Opinnäytetyö AMK

Tietojenkäsittelyn koulutusohjelma

Multimediatuotanto

2011

Jari Kauppila

LOMAKEMODUULI MODX-PÄIVITYSJÄRJESTELMÄÄN

– graafinen lomakkeenluontityökalu

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

Turun ammattikorkeakoulu

Tietojenkäsittely | Multimediatuotanto

Marraskuu 2011 | Sivumäärä: 48

Ohjaaja: Pasi Iivonen

Jari Kauppila

LOMAKEMODUULI MODX – PÄIVITYSJÄRJESTELMÄÄN

Työssä toteutettiin graafinen lomakkeidenluontityökalu toimeksiantona MODx -päivitysjärjestelmään. Aluksi työssä käydään läpi internetin kehitystä ohjelmistoalustaksi ja tekijöitä, jotka ovat vaikuttaneet tällaiseen kehitykseen; muun muassa kaupallista kilpailua, ja siihen liittyvää selainten kehitystä. Työssä pohditaan myös lyhyesti web-yhteisöjen roolia avoimen lähdekoodin sovellusten kehityksessä ja avoimen lähdekoodin merkitystä osana yritysten liiketoimintaa.

Työssä esitellään lyhyesti MODx –järjestelmän peruskäyttö ja sivuston päivitys, sekä moduulin asennus järjestelmään. Lomakemoduulin toteutuksessa kerrotaan moduulin suunnittelusta, ja käydään läpi moduulin toiminnot. Teknisestä toiminnasta kuvataan moduulin tietojen käsittely olennaisilta osin, sekä tietokantaan tallennettavat arvot ja kentät. Moduulin ”raahaus ja pudotus” –käyttöliittymän toteutus käydään läpi omassa kappaleessaan.

Teknisessä osassa määriteltyjen teknisten ja toiminnallisten vaatimusten pohjalta toteutettiin järjestelmään toimiva moduuli.

ASIASANAT:

Sisällönhallinta, päivitys, verkko-ohjelmointi, avoin lähdekoodi

BACHELOR´S THESIS | ABSTRACT
TURKU UNIVERSITY OF APPLIED SCIENCES

Infromation Technology | Multimediaproduction

November 2011 | Total number of pages: 48

Instructor: Pasi Iivonen

Jari Kauppila

MODULE FOR MODX CONTENT MANAGEMENT SYSTEM – FORM CREATION TOOL

This work is about graphic form creation tool for MODx Content Management System. Initially, the work will go through the development of Internet towards software platform and the factors that have affected this kind of development: commercial competition, and the associated development of browsers. The work will also briefly go through the role of web- communities and open source development importance as part of the business.

The work goes trough basic usage of MODx: content editing and basics of designing MODx -template. The work also runs through installation process of a module. The technical operation of the module describes the processing of data. “Production of the module” -section includes planning, functions and database design. Module uses so called “drag & drop –interface” and it goes through its own chapter.

Module was finally implemented based on the technical and operational requirements introduced at the start of the planning process.

KEYWORDS:

Content management system, web development, open source, browser development

SISÄLTÖ

LIITTEET

KUVAT

SANASTOA

Content Management System = Julkaisujärjestelmä

Template = mallipohja, malli

Julkaisujärjestelmällä (Content Management System) tarkoitetaan tässä palvelinsovellusta, jolla voidaan muokata internetsivustojen sisältöä, ulkoasua ja toiminnallisuutta. Julkaisujärjestelmä on tarkoitettu nopeuttamaan sisällön ylläpitoa ja päivitystä, sekä helpottamaan uuden sisällön tuotantoa.

CMF (Content Management Framework)

Framework = puitteet, puiteohjelma

CMF:llä tarkoitetaan yleisesti julkaisujärjestelmää, johon on tehty valmis API (Application Programming Interface). API mahdollistaa järjestelmän liittämisen ulkoisiin moduuleihin tarjoten valmiit liittymät järjestelmän toimintojen käyttöön.

EForm = MODx –julkaisujärjestelmään kehitetty HTML-lomakkeiden syötteentarkistustyökalu

EForm on MODxiin kehitetty laajennos, joka tarkistaa lomakekenttiin syötetyt tiedot mm. kentissä käytettyjen merkkien oikeellisuuden ja puutteellisuuden palvelimella siinä vaiheessa, kun loppukäyttäjä on syöttänyt tiedot lomakkeelle ja painanut lomakkeen lähetys- tai tallennusnappia.

JavaScript

JavaScript on laajalti selaimissa käytetty ohjelmointikieli. JavaScriptillä toteutetaan tässä työssä tehtävän moduulin käyttöliittymässä mm. HTML-elementtien raahaus- ja pudotusominaisuudet, sekä HTML-lomake-elementtien luominen.

HTML 4.01 ja HTML DOM (Document Object Model)

HTML 4.01 eli Hyper Text Markup Language on dokumenttien merkkauskieli (World Wide Web Consortium -standardi) lähinnä internetselaimia varten

JQuery

JQuery on JavaScript –kirjasto ja ns. Framework (puiteohjelma) joka nopeuttaa ja yksinkertaistaa JavaScriptin DOM -ominaisuuksien käyttöä tuotannossa mm. vähentäen koodin kirjoittamisen tarvetta.

PHP (Hypertext Preprocessor)

PHP on laajalti palvelinympäristöissä käytetty ohjelmointikieli. PHP vaatii oman tukensa asentamista palvelimelle.

MySQL

MySQL on relaatiotietokantaohjelmisto, jota MODx käyttää tietojen tallentamiseen.

1 JOHDANTO

Avoimen lähdekoodin päivitysjärjestelmät ja ohjelmistokehykset toimivat usein alustana Internetsuunnittelua ja tuotantoa tarjoavan yrityksen ohjelmistoratkaisuissa. Tämä työ tehtiin toimeksiantona Mediascope Oy:lle, joka käyttää tällaista järjestelmää yhtenä internetsivustojen päivitykseen liittyvistä ratkaisuistaan. Aiheena oli toteuttaa MODx-päivitysjärjestelmään graafinen työkalu, jolla voidaan muokata ja luoda uusia web-lomakkeita. Vastaavissa muissa päivitysjärjestelmissä tämän tyyppisiä työkaluja oli jo olemassa, mutta tästä järjestelmästä kyseinen ominaisuus puuttui.

Pohdin aiheen taustaksi myös hieman sitä, millaiset asiat ovat mahdollistaneet avoimen lähdekoodin päivitysjärjestelmien kehityksen. Päivitysjärjestelmien kehittymistä sinänsä on edesauttanut, paitsi internetin tekninen kehitys, myös erilaiset internetyhteisöt. Internetyhteisöistä ja avoimen lähdekoodin järjestelmästä ovat hyötyneet PK–sektorin lisäksi myös suuryritykset, sekä ratkaisujen käyttäjänä että kehittäjänä. Pohdin myös lyhyesti sitä, millä tavalla nämä järjestelmät ja yhteisöt voivat luoda uusia liiketoimintamahdollisuuksia.

Kuvailen työssä MODx -päivitysjärjestelmän perusasennuksen ja yleisimpien ominaisuuksien käytön pääpiirteissään, sekä moduulin asennukseen vaadittavat valmistelut. ”Käytetyt tekniikat ja sanastoa”–osiossa on kuvailtu työssä käytetyt tekniikat. Moduulin toteutus kuvataan luvussa 4.

INTERNETIN KEHITYS OHJELMISTOALUSTAKSI

1.1.1 Internet kehittyy interaktiiviseksi

Internetin (World Wide Web) ideana on alusta alkaen ollut tiedon jakaminen ja päivittäminen. Internetiä ei alun perin ajateltu varsinaisesti ohjelmistoalustaksi vaan html-dokumenttien (lähinnä tekstidokumenttien) jakoon ja niiden päivittämiseen.

Alun perin internetsivuilla julkaistava sisältö kirjoitettiin tekstitiedostoon html-merkkausta apuna käyttäen ja siirrettiin palvelimelle. Tiedon lukemiseen palvelimelta tarvittiin selainohjelma, kuten nykyäänkin. Selaimia oli aluksi muutamia erilaisia eri käyttöjärjestelmille. Selaimet eivät pystyneet esittämään html-dokumenteissa kuvamateriaalia, eivätkä muita multimediaelementtejä, kuten liikkuvaa kuvaa tai ääntä. Kuvat oli linkitettävä palvelimella olevaan tiedostoon ja näytettävä erikseen omassa ikkunassaan. (Calore, M. 2011.)

Internetin suosio laajeni 1990-luvun alkupuolella selainten ominaisuuksien parantuessa. Suuri edistysaskel ja netin suosion kasvuun vaikuttanut tekijä koettiin vuonna 1993, kun ryhmä opiskelijoita Illinoin yliopistosta, National Center for Supercomputing Applicationsista, kehitti Mosaic-nimisen selaimen, joka kykeni esittämään kuvamateriaalia samalla sivulla ja samassa selainikkunassa muun hypertekstin kanssa. (Calore, M. 2011). Mosaicista tuli tämän ominaisuuden ansiosta erittäin suosittu.

Pian osa Mosaicin kehittäjistä siirtyi yritykseen nimeltä Netscape. Tavoitteena oli hyödyntää Mosaicista saatua kokemusta ja kehittää astetta edistyneempi selain. Netscape-selaimesta haluttiin tehdä myös interaktiivinen, koska työpöytäohjelmistot olivat interaktiivisia, mutta selainympäristö oli näihin

verrattuna melko alkeellinen. Netscapesta tuli myöhemmin vielä suositumpi kuin Mosaicista lukuisten uusien ominaisuuksien sekä interaktiivisuutensa ansiosta.

Interaktiivisuutta varten selaimeen oli kehitettävä oma ohjelmointikieli. Netscapeen kehitettiin oma selainohjelmointikieli, joka sai nimekseen LiveScript. LiveScriptiin otettiin vaikutteita muun muassa Java- (syntaksi), Scheme- (funktiomalli) ja Self- (prototyyppiobjektimalli) kielistä (Crockword 2010). Kyseessä oli ensimmäinen versio selainohjelmointikielestä, joka tunnetaan nykypäivänä parhaiten nimellä JavaScript.

Erilaisten nimeämis-, lisenssi- ja standardointiongelmien takia Netscape halusi LiveScript-kielestään virallisen standardin ja päätyi lopulta hakemaan sitä European Computer Manufacturers Associationilta. Virallinen nimi kielelle, joka tunnetaan nykyään JavaScriptinä, on ECMAScript. Microsoftilla on omassa Internet Explorer -selaimessaan käytössä kieli, nimeltä JScript. Myös interaktiivisuutta web-selaimiin 2000 –luvun alusta tuonut Flash-tekniikka (ActionScript) pohjautuu ECMAScriptin 262 –standardiin (Adobe 2011).

Kaikki edellä mainitut tarkoittavat siis samaa asiaa, eli kansan suussa JavaScriptiä. JavaScript-nimi on virallisesti tällä hetkellä Oracle-yhtiön omistama tuotemerkki (Crockford 2010). Internetselainten interaktiivisuus onkin ollut enemmän tai vähemmän sidoksissa ECMAScript-kieleen.

