Esko Tapio Savolainen

# Umbrella 4: Lomakemoduuli

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Opinnäytetyö

22.9.2013

Metropolia

| | |
|---|---|
| Tekijä(t)<br>Otsikko<br><br>Sivumäärä<br>Aika | Esko Tapio Savolainen<br>Umbrella 4: Lomakemoduuli<br><br>48 sivua<br>22.9.2013 |
| Tutkinto | insinööri (AMK) |
| Koulutusohjelma | Tietotekniikan koulutusohjelma |
| Suuntautumisvaihtoehto | Ohjelmistotekniikka |
| Ohjaaja(t) | FT, lehtori Vesa Ollikainen<br>Asiantuntija, MBA Ville Mäki, Atao Oy |

Opinnäytetyön tavoitteena on kehittää Educrane Oy:n intranet- ja pilvisovellukseen, Umbrella Interactiveen, lomakemoduuli, jolla voidaan tehostaa asiakasorganisaatioiden toimintaan liittyvän tiedon selainpohjaista keruuta, käsittelyä sekä raportointia.
Umbrella Interactiven aikaisempien versioiden arkkitehtuurien ja vaikean laajennettavuuden vuoksi Educrane Oy päätti kehittää sovelluksesta uuden version, 4.0, johon työn aiheena oleva lomakemoduuli rakennettiin. Sovelluksen palvelinpuolen toiminnallisuus on kehitetty PHP-kielellä ja käyttöliittymien toiminnallisuus Javascript-komentosarjoilla.
Lomakemoduulin käyttöliittymä on aikaisempiin Umbrella Interactiven moduuleihin nähden helpompi käyttää ja sisältää uusia ominaisuuksia. Uudessa versiossa on vähennetty tarvittavia sivulatauksia, jotka johtuivat muokkauksen aikana tapahtuvista tallennus- ja latastoiminnoista.
Uuden Umbrella Interactiven lomakemoduulin käytettävyys on mielletty edeltäjäänsä huomattavasti paremmaksi, mutta parannettavaa löytyy vielä.

| | |
|---|---|
| Avainsanat | Umbrella, www, web-lomake, php, mysql, javascript |

| | |
|---|---|
| Author(s)<br>Title<br><br>Number of Pages<br>Date | Esko Tapio Savolainen<br>Umbrella 4: Form Module<br><br>48 pages<br>22 September 2013 |
| Degree | Bachelor of Engineering |
| Degree Programme | ICT |
| Specialisation option | Software Engineering |
| Instructor(s) | PhD., Senior Lecturer Vesa Ollikainen<br>Specialist MBA Ville Mäki, Atao Oy |

The purpose of this thesis was to develop a web form module to Educrane Oy's intranet and cloud application Umbrella Interactive to be used to enhance client organizations' capabilities in web based information collection, management and reporting.
Because of the limitations in architecture and difficulty in expandability in the older versions of Umbrella Interactive, Educrane Oy decided to develop a new version of it. The subject of the present work, Forms module, was built into version 4.0 of Umbrella Interactive. The server functionality of the application was developed with PHP and the user interface with Javascript.
The user interface of the form module is now easier to use and offers more features than the previous versions. The older versions required several page loads for saving and loading during editing. This has been addressed and cut to a minimum in the new version.
The usability of the new version has been found better in comparison to the older versions, eventhough there is still room for improvement.

| | |
|---|---|
| Keywords | Umbrella, www, web form, php, mysql, javascript |

**Sisällys**

# 1 Johdanto

Tässä työssä perehdytään Umbrella Interactiven dynaamiseen lomake -toiminnallisuuteen. Umbrella Interactive on Educrane Oy:n ohjelmistotuote joka on tarkoitettu yrityksen sisäiseen viestintään sekä työohjeiden sekä laatudokumentaation ylläpitoon. Lomakkeet Umbrella Interactivessa toimivat kuten reaalimaailmassakin: tietoa keräävä taho laatii järjestelmään lomakkeen vastauspohjan, joita annetaan toimijoille täytettäväksi. Lomakkeilla voidaan toteuttaa esimerkiksi anonyymejä palautekyselyitä, verkko-oppimisen pisteytettäviä kokeita tai seurata prosessin edistymistä yrityksessä.

