acid-nimiset testit. Tätä kirjoitettaessa Acid2:n (julkaistu 2005) ovat yleisesti käytetyistä selaimista läpäisseet vain Opera, Konqueror ja Safari.[4]

2.3 Dynaamisen webin teknologioita

Dynaamisella webillä tarkoitetaan tapaa vuorovaikuttaa sovelluksen kanssa selaimen kautta. Tekniset ratkaisut voidaan jakaa kahteen ryhmään:

1. Asiakaspään skriptaus (client-side scripting): ohjelmakoodi suoritetaan selaimessa, ja se muokkaa käyttäjälle näkyvää käyttöliittymää.

2 Safarin ytimenä on vapaan lähdekoodin WebKit, mutta muu osa selaimesta on normaali omistusoikeudellinen ohjelmisto.

3 Alunperin Netscape Navigator oli saatavilla lähes kaikille yleisille käyttöjärjestelmille mutta viimeiset versiot (tuki loppui vuonna 2008) olivat vain Windowsille. Uusimmat versiot pohjautuivat Mozilla-projektin vapaan lähdekoodin selaimeen.

4 Tässä ei oteta huomioon kehitysversioita vaan ainoastaan lopulliset julkaisuversiot.

2. Palvelinpään skriptaus (server-side scripting): ohjelmakoodi suoritetaan web-palvelimella sivuhakujen välissä. Parametrit voidaan saada suoraan URL-osoitteesta, istunnon tilasta tai lähetetyn lomakkeen kenttien arvoista.

Monet käytetyistä tekniikoista ovat itse asiassa näiden yhdistelmiä; esimerkiksi lomakkeen tiedot voidaan tarkistaa jo selaimessa JavaScript-ohjelmalla ennen lähetystä, jolloin palvelin saa vain oikeanmuotoista dataa.[5] [3]

CGI (Common Gate Interface) on standardi web-pohjaisten ohjelmistojen asiakas-palvelin-rajapinnan toteuttamiseen. Se ei edellytä erityistä teknologiaa käyttäjän selaimelle; sivujen sisältö generoidaan palvelimella syötteiden perusteella. Se ei myöskään määrää palvelimella CGI-ohjelman toteuttamiseen käytettyä teknistä ratkaisua, mutta hyvin usein nämä skriptit toteutetaan Perl-ohjelmointikielellä. [4, s. 271-272]

Koska toiminnallisuus on palvelimella, CGI:n käyttö ei edellytä käyttäjän ohjelmistoilta erityisiä liitännäisiä tai muita tekniikoita. CGI:llä toteutetun sivun lataaminen voi kuitenkin olla palvelimelle raskas toiminto, sillä ohjelmointikielen tulkin ja sen kirjastomoduulien lataaminen voi viedä enemmän aikaa, kuin itse sivun luovan ohjelmakoodin suoritus. [9, s. 17]

PHP on erityisesti selaimen kautta käytettävien sovellusten toteuttamiseen tarkoitettu komentosarja- eli skriptikieli, jota tulkitaan palvelimella.

PHP:n ensimmäisen version julkaisi Rasmus Lerdorf vuonna 1994; tuolloin PHP (”Personal Home Page Tools”) koostui sarjasta CGI-skriptejä, jotka hän julkaisi vapaina ohjelmistona[6] seuraavana vuonna. Yhteisöllinen kehitystyö tuotti version 2 vuonna 1996 ja lähes täydellisen uudelleenkirjoituksen jälkeen versio 3 (nykyisellä nimellä) julkaistiin 1998. Tätä kirjoitettaessa (marraskuu 2007) PHP:n uusin versio on 5.2.4.

Tyypillisessä PHP-tiedostossa PHP-ohjelmakoodi on upotettu HTML-koodin sekaan. Ohjelmakoodi erotetaan HTML-sisällöstä `<?php ?>` -ilmaisulla.

Seuraavat esimerkit tuottavat selaimessa identtisen tuloksen:

5 Tässä tapauksessa on tärkeää tarkistaa datan oikeellisuus myös palvelimella; on mahdollista että käyttäjän selaimessa ei ole JavaScript-tuki päällä tai on tapahtunut jokin virhe.
6 GNU General Public Licenseä käyttäen

```
<h1>Page Title</h1>

<?php
  $title = 'Page Title';
  $base_tag = 'h1';
  echo ”<$base_tag>$title</$base_tag>”;
?>
```

Jälkimmäisessä esimerkissä HTTP-palvelinohjelmistossa ajettava PHP-tulkki suorittaa yksinkertaisen tagin sisällä olevan ohjelmakoodin, joka tulostaa vastaavanlaisen otsikon kuin edellinen HTML-koodi.

EJB (Enterprise Java Beans) on Java-ohjelmointikieleen (Java 2 Enterprise Edition, J2EE) perustuva komponenttimalli, joka määrittää ympäristön, jossa ohjelmisto- ja tietokantapalvelimen yritystason ohjelmistokomponentit toimivat. Se tarjoaa mahdollisuuden kehittää hajautettuja järjestelmiä, jonka osat ovat uudelleenkäytettäviä ja vaihdettavia.

EJB:n soveltuvuus hajautettujen ja toimintavarmojen yritysohjelmistojen rakentamiseen selittää suurelti sen suosion. CORBA-komponentteja sekä muita Javalla ohjelmoituja ohjelmiston osia voidaan käyttää EJB-komponenttien kanssa. [11, s. 45-49]

AJAX (Asynchronous Javascript and XML) on *Web 2.0*:n erittäin yleisesti käytetty tekniikka dynaamisten web-sivujen luomiseen ja verkkosovellusten käyttöliittymien toteutukseen.

ASP (Active Server Pages) on Microsoftin suljetun lähdekoodin teknologia, joka yhdistää HTML-sivuja ohjelmakoodiin. ASP:n yhteydessä käytetään ohjelmointikielenä yleensä VBScriptiä tai JScriptiä, mutta periaatteessa muitakin kieliä voi käyttää. ASP-tulkki toimii itse WWW-palvelimessa (käytännössä Microsoftin Internet Information Server) ja tarjoaa yleensä paremman nopeuden kuin CGI-skriptit.

ASP:ia ei pidä sekoittaa ASP.NET:iin joka on sitä muistuttava teknologia, jossa koodi kuitenkin käännetään ja joka toteuttaa Microsoftin .NET-sovelluskehystä.

Template Toolkit (TT) on Perlillä toteutettu vapaan lähdekoodin mallinemoottori, jolla toteutetaan web-sivujen mallineita yhdistämällä TT:n omaa yksinkertaista skriptikieltä ja HTML-koodia.

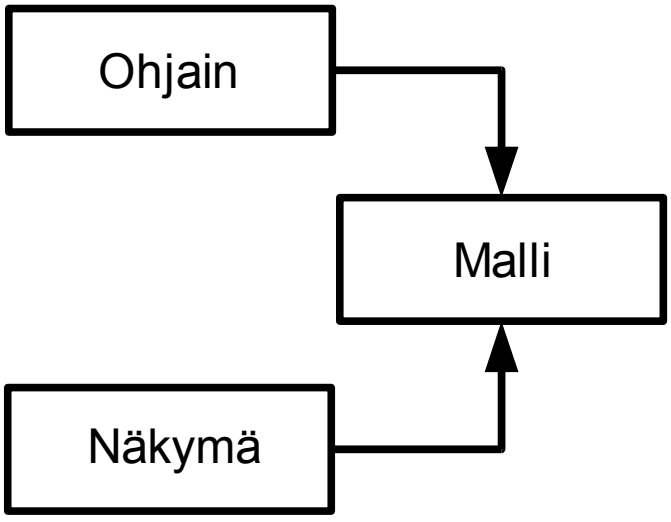
TT:tä käytetään yleisesti Catalyst-sovelluskehyksen kanssa, kuten tämän työn yhteydessä. Tällöin se toimii MVC-mallin näkymäkomponenttina, jonka ohjain pyytää näyttämään tietyn tietosisällön. TT:llä on mahdollista toteuttaa myös muita dokumentteja kuin web-sivuja, esimerkiksi PDF- tai LaTeX-muotoisia asiakirjoja. [7]

3 MVC-ARKKITEHTUURI WEB-SOVELLUSKEHITYKSESSÄ

MVC-arkkitehtuuri[7] kuvattiin ensimmäisen kerran vuonna 1979 Smalltalk-olio-ohjelmointikielen kehittäjän Trygve Reenskaugin toimesta ja sen ensimmäinen toteutus tehtiin Smalltalk-80:lla. Arkkitehtuurissa sovelluksen eri osat eriytetään omiksi komponenteikseen, mikä helpottaa nykyaikaisten oliopohjaisten sovellusten kehitystä ja niiden rakenteen ymmärtämistä [6].

3.1 Yleiskuvaus

Tyypillisellä sovelluksella on jonkinlainen käyttöliittymä (käyttäjärajapinta) ja sovelluksella voidaan katsoa olevan tietovarasto, jonne se tallettaa käsiteltävän datan.



Kuva 2: Minimaalinen MVC-järjestelmä

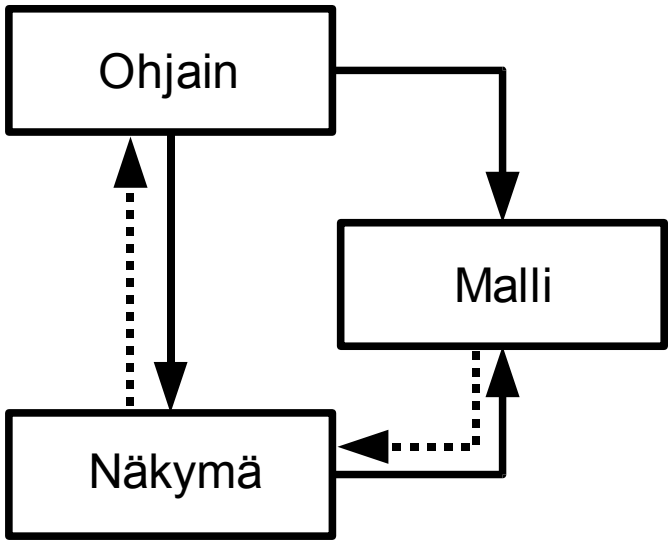
Model-view-controller-arkkitehtuurissa sovelluksen eri osat toteutetaan eri komponentteina (kuva 1):

1. **Malli** (model) kuvaa sovelluksen käyttämän tietorakenteen ja toteuttaa sen käsittelyyn liittyvän osan sovelluksesta.

7 Suomeksi joskus malli-näkymä-ohjain-arkkitehtuuri [5, s. 142]. Tässä työssä käytetään kuitenkin englanninkielistä nimeä.

2. **Näkymä** (view) määrittää tavan, jolla tietosisältö saatetaan käyttäjälle. Yhdelle mallille voi olla useita eri näkymiä. Näkymä myös ohjaa käyttäjän syötteen ohjaimelle.
3. **Ohjain** (controller) on sovelluksen ohjaava osa, joka ottaa näkymältä vastaan syötteitä, vastaa niihin ja määrää näkymän ja mallin toimintaa. [5, s. 142]

Web-sovelluskehityksessä MVC-malli on suosittu. Tällöin näkymää vastaa selaimeen tuotettu HTML-muotoinen sivu, joka sisältää myös sovelluksen hallintaan tarvittavat toiminnot (linkit ja lomakkeet), ohjain taas on sovelluksen toimintaa ohjaava ohjelma (esimerkiksi Perl-skripti) ja malli taas on sovelluksen tietokanta (esimerkiksi MySQL-kanta tai sovelluksen hakemisto palvelimen tiedostojärjestelmässä) [6].