Netscapen selaimesta tuli lopulta erittäin suosittu, ja tämä patisti suuren Microsoft-yhtiön mukaan kehitykseen. Microsoft ei ollut aiemmin ajatellut internetiä tällaisena interaktiivisena alustana, ja se ei ollut tuonut aiemmin markkinoille omaa kunnollista selaintaan. Microsoft julkaisi pian Internet Explorer -selaimensa vastaiskuna Netscapen suosiolle. Se pystyi levittämään sitä oman käyttöjärjestelmäversionsa mukana, suurimman käyttöjärjestelmätoimittajan asemassa erittäin laajalle. Microsoft alkoi toimittaa aina 1990-luvun puolivälistä Internet Explorer-selaimen käyttöjärjestelmiensä mukana esiasennettuna, ja ilmaisena ohjelmistona. IE sai suurimman

markkinaosuuden käyttöjärjestelmien selaimena ennen vuosituhannen vaihdetta (Glasner J. 1999). 2000-luvun puoliväliin tultaessa tämä murskasi Netscapen.

1.1.2 Selainten kaupallinen kilpailu vaikuttaa kehitykseen

Selaimia kehittävät pääosin kaupalliset tahot, ja kehitystä jarrutti pitkään eri tahojen kaupallinen kilpailu. Kilpailu näkyi käytännössä niin, että samaa standardia tuettiin eri tavoin erilaisissa selaimissa. Hypertekstiin käytetyn html – merkkauksen kirjoittamisessa tehtiin useimmiten virheitä, ja eri selaimet kilpailivat muun muassa siitä, miten nämä virheet voitiin ennakoida ja korjata sivua näytettäessä niin, että netin selaaja ei näe virheitä. Markkinoille tuotiin myös uusia ominaisuuksia, joita vain oma selain tuki. Näin yritettiin erottautua kilpailijoista.

Niin sanottu ”selainsota” jatkuu edelleen. Tällä hetkellä on kehitteillä uusi HTML-standardi: HTML5 (W3C 2011.), joka tuo mukanaan lukuisia uusia ominaisuuksia. Standardi ei ole vielä valmis, joten on vaikea sanoa, mitkä ominaisuudet tulevat olemaan lopullisessa versiossa. Vaikka HTML5 onkin keskeneräinen standardi, on se jo käytössä lukuisilla internetsivustoilla. Selainvalmistajat kilpailevatkin jo siitä, kuka tukee HTML 5:n uusia ominaisuuksia parhaiten.

Selainten kehitys on mahdollistanut näin työpöytäohjelmisto-tyyppisen lähestymistavan internetiin. Tulossa oleva HTML 5 -standardi mahdollistaa lukuisia elementtejä, jotka tuovat Internetin käyttöliittymät ja sovellukset entistä lähemmäs työpöytäsovelluksia. Internetselainta voidaan toisin sanoen nykypäivänä käyttää käyttöliittymänä ohjelmistolle, jonka alustana toimii internetpalvelin. Useat työpöytäohjelmistot ovatkin siirtyneet internetiin selaimen sisällä käytettäväksi ohjelmistoksi (Microsoft 2011).

1.1.3 Web-yhteisöjen rooli dynaamisen internetin kehityksessä

Suuressa roolissa avoimen lähdekoodin internetpohjaisten ohjelmistojen kehittämisessä ovat olleet kaupalliset tahot, mutta myös internetiin muodostuneet erilaiset yhteisöt, ei-kaupalliset tahot. Avoimen lähdekoodin käyttö on nykyisin erilaisissa internetin sovelluksissa onkin hyvin yleistä. Arviolta kolmessa neljäsosassa kaikista internetsivustoista käytetään hyväksi avoimen lähdekoodin PHP–kieltä (Tapscott, D. & Williams, Anthony D. 2008, 84).

Ei-kaupallista tahoa voisi ajatella web-yhteisönä, joka on alkanut kehittää järjestelmää omiin tarpeisiinsa tietyn projektin ympärille syntyneen yhteisön voimin. Tarkoituksena on ehkä ennemminkin ollut ratkaista projektin puitteissa jokin ongelma kuin ajatella järjestelmän kaupallisia mahdollisuuksia. Kaupallisuuden ja ei-kaupallisuuden raja on esimerkiksi erilaisia sisällöntuotantosovelluksia ajatellen kuitenkin erittäin häilyvä: useita yhteisöjen kehittämiä ei-kaupallisia järjestelmiä käytetään osana kaupallisia sovelluksia, ja toisaalta liiketoimintaa kehitetään ei-kaupallisten järjestelmien varaan. Nykypäivänä suurin osa mm. Liiketoiminnassa käytettävistä ohjelmistoista on saatavissa avoimen lähdekoodin versiona (Tapscott, D. & Williams, Anthony D. 2008, 85.).

Ohjelmiston hankkijan tai loppukäyttäjän näkökulmasta on vaikea sanoa, mikä olisi paras valinta ohjelmistoa hankkiessa: ns. ei-kaupallinen avoimen lähdekoodin järjestelmä, vai jonkin yrityksen kehittämä kaupallinen järjestelmä. Ohjelmistotuotteen hankkijan olisi hyvä ajatella, millainen rooli yhteisöllä on järjestelmän kehittämisessä, ylläpidossa tai tuotetuessa. Ajatusmalli on eräällä tavalla erilainen kuin kaupallisen lähdekoodin järjestelmässä: yritykset voivat mennä konkurssiin, tuotetuki ja kehitys voi loppua kuin seinään. Avoimen lähdekoodin yhteisöä, tai projektia, ei voida ajatella aivan samalla tavalla. Web-yhteisöön eivät välttämättä päde liiketoiminnan lainalaisuudet.

Yhteisöt käyttävät tuottamissaan järjestelmissä yleensä avoimen lähdekoodin erityyppisiä lisenssejä, joiden puitteissa järjestelmän, tai järjestelmän osien käyttö sallitaan osana kaupallista järjestelmää. Yhteisön tuottamaa järjestelmää sallitaan usein myös myydä eteenpäin osana kaupallista järjestelmää, tai muita palveluita. Yhteisöt saattavat itse kehittää järjestelmästä myös kaupallisen version, jonka lisäpalveluina tarjotaan usein tuotetukea tai muita ylläpitopalveluita. Tällaisen kaupallisen version ohella tarjotaan tuotteesta useimmiten ns. yhteisöversiota (community), jolla ei näitä lisäpalveluita ole, vaan tukena toimii taustalla toimiva yhteisö ja käyttäjäyhteisö itsessään.

Esimerkkejä suurten kaupallisten järjestelmien avaamisesta yhteisöjen kehitettäväksi on lukuisia. Muun muassa suuryritykset, kuten Nokia ja Google, ovat kehittäneet puhelimiaan varten omat kilpailevat käyttöjärjestelmänsä tunnetun avoimen lähdekoodin järjestelmän, Linuxin päälle. Nokia avasi myös kehittämänsä suljetun Symbian–käyttöjärjestelmänsä sekä perusti säätiön tukemaan kyseistä projektia, mutta tästä on sittemmin luovuttu Nokian siirryttyä käyttämään Microsoftin Windows–käyttöjärjestelmää tulevissa puhelinmalleissaan.

Syitä tällaiselle avoimemmalle ilmapiirille voi etsiä siis yhteisön tuen tärkeydestä. Yritykset ovat ymmärtäneet verkkoyhteisöjen voiman ja hyödyn suurissa projekteissa. Osasyynä Nokian kaltaisten yritysten siirtymiseen avoimempaan ilmapiiriin on eri yhteisöjen tuottamat lukuisat ohjelmistot näiden käyttöjärjestelmien päälle, olivatpa ne sitten kaupallisia tai ei-kaupallisia tahoja.

Voidaan ajatella, että suuryritykset tarvitsevat joissain tilanteissa erilaisten web-yhteisöjen tukea kehittäessään ohjelmistoja ja alustoja uusille palveluille. Esimerkiksi Nokian siirryttyä käyttämään suljettua Windows–käyttöjärjestelmää, tarjotaan ohjelmistojen kehittäjille kuitenkin ilmaiseksi kaikki työkalut, joilla ohjelmistoja voidaan kehittää tälle alustalle. Myös muita vastaavan tyyppisiä käyttöjärjestelmiä, kuten Nokian MeeGoa tai Googlen Androidia varten tarjotaan

kehittäjille viimeisimmät kehitysohjelmistot, eli samat ohjelmistot, joita yritykset käyttävät itse kehitystyössään.

1.2 Avoin lähdekoodi osana PK–yrityksen liiketoimintaa

Ohjelmiston suunnittelu ja kehitys on useimmiten suuri projekti. Avoimen lähdekoodin sisällönhallintajärjestelmät ovat kehittyneet useiden vuosien ajan, ja lisenssit sallivat useimmiten järjestelmien kaupallisen käytön. Järjestelmät mahdollistavat näin liiketoiminnan usein yrityksille tuote, tai palveluratkaisuna. Pitkälle kehittyneet järjestelmät isoine käyttäjäyhteisöineen tarjoavat tietynlaisen mallin vakaan liiketoiminnan harjoittamiseen tällaisten tuotteiden puitteissa.

Yritykset siis hyötyvät avoimen lähdekoodin ympärille syntyneistä yhteisöistä useilla eri tavoilla: tuotteen ympärille syntynyt yhteisö tekee osaltaan suuren työn tuotekehityksen, testauksen ja markkinoinnin osalta. (Tapscott, D. & Williams, Anthony D. 2008, 85-88). Avoimen lähdekoodin projektit ovat näin mahdollistaneet erilaisia liiketoimintamalleja myös pienille ja keskisuurille yrityksille, joilla ei ole resursseja massiiviseen tuotekehitykseen. Tilaajan kannalta web-yhteisöjen tuki saattaa olla myös ostopäätökseen positiivisesti vaikuttava tekijä.

2 MODX-PÄIVITYSJÄRJESTELMÄ

Julkaisujärjestelmät ovat helpottaneet sisällön tuotantoa, mutta myös mahdollistaneet varsinaisen sisällön ja tekniikan erottamisen toisistaan. Näin sisällön tuottajan ei tarvitse välittää teknisestä toteutuksesta. Hallintajärjestelmän kehittäjät voivat myös keskittyä itse järjestelmän ominaisuuksien kehittämiseen jo olemassa olevan järjestelmän päälle, kajoamatta varsinaiseen sisältöön.

2.1 Järjestelmän esittely

MODx CMF on samalla sekä julkaisujärjestelmä että ns. Framework. Julkaisujärjestelmän suunnittelussa on yleisesti otettu huomioon sisältömateriaalin ja esittämiskerroksen erottaminen toisistaan. Sisältöä voidaan julkaista järjestelmällä halutussa muodossa erilaisille laitetyypeille (tietokone, matkapuhelimet jne), ja erikokoisille näytöille. Sisältö esitetään useimmiten erillistä mallipohjaa (template) apuna käyttäen, mikä helpottaa esityskerroksen rakenteen pysymistä selkeänä ja yhtenäisenä. Mallipohjaa vaihtamalla voidaan myös vaikuttaa nopeasti sisällön esitystapaan, varsinaiseen sisältöön kajoamatta.