Työssä keskitytään pääasiallisesti lomakkeiden käyttöliittymään sekä käytettyyn tallennustapaan. Tallennustavalle haasteita asettaa muutokset vastauspohjassa joka ei saa vaikuttaa jo täytettyjen lomakkeiden tietojen häviämiseen.

Luvussa 2 esitellään Educrane Oy, yrityksen ohjelmistotuote Umbrella Interactive ja kerrotaan lyhyesti sen sisältämistä modulaarisista toiminnallisuuksista. Tavoitteissa kerrotaan työn kohteena olevan moduulin vaatimuksista tarkemmin ja läpikäydään siihen liittyviä haasteita.

Kolmannessa luvussa tutkitaan muiden, jo olemassa olevien, kilpailevien ratkaisujen ominaisuuksia ja vertaillaan niitä Umbrella Interactiveen. Luvussa perustellaan myös, miksi Umbrella Interactive on asiakasyritykselle kannattavampi vaihtoehto kuin kilpailijansa.

Umbrella Interactiven käyttämiä kolmansien osapuolien tuottamia sovelluksia ja tekniikoita käydään läpi työn neljännessä luvussa. Luvussa esitellään myös työn aiheena olevan lomakemoduulin termistöä.

Viidennessä luvussa perehdytään Umbrella Interactive -sovelluksen aiempiin versioihin, niiden välisiin eroihin sekä tarpeisiin, joiden perusteella on päätetty aloittaa uuden arkkitehtuuriversion suunnittelu ja toteutus.

Työn kuudennessa luvussa käydään tarkemmin läpi työn varsinaista aihetta, toteutetun lomakemoduulin ominaisuuksia, käyttöliittymää, toimintalogiikkaa ja tietorakenteita.

Seitsämännessä luvussa kerrotaan valmiin moduulin käyttökokemuksista, havaituista ongelmista sekä seuraavien versioiden jatkokehitysideoista.

# 2 Tausta ja tavoitteet

## 2.1 Educrane Oy ja Umbrella Interactive

Educrane Oy on konsultointi- ja teknologiayritys joka on erikoistunut hallinta- ja laatujärjestelmiin. Educranen tarjoama tuote, Umbrella Interactive, tarjoaa työkalut yrityksen laatujärjestelmän ylläpitoon, prosessien kuvaukseen, raportointiin sekä verkko-oppimisjärjestelmän.

Umbrella Interactive on modulaarinen web-ohjelmisto, joka koostuu moduuleista kuten sivuista, dokumenteista, lomakkeista, raporteista ja työjonosta. Sivuilla järjestelmään luodaan luettavaa sisältöä ja sitä voidaan täydentää dokumenteilla, jotka toimivat liitetiedostojen tapaan. Lomakkeilla ja raporteilla voidaan kerätä tietoa ja tuottaa johdettua tietoa esimerkiksi johtajien tarpeisiin. Työjonon avulla voidaan toteuttaa järjestelmän sisäiset työt, kuten uusien sivujen oikeellisuuden tarkistus ja julkaisu.

## 2.2 Työn tavoitteet

Työn tavoitteena on suunnitella ja toteuttaa lomakemoduuli Umbrella Interactiven versioon 4. Lomakemoduulin tulee toteuttaa kaikki Umbrella Interactive -version 3.X toiminnot sekä tarjota laajemmat jatkokehitysmahdollisuudet joilla voidaan tulevaisuudessa laajentaa lomakkeiden toiminnallisuuksia. Keskeisin uudistus lomakemoduulissa on dynaamisempi käyttöliittymä, jonka tarkoituksena on vähentää tarvittavien sivulatausten määrää eli hoitaa kaikki muokkaustoimenpiteet yhdellä sivulatauksella ja tallennuksella.