Kuva 3: Monimutkainen MVC-malli suuremmalla määrällä vuorovaikutustapoja [8]

MVC:n erottaa kolmitasoarkkitehtuurista se, että sen komponentit muodostavat kehän; kolmitasoarkkitehtuuri taas on lineaarinen. MVC:ssä erityisesti näkymä ja malli vuorovaikuttavat suoraan ilman ohjainta – tämä on kolmitasoarkkitehtuurissa mahdotonta. [10]

Päivityksiä varten MVC-sovellus kuitenkin tarvitsee komponenttien ulkopuolisen tarkkailijakomponentin (observer). Tarkkailija muodostaa yksi-moneen-suhteen komponentteihin ja päivityksen tapahtuessa antaa muille komponenteille tiedon, että tieto on muuttunut ja uusi versio pitää hakea.

3.2 Toteutuksia

Koska MVC-arkkitehtuurimalli soveltuu hyvin verkkosovellusten suunnitteluun, sen pohjalta on kehitetty useita web-sovelluskehityksiä, jotka helpottavat sovelluksen rakentamista tarjoamalla valmiit rajapinnat komponenttien väliseen vuorovaikutukseen.

Näillä sovelluskehyksillä on useita yhteisiä piirteitä:

- toteutus skriptikielellä
- avoin lähdekoodi
- kevyt, varsinaisesta WWW-palvelimesta erillinen palvelinohjelma kehitystä varten
- säännöllisiin lausekkeisiin (regular expressions) perustuva osoitteiden käsittely
- modulaarisuus, hyvä laajennettavuus

Tässä tarkastellaan yleisimmin käytettyjen toteutusten erityispiirteitä.

Django on vuonna 2005 julkaistu sovelluskehys, joka oli alun perin tarkoitettu uutispohjaisten sivustojen julkaisemiseen. Se on toteutettu Python-kielellä.

Ruby on Rails on Rubylla toteutettu sovelluskehys, joka käyttää erityisesti JavaScriptiä ja AJAX:ia dynaamisten sovellusten rakentamiseen. Ruby on Rails on sisällytetty Apple MacOS X 10.5 -käyttöjärjestelmään.

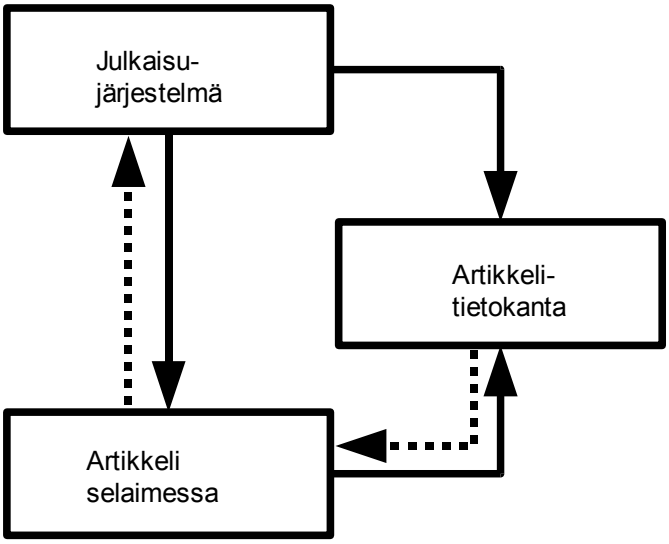
Catalyst on Perlillä toteutettu MVC-mallia kattavasti seuraava web-sovelluskehys, joka painottaa Perl-moduulien uudelleenkäyttöä siten, että se ei toteuta uudelleen tyypillisiä web-sovelluskehityksen komponentteja. Siten MVC-mallin mallikomponentti toteutetaan esimerkiksi Perlin DBIx::Class-moduulilla ja näkymäkomponentti Template Toolkitillä.

Drupal on suosittu, PHP:llä toteutettu julkaisujärjestelmän ja web-sovelluskehyksen yhdistelmä. Se toimii paitsi Apache-palvelinohjelmiston, myös Microsoftin Internet Information Server -ohjelmiston kanssa ja vaatii seurakseen tietokannan. Drupalin ensimmäinen vapaan lähdekoodin versio julkaistiin jo vuonna 2001.

Koska Drupalissa on valmiit rajapinnat erilaisten verkkopalveluiden komponenteille (keskustelupalstat, lomakkeet, käyttäjätilit), ohjelmoijan ei välttämättä tarvitse toteuttaa näitä uudestaan.

3.3 MVC-järjestelmän komponentit verkkolehdessä

MVC-järjestelmänä toteutetussa verkkolehdessä (kuva 4) merkittävin osa sovelluslogiikasta sijoittuu ohjaimeen. Mallia vastaa artikkelit sisältävä tietokanta ja näkymänä toimii verkkolehden selaimessa näkyvä sivu. Toimittajien kokeiluja varten voi tietokannasta olla samaan aikaan toiminnassa testiversio, johon voidaan ylläpitoliittymässä vaihtaa lennosta. Vastaavasti artikkelinäkymästä voi olla eri versiot esimerkiksi mobiilipäätelaitteille ja tulostusta varten. Erikoisversioita sisällöstä voi tuottaa kokonaan erillinen logiikka, joka käyttää kuitenkin samaa tietokantaa ja samoja näkymäkomponentteja.



Kuva 4: Eräs verkkolehden näkymä komponentteina

AJAX-tekniikalla voidaan toteuttaa tehokkaita, dynaamisia käyttöliittymiä, joissa näkymän osat viestivät suoraan tietokannan kanssa ilman, että weblomaketta lähetetään ohjainkomponentin käsiteltäväksi. Tällöin voidaan toteuttaa MVC-järjestelmissä usein oleva tarkkailija- eli observer-komponentti, jonka avulla komponentit saavat jatkuvasti ajan tasalla olevia tietoja toisiltaan [5, 142].

4 JULKAISUJÄRJESTELMÄN SUUNNITTELU

4.1 Vaatimusmäärittely

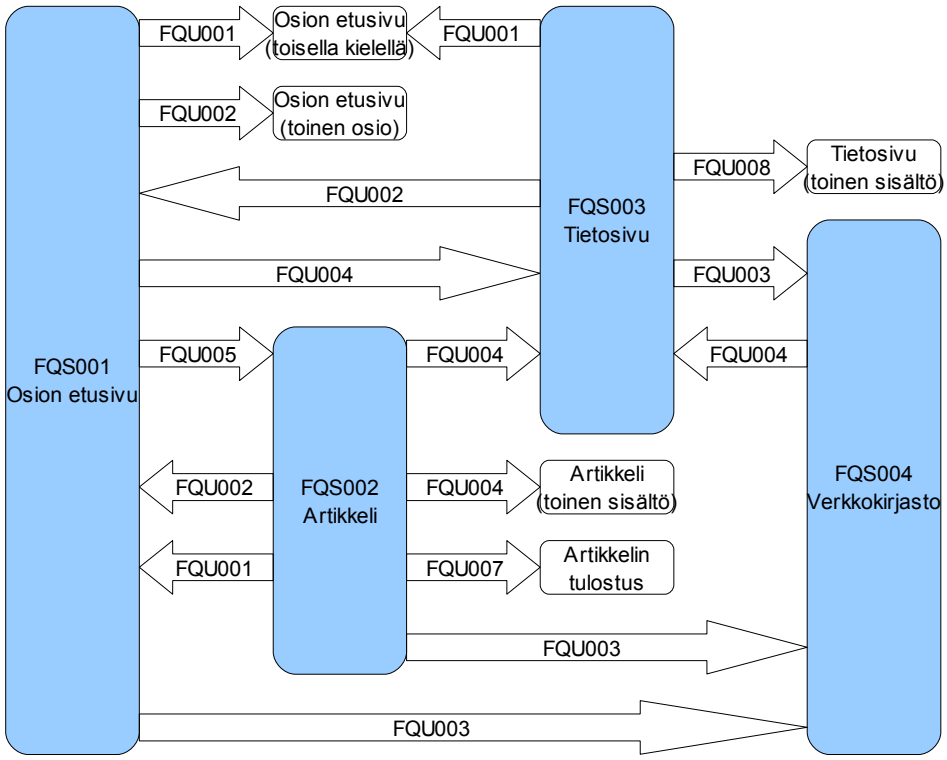
Järjestelmän teknisten ratkaisujen tulee tukea sen käyttötarkoitusta; tässä tapauksessa verkkolehden sisällön välittämistä lukijalle. Tästä voidaan johtaa seuraavankaltaiset vaatimukset julkaisujärjestelmälle:

1 Järjestelmän tulee toimia Apache-palvelinohjelmiston moduulina Ubuntu Linux 7.10 -käyttöjärjestelmässä.
2 Järjestelmän tulee toteuttaa seuraavat käyttöliittymät:
  2.1 Peruskäyttäjän käyttöliittymä (näkyvä verkkolehti)
    2.1.1 Osion etusivu / artikkelilistaus
    2.1.2 Artikkelinäkymä
    2.1.3 Verkkokirjasto
    2.1.4 Tietosivut
  2.2 Ylläpidon käyttöliittymä (sisällön muokkaus)
    2.2.1 Osioiden ylläpito
    2.2.2 Artikkelien ylläpito
    2.2.3 Käyttäjien ylläpito
3 Ylläpidon käyttöliittymän tulee edellyttää käyttäjältä tunnusta ja salasanaa ja ylläpitää istuntoa evästeiden avulla.
4 Järjestelmän tulee pystyä palvelemaan jatkuvasti 10 artikkelisivua kuvineen sekunnissa.
5 Verkkolehden sisällön tulee olla luettavissa kaikilla selaimilla.
  5.1 Peruskäyttäjän käyttöliittymä ei saa edellyttää lukijan selaimelta JavaScript- tai evästetukea.
  5.2 Kunkin normaalisivun tulee olla säännönmukainen (validi) XHTML-dokumentti.
  5.3 Sivut tulee voida tuottaa myös kevytversioina (ei grafiikkaa) ja PDF-muodossa.
6 Selaimeen siirrettävän datan määrä, ottamatta huomioon sisältöön liitettyjä kuvia, ei saa ylittää 100 kB / yksi siirretty sivu.
  6.1 Yhden tuotetun sivun HTML-koodi ei saa olla kooltaan yli 50 kilotavua.
  6.2 Käyttöliittymän joka sivulla toistuva perusgrafiikka ei saa olla kooltaan yli 50 kilotavua.
7 Käyttöliittymä tulee voida näyttää eri kielillä.

4.2 Peruskäyttäjän käyttöliittymä

Peruskäyttäjän käyttöliittymä, (kuva 5), jota kaikki sivuston käyttäjät poislukien ylläpitäjät käyttävät, jakautuu neljäntyyppisiin käyttäjälle näkyviin osiin:

1. **Osion etusivu** (FQS001) näyttää tietyn aihepiirin artikkelien otsikot sekä artikkelien esittelytekstit.
2. **Artikkeli** (FQS002) on sivu, jolla verkkolehden artikkelisisällöt näytetään.
3. **Tietosivu** (FQS003) sisältää pysyvää informaatiota verkkolehdestä ja sen esittelyn.
4. **Verkkokirjasto** (FQS004) on käyttöliittymä palvelimella ylläpidettyyn PDF-muotoiseen aineistoon ja sen esittelysivuihin.



Kuva 5: Peruskäyttäjän käyttöliittymän tilat ja niiden väliset siirtymät

Normaalikäyttäjän käyttöliittymän tilojen ja siirtymien täydellinen lista on liitteessä 1.

4.3 Ylläpitäjän käyttöliittymä

Verkkolehden ylläpito tapahtuu ylläpidon käyttöliittymän (kuva 6) kautta. Käyttöliittymä vaatii sisäänkirjautumisen ja tämän jälkeen luodun istunnon aikana voidaan käyttää ylläpitotoiminteita.