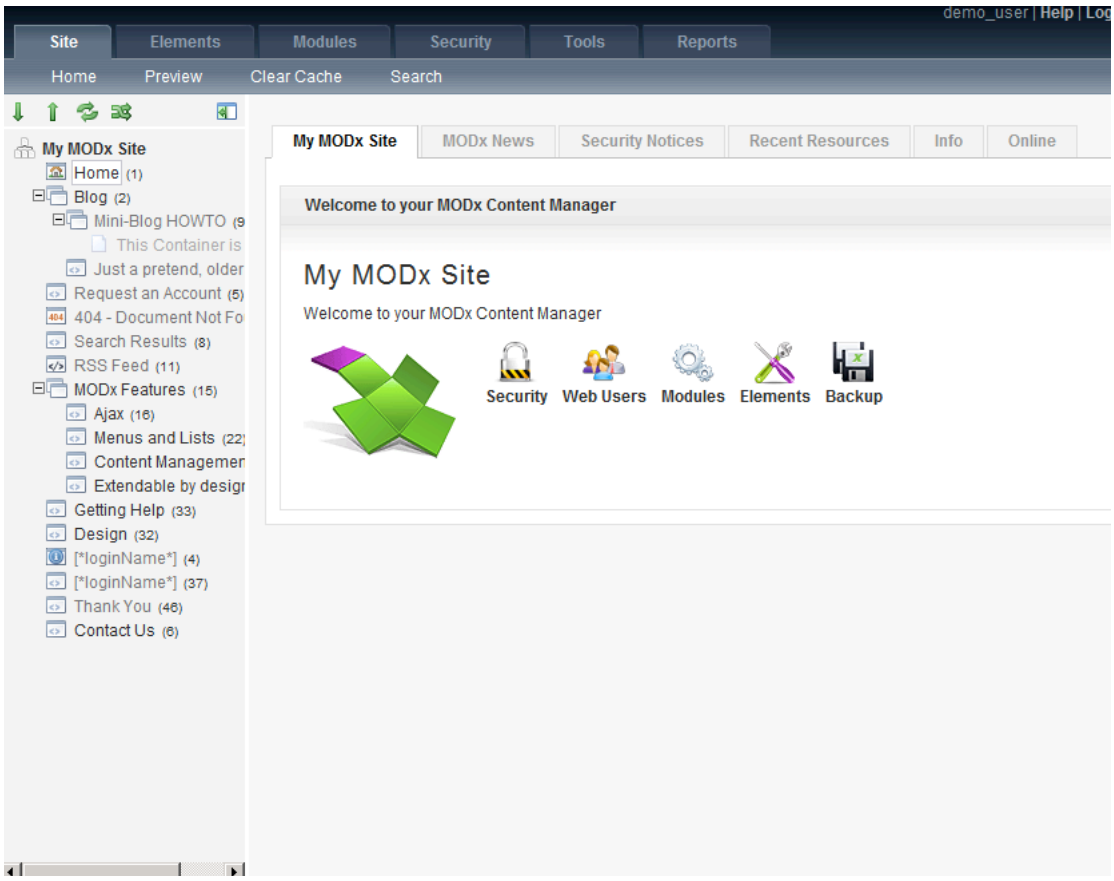
2.1.1 Perusominaisuudet ja sivuston päivitys

Ohessa kuva MODx Evolution -järjestelmästä, kun järjestelmään on kirjauduttu sisään (Kuva 1). Vasemmalla on nähtävissä sivuston rakenne puurakenteena. Päivitettävä sivu valitaan vasemmalta, ja sivun tiedot sekä päivitettävät kentät avautuvat oikealle puolelle. Tekstisisältö ja kuvat liitetään kohtaan ”Resource content”. Sivu voi sisältää myös muita erityyppisiä päivitettäviä kenttiä ja elementtejä, kuten kuvasisältöä tai videokuvaa.

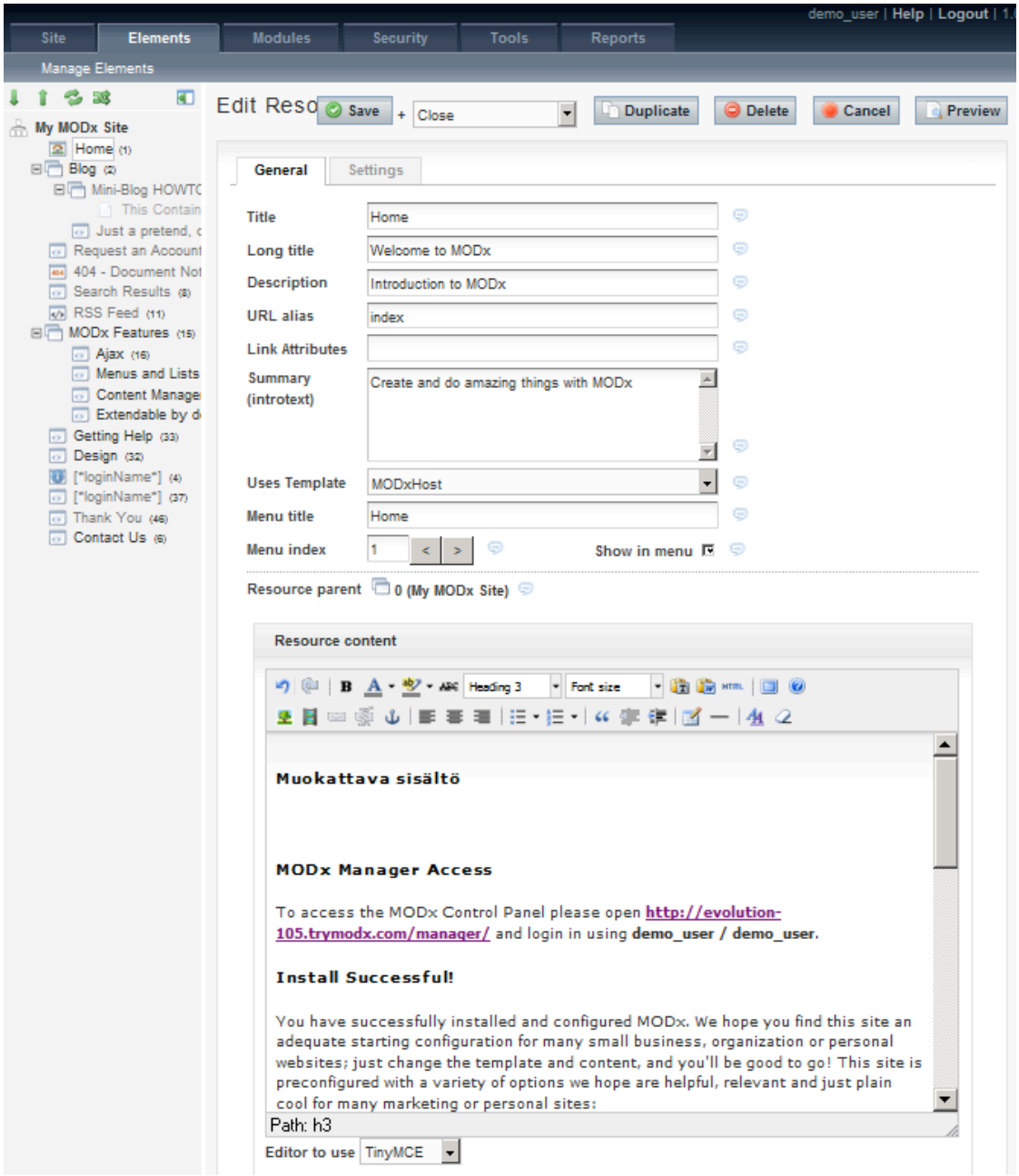
Kullekin sivulle voidaan määritellä erilaisia asetuksia aina ns. META–hakusanoista kyseisen sivun sisällön kuvaukseen. Sivuja voidaan myös ajastaa näkymään vain jonkun tietyn ajanjakson.

Kukin sivu käyttää ulkoasun esittämiseen mallipohjaa (Kuva 3). Mallipohjan koodi on useimmiten HTML- ja JavaScript–muodossa. Sivuston mallipohjaan haluttuihin kohtiin lisätään MODxin omia tageja, jotka kuvaavat dynaamisia alueita, joita voidaan muokata sivustoa päivitettäessä. Kuvassa 3 on varjostettu ns. TITLE –tagien välissä kohta, joka sisältää MODx-tagin. Esimerkiksi tätä kohtaa voidaan kullakin sivulla päivittää. Mallipohjien etuna on, että sivuston rakenne pysyy samanlaisena. Tarvittaessa voidaan vaihtaa sivun käyttämää mallipohjaa sisältöön kajoamatta kohdasta ”Uses template” (Kuva 2).

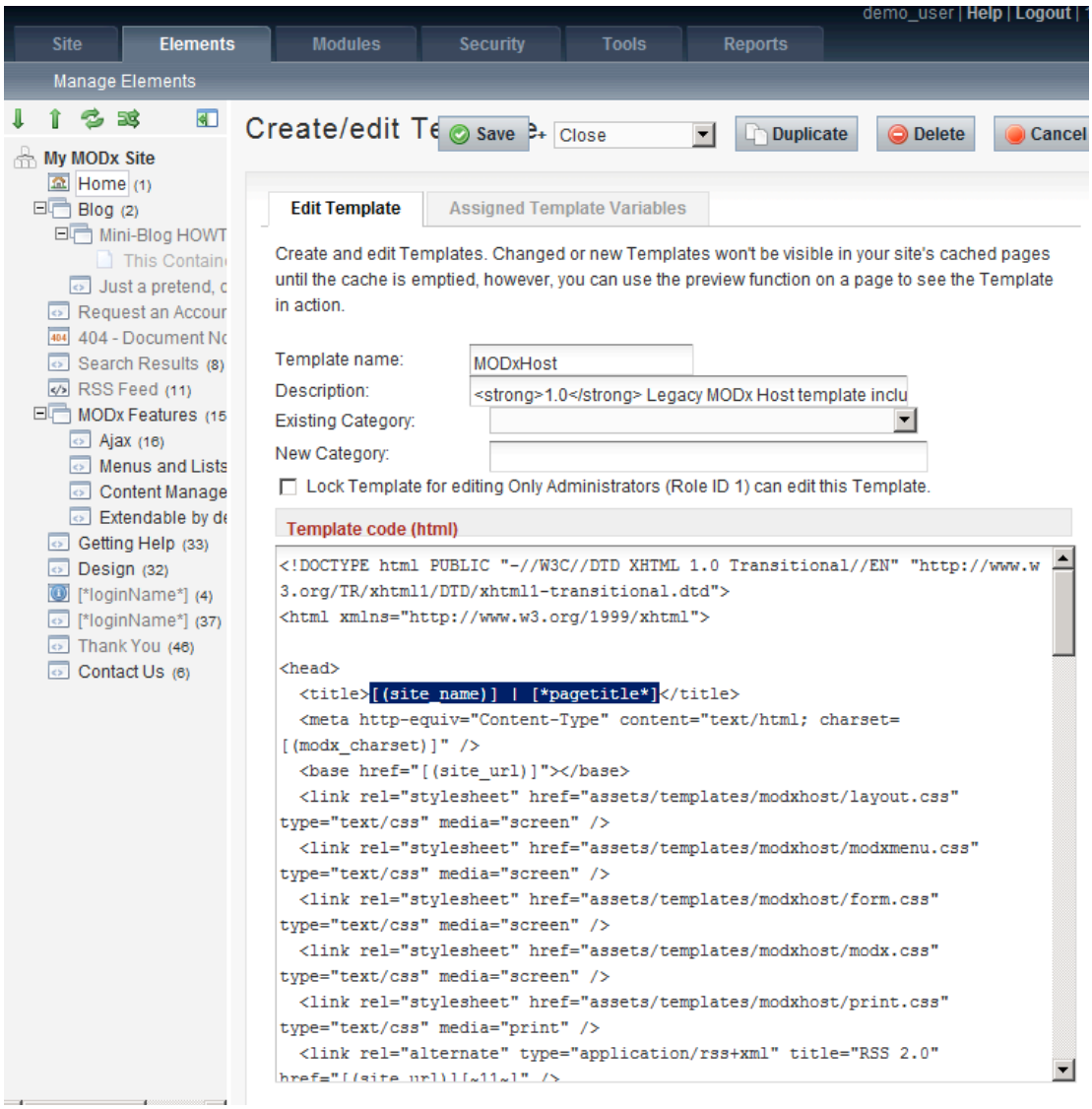
Sivuston toteutus MODx–järjestelmään on melko suoraviivaista; ensin tehdään mallipohja, johon kirjoitetaan html-koodi ja paikat päivitettäville elementeille sekä sisällölle. Tämän jälkeen luodaan itse sivu, ja asetetaan sivu käyttämään tiettyä mallipohjaa. Sivun sisältöelementteihin kirjoitetaan sisältö, ja valitaan asetuksista sisällön julkaisu. Julkaistu sivu on heti selattavissa.