Laajemmilla jatkokehitysmahdollisuuksilla halutaan jättää Educrane Oy:lle mahdollisuus jatkaa lomakemoduulin kehitystä tulevaisuuden asiakastoiveiden mukaan. Moduulin käyttämät luokat ja tietorakenteet on suunniteltava geneerisiksi, yleiskäyttöisiksi sekä mahdollisimman joustaviksi, jolloin sovelluslogiikan muuttaminen ja uusien toiminnallisuuksien lisääminen myöhemmin on helpompaa.

Lomakemoduulin on oltava käytettävissä niillä selaimilla, jotka on ilmoitettu Umbrella Interactive 4:n tukemiksi. Tällaisia selaimia ovat Internet Explorer 7.0 ja 8.0, Mozilla Firefox 3.0 sekä Chrome 2.0. Valitut vähimmäisvaatimukset tuetuille selaimille aiheuttavat potentiaalisia haasteita kehitykselle, sillä niiden tukemat ominaisuudet ovat suhteessa rajallaset uudempiin selaimiin. Vanhemmat selainversiot kuitenkin valittiin tuettujen joukkoon johtuen yritysten hitaasta siirtymätahdista uudempien versioiden pariin.

Tässä työssä sovelletaan yleisiä verkkotekniikoita ja kolmansien osapuolten ohjelmistoja, kuten XHTML-kuvauskieli, Javascript-komentosarjat, AJAX-tiedonsiirrot sekä PHP-ohjelmointikieli. Kyseisillä tekniikoilla voidaan toteuttaa moderneja selainpohjaisia sovelluksia.

## 3 Olemassa olevat ratkaisut

Umbrella Interactiven lomakemoduulin toiminnallisuutta vastaavia verkossa toimivia lomakejärjestelmiä on markkinoilla tarjolla niin ilmaisia kuin kaupallisiakin tuotteita. Tällaisia ovat esimerkiksi Solinum Oy:n sähköiset lomakkeet [1] tai CoffeeCup Softwaren Web Form Builder [2]. Mainitut järjestelmät ovat kuitenkin erikoistuneet vain lomaketoiminnallisuuteen, siinä missä Umbrella Interactiven kantava idea on tarjota yritykselle mahdollisimman kattavasti kaikki tarvittavat työkalut yrityksen sisäiseen viestintään, tiedonkeruuseen sekä raportointiin.

Umbrella Interactiven lomakemoduuli sisältyy Umbrellaan Asiakkaan ei tarvitse tiedonkeruuta ja raportointia varten ottaa käyttöön enää toisia sovelluksia. Etuina muihin lomake- ja raportointijärjestelmiin Umbrella Interactiven lomakemoduuli käyttää järjestelmän keskitettyä käyttäjien ja käyttäjäryhmien hallintaa, jolloin käyttäjätunnuksia ja pääsyoikeuksia ei tarvitse luoda useisiin sovelluksiin.

## 4 Termistö ja tekniikat

Koska Umbrella Interactive on web-pohjainen sovellus, tarvitaan useita erilaisia tekniikoita, jotta järjestelmä toimisi, kuten www-palvelinsovelluksen, kooditulkin sekä käyttö-

liittymän esittämisestä vastaavan kuvauskielen. Seuraavissa luvuissa käydään läpi järjestelmään liittyvä sanasto sekä käytetyt tekniikat.

## 4.1 Termistö

Umbrella Interactiven lomakemoduulin termistö on pyritty pitämään mahdollisimman loogisena ja vastaavana kuin mitä fyysisissä paperilomakkeissa on totuttu käyttämään. Kaikki lomakemoduulin konseptit eivät kuitenkaan ole peräisin perinteisistä paperilomakkeista, joten järjestelmälle ominaisia termejä on syntynyt, ja ne ovat selitettynä taulukossa 1.