Kaikilla ylläpitäjäkäyttäjillä ei ole oikeuksia käyttää kaikkia toiminteita; valikoissa näytetään vain ne toiminteet, joihin käyttäjällä on oikeus. Käyttäjätasot toteutetaan rooleilla, jotka ovat:

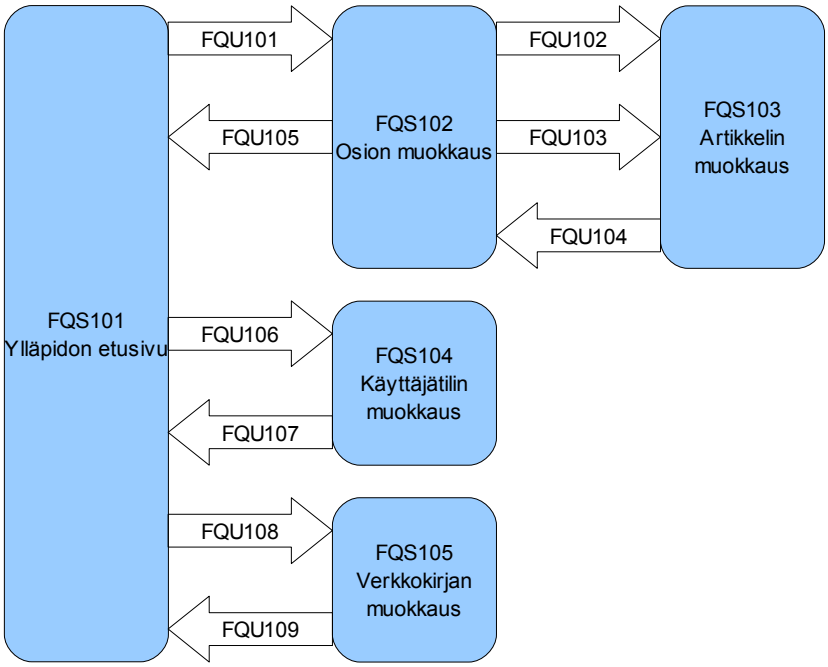
1. **Toimittaja** voi ainoastaan lisätä artikkeleita ja muokata itse lisäämiään artikkeleita.
2. **Ylläpitäjä** voi muokata osioita ja kaikkia artikkeleita.
3. **Pääkäyttäjä** voi muokata käyttäjätilejä ja antaa käyttäjille rooleja.

Istunto lopetetaan uloskirjautumisella, jolloin palataan peruskäyttäjän näkymään ja selaimessa oleva kirjautumisesta kertoma eväste poistetaan.

Ylläpitäjän käyttöliittymä voidaan jakaa seuraavaan viiteen osaan:

1. **Ylläpidon etusivu** (FQS101) listaa verkkolehden aihepiirit (osiot) ja ylläpitäjät ja antaa mahdollisuuden valita jonkun näistä muokattavaksi tai luoda uusia.
2. **Osion muokkaus** (FQS102) listaa osion tiedot sekä artikkelit ja antaa mahdollisuuden muokata jotain niistä tai luoda uuden artikkelin kyseiseen osioon.
3. **Artikkelin muokkaus** (FQS103) mahdollistaa artikkelien parametrien ja sisällön muokkauksen ja sisällön esikatselun.
4. **Käyttäjän muokkaus** (FQS104) mahdollistaa käyttäjätilin ominaisuuksien muokkaamisen ja uusien käyttäjien lisäämisen ylläpitojärjestelmään.
5. **Verkkokirjan muokkaus** (FQS105) mahdollistaa verkkokirjaston kirjan ominaisuuksien muokkaamisen.

Ylläpitäjien käyttöliittymän tilojen ja siirtymien täydellinen lista on liitteessä 1.

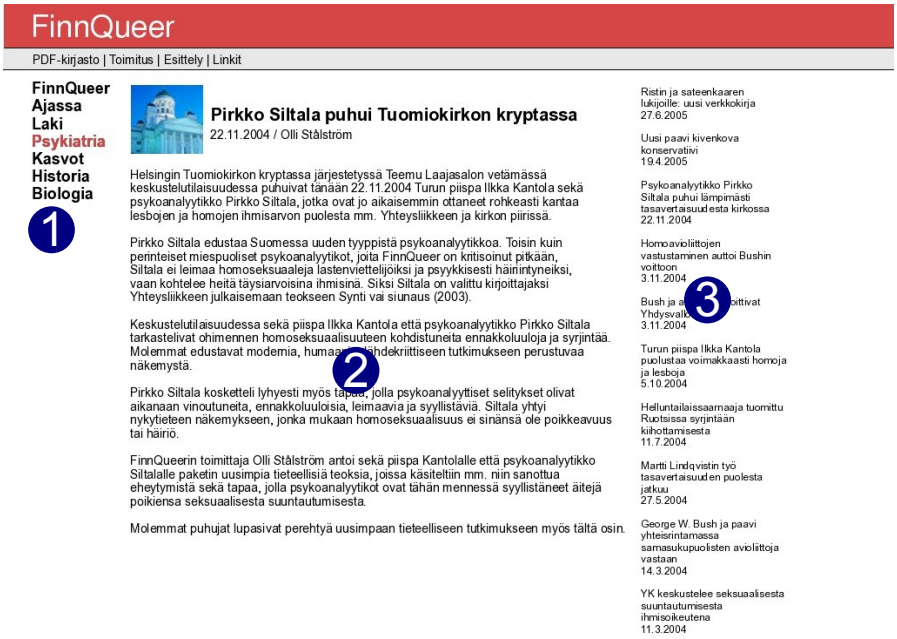


Kuva 6: Ylläpitäjän käyttöliittymän tilat ja niiden väliset siirtymät

4.4 Käyttöliittymän ulkoasu

Vaatimusmäärittelyn kohdat 5 ja 6 alakohtineen määrittävät, että sivuston tulee olla kevyt ja toimiva eri selaimilla. Tästä voidaan edelleen johtaa edellytys, että verkkolehden käyttöliittymän ulkoasun tulee olla sellainen, että sen kuvatiedostot eivät ole tiedostokooltaan kovin suuria. Toisaalta myös verkkolehden sisällön (tässä tapauksessa kyseessä on yleistajuinen sosiologiaa ja seksologiaa ja lakiakin käsittelevä verkkolehti) kannalta on olennaista, ettei visuaalinen ilme ole liian räväkkä.

Vaatimus kevyestä käyttöliittymästä on ratkaistu siten, että yhdellä Finn-Queer-verkkolehden sivulla on vain yksi käyttöliittymään liittyvä kuvatiedosto - nimittäin lehden logo vasemmassa yläkulmassa. Kaikki muut graafiset elementit on tuotettu CSS-tyyleillä.



Kuva 7: Sivun ohjelmallisesti tuotetut elementit

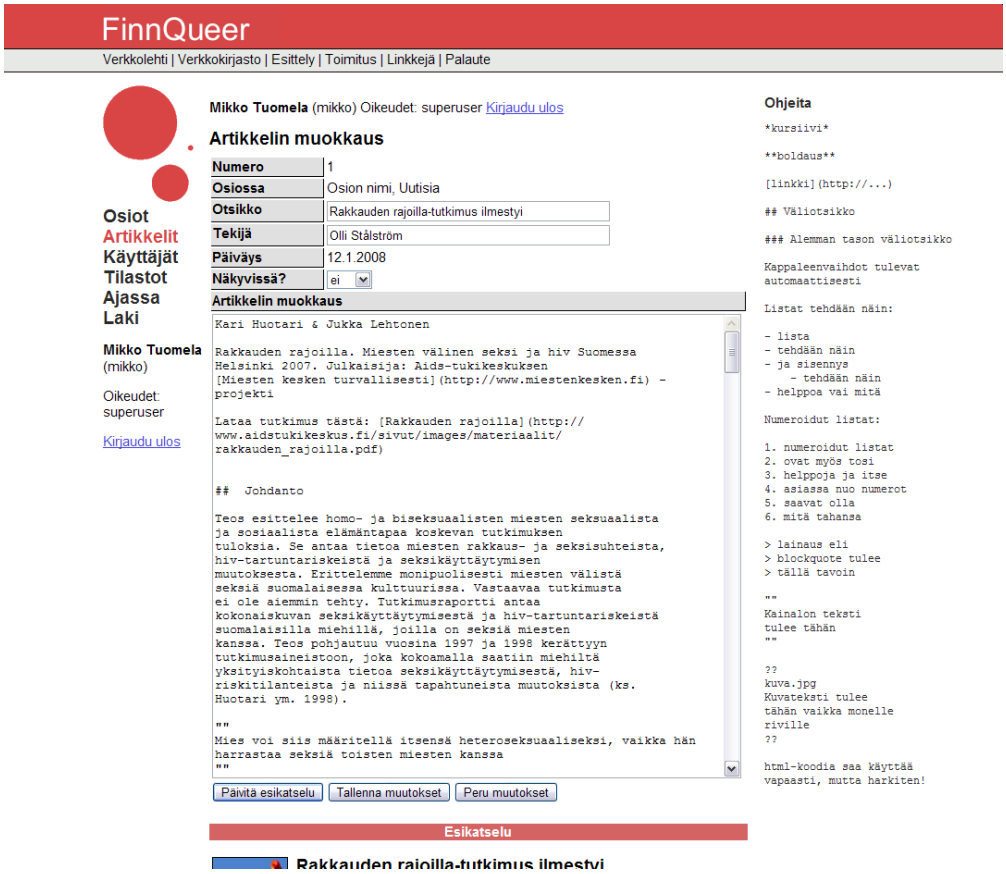
Kullakin peruskäyttäjän käyttöliittymän sivulla[8] (kuva 7) on seuraavat ohjelmallisesti tuotetut elementit:

1. osiolistaus
2. sisältöosa
3. kontekstivalikko.

Aktiivisesta sivusta riippuu, mitä sisältöä näissä näytetään, mutta tyypillisesti osiolistaus sisältää linkkejä muihin aihepiireihin, sisältöosa näyttää kyseisen sivun varsinaisen sisällön ja kontekstivalikko sisältää linkit muihin saman osion artikkeleihin tai näyttää aiheeseen liittyviä lisävalintoja.

Verkkolehden tavoitteista ja toimituksesta kertovat tietosivut (harmaalla taustalla olevat yläpalkin linkit) ovat itse asiassa normaaleja artikkeleita, jotka vain esitetään eri yhteydessä ja joita voi muokata ylläpidon käyttöliittymän kautta.

8 Käyttöliittymällä tarkoitetaan tässä template-näkymäkomponentin kautta tuotettuja selaimen kautta käytettäviä sivuja. Muiden näkymäkomponenttien tuotokset eivät sisällä käyttöliittymän kaikkia elementtejä.



Kuva 8: Artikkelin muokkausnäkymä

Ylläpidon käyttöliittymässä (kuva 8) oikeanpuoleista palstaa (kontekstivalikko) voidaan käyttää erilaisten ohjeiden välittämiseen toimittajille. Artikkelin muokkausnäkymässä siinä näytetään artikkeleiden kuvauskielen ohjeet.

Kaikki julkaisujärjestelmän selaimeen tuottamat HTML-dokumentit ovat XHTML 1.0 Strict -standardin mukaisia. Standardinmukaisuuteen pyrkimisen eräs tärkeimmistä syistä on se, että W3C:n validointipalvelun[9] avulla voidaan sivun HTML-koodin eheys tarkistaa ja tällä tavoin löytää mahdolliset virheet (jotka usein johtavat myös yhteensopivuusongelmiin eri selainten välillä).