Kuva 1. MODxin aloitussivu (Ruutukaappaus asennetusta järjestelmästä).



Kuva 2. Sisällön muokkaus ja päivitys (Ruutukaappaus asennetusta järjestelmästä).



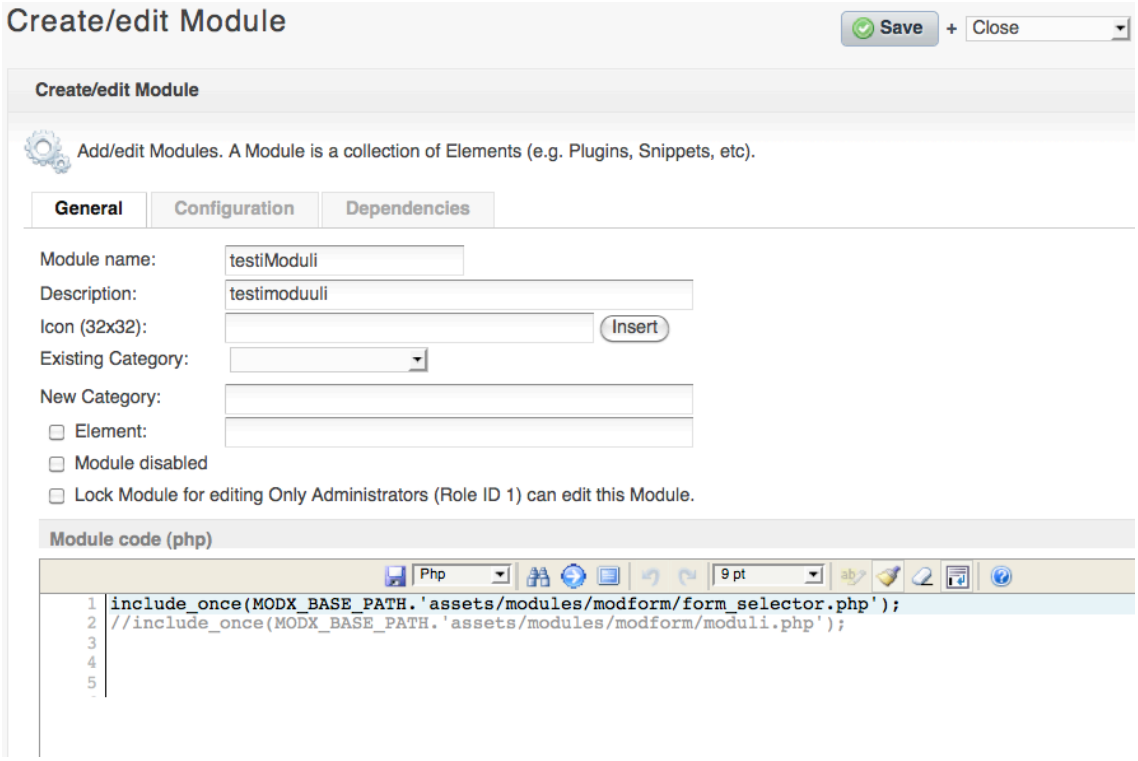
Kuva 3. Mallipohjan muokkaus (Ruutukaappaus asennetusta järjestelmästä).

2.2 Moduulin luonti MODxiin ja käyttöoikeuksien määrittäminen

Lomakemoduulia käytetään MODx–päivitysjärjestelmän sisällä. Moduulia voidaan käyttää, jos päivitysjärjestelmässä on ylipäänsä määritelty käyttäjälle oikeudet käyttää moduuleja. Päivitysjärjestelmän ylläpitäjä voi luoda

käyttäjäryhmän, jolla on oikeus ajaa moduuleita, ja käyttäjä voidaan liittää tähän ryhmään: Security-> Roles -> Module management -> Run Module.

Jotta moduuli saadaan toimimaan MODxin sisällä, eli moduulissa voidaan käyttää MODxin frameworkin APIa ja määrittää oikeudet moduulin käyttöä varten, moduulille on luotava MODxissa ns. pohja: Module-> Manage Modules -> New Module (Kuva 4).



Kuva 4. Moduulin luonti MODxiin (Ruutukaappaus asennetusta järjestelmästä).

Moduulille annetaan nimi, joka ilmestyy myös linkiksi MODxin käyttöliittymään. Moduulille voidaan antaa myös kuvaus *Description*–kenttään. Varsinainen moduulin koodi kirjoitetaan *Module code (php)* –osioon. Mikäli kyseessä on pieni sovellus, voidaan koko sovelluksen koodi liittää tähän osioon. Isommissa sovelluksissa on koodi useimmiten jaettu eri tiedostoihin muun muassa

selkeyden säilyttämiseksi, kuten tässäkin työssä, joten Module code –osiota käytetään vain lataamaan tai liittämään tarvittava koodi moduuliin (Kuva 4).

2.3 MODxin API

Sisällönhallintajärjestelmän API tarjoaa useimmiten valmiita ratkaisuja ja ominaisuuksia helpottaen kehittäjien työmäärää uusia ominaisuuksia kehitettäessä. Toisinaan APIn avulla on haluttu ohjata kehittäjiä käyttämään tietyntyyppisiä ominaisuuksia ja välttämään järjestelmän ominaisuuksien käyttämistä väärin. Esimerkiksi MODx–järjestelmän API mahdollistaa SQL-kyselyiden suorittamisen tietokannassa omien muuttujiensa kautta. Tämä säästää paitsi aikaa, mutta tarjoaa myös mahdollisuuden syötteiden oikeellisuuden tarkastamisen, sekä estää haitallisten tietojen liittämistä kyselyyn.

APIa hyväksi käyttäen, voidaan moduulissa esim. seuraavalla tavalla hakea etusivun (tässä tapauksessa id: 1) sisältöosa tietokannasta, ja tulostaa se näytölle:

```
$doc = $modx->getDocument(1);

echo $doc['content'];
```

Sama onnistuisi myös suoraan SQL–kyselylauseina.

3 LOMAKEMODULIN TOTEUTUS

Suunnittelu, toteutus ja testaus ovat kulkeneet tässä työssä päällekkäin. Kuvaan aluksi lyhyesti moduulin suunnittelua ja sen pohjalta tehtyjä toiminnallisia ja teknisiä vaatimuksia. Käyn myös läpi moduulin toiminnot ja käytön pääpiirteissään.

Moduulin käytöstä ja toiminnasta etenen tarkemmin tekniseen toteutukseen: miten moduuli on toteutettu, ja mitä tekniikoita toteutuksessa on käytetty. Yksityiskohtaiset ohjeet moduulien asentamiseen ja käyttöoikeuksien määrittämiseen löytyvät dokumenteista ja käyttöoppaasta MODxin viralliselta sivustolta (MODx 2011). Moduulin asennus ja perusasetukset on kuvattu edellä kappaleessa ”Moduulin luonti ja käyttöoikeuksien määrittäminen”.

Luvussa 5 pohdin lopuksi, miten olen onnistunut työssä ratkaisemaan alussa asetetut toiminnalliset ja tekniset vaatimukset, minkälaisia haasteita työssä tuli vastaan sekä, mitä minun olisi pitänyt tehdä toisin.

3.1 Suunnittelu ja työn aloittaminen

Työn suunnittelu eteni siten, että pidimme aloituspalaverin, jonka pohjalta lähdin hahmottelemaan moduulin käyttöliittymää. Suunnittelussa ei kartoitettu toimintoja kovinkaan tarkasti, mutta päälinjat mainituista toiminnallisista vaatimuksista sovittiin aluksi.

Myös tekniikat, kuten jQueryn käyttö käyttöliittymän pudotus- ja raahaustoiminnossa, (kuva 9) mietittiin jo alussa, sekä mahdollisuus käyttää EFormia käyttäjäsyötteiden tarkastuksessa. Hahmottelimme myös, mitä tietoja tietokantaan tulisi tallentaa.

Lähdin tämän perusteella selvittämään aluksi, kuinka käyttöliittymän raahaus- ja pudotusominaisuudet olisivat toteutettavissa vaivattomasti, ja minkälaisia ratkaisuja jQuery tähän tarjoaa. Tämän jälkeen hahmottelin, millä tavoin tarvittavat tiedot saadaan haettua luodusta lomakkeesta, ja tallennettua tietokantaan.

Aloituspalaverin ja keskustelujen pohjalta hahmottelin toiminnallisia ja teknisiä vaatimuksia, joita on listattu tarkemmin seuraaviin kappaleisiin.

3.2 Yleiskuvaus ja tavoitteet

Tarkoituksena oli toteuttaa html-lomakkeiden luonti ja muokkaustyökalu MODx Evolution –päivitysjärjestelmään. Työkalulla voidaan luoda html-lomakkeita, mutta myös muokata lomakkeiden elementtejä, html-attribuutteja, lomake-elementteihin liitettäviä tekstejä, sekä mahdollistaa käyttäjäsyötteiden tarkastus EForm–liitännäistä varten.