Taulukko 1. Umbrella Interactiven ja lomakemoduulin sanasto

| | |
|---|---|
| Umbrella, Umbrella Interactive | Educrane Oy:n tuottama web-pohjainen toimintajärjestelmäsovellus. |
| Rekisteri | Vanhempien Umbrella-sovellusten nimitys lomakepohjille.<br>ks. lomakepohja |
| Lomakepohja | Käyttäjän määrittelemä kokonaisuus vaiheita ja kenttiä. |
| Lomake | Lomakepohjan perusteella muodostettu web-lomake (form), johon käyttäjä syöttää tietoa. |
| Vaihe | Kokoelma lomakkeen kenttiä, jolla voidaan määritellä myös lomakkeen täyttöjärjestys. |
| Kenttä | Lomakkeen komponentti, joka vastaanottaa, validoi ja tallentaa käyttäjän syöttämää tietoa. |
| Vaiheistus | Lomakepohjan toimintamalli, jolla voidaan määritellä lomakkeelle käyttäjiä, jotka täyttävät lomakkeesta heille kuuluvat vaiheet. |
| Työjono | Umbrella Interactiven ylläpitämä, käyttäjäkohtainen lista järjestelmän sisällä suoritettavista työtehtävistä, joita käyttäjälle on annettu. |
| Lomakkeen sulkeminen | Suljettu lomake ei ole enää käyttäjien muokattavissa tai jatkotäytettävissä. Yleensä lomakkeet suljetaan, kun niiden käsittely on saatettu loppuun. |

## 4.2 Tekniikat

Seuraavissa luvuissa käsitellään tekniikoita, joiden päälle Umbrella Interactiven toiminnallisuudet on rakennettu. Käytetyt tekniikat ovat yleiskäyttöisiä ja soveltuvat erilaisiin käyttökohteisiin. Umbrella Interactiven kehityksen kannalta tärkeää on ollut avoimuus ja

jatkuvuus. Tällä on pyritty takaamaan se, että käytetyt tekniikat eivät tule poistumaan käytöstä ylläpitonsa tai lisensoinnin takia.

### 4.2.1 UTF-8

UTF-8, 8bit Unicode Transformation Format, on käytettyjen kirjoitusmerkkien koodaustapa ja on taaksepäin yhteensopiva ASCII:n (American Standard Code for Information Interchange, myös merkistökoodaustapa) kanssa. UTF-8 on saavuttanut suosiota oletuskoodauksena www-sivuilla ja sähköposteissa, sillä sen avulla on helppo esittää lähestulkoon mitä hyvänsä merkkejä, jopa laajennettuja 32-bittisiä unicode-merkistön symboleja.

UTF-8:ssa ensimmäisen tavun ensimmäiset bitit ilmaisevat, onko koodattu merkki laajennettu, eli ASCII-standardin 127 ensimmäisen merkin ulkopuolella. Esimerkiksi merkki 'a' tarvitsee vain yhden tavun tallennettaessa, mutta 'ä' tarvitsee kaksi. Merkin koodauksessa käytettyjen tavujen määrä ilmaistaan ensimmäisen tavun eniten merkitsevien bittien ykkösten määrällä. Ykkösjono ja itse merkkidata erotetaan ykkösiä seuraavalla nolla-bitillä (taulukko 2). [3.]