4.5 Käytettävyysnäkökohtia

FinnQueer-verkkolehden käyttöliittymässä on aina pyritty selkeyteen - käytetään vain vähäistä määrää värejä (poislukien artikkeliin liittyvät täysväriset kuvat) ja pientä määrää erikokoisia tekstityylejä.

Kirjassaan *Web-suunnittelu* Jukka K. Korpela ja Tero Linjama pitävät hyvin tärkeänä sitä, että sivuston etusivulta saa selvää, mikä sivusto on kyseessä.

9 W3C Markup Validation Service, http://validator.w3.org/

FinnQueerin tapauksessa on tärkeää, että käyttäjä näkee sen olevan FinnQueer-niminen verkkolehti. Siksi ensimmäiset asiat käyttöliittymän vasemmassa ylänurkassa ovat juuri lehden logo ja teksti "Verkkolehti". [4, s. 357-358]

Korpela ja Linjama kirjoittavat, että web-suunnittelussa on hyvä pitäytyä yhdessä tekstipalstassa, toisin kuin sanomalehdissä, joissa usein on useita rinnakkaisia palstoja. Myös tasaus molempiin reunoihin on ongelmallinen muun muassa siksi, että selaimet eivät osaa tavuttaa sanoja. FinnQueerin tapauksessa artikkeleiden pituudet ja myös tyylit vaihtelevat paljon, jolloin ulkoasua ei voi sitoa tietynpituiseen tai -tyyliseen artikkeliin.

Koska selainikkunoiden koot vaihtelevat myös valtavasti (500:sta jopa 2000 pikseliin), voisi vakiolevyinen sivusto näkyä monilla joko liian leveänä tai liian kapeana. FinnQueerin tapauksessa käyttöliittymälle on määritelty minimileveys CSS-tyyleillä:

```
div.main {
    width:     80%;
    min-width: 600px;
}
```

Tämän ansiosta sivusto mukautuu selainikkunan leveyteen hyvinkin erilaisissa tapauksissa. Käyttöliittymän sisältävä alue on siis vähintään 600 pikseliä leveä ja enintään 80 % selainikkunan näyttöalueen leveydestä.

Ilman tyylitiedostoa ja jopa tekstiselaimella FinnQueerin artikkeleiden lukeminen onnistuu varsin hyvin. Täten mobiiliversion tuottaminen olisi jo normaalinäkymästäkin helppoa, mutta erillinen toiminto kevytversioon siirtymiseksi on tarpeen, sillä webissä ei valitettavasti voi luottaa siihen, että käyttäjä tai hänen käyttämänsä selain osaisivat tehdä muunnoksen automaattisesti.

5 JULKAISUJÄRJESTELMÄN TEKNINEN TOTEUTUS

Julkaisujärjestelmä on toteutettu käyttäen ainoastaan vapaita ohjelmistoja ja sen toimintaympäristönä tuotantokäytössä on palvelin, jonka käyttöjärjestelmänä on Ubuntu 7.10[10].

10 Ubuntu-projektin kotisivuilla käyttöjärjestelmästä käytetään nimeä ”Ubuntu” - se kuitenkin pohjautuu Linux-ytimeen ja Ubuntua voisi varmaan kuvailla myös GNU/Linux -järjestelmäksi sen käyttämien GNU-ohjelmistojen vuoksi.

5.1 Järjestelmän vaatimukset ja edellytykset

Palvelimessa, jolla julkaisujärjestelmää ajetaan, on Pentium 4 -prosessori ja Linux-ytimen i686-versio, eli käytännössä valtavirran mukainen x86-käskykanta. Projektissa on käytetty ainoastaan sellaisia ohjelmistoja, joiden lähdekoodi on avointa ja joista on olemassa versiot myös muille käyttöjärjestelmille (ja samalla muille prosessoriperheille).

Itse julkaisujärjestelmä ei ole vapaa ohjelmisto eikä toistaiseksi suunnitelmia sen julkaisemiseksi esimerkiksi GPL-lisenssillä ole. Syitä projektin toteuttamiseen yksinomaan vapaita ohjelmistoja käyttäen oli useita:

- yleinen saatavuus ja ilmaisuus
- toimivuus eri järjestelmissä (kehityksessä käytettiin sekä Linux- että Windows-käyttöjärjestelmällä toimivia työasemia)
- yhteensopivuus toistensa kanssa
- standardinmukaisuus
- yleisyys tuotantokäytössä (esim. Apache, MySQL) ja todennettu toimivuus.

Kehitystyössä tarvitut ohjelmistot käytettiin Ubuntun APT-käyttöliittymän kautta. Asennettujen ohjelmistojen versiot olivat:

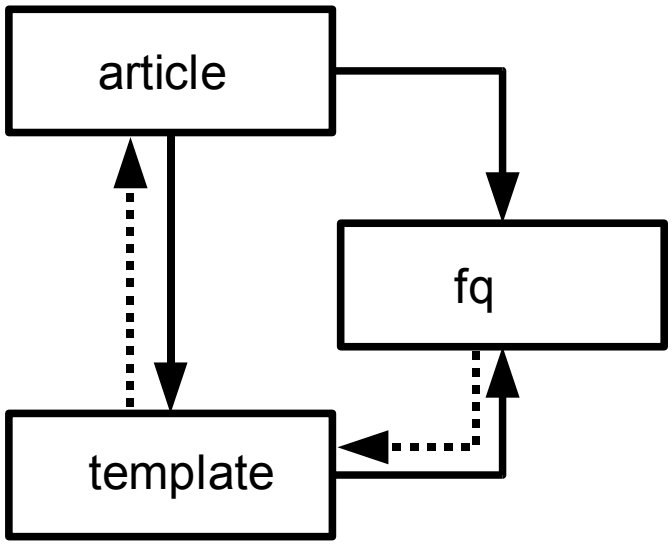
- Ubuntu 7.10
- Linux 2.6.22-14
- Catalyst 5.7007
- Perl 5.8.8
- Apache 2.2.4
- Template Toolkit 2.19
- MySQL 5.0.45

Ohjelmiston kehitysversio toimii Catalystin oman web-palvelimen ympärillä, mutta sen lopullinen versio toimii Apachen moduulina ja käynnistyy automaattisesti Apachen käynnistyessä.

5.2 MVC-järjestelmän komponentit

Catalyst toteuttaa MVC-mallia ja käyttäjän luomat komponentit sijoittuvat tämän mukaan hakemistoihin Model, View ja Controller. Siten jokaisessa Finn-Queerin näkymässä on aktiivisena yksi komponentti kutakin sorttia (joidenkin hakujen yhteydessä periaatteessa useitakin). Kuvassa 9 on esimerkki tyypil-

lisestä MVC-konfiguraatiosta käytettäessä verkkolehden normaalia käyttöliittymää.



Kuva 9: FinnQueerin MVC-malli artikkelinäkymässä: *fq* on malli, *template* on näkymä ja *article* on ohjain.

Sovelluskehys tarjoaa useita palveluita, kuten tietokannan käsittelyn, joten näissä komponenteissa ei välttämättä ole kovin paljon ohjelmakoodia.

5.2.1 Malli

FinnQueerDB (koko nimeltään *Catalyst::View::FinnQueerDB*) on julkaisujärjestelmän ainoa mallikomponentti, sillä projektissa tuotettu julkaisujärjestelmä on suunniteltu käyttämään vain yhtä tietokantaa, siis verkkolehden sisällön sisältävää kantaa.

Teoriassa olisi täysin mahdollista ohjelmoida rinnalle myös esim. *FinnQueerOldDB*-malli, joka käyttäisi FinnQueerin vanhan version tiedostojärjestelmässä olevia artikkelitiedostoja vastaavilla hauilla ja jota voisi siten käyttää uudella käyttöliittymällä. Tällöin vanhan mallin vaatimat tiedosto-operaatiot voitaisiin kokonaan määrittää *FinnQueerOldDB.pm*-tiedostossa. Tässä projektissa siihen ei kuitenkaan ollut tarvetta, sillä verkkolehden uusi versio korvaa kokonaan vanhan ja vanhan sisältö siirretään uuteen tehtävään erikseen kirjoitetulla Perl-skriptillä.

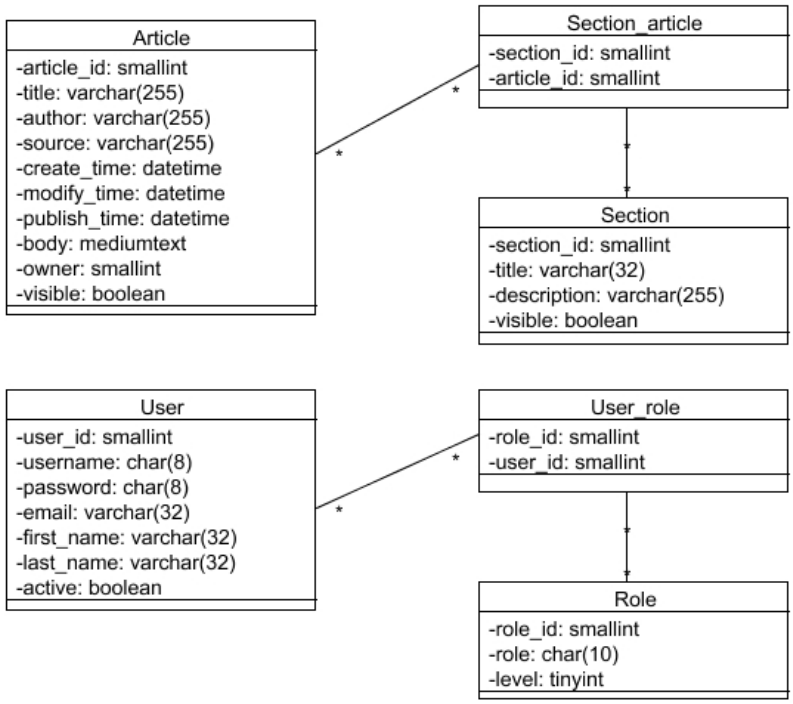
Koska tässä tapauksessa tietokannan hakujen tuottamisessa ei käytetä mitään erikoisia keinoja, tiedosto *FinnQueerDB.pm* on kokonaisuudessaan seuraavanlainen:

```
package FinnQueer::Model::FinnQueerDB;

use strict;
use base 'Catalyst::Model::DBIC::Schema';

__PACKAGE__->config(
    schema_class => 'FinnQueerDB',
    connect_info => [
        'dbi:mysql:fq', 'username', 'password',
        { AutoCommit => 1 },
    ],
);
```

Tämä ohjelmakoodi siis perii Catalystin tietokantaolion *Catalyst::Model::DBIC::Schema* ja määrittää tietokannan parametrit. Tässä tapauksessa käytetään järjestelmän MySQL-tietokannan kantaa ”fq” (kuva 5).



Kuva 10: Tietokanta ”fq”

Varsinainen tietokantalogiikka, joka määrittelee taulujen keskinäiset suhteet, sijaitsee FinnQueerDB-hakemiston moduuleissa. Esimerkiksi *Article.pm* määrittelee artikkeleiden suhteen verkkolehden osioihin:

```
package FinnQueerDB::Article;

use base qw/DBIx::Class/;

# Table info
__PACKAGE__->load_components(qw/PK::Auto Core/);
__PACKAGE__->table('articles');
```

```
__PACKAGE__->add_columns(qw/article_id title author
    source create_time modify_time publish_time body
    owner visible/);
__PACKAGE__->set_primary_key(qw/article_id/);

# Relationships:
# One article can be in many sections
__PACKAGE__->has_many(
    article_sections => 'FinnQueerDB::SectionArticle',
   'article_id');

# Many sections can have an article
__PACKAGE__->many_to_many(sections => 'article_sections',
    'section');

1;
```

Tämän moduulin määrittelyt asettavat artikkelit sisältävän taulun pääavaimen, taulun sarakkeet sekä määrittävät kaksi suhdetta:

1. Artikkeli voi sijaita monessa eri osiossa: *Articles*-taulun suhde *sections*-tauluun määritellään luokassa *FinnQueerDB::SectionArticle*, jossa *article*-taulun pääavaimena toimii sarake *article_id*.
2. Yhdessä osiossa voi sijaita monta artikkelia: *Sections*-taulun (joka sisältää osioiden tiedot, kuva 5) suhde *Articles*-tauluun määritellään taulussa *article_sections* ja tätä suhdetta SectionArticle.pm-moduulissa vastaa kohta ”section”.