Lomaketyökalu palvelee sekä toimeksiantajaa että mahdollisesti toimeksiantajan asiakkaita. Kohderyhmä laajenee näin tietojenkäsittelyn ammattilaisista mahdollisiin päivitysjärjestelmän loppukäyttäjiin. Työkalun on tarkoitus olla käytettävyydeltään laadukas: nopea ja näppärä käyttää, sekä olla käyttöliittymältään selkeä. Käytettävyyttä haettiin muun muassa ”raahaus & pudotus” -käyttöliittymällä.

Lomakkeiden muokkaus toimii siten, että haluttuja lomake-elementtejä raahataan käyttöliittymästä lomakepohjalle (Kuva 9), jossa niitä voidaan järjestellä ja muokata. Elementeille voidaan määrittää muun muassa erilaisia tekstikenttiä (otsikkotekstit ja kuvaustekstit), sekä muokata html-elementtien attribuutteja (esim. Id, name), joita tarvitaan myöhemmin, kun lomake tulostetaan tietojen pohjalta halutulle internetsivulle.

3.2.1 Toiminnallisia vaatimuksia

- Nopea käyttää (raahaus ja pudotus -toiminto)
- Selkeä graafinen käyttöliittymä
- Mahdollisuus muokata otsikkoja ja kuvaustekstejä
- Mahdollisuus muokata lomakkeen esittämisessä tarvittavia html-elementtejä, kuten *id* ja *class*
- Mahdollisuus muokata lomake-elementtien järjestystä lomakepohjalla (raahaus ja pudotus -toiminto).

3.2.2 Teknisiä vaatimuksia

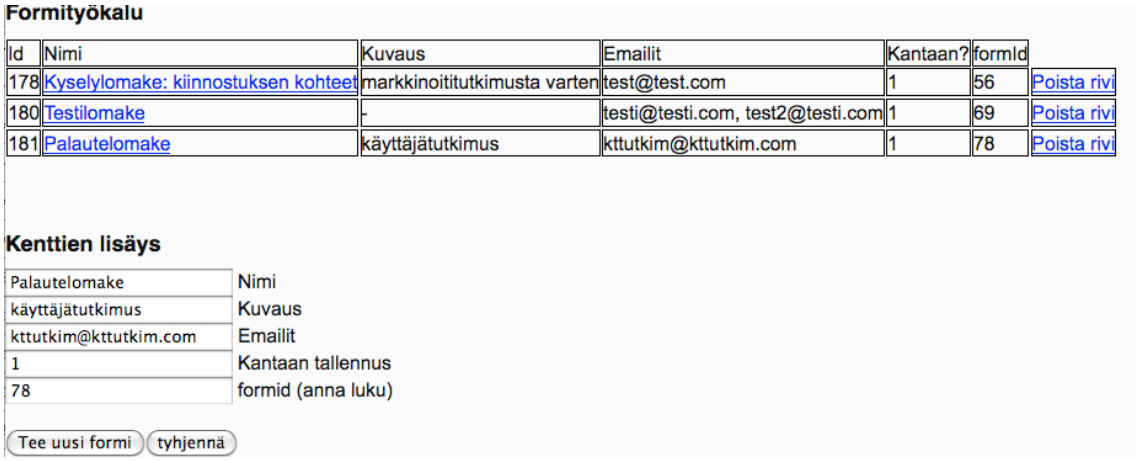
- Toteutettava MODx-järjestelmän kanssa yhteensopivaksi
- Oltava yhteensopiva MySQL:n kanssa
- Toteutettava niin, että tietoja ei poisteta MySQL:stä
- Mahdollisuus moduulin käyttöoikeuksien määrittämiseen MODx-järjestelmällä
- Mahdollisuus käyttää MODxin API:a muun muassa tietoturvan parantamiseksi
- Toimittava jQueryn kanssa.

3.3 Toiminnot

Lomakkeen suunnittelussa on pyritty antamaan mahdollisimman samanlainen tuntuma lomakkeen muokkaustilassa kuin mitä lomakkeen lopullinen ulkoasu tulee olemaan. Loppukäyttäjälle näkyvät tiedot voidaan tarkistaa lomakepohjalta (kuva 6), ja muut lomakkeen luomisessa ja eteenpäin lähettämisessä tarvittavat tietokannan tiedot näkyvät muokkaimessa, mutta eivät häiritse lomakkeen ulkoasun suunnittelua.

3.3.1 Lomakkeen valinta ja uuden lomakkeen luonti

Kun lomakemoduuli avataan, ensimmäisessä näytössä voidaan joko valita jokin lomake muokattavaksi tai luoda uusi lomake (kuva 5). Lomakkeen tiedot kirjoitetaan kohtaan: ”kenttien lisäys”. Lomakkeelle voidaan antaa useampi sähköpostiosoite, johon tiedot lopulliselta lomakkeelta tullaan lähettämään. Merkitään myös tallennetaanko lopulliset tiedot tietokantaan sähköpostin sijaan. ”Poista rivi” -linkistä voidaan poistaa kyseinen rivi. Rivin poistaminen poistaa vain kyseessä olevan rivin tiedot, mutta ei kuitenkaan poista tietoja tietokannasta, jonne eri lomake-elementit tallennetaan. Alkutietojen antamisen jälkeen päästään muokkaamaan varsinaista lomaketta lomakepohjalla klikkaamalla haluttua nimeä.



Kuva 5. Lomakemoduulin etusivu (Ruutukaappaus asennetusta järjestelmästä).

3.3.2 Lomakkeen muokkaus lomakepohjalla

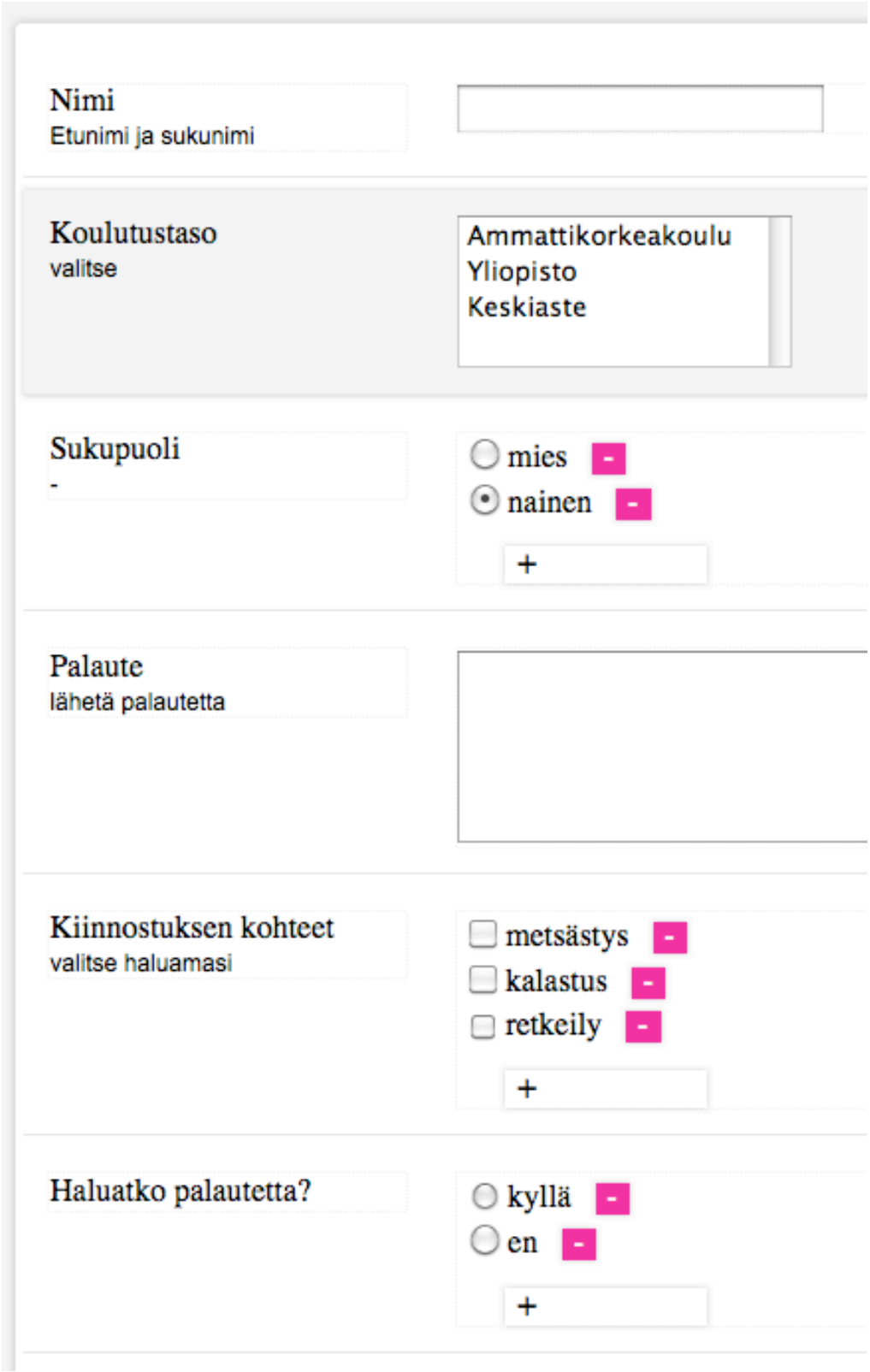
Muokattava elementti voidaan valita lomakepohjalta näpäyttämällä sitä, jolloin elementin taustaväri muuttuu harmaaksi (kuva 6). Valitun elementin sisältämät tiedot näkyvät muokkaimessa (kuva 7). Kuvassa 6. on valittuna rivi, jonka otsikko on ”Koulutustaso”. Valinta näkyy harmaalla pohjalla. Rivin otsikko näkyy muokkaimessa ensimmäisenä.

Tietoja, jotka näkyvät lomakkeen loppukäyttäjälle, ovat muun muassa ”Rivin otsikko” ja tarkempi kuvaus tai alaotsikko: ”Kuvaus / tooltip”. Checkbox- ja radioelementeille voidaan kirjoittaa oma tekstinsä näpäyttämällä ensin haluttua tekstiä lomakepohjalla checkbox-, tai radionapin vieressä, jolloin valittu teksti muuttuu keltaiseksi ja tekstin muokkaus onnistuu muokkaimesta kohdasta ”Radio / checkbox inner”.