Taulukko 2. UTF-8

| Tavu 1 | Tavu 2 | Tavu 3 | Tavu 4 | |
|---|---|---|---|---|
| 0xxxxxxx | - | - | - | Yksitavuiset merkit: a-z, A-Z, joitakin symboleja (esim. $), ASCII |
| 110yyyxxx | 10xxxxxx | - | - | Kaksitavuiset merkit: Aksentit (esim. ä, é, ŝ) |
| 1110yyyy | 10yyyyxx | 10xxxxxx | - | Kolmetavuiset merkit: Erilaisia symboleja, kuten € |
| 11110zzz | 10zzyyyy | 10yyyyxx | 10xxxxxx | Neljätavuiset merkit: mm. aasialaisia kirjoitusmerkkejä. |

Umbrella Interactiven kannalta UTF-8 merkistö oli käytännöllinen valinta, koska sovellukseen haluttiin jättää mahdollisuus lokalisoida käyttöliittymä mahdollisimman monelle kielelle, jolloin sovelluksen pitää pystyä esittämään suuri joukko erilaisia kirjoitusmerkkejä.

### 4.2.2 PHP

Php eli ”PHP: Hypertext Preprocessor” on pääosin web-palvelujen tuottamiseen tarkoitettu komentosarjakieli. PHP:in ominaisuuksiin kuuluvat käyttöjärjestelmäriippumattomuus, olio- ja proseduraaliset paradigmat, heikko tyypitys sekä laaja luokkakirjasto. PHP-sovellukset suoritetaan yleensä www-palvelimella ja saatu tuloste (yleensä HTML tai vastaava) lähetetään asiakkaan selaimelle näytettäväksi. [4.]

Heikosta tyypityksestään johtuen muuttujien arvo on riippuvainen kontekstistaan. Esimerkiksi tyhjät merkkijonot ja lukuarvo 0 voivat käyttäytyä vertailuissa epätosina (kuva 1).

```
<?php
// muuttujien arvojen asetus
$i = 0;
$k = 9001;
$s = ”testi”;
if ( !$i && $k && $s ) // muuttujien vertailu
{
  echo ”Tämä tulostuu koska:”;
  echo ”\$i on nolla (epätosi)”;
  echo ”\$k on nollaa suurempi (tosi)”;
  echo ”\$s ei ole tyhjä merkkijono”;
}

if ( $i === false )
{
  echo ”Ei tapahdu, koska === vertailee myös tietotyyppiä ja
numero ($i) ei ole totuusarvo (boolean)”;
}
?>
```

Kuvio 1. Esimerkki PHP-koodista

Umbrella Interactiven toteutuskieleksi PHP valittiin sen saatavuuden ja avoimuuden takia. PHP-tulkki on saatavilla eri käyttöjärjestelmille ja www-palvelinsovelluksille. Avoimen lähdekoodin ja lisensointinsa ansiosta PHP:n kehitys ja ylläpito on luotettaval-la pohjalla.

### 4.2.3 XML

XML:n suunnittelutavoitteina oli saada luotua yksinkertainen ja joustava tietoformaatti, jolla voidaan esittää ja siirtää jäsennettyä tietoa internetissä ohjelmien ja järjestelmien

välillä. Kieli on syntaksiltaan samankaltainen kuin HTML ja on siten myös ihmisluettavaa.

Dokumentti koostuu elementeistä, jotka ovat käytännössä säiliöitä toisille elementeille, tekstille tai attribuuteille. Elementeillä pyritään kuvaamaan hierarkista rakennetta jossa isommat kokonaisuudet koostuvat pienemmistä osasista. Elementti itsessään koostuu tagista, attribuuteista ja sisällöstä. Tagi voi olla esimerkiksi <foo>, joka on foo-elementin avaava tagi. Elementit kuuluu aina myös sulkea; sulkeminen tapahtuu joko sulkevalla tagilla </foo> tai kirjoittamalla avaustagi muotoon <foo />, jolloin elementti suljetaan heti ja se ei voi täten sisältää toisia elementtejä tai tekstisisältöä. [5; 6; 7.]

Attribuutilla voidaan antaa lisää tietoa elementistä. Esimerkiksi <foo attribuutti=”arvo” />. Attribuutin tarjoaman tarjoaman tiedoin voi myös esittää elementtinä. [8.]