Artikkelien ja osioiden suhteen määrittävän taulun *article_sections* moduuli *SectionArticle.pm* käyttää Catalystille ominaista tapaa määrittää tietokantasuhde:

```
package FinnQueerDB::SectionArticle;

use base qw/DBIx::Class/;

# Table info
__PACKAGE__->load_components(qw/PK::Auto Core/);
__PACKAGE__->table('section_articles');
__PACKAGE__->add_columns(qw/article_id section_id/);
__PACKAGE__->set_primary_key(qw/article_id section_id/);

# Relationships
__PACKAGE__->belongs_to(article => 'FinnQueerDB::Article',
    'article_id');
__PACKAGE__->belongs_to(section => 'FinnQueerDB::Section',
    'section_id');

1;
```

Tällä tavalla tietokannan *Article*- ja *Section*-olioille on määritelty many-to-many-suhde, jota voidaan käyttää tietokantahauissa sovelluksen sisällä. Esimerkiksi verkkolehden osion etusivulla haetaan aktiivisen osion tiedot:

```
$c->stash->{data} =
    $c->model('FinnQueerDB::Section')->find($section_id);
```

Nämä tiedot ohjataan edelleen mallinnemoottorille muuttujassa "data". Osion etusivulla listataan tämän osion kaikki artikkelit, ja määritellyn many-to-many-suhteen ansiosta tämä lista ja tieto kunkin artikkelin kirjoittajasta sekä julkaisupäivämäärästä on saatavissa suoraan haetusta tietorakenteesta mallineessa *section.tt*:

```
[% FOREACH article = data.articles %]
<a href="/article/[% article.article_id %]">[% article.tit-
le %]</a>
[% article.create_time %] / [% article.author %]<br />
[% END %]
```

5.2.2 Näkymät

Näkymillä määritetään näytettävän tietosisällön lopullinen ulkonäkö.

Template (koko nimeltään siis *Catalyst::View::Template*) on verkkolehden oletusarvoinen näkymäkomponentti ja se vie näytettävän artikkelin Template Toolkit – mallineelle, joka tuottaa siitä selaimelle HTML-koodia. Itse näkymäkomponentin koodi on tässä tapauksessa yksinkertainen:

```
package FinnQueer::View::Template;

use strict;
use base 'Catalyst::View::TT';

__PACKAGE__->config({
        CATALYST_VAR       => 'Catalyst',
        TEMPLATE_EXTENSION => '.tt',
});

1;
```

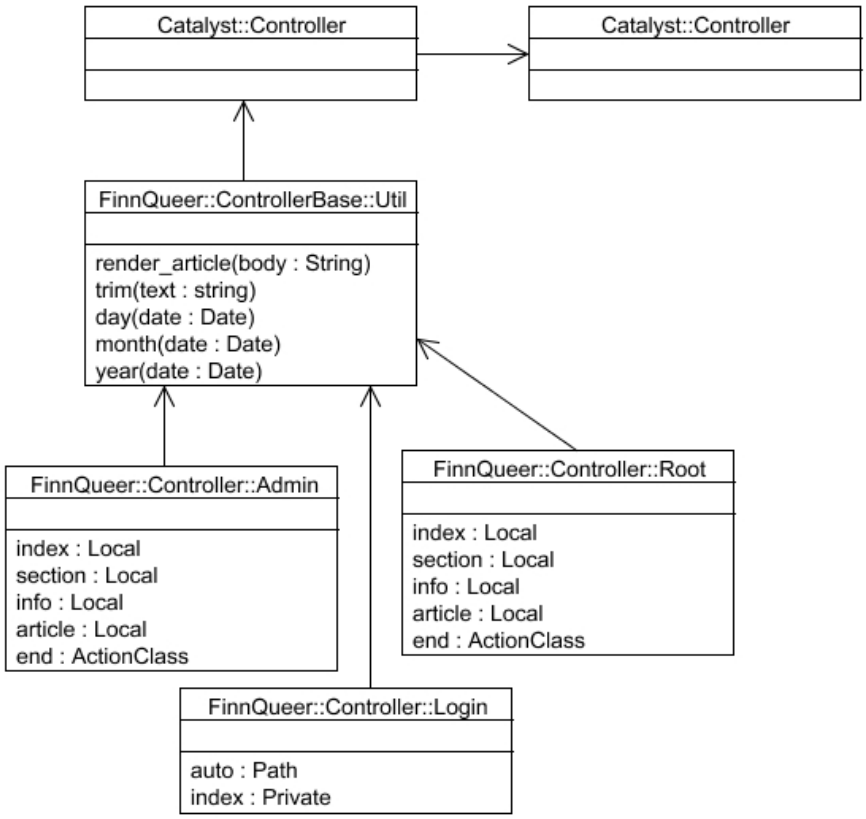
PDF-näkymäkomponentti tuottaa artikkelista Portable Document Format -muotoisen tiedoston, joka soveltuu erityisesti tulostamiseen, sillä siinä voi dokumentille määritellä tarkan ulkoasun.

Mobile on HTML-koodin mobiilipäätelaitteita varten tuottava näkymäkomponentti. Myös se käyttää Template Toolkitiä, mutta valitsee eri mallineet sisällön pohjaksi.

5.2.3 Ohjaimet

Verkkolehden ohjainkomponentit vastaavat jokseenkin käyttöliittymän eri tiloja. Ne käsittelevät käyttäjän lähettämiä parametreja ja suorittavat tietokantahakuja ja ohjaavat sisällön ohjaamista käyttäjälle.

Normaalisti Catalystissa eri ohjaimet perivät suoraan *Catalyst::Controller*-luokan, joka on kaikkien ohjainten perusluokka. Koska FinnQueerissa useat eri ohjaimet tarvitsevat samoja aliohjelmia, nämä yhteiset ohjelmakoodin osat on sijoitettu omaan väliluokkaansa *FinnQueer::ControllerBase::Util*, jonka ohjaimet perivät. Uusi väliluokka perii *Catalyst::Controller*-luokan (kuva 11).



Kuva 11: FinnQueerin ohjainkomponenttien periytyminen

Verkkolehden ohjainkomponentteja vastaavat URL-osoitteen polku - esimerkiksi osoitteessa *http://www.finnqueer.net/admin/article/56* olennainen polku on "*/admin/article/56*", joka tulkitaan siten, että valittu ohjain on *admin*, toiminto on *article* ja tämän parametri on 56.

Root on päätason ohjainkomponentti, joka sisältää seuraavat toiminnot (actions):

- *index* eli koko sivuston etusivu. Käytännössä tästä on edelleenohjaus ensimmäisen osion artikkelilistaukseen.

- *section* eli osion etusivu. Toiminto hakee mallin kautta aktiivisen osion tiedot ja relaation kautta myös kaikki osioon kuuluvat artikkelit.
- *info* vastaa verkkolehden tietosivuista, toimituksen esittelystä jne. Ohjaimeen on kovakoodattu tieto siitä, mitkä artikkelit vastaavat näitä tietosivuja. (Ylläpidon liittymässä tietosivut näkyvät omassa osiossaan, joka ei näy normaalissa näkymässä.)
- *article* on nimensä mukaisesti se toiminto, joka tuottaa yksittäisen artikkelin sisällön selaimelle. Ohjain hakee tietokannasta artikkelin lähdekoodimuotoisena, ajaa sen *render_article*-metodin läpi ja siirtää eteenpäin näkymäkomponentille. Sivulle lisätään myös lista osiosta sekä kymmenestä uusimmasta aktiivisen osion artikkelista.

Login ja **Logout** toteuttavat sisäänkirjautumistoiminnon, joita tarvitaan ylläpidon käyttöliittymässä. Jälkimmäisen ainoa tehtävä on suorittaa Catalyst-kontekstin metodi *logout* ja sen jälkeen ohjata käyttäjä sivuston päätasolle.

Admin on ylläpidon toiminnoista huolehtiva ohjainkomponentti. Sen tuottamiin sivuihin vaaditaan sisäänkirjautuminen - jos käyttäjä ei ole kirjautunut sisään, hänet ohjataan Login-ohjaimen kautta sisäänkirjautumissivulle.

- *section* vastaa osion muokkauksen toiminnallisuudesta.
- *article* mahdollistaa verkkolehden artikkeleiden muokkauksen ja lisäämisen. Toiminnolla on itse asiassa kolme moodia (joista toisen kautta ei välttämättä kuljeta):
  1. Uusi sivu: tiedot haetaan mallista
  2. Esikatselu: sivu ladataan uudestaan mutta tiedot päivitetään lomakkeesta mallin sijaan
  3. Tietojen tallennus: lomakkeen tiedot tallennetaan malliin ja käyttäjä ohjataan takaisin osion muokkaussivulle onnistuneen kanta-operaation jälkeen.
- *user* toteuttaa käyttäjätilien muokkaukseen liittyvän toiminnallisuuden. Tämä toiminto on vain pääkäyttäjän käytettävissä, joten käyttäjän rooli tarkistetaan jokaisen kutsun yhteydessä.
- *library* on verkkokirjaston muokkaustoiminto, jossa säädetään kunkin verkkokirjan tiedot sekä osoitetaan tiedostojärjestelmästä PDF-muotoinen tiedosto, joka käyttäjän annetaan ladata.

Library tuottaa verkkokirjastoon liittyvät toiminnot:

- *list* listaa verkkokirjastoon kuuluvat verkkokirjat ja niiden olennaisimmat tiedot

- *view* tarjoaa mahdollisuuden tarkastella verkkokirjan pitää kuvausta ja listaa linkit kirjan PDF-version lataamiseksi.

Tämän projektin puitteissa ei vielä toteutettu verkkolehden kevytversiota eikä automaattista PDF-tiedostojen luomista artikkeleista, mutta ne olisi mahdollista toteuttaa omana ohjainkomponenttinaan tai *Root*-komponentin toimintoina. MVC-mallin näkökulmasta elegantein tapa olisi tehdä pelkästään uusi näkymäkomponentti, mutta käytännön syistä näytettävä sisältö vaatii jonkin verran käsittelyä eri tarkoituksiin, jolloin nämä uudet näkymät vaatisivat myös uuden ohjaimen.

5.3 Parametrien suodatus

Eräs olennainen verkkosovelluksen toimintaan liittyvä seikka on sen vastaanottamien lomakkeelta tai URL-osoitteesta tulevien parametrien tarkistaminen. XSS- eli Cross Site Scripting -tekniikassa näiden parametrien joukkoon lisätään JavaScript-koodia, joka suoritetaan selaimessa [14].

Yleisluontoinen esimerkki tämäntyyppisestä kiusanteosta voisi olla käyttäjälle näkyvä viesti, joka voitaisiin toteuttaa näin:

http://www.verkkolehti.com/sivu.cgi?sivu=<script>alert('hah hah haa!');</script>

Vaikka tämäntyyppinen kiusanteko ei varsinaisesti muokkaa verkkolehden sisältöä, sen avulla voidaan saada tuotettua sivuja, jotka osoitteen perusteella näyttävät olevan autenttisia verkkolehden sivuja, mutta joiden sisältö ladataan joltain toiselta palvelimelta.