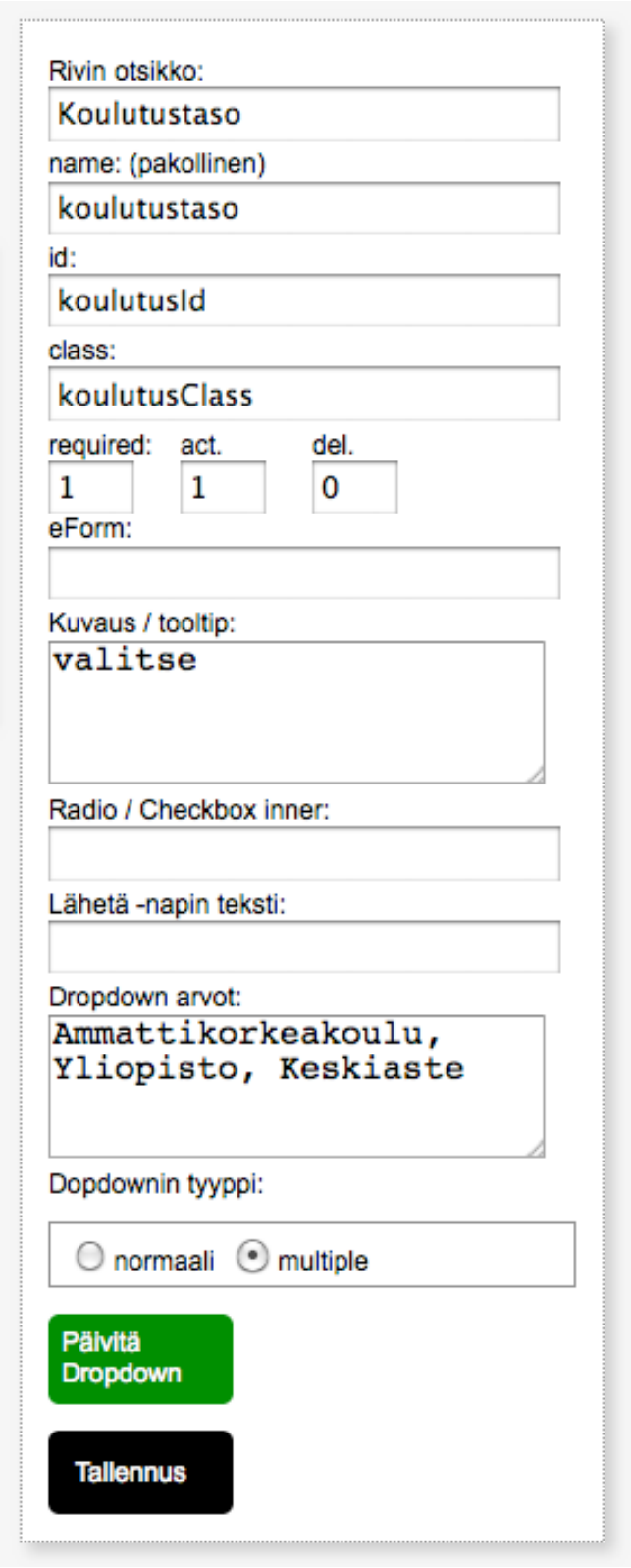
Select–elementille löytyy muokkaimesta oma kenttänsä, johon valittavaksi asetettavat arvot voidaan syöttää pilkuilla erotettuina. Koska select-elementtiä on kaksi erilaista ns. pudotusvalikko, ja valikko, joista voidaan valita useampi elementti kerrallaan, täytyy select–elementin tyyppi valita ”Dropdownin tyyppi”-valikosta ja päivittää alla löytyvällä ”Päivitä dropdown”-napilla. Muutokset näkyvät lomakepohjalla heti päivityksen jälkeen.

Lomakkeen elementtejä voidaan järjestellä raahaamalla lomakepohjalla. Radio ja checkbox–elementtejä voidaan luoda lisää ”+”-napista tai poistaa ”-”-napista lomakepohjalla.

Valmis lomake tallennetaan lopuksi ”Tallennus”–napilla, jolloin tiedot tallentuvat tietokantaan.



Kuva 6. Lomakepohja. Valittuna rivi: ”koulutustaso”



Kuva 7. Koulutustasoa vastaavan rivin tiedot muokkaimessa.

Tietoja, jotka eivät näy lomakepohjalla, eivätkä näin loppukäyttäjälle, ovat muun muassa Rivin otsikon alta löytyvät *name, id ja class,* -kentät, joita käytetään html -tagien attribuuteissa. Näiden alta löytyviin *required, act, del ja eForm*–kenttiin tulevat arvot tallennetaan niin ikään tietokantaan lisätietoina. Kuvaus näiden merkityksestä löytyy seuraavasta ”Tiedot ja tietokanta”-kappaleesta.

3.4 Tiedot ja tietokanta

Tiedot tallennetaan tietokantaan niin, että lomake-elementtien html-koodia ei tallenneta, vaan tietokantaan tallennetaan tieto siitä, minkä tyyppinen lomake-elementti on kyseessä, esim. Radio, select, textarea, submit jne. Tämän tiedon perusteella voidaan myöhemmin luoda halutun tyyppinen html-elementti tunnistamalla kannasta haettu elementti ja tulostamalla siihen haluttu html-koodi. Näin voidaan vähentää tietokantaan tallennettavan tiedon määrää ja tehdä lomakkeen tulostamisesta kieliriippumatonta, vaihtuipa sitten html–versio tai palvelimen kielen tyyppi.

Lomake-elementtien tiedot tallennetaan kuvan 8. mukaiseen tietokantatauluun.

3.4.1 Tietokannan kentät ja elementtien mukana tallennettavat tiedot

Käyn kentät järjestyksessä kuvan 8. mukaan ylhäältä alas. Kuvaan mihin kutakin kenttää käytetään. Koko systeemi käsittää kaksi tietokantataulua, jotka eivät ole missään suhteessa toisiinsa. Toisen tietokantataulun rakenne selviää kuvasta 5. kohdasta ”Formityökalu”. Kentät ovat tuossa toisessa taulussa samat kuin Formityökalun sarakkeet.

colId

Yksilöllinen numero riville. Juokseva numero joka lisätään automaattisesti tallennettavalle riville (auto_increment).

colOrder

Tietoja tulostettaessa tarvitaan elementin järjestysnumero (colOrder) ja yksilöllinen numero (colUniqueId), jotta tietty elementti voidaan hakea kannasta ja tulostaa näytölle oikeassa järjestyksessä.

colName

Rivin kentän nimi. Otsikkotieto, joka liittyy lomakkeen rivin tietoihin. Tämä näkyy lomakkeen rivin otsikkona loppukäyttäjälle eli lomakkeen täyttäjälle. Tätä vastaa kenttä *colLabel*, johon voidaan tallentaa otsikkotieto lomakkeen lopullista tarkastajaa varten, eli sille, jolle tiedot lähetetään.

colType

Tähän kenttään tallennetaan lomakkeen elementin tyyppi, jotta tietoihin voidaan liittää html-sivulle tulostettaessa oikeanlainen html-lomake –elementti.

colMultiVals ja colMultiOps

Joillekin elementeille, kuten *select* voidaan antaa useita arvoja valittavaksi. Näihin kenttiin tallennetaan numerotieto (colMultiVals) sekä kutakin numeroa vastaava tekstitieto (colMultiOps).

Näitä kenttiä käytetään myös checkbox- ja radioelementeissä samaan tarkoitukseen.

colCreatedOn

MySQL tallentaa tähän kenttään automaattisesti aikaleiman, jolloin kenttä tai rivi on luotu. Tätä tietoa voidaan käyttää myöhemmin esimerkiksi poistettaessa tietoja suoraan tietokannasta.

colDeleted ja colActive

Näitä tietoja voidaan käyttää poistettaessa tietoja kannasta tai tulostettaessa lopullista lomaketta näytölle. Näihin kenttiin merkitty luku edustaa sitä, näytetäänkö kenttää ollenkaan lopulliselle käyttäjälle.

colRequired

Kenttään tallennetaan tieto loppukäyttäjää varten siitä, onko kyseiseen elementtiin valittava tai täytettävä jokin tieto. Toisin sanoen kenttää voidaan käyttää myöhemmin, ilmoittamaan käyttäjälle, että kyseinen tieto on pakollinen ja lomaketta ei lähetetä eteenpäin ennen kuin ko. tieto on annettu.

colLabel

Kentän tietoa käytetään tunnisteena lähetettäessä tiedot lomakkeelta lopulliseen määränpäähänsä, esimerkiksi haluttuun sähköpostiin tai tietokantaan. Tässä voitaisiin periaatteessa käyttää samaa tietoa kuin *colName*–kentässä, mutta tämä tieto saattaa hieman poiketa *colName*–kentästä, jonka käyttötarkoitus on hieman toinen.

colTooltip

Tätä kenttää käytetään *colName- ja colLabel*–kenttien tarkenteena tai alaotsikkona. ”Tooltip”–nimi viittaa siihen, että joissain lomakkeissa tämän tekstin saa esiin viemällä osoittimen esimerkiksi otsikkokentän päälle.

colFormId

Koska eri lomakkeet tallennetaan samaan tauluun, voidaan tämän tunnisteen perusteella hakea ja tallentaa tiettyyn lomakkeeseen kuuluvat kentät, sekä muokkainta varten, että loppukäyttäjää varten.

colUniqueId

Jokaisella elementillä on yksilöllinen *id,* joka muodostuu *colFormId*:stä ja loppuosasta. Tätä tietoa käytetään tallennettaessa kenttiä tietokantaan, jolloin joissain tapauksissa tietyt muuttuneet tiedot vain päivitetään riville.

colSelectType

Select-elementille on kaksi eri esitystapaa: pudotusvalikko ja valintalaatikko, josta voidaan valita monta elementtiä kerrallaan. Jotta osataan tulostaa näytölle oikeanlainen select–elementti, on tallennettava myös tieto siitä, kumpaa tyyppiä ollaan tulostamassa.

colElementId ja colElementClass

Id–attribuutti on elementille html-koodissa annettava yksilöllinen nimi, johon voidaan myöhemmin viitata CSS- tai JavaScript–kieltä apuna käyttäen. CSS–kielellä voidaan viitata, sekä Id, että Class –attribuutteihin. Id- ja Class-attribuutit eroavat käytännössä siten, että Id on yksilöllinen, tiettyyn elementtiin viittaava ja Class –arvo voi olla samanlainen useilla elementeillä. Näin voidaan esimerkiksi CSS:llä muokata juuri halutun elementin ulkoasua (Id) yksin tai useamman elementin ulkoasua kerralla (Class).

colElementEform

Kenttään tallennetaan EForm -syötteentarkistuksessa käytettävä koodi. Kun loppukäyttäjä lähettää lomakkeen eteenpäin, EForm käy lomakkeen läpi ja tarkastaa palvelimella syötteet tämän merkinnän pohjalta.

Sarake	Tyyppi	Tyhjä	
colId	int(10)	Ei	
colOrder	bigint(20)	Ei	
colName	varchar(255)	Ei	
colValue	varchar(255)	Ei	
colType	varchar(15)	Ei	
colMultiVals	varchar(255)	Ei	
colMultiOps	varchar(255)	Ei	
colCreatedon	timestamp	Ei	C
colEditedon	timestamp	Ei	0
colDeleted	int(1)	Ei	0
colActive	int(1)	Ei	1
colRequired	int(1)	Ei	0
colLabel	varchar(255)	Ei	
colTooltip	text	Ei	
colFormId	int(255)	Ei	
colUniqueId	int(255)	Ei	
colSelectname	varchar(30)	Ei	
colSelecttype	varchar(15)	Ei	n
colElementId	varchar(30)	Ei	
colElementClass	varchar(30)	Ei	
colElementEform	varchar(100)	Ei	

Kuva 8. Tietokantataulu, johon arvot tallennetaan.

3.5 Tekninen toteutus

Käyn läpi pääkohdat moduulin teknisestä toiminnasta ja koodin oleellisimmat kohdat, joissa kuvataan tiedon kulkua moduulin käyttöliittymän ja tietokannan välillä.