Kuvassa 2. esitellään XML-dokumenttia jossa on kuvattu yksinkertainen, geneerinen dokumentti sekä käytännön esimerkki musiikkikirjaston mahdollisesta esitystavasta. [Kuva 2.]

```
<!-- geneerinen esimerkki XML-dokumentista -->
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<esimerkkidokumentti>
  <tagi attribuutti=”arvo”>
    Hei maailma
  </tagi>
</esimerkkidokumentti>

<!-- Käytännön esimerkki -->
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<musiikkikirjasto>
  <esittäjä nimi=”Django Reinhardt”>
    <levy nimi=”Nuages” vuosi=”1988”>
      <kappale numero=”1” nimi=”Nuages”
uri=”file:///media/data/foo1.mp3” />
      …
    </levy>
    <levy nimi=”The Best of Django Reinhardt” vuosi=”1996”>
      …
    </levy>
  </esittäjä>
</musiikkikirjasto>
```

Kuvio 2. Esimerkki XML-dokumentista

XML-dokumenteille voidaan määritellä myös DTD, Document Type Definition, jolla voidaan määritellä miten dokumentin tulee rakentua. Käyttämällä sopivaa DTD-määrittelyä voidaan XML-dokumentista muodostaa esimerkiksi tietyt muotosäännöt XHTML, josta kerrotaan seuraavassa luvussa.

### 4.2.4 XHTML

XHTML, Extensible Hypertext Markup Language, on www-sivujen sisällön rakentamiseen käytetty merkintäkieli, joka on kehitetty vanhemman HTML:n pohjalta. XHTML täyttää XML:n muotovaatimukset ja on täten vaativampi syntaksin suhteen kuin HTML. XHTML eroaa HTML:sta myös laajennettavuutensa ansiosta. Tarvittaessa voidaan määritellä kokonaan uusia elementtejä. [6; 9.]

Kuvassa 3 esitellään XHTML-muotoinen esimerkkisivu joka sisältää sivun nimen sekä yksinkertaisen Hei maailma sisällön.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" lang="fi"
xml:lang="fi">
  <head>
    <meta http-equiv="content-type" content="text/html; char-
set=UTF-8" />
    <title>Esimerkki XHTML-sivusta</title>
  </head>
  <body>
    <p>Hei maailma</p>
  </body>
</html>
```

Kuvio 3. Esimerkki XHTML-sivusta

XHTML valittiin Umbrella Interactiven käyttöliittymien kuvauskieleksi koska se on hyvin tuettu eri selaimissa ja sen muotovaatimusten vuoksi käyttöliittymän lähdekoodi pysyy yhtenäisen näköisenä ja selkeälukuisena.

### 4.2.5 CSS 2.1

Cascading Style Sheet on tekniikka, jolla XML-tyyppisen tiedon ulkoasu määritellään. CSS:n kantava idea on, että sisältö ja sen muotoilu erotetaan, jolloin sisältö voidaan esittää erityylisesti riippuen esimerkiksi esitysmediasta. Esimerkiksi paperitulosteisiin voidaan haluta erilainen ulkoasu kuin tietokoneen ruudulle (kuva 4). [10.]