Tämän tyyppiset hyökkäykset voidaan torjua siten, että kaikki hakuun liittyvät parametrit suodatetaan, esimerkiksi seuraavien ohjeiden perusteella:

- Jos parametrin pitäisi olla määrämuotoinen (esimerkiksi numero) niin se pakotetaan tähän muotoon suodattamalla ylimääräiset merkit ja ottamalla vain ensimmäinen merkki huomioon.
- Tekstikentissä merkit < ja > muutetaan muotoon < ja >.
- Myös merkit (,), # ja & kannattaa muuttaa vastaaviksi HTML-entiteeteiksi. [14]

Vaikka verkkolehti FinnQueer ei todennäköisesti ole hyökkääjien listalla kovin korkealla, on silti syytä testauksen yhteydessä kokeilla järjestelmän lomakkeissa erilaisia sisältöjä, jotka voisivat johtaa XSS-haavoittuvuuteen tai muihin ongelmiin.

5.4 Artikkelien muoto

5.4.1 FinnQueerin vanha järjestelmä

FinnQueerin vanhassa julkaisujärjestelmässä artikkelitekstit ovat palvelimen tiedostojärjestelmässä omina tekstitiedostoinaan, kuten myös osioiden artikkelilistaukset ja kuvaukset. Yksi artikkelitiedosto sisältää seuraavat osat:

1. otsikkotiedot
2. ingressi
3. leipätekstiosuus.

Artikkelin ensimmäinen kappale tulkitaan ingressiksi ja sen ensimmäiset 4 riviä näytetään myös osion etusivulla. Tämä voi olla ongelmallista monista eri syistä: jos esimerkiksi artikkelin rivit ovat ylipitkiä, voivat nämä neljä ensimmäistä riviä sisältää useita satoja merkkejä, kokonaisia kappaleita. Ingressin neljä ensimmäistä riviä eivät myöskään välttämättä ole paras mahdollinen mainos osion etusivun artikkelilistaan laitettavaksi, joten erillinen kenttä esittelytekstille on uuden järjestelmän suunnitelmissa.

Esimerkki artikkelitiedoston alkuriveistä:

```
Mitä ovat Yogyakartan periaatteet?
7.4.2008
Mikko Väisänen
yogyakarta.png
Yogyakartan periaatteet ovat johtavien ihmisoikeusjuristien
muotoilemia tasavertaisuusperiaatteita, joille nyt pyritään
saamaan virallinen tunnustus YK:n tasolla. Sitä kautta ne
voivat vaikuttaa jokaisen maan lainsäädäntöön, siten että
```

Otsaketiedoissa on seuraavat tietosisällöt:

1. artikkelin otsikko
2. julkaisupäivämäärä
3. kirjoittaja / lähde
4. aihepiiriin liittyvä ikoni (ei pakollinen)

Itse leipätekstin muotoilusta suurin osa on tehty tekstin sekaan HTML:llä. Väliotsikoiden ja kuvien vaatimien tagien tuottamiseen on käytetty omia ta-

geja. Esimerkiksi kuva.jpg-niminen kuvatiedosto liitetään artikkelin oikeaan reunaan näin:

```
!K v kuva.jpg
Kuvalle pitkä kuvateksti tällä rivillä
```

Vanhan järjestelmän ongelmana oli muun muassa juuri kuvatekstille varatun tilan lyhyys - kuvateksti piti saada mahtumaan kahdelle riville. Uudistetussa verkkolehdessä vanhat koodit on hylätty ja on siirrytty käyttämään valmista avoimen lähdekoodin Markdown-kieltä omin laajennoksin.

5.4.2 Markdown ja MultiMarkdown

Markdown on kevyt avoimen lähdekoodin merkkauskieli, joka on alun perin tarkoitettu muistiinpanojen rakenteelliseen kirjoittamiseen. Se on lähdekoodimuodossaankin varsin luettavaa ja helposti kirjoitettavaa ja siitä voi tuottaa helposti esimerkiksi HTML-koodia.

Yksittäinen teksti ajetaan Markdownin läpi Perl-komennolla

```
$result = $c->markdown->markdown($text);
```

missä $c on Catalyst-olio.

FinnQueerissa käytetään Markdownin viritettyä MultiMarkdown-versiota, joka lisää ominaisuuksiin muun muassa automaattiset lähdeluettelot, alaviitteet ja paremman kuvien käsittelyn. Markdown on määritelty yhdeksi Catalystin liitännäiseksi, jolloin kirjasto on käytössä kaikissa komponenteissa. MultiMarkdown on saatu käyttöön *Catalyst::Plugin::Markdown*-olion toteuttavasta Perl-moduulista muuttamalla Markdown-olion luomisen suorittava rivi muotoon

```
__PACKAGE__->markdown(Text::MultiMarkdown->new);
```

Tämän lisäyksen jälkeen MultiMarkdown toimii normaalin Markdownin tavoin kaikissa yhteyksissä. Markdownista ja MultiMarkdownista on lisätietoja liitteessä 2.

5.4.3 Omat laajennokset

Vaikka Markdown tukeekin kuvia, FinnQueeria varten tarvitaan mahdollisuus (halutun pituiselle) kuvatekstille. Siksi uudistetun FinnQueerin artikkeleissa kuvat merkitään tällä tavalla:

```
??
kuva.jpg
Kuvan pitkä kuvateksti usealla
rivillä *muotoilujen kera*
??
```

Koska myös kuvatekstissä pitää voida käyttää Markdownin muotoiluja, ajetaan Markdown sille erikseen.

"Kainalotyyppiset" lainaukset toteutetaan vastaavasti:

```
""
Pitkä kainaloteksti tässä, **muotoiluja** voi käyttää
ja samoin useaa riviä.
""
```

Itse Markdownin ohjelmakoodiin ei ole kajottu, vaan kaikki poikkeukset sen toimintaan on toteutettu ohjainkomponenttiin *FinnQueer::Controller-Base::Util*, jonka ohjaimet perivät.

5.5 Kehitysympäristön asetukset

Kehitysympäristö edellyttää käyttäjäkohtaista Apache-palvelinohjelmiston instanssia. Catalyst-kehitysympäristö asentuu käyttäjän kotihakemiston alle ja lopulta muodostuvat seuraavanlaiset hakemistot:

/FinnQueer	päähakemisto
/FinnQueer/lib	komponenttimoduulit
/FinnQueer/lib/FinnQueer	MVC-komponentit
/FinnQueer/lib/FinnQueer/ControllerBase	ohjainluokkien yläluokat, joissa yleiskäyttöisiä metodeja
/FinnQueer/lib/FinnQueer/Controller	sovelluksen ohjainkomponenttien luokat
/FinnQueer/lib/FinnQueer/Model	sovelluksen malliluokat
/FinnQueer/lib/FinnQueer/View	sovelluksen näkymäluokat
/FinnQueer/lib/FinnQueerDB	tietokannan taulujen määritysluokat
/FinnQueer/root	web-palvelimen juurihakemisto, josta löytyvät päätason mallineet, alihakemistoista (kuten admin) eri ohjainten mallineet
/FinnQueer/t	tiedostoja automaattisia testitapauksia varten

Näiden lisäksi juuressa on hakemistot *pic* ja *styles*, jotka sisältävät käyttöliittymän tarvitsemat kuvat ja tyylitiedostot.

5.6 Varmuuskopiot ja varmentaminen

Palvelin, jolla julkaisujärjestelmää ajetaan, käyttää kahta SATA-kiintolevyä, jotka on konfiguroitu RAID1-pakaksi. Tämä parantaa järjestelmän vikasietoisuutta, sillä palvelin voi jatkaa normaalia toimintaansa, vaikka toinen kiintolevyistä lakkaisi toimimasta. Tällöin verkkolehden tiedot voitaisiin väliaikaisesti siirtää toiselle palvelimelle siksi aikaa, kun pääpalvelinta huolletaan.

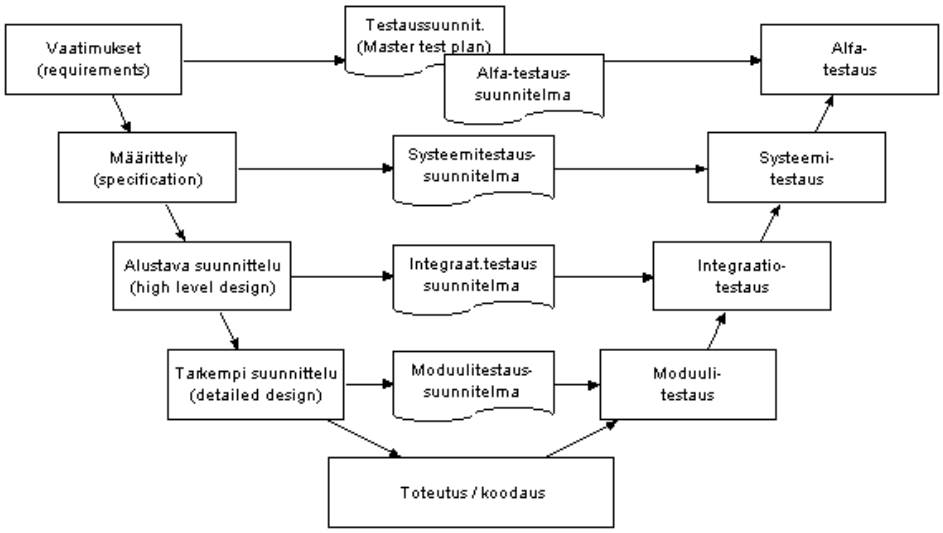
Palvelin myös suorittaa kerran vuorokaudessa tietokannan kopioinnin hakemistoon, josta sen voi manuaalisesti noutaa toiselle tietokoneelle varastointia varten.

6 JULKAISUJÄRJESTELMÄN TESTAUS

Ohjelmistotestauksen tavoite on varmistua siitä, että ohjelmisto toimii oikein sekä löytää mahdolliset virheet ja ongelmat. Käyttöliittymän sisältävän ohjelmiston, jollaisia verkkosovellukset yleensä ovat, testaukseen sisältyy käyttöliittymätapausten läpikäynti ja erilaiset testikäyttäjät (oikeat ihmiset tai tekoälyllä toimivat kuormitustestauksissa käytettävät virtuaalikäyttäjät).

6.1 Verkkojulkaisujärjestelmän testausmallit

Ohjelmistojen testaus jaetaan yleensä moduulitestaukseen, integraatiotestaukseen ja järjestelmätestaukseen. Kun käydään läpi järjestelmän vaatimusmäärittelyä testaussuunnitelman muodossa, kyseessä on lähinnä järjestelmätestaus, joka sopiikin hyvin verkkosovellukselle.



Kuva 12: V-tyyppinen ohjelmistokehitys [12]

Testausprosessin suhdetta ohjelmointiprosessiin kuvataan usein V-mallisella kuviolla ohjelmistotuotantoprosessista (kuva 12). Siinä kutakin määrittelyvaihetta vastaa omanlaisensa testausmalli, mikä on omiaan suurissa sovelluksissa, joiden määrittelyprosessissa on useita vaiheita ja jonka toimivuus tulee todeta monella eri tasolla.

Tämän insinöörityön tapauksessa ohjelmistokehitysprosessi ei ole ollut yhtä monisyinen, vaan ohjauksen yhteydessä on prototyyppejä testattu käytännössä ja iteroitu toteutusta. Käyttötapauksia sekä ohjelmistokomponenttien toimintaa voidaan silti käydä läpi testaussuunnitelman pohjalta. Kuvan 12 prosessissa kyseessä olisi lähinnä systeemitestaus, poislukien ne kohdat, joissa toteutetaan ohjelmallista moduulitestausta.