3.5.1 Raahaus ja pudotus –käyttöliittymän toteutus

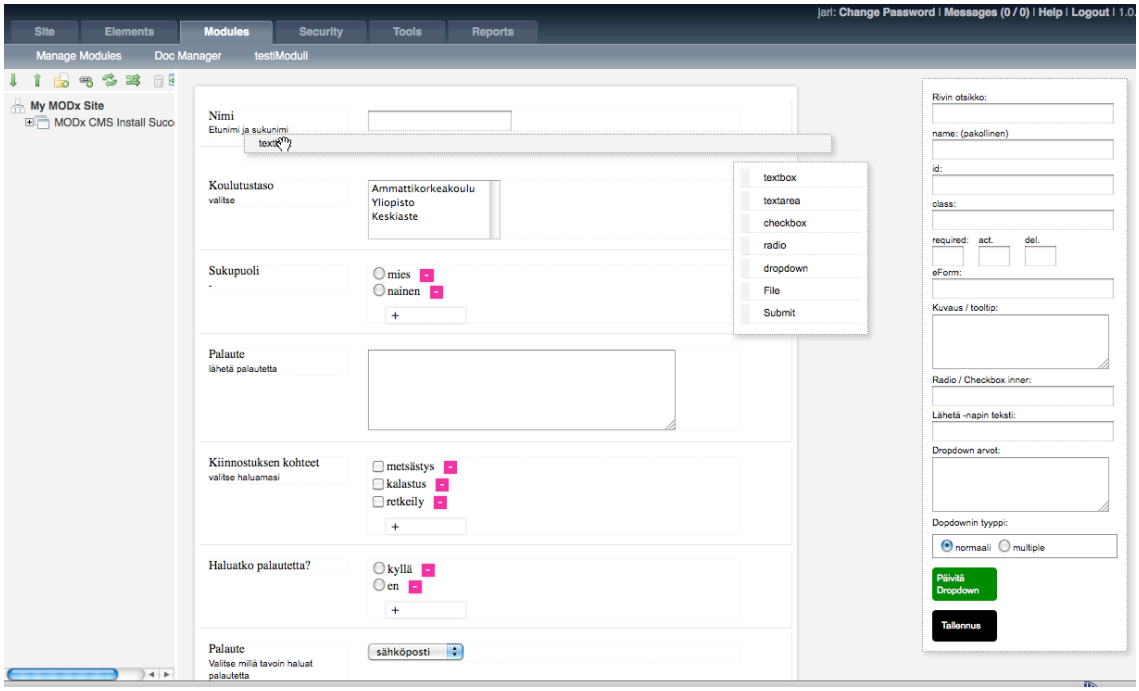
Moduulin käyttöliittymässä käytetään raahaus ja pudotus ominaisuuksien toteutuksessa jQuery–frameworkia. JQuery on JavaScript–kirjasto ja ns. Framework, joka nopeuttaa ja yksinkertaistaa JavaScriptin DOM-

ominaisuuksien käyttöä tuotannossa mm. vähentäen koodin kirjoittamisen tarvetta. JQuery tarjoaa myös lukuisia valmiita JavaScript –luokkia elementtien raahaamiseen ja pudottamiseen. JQueryä voi käyttää niin, että se ei rajoita tavallisen JavaScriptin kirjoittamista, vaan päinvastoin mahdollistaa JQuery–kirjaston käytön aina tarvittaessa. JQueryä ylläpidetään ja kehitetään jatkuvasti joten sen avulla toteutetut JavaScript -toiminnallisuudet toimivat aina useimmissa selaimissa.

Moduulin käyttöliittymässä ja ulkoasussa käytetään HTML 4.01 –merkkausta. HTML 4.01 on lähes kaikkien selainmallien tukema merkkauskieli. HTML:stä on vastikään julkaistu myös uudempi versio: HTML5, jota olisi myös mahdollista käyttää moduulissa, mutta sitä ei ole vielä virallisesti otettu käyttöön kaikissa selaimissa, ja jotkin ominaisuudet saattavat muuttua lähitulevaisuudessa.

HTML –dokumentin koodia on päivitettävä reaaliaikaisesti sitä mukaa, kun moduulilla muokataan dokumentin ulkoasua ja lomake-elementtejä. JavaScriptin DOM –luokkien avulla, voidaan halutut HTML-elementit hakea dokumentista, ja muokata niitä halutulla tavalla: HTML DOM (World Wide Web Consortium -standardi) eli Dokumentin oliomalli tarjoaa alustasta tai ohjelmointikielestä riippumattoman, standardin tavan HTML–dokumentin muokkaamiseen. DOM esittää HTML–dokumentin hierarkkisena puurakenteena, joka sisältää juurielementin ja erilaisia lapsielementtejä. Elementeillä voi olla eri ominaisuuksia, attribuutteja. Elementtien ominaisuuksia voidaan muuttaa, ja elementtejä voidaan poistaa tai lisätä haluttuihin puurakenteen kohtiin JavaScriptillä.

Raahaus ja pudotus (kuva 9) on toteutettu jQueryä apuna käyttäen. JQueryn avulla voidaan tietyt html-elementit määrittää raahattaviksi. Samoin on määritettävä alue, jolle raahattavat elementit voidaan pudottaa. Molemmat merkitään id- tai class–attribuuteilla.



Kuva 9. Lomakkeiden luonti. Elementin raahaus käyttöliittymästä lomakepohjalle.

Lomaketyökalun käyttöliittymän html-elementit on merkitty jQueryn- raahaus ominaisuutta varten luokalla *.formMenu_b,* tiedostossa demo.php.

Muuttujalla *dragOptionsMenu_b* annetaan raahattavalle elementille erilaisia ominaisuuksia. Viimeinen ominaisuus *stop* kertoo, mitä tapahtuu, kun raahaus lopetetaan. Tässä tapauksessa kutsutaan funktiota *build* (jQuery 2011).

```
$(".formMenu_b").draggable(dragOptionsMenu_b);

var dragOptionsMenu_b = {
        connectToSortable: 'div#content',
        helper: 'clone', // tehdään klooni siirrettävästä
        elementistä, jotta alkuperäinen ei häviä
        cursor: 'move', // muutetaan kursori raahauskuvaksi
        scope: 'a' ,
        stop: build // kun raahaus lopetetaan kutsutaan build
        -funktiota
    };
```

Elementti pudotetaan alueelle, joka on merkitty id:llä *content*.

```
$("#content").droppable(dropOptions)
          .sortable({
          cursor: 'move',
          });
```

Kun elementti on raahattu lomakepohjalle content-elementin päälle ja pudotettu, kutsutaan build–funktiota. Build–funktio tunnistaa raahatun elementin luokan (esimerkistä jätetty pois if-lauseen alku):

Tiedosto formBuilder.js rivi 57.

```
else{ var textArea = $(this).hasClass("drag_textarea");}
```

textArea –muuttujan arvo on nyt ”true”, eli se on tosi.

Rivillä 138 kutsutaan textarea -funktiota, jos textArea -muuttuja on tosi.

```
if (textArea || textArea == 'roll' ){

textarea(counter, textArea, names, labels, tooltips, requireds,
active, deleted, elementid, elementclass, elementeform);}
```

Rivillä 436, luodaan itse html-textarea-elementti, joka sijoitetaan tekstinä inputElement –muuttujaan. InputElement–muuttuja sijoitetaan formContent–muuttujaan muiden elementtien kanssa. InputElementin kanssa lisättävät muut muuttujat ovat html-merkkausta. Näiden muuttujien sisältö on alustettu valmiiksi formBuilder–tiedoston alussa. FormContent–muuttuja sisältää nyt kaiken tarvittavan lomake-elementin luomiseksi.

```
var inputElement = '<textarea rows="5" cols="40" id="'+
uniqueInputId +'" name="'+ labels +'"></textarea>';

// inputForm builder

var formContent =
rowWrapTop +
```

```
descriptContainerTop + descriptLabelContainerTop +
labelContainerTop + labelText + labelContainerBottom +
descriptLabelContainerBottom + descriptTextContainerTop +
tooltipText + descriptTextContainerBottom +
descriptContainerBottom +


formElementContainerTop + inputElement +
formElementContainerBottom +hiddenPlaceholder +
rowWrapBottom;
```

Kun käyttöliittymän elementti on raahattu ja pudotettu ns. Lomakepohjalle (kuva 9) content-elementin päälle, elementti säilyttää vielä luokkatietonsa (class), joka on tässä tapauksessa *drag_textarea.* Alla olevilla riveillä etsitään *drag_textarea,* paikka, johon elementti pudotettiin, ja korvataan tämä formContent –muuttujan sisällöllä.

```
// placeholder - hekee paikan ja korvaa välittömästi uudella

var placeholder = $("#content").find(".drag_textarea");

if (switcher != 'roll'){

          placeholder.replaceWith(formContent);

}
```

3.5.2 Tietojen lähetys PHP-skriptille

Luodun lomakkeen tiedot voidaan tallentaa ”Tallennus”–napilla, muokkaimesta.

Tallennusnappi on merkitty id:llä *saveto,* ja siihen on määritelty toiminto nappia painettaessa. Luotujen html-elementtien koodi kerätään content–elementistä *tieto* -muuttujaan.

```
$("#saveto").click(function(){

// lisätään muuttujaan lomakepohjalle luotujen elementtien html-
```

koodi kokonaisuudessaan:

```
var tieto = $("#content").html();

// thefid –muuttujan tieto, lomakkeen yksilöivä tunniste on
muistissa ja se lisätään muuttujaan

var thefid = '<?php if (isset($_GET['id']) ){
          $formid = $_GET['id']; echo $formid;} ?>';

// tiedot lähetetään funktiolle lahetaTieto

lahetaTieto(tieto, thefid);

// tietojen lähetys tekstimuodossa php-skriptille

function lahetaTieto(str, ffid){

$.post("http://localhost/assets/modules/modform/moduli.php",{
tieto: str, thefid: ffid }, "text");
}
```

thefid –muuttujaa ei voi ymmärtää tämän esimerkin perusteella, mutta tieto on käytössä tässä vaiheessa. Ko. tieto jää muistiin, kun lomakemoduulin etusivulla määritellään lomakkeelle tunniste (formid) ja näpäytetään lomakkeen otsikkoa Formityökalussa.

HTML –koodi (*str),* sekä *formid* lähetetään lopulta eteenpäin PHP -skriptille (moduli.php) jQueryn post -ominaisuutta apuna käyttäen.

3.5.3 PHP-skriptille lähetettyjen tietojen käsittely ja tallennus tietokantaan

Tietojen käsittely tapahtuu palvelimella moduli.php-tiedostossa. PHP:tä käytetään tietokantayhteyksissä, tiedon tallentamisessa ja hakemisessa tietokannasta. Tässä työssä PHP ottaa vastaan käyttäjän tallentaman lomakkeen koodin, hakee tiedon joukosta lomakkeeseen muokatut arvot, tekstit, sekä elementtien tyypit, ja tallentaa nämä tiedot tietokantaan.