```
<style type="text/stylesheet" media="screen">
  /* Tyylit näytöllä esitettävälle tervehdykselle */
  p.greeting {
    font-variant: cursive;
  }
</style>
<style type="text/stylesheet" media="print">
  /* Tyylit paperitulosteen tervehdykselle */
  p.greeting {
    font-variant: bold;
}
</style>
<p class="greeting">
  Tämä teksti on näytöllä kursivoitua ja
  tulostettaessa lihavoitua
</p>
```

Kuvio 4. Kappaleelle on määritelty kursivoitu fontti tietokoneen ruudulle ja lihavoitu tulosteelle

Www-sivustojen ulkoasun määrittelyihin ei ole monia erilaisia tapoja, joten CSS oli Umbrella Interactiven kehityksen kannalta ainoa mahdollinen, sillä ainoa toinen vaihtoehto olisi ollut määritellä kiinteästi jokaisen käyttöliittymäkomponentin ulkoasu XHTML-sivujen lähdekoodiin. Tämä olisi tehnyt kehityksestä ja ylläpidosta huomattavan työlästä.

### 4.2.6 Javascript ja Ajax

Javascript on komentosarjakieli, jolla voidaan lisätä toiminnallisuutta ja muokata sisältöä muuten staattisissa HTML/XHTML-dokumenteissa. Javascriptin syntaksi on C-kielen tyyppistä ja nimestään poiketen sillä ei ole juurikaan mitään tekemistä Java-ohjelmointikielen kanssa (kuva 5). [11; 12.]

```
function helloWorld() //funktion esittely
{
  var message = ”Hei maailma!”; //muuttujan määrittely
  alert(message); //funktion kutsu parametrin kanssa
}
helloWorld(); //funktion kutsu

var Olio = function() //luokan konstruktori
{
  this.data = ''; //jäsenmuuttujan alustus
};
Olio.prototype = { //luokan jäsenfunktiot
  funkA: function(parametri)
  {
    this.data = parametri; //luokan jäsenmuuttujan arvon asetus
    this.funkB(” world”); //luokan jäsenfunktion kutsu
  },
  funkB: function(parametri)
  {
    alert(this.data + parametri); //näyttää tekstin
  }
};

var o = new Olio(); //luokan instanssin luonti

// luokan jäsenfunktion kutsu, näyttää ”Hello World” -
ilmoituksen
o.funkA(”Hello”);
```

Kuvio 5. Esimerkki Javascriptistä

AJAX (Asynchronous Javascript And XML) on kokoelma tekniikoita, joilla esimerkiksi selaimessa esitetyn web-sivun käyttöliittymä voi vaihtaa tietoa palvelimen kanssa, tarjoten näin käyttäjälle uutta sisältöä lataamatta koko sivuston näkymää uudestaan ja näin ollen parantaen sivuston vuorovaikutteisuutta. Nimensä mukaisesti AJAX yleensä käyttää XML-kieltä tiedonsiirtoon palvelimen kanssa, mutta käytännössä voidaan käyttää mitä hyvänsä tietomuotoa. Esimerkkeinä ovat JSON tai raaka tekstidata. Kuvassa 6. on AJAX esimerkki jossa käyttäjälle näytetään hänen valitsemansa tuotteen hinta.

```
function getProductPrice(productID)
{
  /* Suorittaa pyynnön kohteeseen ./ajax/handler.php palveli-
mella ja toimittaa saadun vastineen showProductPrice() -
funktiolle parametrina */
  new Ajax.Request('./ajax/handler.php', {
    method: 'post',
    encoding: 'UTF-8',
    parameters: {
      'mode': 'product',
      'id': productID
    },
    onSuccess: showProductPrice //viite funktioon
  });
}
function showProductPrice(result)
{
  var data = eval('(' + result.responseText + ')');
  alert('Tuote maksaa ' + data['productPrice']);
}
getProductProce(1337); //suoritetaan
```

Kuvio 6. Esimerkki AJAX:n käytöstä Prototype-kirjaston kanssa

Kuvan 6 esimerkissä getProductPrice()-funktiossa luodaan uusi Ajax-pyyntö joka ohjataan ./ajax/handler.php-osoitteeseen. Pyynnössä lähetetään POST-datana parametrit mode ja id, joiden perusteella pyynnön käsittelevä PHP-skripti etsii oikean tuotteen ja palauttaa siihen liittyvät tiedot. Palautetut tiedot käsitellään showProductPrice()-funktiossa, jossa haettu data jäsennetään javascriptin ymmärtämiksi muuttujiksi ja esitetään käyttäjälle.

Javascript valittiin Umbrella Interactiven käyttöliittymän toiminnallisuuden toteuttamiseen koska se ei vaadi selaimiin erikseen asennettavia lisäsovelluksia. Kielen ongelmiksi tunnistettiin eri selainten väliset erot komentosarjojen suorituksessa, joten kehityksen yksinkertaistamiseksi päätettiin käyttää myös javascript kirjastoja, kuten Prototype ja Jquery, jotka esitellään myöhemmissä luvuissa.

### 4.2.7 JSON

JSON (JavaScript Object Notation) on tapa esittää tietoa tietorakenteina ja assosiatiivisina taulukoina, kuitenkin niin, että se on helppoa lukea ilman erillistä tulkkia. JSON:ia voidaan käyttää tiedon tallentamiseen ja käsittelyyn javascriptissa kuin myös tietomuotona AJAX-tiedonsiirrossa (kuva 7). [13; 14.]