Kun kyseessä on verkkosovellus, olennaista on järjestelmän vasteaika käyttäjälle. Vasteaikaan kuitenkin vaikuttavat myös erilaiset verkon häiriöt ja tiedonsiirtonopeudet. Siten kattavan kuvan saaminen järjestelmän suorituskyvystä on hyvin vaikeaa. Eräs ratkaisu on LoadRunnerin kaltaisten kuormitustestausohjelmistojen käyttö. Niiden avulla voidaan ohjelmoida erilaisia virtuaalikäyttäjiä, jotka selailevat järjestelmän käyttötapaukset läpi ja joiden määrä voidaan valita testitapauksen mukaan. Näin voidaan simuloida myös harvinaisia virhetilanteita, joiden toistaminen edellyttää suurta kuormitusta ja lukuisia (jopa tuhansia) samanaikaisia käyttäjiä. [26]

6.2 Testaussuunnitelma

FinnQueerin julkaisujärjestelmän testitapaukset voidaan jaksaa kahteen ryhmään:

1. Toiminnallinen testaus: ohjelmiston käyttötapaukset ja toiminta käydään läpi testaussuunnitelman pohjalta ja jokaisen toiminnon oikea toiminta varmistetaan.
2. Ohjelmallinen testaus: ohjelmiston moduulien toiminteita käytetään ohjelmakoodista. Pitää sisällään myös kuormitustestauksen, joka toteutetaan ohjelmallisesti.

Näistä toiminnallinen testaus tarkoittaa käytännössä ohjelmiston toimintojen läpikäyntiä testaussuunnitelman ja käyttötapauksien pohjalta.

Luvussa 4.1 esitetystä vaatimusmäärittelystä voidaan johtaa testaussuunnitelma, jonka läpikäymällä saadaan arvio julkaisujärjestelmän toimivuudesta

ja mahdollisista ongelmista. Suunnitelman kuhunkin kohtaan liitetään testauksen tulos, joka on ei toimi / toimii osittain / toimii.

1. Apache-palvelinohjelmisto suorittaa käynnistyessään automaattisesti myös FinnQueer-julkaisujärjestelmän ja tarjoaa sen osoitteessa http://www.finnqueer.net/ portissa 80 (testiympäristössä portti 30302).
2. Tilat: käyttöliittymän eri tilat (kuvat 5 ja 6) on toteutettu ja näyttävät tarkoitetun sisällön.
3. Siirtymät: käyttöliittymän siirtymät (kuvat 5 ja 6) toimivat ja välittävät mahdolliset istuntoparametrit (käyttöliittymän kieli) oikein.
4. Ylläpitäjä voi luoda, muokata ja poistaa artikkeleita.
5. Ylläpitäjä voi luoda, muokata ja poistaa osioita.
6. Ylläpitäjä voi luoda, muokata ja poistaa käyttäjiä.
7. Sisäänkirjautuminen ylläpidon käyttöliittymään toimii: luo istunnon ja asettaa evästeen.
8. Uloskirjautuminen toimii ja poistaa evästeen.
9. Käyttäjä, joka ei ole sisäänkirjautunut, ei pääse ylläpidon käyttöliittymään.
10. Käyttäjä, joka on sisäänkirjautunut, mutta jolla ei ole sopivaa roolia, ei pääse ylläpidon käyttöliittymään.
11. Järjestelmän pystyy palvelemaan jatkuvasti 10 artikkelisivua kuvineen sekunnissa.
12. Peruskäyttäjän käyttöliittymä toimii normaalisti, vaikka käytössä ei olisi JavaScriptiä tai evästeitä.
13. Jokainen selaimessa näytettävä sivu on säännönmukainen (validi) XHTML-dokumentti.
14. Määritellyt siirtymät eri näkymiin (PDF, mobiili) toimivat ja näyttävät sisällön määritellyssä muodossa.
15. Käyttöliittymän kielen vaihto onnistuu.
16. Minkään URL-parametrin käsittely ei sisällä XSS-haavoittuvuutta.

Tämän käytännönläheisen lähestymisen tarkoitus on, että loppukäyttäjän näkökulmasta sovellus toimii johdonmukaisesti ja oikein.

6.3 Testauksen tulokset

Testauksen tuloksena pitäisi olla paitsi yleinen käsitys järjestelmän toiminnasta ja toimivuudesta, myös kommentteja ja kritiikkiä siitä miten toimiva

systeemi vastaa oletuksia ja testitapauksia, ja mitä voisi parantaa vaikka toimisikin.

Testauksen voi aloittaa jo ohjelmiston ollessa edelleen toteutusvaiheessa, vaikka ohjelmisto olisi keskeneräinenkin. Ohjelmiston tai sen osien toiminta voi olla jo lähellä lopullista ja näin voidaan ongelmiin puuttua jo ennen kuin ohjelmiston ensimmäiset testiversiot toimitetaan asiakkaalle testattavaksi.

FinnQueerin tapauksessa kaikkia toimintoja ei toteutettu valmiiksi asti tämän insinöörityöprojektin puitteissa, mikä näkyy testauksen tuloksista. Julkaisujärjestelmän prototyyppi kuitenkin toimii ja sitä voidaan käyttää jo olemassa olevien artikkelien viimeistelyyn ennen uuden version julkaisua.

1	(ei toteutettu)	**9**	toimii
2	toimii osittain	**10**	ei toimi
3	toimii osittain	**11**	(ei testattu)
4	toimii	**12**	toimii
5	toimii	**13**	toimii
6	toimii osittain	**14**	(ei toteutettu)
7	toimii	**15**	(ei toteutettu)
8	toimii	**16**	toimii

7 JOHTOPÄÄTÖKSET

Työn tarkoituksena oli suunnitella ja toteuttaa verkkolehdelle julkaisujärjestelmä, joka mahdollistaisi toimittajajoukolle sen helpon ja käytännöllisen ylläpidon ja tuottaisi lehden lukijalle sisällön nopeasti ja erilaisiin tilanteisiin (selain, mobiililaitteet, tulostus jne.).

MVC-arkkitehtuurimalli soveltuu hyvin verkkosovellusten kehittämiseen ja sitä toteuttava Catalyst-sovelluskehys on soveltuva alusta verkkolehden julkaisujärjestelmän kehittämiseen ja myös muuhun tuotantokäyttöön. MVC-ajattelu tarjoaa verkkosovelluksen kehittäjälle mahdollisuuksia toteuttaa ohjelmiston komponentteja modulaarisesti ja useissa tapauksissa se tukee olioajattelua erinomaisesti.

Dynaamisten web-sivujen mallineiden rakentamiseen Template Toolkit on sovelias työkalu ja se integroituu Catalystiin hyvin.

Verkkolehden artikkelien kuvaamiseen tarvitaan jokin merkkauskieli ja vaihtoehtoina ovat muun muassa raaka HTML, XML, jokin oma tekstipohjainen

kuvauskieli tai Markdown. Kun järjestelmän suunnittelukriteerinä on toteuttamisen ja toimittamisen keveys, on Markdownin kaltainen kevyt merkkauskieli soveltuva, mutta sen ominaisuuksien laajentaminen vaatii julkaisujärjestelmältä logiikkaa, jotta verkkolehdelle ominaiset taittoratkaisut ovat mahdollisia.

Verkkosovelluksen testaukseen käytännönläheinen, käyttötapauksiin perustuva testaussuunnitelma on sopiva, mutta moduulitestaus on haastavampaa. Tietoturvan kannalta on kiinnitettävä huomiota yleisimpiin tietoturvaongelmiin ja URL-parametrien suodatukseen, sillä ongelmien ja niiden ratkaisujen triviaalius ei estä esimerkiksi rikollisia kokeilemasta kaikkia mahdollisia aukkoja ja löytämästä huolimattomasti toteutetusta järjestelmästä aukkoja, jotka mahdollistavat vähintäänkin kiusanteon.

Kehitystyössä web-sovelluskehystä käyttäen työn painopiste siirtyy pelkästä ohjelmoinnista järjestelmäsuunnittelun suuntaan, mikä on hyvä asia. Sovelluskehyksen hoitaessa monet arkiset operaatiot on varsinkin sovelluksen koon kasvaessa tärkeää kiinnittää huomiota tietokantasuunniteluun ja sovelluksen jakamiseen komponentteihin ja näiden komponenttien väliseen vuorovaikutukseen.

VIITELUETTELO

[1] Stålström, Olli - Nissinen, Jussi, The Spitzer Study and the Finnish Parliament. *Journal of Gay and Lesbian Psychotherapy 3*, 2003.

[2] Net Applications. *Browser Market Share for February, 2008* [verkkodokumentti] 2008 [viitattu 25.3.2008]. Saatavissa: http://marketshare.hitslink.com/report.aspx?qprid=0.

[3] Wise, Rosemarie, *Client or Server Side Scripting, What's the Difference?* [verkkodokumentti]. 25.12.2005 [viitattu 20.4.2008]. Saatavissa: http://websiteowner.info/articles/cgi/whichside.asp.

[4] Korpela, Jukka K. - Linjama, Tero, *Web-suunnittelu*. Jyväskylä: Docendo. 2005.

[5] Koskimies, Kai - Mikkonen, Tommi, *Ohjelmistoarkkitehtuurit*. Helsinki: Talentum. 2005.

[6] Wikipedia. *Model-view-controller* [verkkodokumentti] päivitetty 1.1.2008 [viitattu 2.1.2008]. Saatavissa: http://en.wikipedia.org/wiki/Model-view-controller

[7] Wardley, Andy, *Template Toolkit Documentation* [verkkodokumentti] 2007, päivitetty 16.8.2007 [viitattu 15.11.2007]. Saatavissa: http://template-toolkit.org/docs/.

[8] Keränen, Vesa - Lamberg, Niko - Penttinen, Jukka, *Web-julkaiseminen & multimedia*. Jyväskylä: Docendo. 2006.

[9] Vainionpää, Veijo, N*äkökohtia tehokkaan web-sovelluksen suunnitteluun* [verkkodokumentti] Insinöörityö, Helsingin ammattikorkeakoulu Stadia, 2002 [viitattu 21.12.2007]. Saatavissa: http://www.stadia.fi/palvelut/julkaisutoiminta/c-sarja/Opinnaytetyöt_5_verkkojulkaisu.pdf.

[10] Vesterinen, Laura, *MVC-arkkitehtuurin toteuttaminen suunnittelumalleja käyttäen*. Insinöörityö. Helsingin ammattikorkeakoulu Stadia. 2007. Saatavissa: https://oa.doria.fi/handle/10024/5550.

[11] Vainio, Ville, *Enterprise JavaBeans -komponenttimalli ja sen vaikutus ohjelmistonkehitykseen*. Pro gradu -tutkielma. Jyväskylän yliopisto. 2001. Saatavissa: http://urn.fi/URN:NBN:fi:jyu-2001864605.

[12] Kautto, Tuomas, *Ohjelmistotestaus ja siinä käytettävät työkalut* [verkkodokumentti]. 21.11.1996 [viitattu 20.4.2008]. Saatavissa: http://www.mit.jyu.fi/opiskelu/seminaarit/ohjelmistotekniikka/testaus/.

[13] *HP LoadRunner data sheet* [verkkodokumentti]. 2008 [viitattu 20.4.2008]. Saatavissa: https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-126-17^8_4000_100__.

[14] CGI Security. *Cross Site Scripting (XSS) questions and answers* [verkkodokumentti]. 2003 [viitattu 21.4.2008]. Saatavissa: http://www.cgisecurity.com/articles/xss-faq.shtml.