PHP:ssä käytetään tietojen hakemiseen lomakekoodista PHP:n omia DOM-luokkia. PHP:n DOM-luokat tukevat HTML4.01–merkkausta (PHP 2011).

Moduli.php tiedostossa tarkistetaan ensin onko *tieto* –muuttuja asetettu. Tälle skriptille lähetetyt tiedot ladataan omiin muuttujiinsa. Skriptille lähetettyyn tietoon on lisättävä HTML-dokumenttityyppitieto, jotta PHP tietää, miten tietoa tulisi käsitellä. Jos dokumenttityyppiä ei lisätä, tietoa ei voida käsitellä PHP:n DOM–luokilla.

Kappaleen koodit löytyvät kokonaisuudessaan tiedostosta moduli.php.

```
if (isset($_POST['tieto'])) {

$formid = $_POST['thefid'];
$raakadata = $_POST['tieto'];
$readyhtml =  '<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-
8" /></head>'. $raakadata . '</html>';
$dom = New DOMDocument();
$dom->preserveWhiteSpace = true;
$dom->loadHTML($readyhtml);
$dom->validateOnParse = true;
```

…

Halutut tiedot haetaan läpikäymällä DOM–olioon ladattuja tietoja:

```
foreach ($dom->getElementsByTagName('div') as $element){

        $attrclass = $element->getAttribute('class');

        // haetaan rivin id
          if ($element->hasAttribute('id') &&
          preg_match("/\bcontainerRow\b/i",$attrclass) ) {
          $i++;
          $rowID = $element->getAttribute('id');
```

…

Jokainen elementti käydään läpi erikseen ja tiedot tallennetaan muuttujiin. Lopulta muuttujien sisältämä tieto tallennetaan taulukkoon:

```
$cols = array(

          'colName' => $labelInnerText,
          'colLabel' => $formAttrName,
          'colTooltip'=>$tooltipText,

          );
```

Tämän jälkeen arvot lähetetään funktiolle, joka ottaa yhteyden tietokantaan ja tallentaa ne.

```
connection_modform($cols, $rowIdNumber);
```

Ohessa kokonaisuudessaan funktio, jolle tiedot lähetetään.

```
function connection_modform($cols, $rowIdNumber){
$openConn = mysqli_connect('localhost', 'root', '', 'modx');

    $colKeys = array_keys($cols);
    $colValues = array_values($cols);

    $columKeys = "(" . implode(",", $colKeys) . ") ";
    $columValues = "('" . implode("','", $colValues) . "')";

    $rotla = '';

        foreach ($cols as $key => $value) {
            if (!empty ($cols)){

                    $flds .= $key . "=";
                    $flds .= "'" . $value . "',";

            }
        }

     // korjataan viimeinen pilkku pois
    $rotla = $flds;
    $rotla = substr($rotla, 0, -1);
```

```
 // jos yhteyttä ei saada:
$rowIdNumber1 = "'$rowIdNumber'";

    if( mysqli_connect_errno() ) {

      echo 'yhteytt&auml; tietokantaan ei saatu: ',
      mysqli_error($openConn);
      exit();

    } else {

      echo 'affected'.$openConn->affected_rows.'<br>';

      if ($openConn->affected_rows == 0) {

      $hakulause = "INSERT INTO modx_module_modform
      $columKeys VALUES $columValues ON DUPLICATE KEY UPDATE
      $rotla";

      $qresults = mysqli_query($openConn,$hakulause);

    }

    if ($qresults === TRUE){

        echo '<br>Tiedot p&auml;ivitetty<br>';

    } else {

      echo "Tietojen p&auml;ivitys kantaan ei onnistunut,
      koska: \n", mysqli_error($openConn) ;
    }
// suljetaan yhteys
mysqli_close($openConn);

}

}
```

3.5.4 Tietojen haku tietokannasta ja lomakkeen tulostaminen

Tiedot haetaan tietokannasta lomakkeen id:n avulla: kun lomakemoduulin etusivulla luodaan Formityökalulla lomake ja näpäytetään lomakkeen nimeä, haetaan kyseisen lomakkeen elementit tietokannasta tämän id:n perusteella. Tietokannan tiedot tuodaan PHP-muuttujista JavaScriptin muuttujiin ja lähetetään *formBuilder* – JavaScript -funktiolle.

Demo.php –tiedosto. Id:n siirto JavaScript –muuttujaan:

```
var thefid = '<?php if (isset($_GET['id']) ){ $formid =
$_GET['id'];} ?>';
```

Kunkin elementin tiedot haetaan yksitellen tietokannasta ennen JavaScript–muuttujiin siirtämistä. Tarvittava tieto haetaan tietokannasta olioita apuna käyttäen. Tietokantahaun suorittavat oliot löytyvät renderForm-luokasta tiedostosta testload.php. Ylin rivi pitää sisällään tiedon siitä, minkä tyyppinen HTML-elementti tullaan myöhemmin luomaan.

```
var types = '<?php $types = new renderType(); $types-
>render_modform($formid); ?>';

var names = '<?php $names = new renderName(); $names-
>render_modform($formid); ?>';

var labels = '<?php $labels = new renderLabel(); $labels-
>render_modform($formid); ?>';
```

Tiedot lähetetään formBuilder – funktiolle käsiteltäväksi:

```
build(theType,theName,theLabel …)
```

Funktio tunnistaa tyypin ja ohjaa sen eteenpäin toiselle funktiolle, joka liittää tarvittavat tiedot oikeisiin paikkoihin HTML:ä ja tulostaa HTML-elementin tietoineen näytölle.

```
if (type == "textarea"){textArea = 'roll'; …
```

Jos kyseessä on textarea, lähetetään tiedot textarean tietoja käsittelevälle funktiolle:

```
if (textArea || textArea == 'roll' ){

textarea(counter, textArea, names, labels, tooltips, requireds,
active, deleted, elementid, elementclass, elementeform);

}
```

Elementin ulkoasun tulostaminen tapahtuu samoilla funktiolla ja samalla tavoin kuin lomake-elementin luonnissa “Raahaus ja pudotus –käyttöliittymän toteutus” –kappaleessa. Jotta tietokannasta haetut tiedot saadaan paikoilleen käytetään lisäksi apuna JavaScript–olioita, ennen kuin tieto tulostetaan näytölle. Osa tiedoista tuodaan HTML-elementteihin ns. Piilokenttinä.

Koska tieto tulostetaan näytölle järjestysnumeron perusteella, ei paikkaa tarvitse etsiä, kuten raahaus ja pudotus –käyttöliittymässä, vaan tieto voidaan lisätä seuraavalla tavalla:

```
$("#content").append(formContent);
```

4 YHTEENVETO

Internetin ja internetyhteyksien tekninen kehitys on tuonut netin osaksi työpöytää. Internetistä on tulossa hiljalleen alusta yhä useammille palveluille ja sovelluksille, joita voidaan käyttää internetselaimen avulla. Selainten kehitys onkin ollut, ja tulee olemaan tässä kehityksessä isossa osassa.

Avoimen lähdekoodin sovellusten ympärille muodostuneet erilaiset web-yhteisöt ovat olleet suuressa roolissa internetpalvelujen ja sovellusten kehittämisessä. Näillä yhteisöillä ei välttämättä ole aina kaupallisia päämääriä, mutta näyttää siltä, että yritykset hyötyvät näistä sovelluksista: web-yhteisöt toimivat sovellusten kehittäjinä, testaajina, sekä osaltaan myös markkinoijina. Pieni tai keskisuuri yritys voi kehittää näitä sovelluksia omiin tarpeisiinsa ja luoda uutta liiketoimintaa sovellusten varaan. Tämän opinnäytetyön tuloksena syntynyt pieni lomaketyökalu on myös kehitetty tällaisen mallin myötä.

Työn alkuvaiheessa erittelin lomakemoduulille muutamia toiminnallisia ja teknisiä vaatimuksia. Suurilta osin pystyin toteuttamaan nuo ominaisuudet ja tuloksena oli toimiva moduuli. Suurimpana ongelmana oli kokemuksen puute vastaavista töistä. Arvioin väärin työn määrän ja siihen vaadittavan panostuksen. Eri tekniikoiden harjoitteluun oli varattava oma aikansa. Olisi ollut hyvä tehdä työn alussa erilliset dokumentit ainakin määrittelystä, suunnittelusta ja testauksesta. Nämä olisivat tuoneet varmasti esiin ongelmakohtia jo ennalta, ja auttaneet ajankäytön hallinnassa. Nyt näitä edellä mainittuja asioita tehtiin päällekkäin, mikä teki jälkeenpäin ajateltuna työskentelystä hieman vaikeaa.

Työn dokumentointia olisi myös pitänyt tehdä koko ajan työn edetessä. Näin olisi voitu varmistaa, että mahdollisten korjausten tai uusien ominaisuuksien tekeminen moduuliin olisi vaivatonta. Dokumentointi on jälkeenpäin tietysti mahdollista, mutta aikaa vievää.

Tulevaisuudessa moduulia voidaan kehittää vastaamaan paremmin varsinaisen sisällöntuottajan, eli päivitysjärjestelmän loppukäyttäjän tarpeita. Moduuliin voisi liittää myös toiminnon, jolla käyttäjä voi julkaista lomakkeita haluamallaan sivulla.

LÄHTEET

Adobe. 2011. Getting started with ActionScript. Viitattu 4.11.2011 http://help.adobe.com/en_US/flash/cs/using/WSd60f23110762d6b883b18f10cb1fe1af6-7be9a.html

Calore, M. Wired 2011. April 22, 1993: Mosaic Browser Lights Up Web With Color, Creativity. Viitattu 3.11.2011 http://www.wired.com/thisdayintech/2010/04/0422mosaic-web-browser/

Crockford, D. Crockford on JavaScript - Chapter 2: And Then There Was JavaScript. 2010. Viitattu 3.11.2011 http://yuilibrary.com/theater/douglas-crockford/crockonjs-2/

Glasner J. Wired 1999. Microsoft Leading Browser War. Viitattu 4.11.2011 http://www.wired.com/techbiz/media/news/1999/06/20067

jQuery 2011. UI/API/1.8/Draggable. Viitattu 7. 11. 2011 http://docs.jquery.com/UI/API/1.8/Draggable#event-stop

Microsoft 2011. Office Web Apps. Viitattu 4.11.2011 http://office.microsoft.com/fi-fi/web-apps/

MODx 2011. MODx Evolution Documentation. Viitattu 8.11. 2011 http://rtfm.modx.com/display/Evo1/Home

PHP 2011. Php DOM – Introduction. Viitattu 7.11.2011 http://fi.php.net/manual/en/intro.dom.php

Tapscott, D. & Williams, Anthony D. Wikinomics: how mass collaboration changes everything. 2008. Expanded ed. New York: Penguin Group: Portfolio

W3C. 2011. 1 HTML5 A vocabulary and associated APIs for HTML and XHTML. W3C Working Draft 25 May 2011. Viitattu 4.11.2011 http://www.w3.org/TR/html5/

Lomakemoduuli CD -levy

Lomakemoduulin ja moduulin asennusohjeet sisältävä CD-levy.