LIITE 1: KÄYTTÖLIITTYMÄTAPAUKSET

Peruskäyttäjän käyttöliittymän tilat

FQS001 Osion etusivu	
Suorittajat	Kaikki käyttäjät
Sisältö	Osioon kuuluvien artikkelien listaus

FQS002 Artikkeli	
Suorittajat	Kaikki käyttäjät
Sisältö	Osioon kuuluvien artikkelien listaus

FQS003 Tietosivu	
Suorittajat	Kaikki käyttäjät
Sisältö	Osioon kuuluvien artikkelien listaus

FQS004 Verkkokirjasto	
Suorittajat	Kaikki käyttäjät
Sisältö	Verkkokirjaston ladattavien tiedostojen ja niihin liittyen tietojen listaus

Peruskäyttäjän käyttöliittymän siirtymät

FQU001 Kielen valinta	
Suorittajat	Kaikki käyttäjät
Aloitustilanne	Mikä tahansa sivu paitsi tulostusnäkymä
Kulku	Käyttäjä klikkaa valikosta haluamansa kielen
Lopputilanne	Käyttöliittymä muuttuu halutunkieliseksi ja käyttäjä saa eteensä kyseiselle kielelle määritellyn ensimmäisen osion artikkelilistauksen

FQU002 Siirtyminen osion etusivulle	
Suorittajat	Kaikki käyttäjät
Aloitustilanne	Osion etusivu tai artikkelinäkymä
Kulku	Käyttäjä klikkaa valikosta haluamansa otsikon nimeä
Lopputilanne	Siirrytään halutun osion etusivulle ja näytetään sen artikkelilistaus
Poikkeukset	(Osiolistauksessa on vain saman kielen avoimet osiot)

FQU003 Siirtyminen Verkkokirjastoon	
Suorittajat	Kaikki käyttäjät
Aloitustilanne	Mikä tahansa sivu paitsi tulostusnäkymä
Kulku	Käyttäjä klikkaa sivulla olevaa Verkkokirjasto-linkkiä
Lopputilanne	Siirrytään Verkkokirjaston etusivulle ja näytetään sen sisältämät tiedot

FQU004 Siirtyminen tietosivuille	
Suorittajat	Kaikki käyttäjät
Aloitustilanne	Mikä tahansa sivu paitsi tulostusnäkymä
Kulku	Käyttäjä klikkaa sivun yläpalkissa olevaa tietosivun otsikkolinkkiä
Lopputilanne	Siirrytään tietosivuille ja näytetään sen artikkeli

FQU005 Siirtyminen artikkeliin artikkelilistauksesta	
Suorittajat	Kaikki käyttäjät
Aloitustilanne	Osion etusivu
Kulku	Käyttäjä klikkaa artikkelilistauksesta artikkelia vastaavaa otsikkoa, ikonia tai Lue lisää -linkkiä
Lopputilanne	Siirrytään artikkelisivulle ja näytetään artikkeli

FQU006 Siirtyminen artikkeliin artikkelisivun artikkelilistauksesta	
Suorittajat	Kaikki käyttäjät
Aloitustilanne	Artikkelisivu
Kulku	Käyttäjä klikkaa artikkelilistauksesta artikkelia vastaavaa otsikkoa
Lopputilanne	Siirrytään kyseiselle artikkelisivulle

FQU007 Artikkelin tulostus	
Suorittajat	Kaikki käyttäjät
Aloitustilanne	Artikkelisivu
Kulku	Käyttäjä klikkaa valikosta Tulosta-linkkiä
Lopputilanne	Artikkeli näytetään uudessa ikkunassa ilman valikoita ja annetaan mahdollisuus sen tulostamiseen

FQU008 Siirtyminen tietosivulta toiselle tietosivulle	
Suorittajat	Kaikki käyttäjät
Aloitustilanne	Tietosivu
Kulku	Käyttäjä klikkaa valikosta haluamansa tietosivun nimeä
Lopputilanne	Näytetään haluttu tietosivu

Ylläpitäjän käyttöliittymän tilat

FQS101 Ylläpidon etusivu	
Suorittajat	Toimittajat, ylläpitäjät, pääkäyttäjät
Sisältö	Lista kaikista osiosta; lista kaikista käyttäjätileistä

FQS102 Osion muokkaus	
Suorittajat	Toimittajat (vain artikkelin valinta), ylläpitäjät, pääkäyttäjät
Sisältö	Osioon kuuluvien artikkelien listaus, mahdollisuus luoda uusi artikkeli

FQS103 Artikkelin muokkaus	
Suorittajat	Toimittajat (vain tietyt artikkelit), ylläpitäjät, pääkäyttäjät
Sisältö	Artikkelin sisältö ja parametrit muokkauskentissä

FQS104 Käyttäjän muokkaus	
Suorittajat	Pääkäyttäjät
Sisältö	Käyttäjätilin tietojen listaus muokkauskentissä

FQS105 Verkkokirjan muokkaus	
Suorittajat	Ylläpitäjät, pääkäyttäjät
Sisältö	Verkkokirjaston ladattavien tiedostojen ja niihin liittyvien tietojen listaus muokkauskentissä

Ylläpitäjän käyttöliittymän siirtymät

FQU101 Siirtyminen osion muokkaukseen	
Suorittajat	Toimittajat, ylläpitäjät, pääkäyttäjät
Aloitustilanne	Ylläpidon etusivu
Kulku	Käyttäjä klikkaa osiolistauksesta haluamansa osion
Lopputilanne	Käyttöliittymä listaa halutun osion asetukset ja antaa mahdollisuuden muokata niitä sekä listaa osion artikkelit

FQU102 Siirtyminen artikkelin muokkaukseen	
Suorittajat	Toimittajat, ylläpitäjät, pääkäyttäjät
Aloitustilanne	Osion muokkaussivu
Kulku	Käyttäjä klikkaa artikkelilistauksesta haluamansa artikkelin otsikkoa
Lopputilanne	Käyttöliittymä listaa halutun artikkelin tiedot ja antaa mahdollisuuden muokata niitä
Poikkeukset	Toimittajat saavat muokata ainoastaan "omistamiaan" artikkeleita

FQU103 Uuden artikkelin luominen	
Suorittajat	Toimittajat, ylläpitäjät, pääkäyttäjät
Aloitustilanne	Osion muokkaussivu
Kulku	Käyttäjä klikkaa sivulla olevaa Uusi artikkeli -linkkiä
Lopputilanne	Käyttöliittymä näyttää tyhjän artikkelin muokkaussivun

FQU105 Osion tallentaminen	
Suorittajat	Ylläpitäjät, pääkäyttäjät
Aloitustilanne	Osion muokkaussivu
Kulku	Käyttäjä klikkaa sivulla olevaa "Tallenna muutokset" -painiketta
Lopputilanne	Tallennetaan osion muutokset ja palataan ylläpidon etusivulle

FQU106 Käyttäjän muokkaaminen/luominen	
Suorittajat	Pääkäyttäjät
Aloitustilanne	Ylläpidon etusivu
Kulku	Käyttäjä klikkaa käyttäjätililistauksesta haluamansa käyttäjän nimeä
Lopputilanne	Siirrytään muokkaussivulle ja näytetään käyttäjätilin tiedot

FQU107 Käyttäjän tallentaminen	
Suorittajat	Pääkäyttäjät
Aloitustilanne	Käyttäjän muokkaussivu
Kulku	Käyttäjä klikkaa sivulla olevaa "Tallenna muutokset" -painiketta
Lopputilanne	Siirrytään ylläpidon etusivulle

FQU108 Verkkokirjan muokkaaminen/luominen	
Suorittajat	Ylläpitäjät, pääkäyttäjät
Aloitustilanne	Ylläpidon etusivu
Kulku	Käyttäjä klikkaa verkkokirjalistauksesta haluamansa verkkokirjan nimeä
Lopputilanne	Siirrytään verkkokirjan muokkaussivulle ja näytetään verkkokirjan tiedot

FQU109 Verkkokirjan tallentaminen	
Suorittajat	Ylläpitäjät, pääkäyttäjät
Aloitustilanne	Verkkokirjan muokkaussivu
Kulku	Käyttäjä klikkaa sivulla olevaa "Tallenna muutokset" -painiketta
Lopputilanne	Siirrytään ylläpidon etusivulle

FQU110 Osion luominen	
Suorittajat	Ylläpitäjät, pääkäyttäjät
Aloitustilanne	Ylläpidon etusivu
Kulku	Käyttäjä klikkaa sivulla olevaa "Uusi osio" -linkkiä
Lopputilanne	Siirrytään osion muokkaussivulle

LIITE 2: MARKDOWN JA MULTIMARKDOWN

Markdown on John Gruberin avoimen lähdekoodin merkkauskieli ja siihen liittyvä Perl-moduuli. Markdown soveltuu paitsi muistiinpanojen tekemiseen, myös tekstin muuntamiseen HTML:ksi ja muihin muotoihin. Fletcher T. Penneyn MultiMarkdown laajentaa kieltä usein tavoin ja lisää siihen tuen muun muassa lähdeviitteille, metatiedoille ja alaviitteille.

Markdownin perusmuotoilu tapahtuu seuraavankaltaisesti:

```
Tekstin seassa voi *kursivoida*, **boldata**.
```

Otsikot merkitään taas näin:

```
Pääotsikko          tai     # Pääotsikko
==========

Alaotsikko          tai     ## Alaotsikko
----------

                            ### Kolmannen tason otsikko
```

Myös erilaiset listat luodaan tekstipohjaisesta muotoilusta:

```
- Listan ensimmäinen kohta
- Listan toinen kohta
    1. Sisennetty järjestetty lista
    2. Järjestetyn listan toinen kohta
- Listan kolmas kohta ja [linkki](http://www.finnqueer.net)
  FinnQueer-verkkolehteen
```

Verkkolehdessä on lainauksia esimerkiksi tieteellisen kommentoinnin yhteydessä. Käytettäessä <blockquote>-tagia täytyy muistaa kappaleenjaot, mikä FinnQueerin vanhassa järjestelmässä tuotti ongelmia HTML:n generoinnin kanssa. Markdownissa sisennetty lainaus tuotetaan näin:

```
Sitten hän sanoi:

> ”Olen aina ollut tätä mieltä ja olen edelleen. Joskus
> tuntuu siltä, että olen vielä tulevaisuudessakin.”

Tätä voidaan pitää erikoisena kommenttina.
```

FinnQueerin kannalta MultiMarkdownin merkittävimmät lisäykset ovat tuki alaviitteille ja viiteluetteloille. Otsaketietoja ei tarvita, sillä vastaavat tiedot löytyvät FinnQueerin artikkelitietokannasta. MultiMarkdownin tuki viiteluettelolle toimii samaan tapaan kuin sen tuki dokumentin sisäisille ankkureille ja linkeille:

```
Väite, joka vaatii seurakseen luotettavan lähteen [Sten-
bäck, s. 26][#Ste:1971]...

Viiteluettelo
-------------

[#Ste:1971] Stenbäck, Håkan, *Alan auktoriteettiopus*.
1971. Helsinki: Kirjapaino.
```

Alaviitteet toimivat lisäämällä tekstiin tageja seuraavaan tapaan:

```
Asia on nyt niin, että hämähäkkejä ei saada[^alaviite].

[^alaviite]: Eikä saada myöskään tulevaisuudessa.
```

Luonnollisesti Markdown-koodin seassa voi olla HTML-tageja ja FinnQueerissa muun muassa kuvia varten generoidaan erikseen HTML-koodia, joka ei millään tavoin haittaa artikkelin muuntamista Markdownin avulla.

Markdownin kotisivu: http://daringfireball.net/projects/markdown/

MultiMarkdownin kotisivu: http://fletcherpenney.net/MultiMarkdown