```
var json_esimerkki = {
  ”teksti”: ”tekstiähän se”,
  ”numero”: 42,
  ”objekti”: {
    ”avain”: ”arvo”,
    ”key”: ”value”
  },
  23: ”hei maailma”
};
//Kumpikin seuraavista näyttää ilmoituksen ”tekstiähän se”
alert(json_esimerkki.teksti);
alert(json_esimerkki[”teksti”]);

//näyttää ilmoituksen ”hei maailma”
alert(json_esimerkki[23]);
```

Kuvio 7. Esimerkki JSON:ista

### 4.2.8 Sarjallistus (serialisointi)

PHP:ssa, kuten muissakin ohjelmointikielissä, tietoa voidaan sarjallistaa, jolloin se on helposti tallennettavissa ja ladattavissa esimerkiksi tietokantaa käytettäessä. Sarjallistetun datan kanssa on oltava varma, ettei käytä useampaa merkistöä, sillä esimerkiksi monitavuiset UTF8-merkit muuttuvat ASCII-merkistöä käyttäessä useammiksi merkeiksi, jolloin merkkijonojen pituudet eivät enää vastaa tarkoitettua. Tällöin sarjallistuksen muuntaminen takaisin alkuperäiseksi tiedoksi ei onnistu.

Kuvassa 8. esitellään taulukon sarjallistus merkkijonoksi PHP-kielellä. Taulukkoon sijoitetaan eri tietomuotoisten indeksien alle erilaisia syötteitä jotta voitaisiin havainnollistaa sarjallistetun tiedon tietotyyppien esitystapa.

```
//taulukoitua tietoa
$a = array(
  0 => 'hei',
  'avain' => array('m','a','a'),
  42 => true
);

//sarjallistaminen
$s = serialize($a);

//tulostetaan sarjallistettu tieto
echo $s;

/* tuloste näyttää tältä:
a:3:{i:0;s:3:"hei";s:5:"avain";a:3:{i:0;s:1:"m";i:1;s:1:"a";i:2;
s:1:"a";}i:42;b:1;}
*/

//sarjallistetun tiedon muuntaminen takaisin taulukoksi:
$b = unserialize($s);
```

Kuvio 8. Esimerkki sarjallistamisesta PHP:ssä

### 4.2.9 Prototype

Prototype on JavaScript-sovelluskehys, joka tarjoaa suuren määrän valmiita luokkia ja funktiota, kattavat työkalut olio-ohjelmointiin ja XML-muotoisen tiedon käsittelyyn. [15]

Esimerkki luokan määrittelystä ja sen ilmentymän luomisesta on kuvassa 9. Luokalle määritellään konstruktori sekä yksi jäsenfunktio joka näyttää ilmentymälle annetun tunnisteen. [Kuva 9.]

```
//luokan määrittely
var Luokka = Class.create({
  //Konstruktori
  initialize: function(id)
  {
    this.id = id;
  },
  //jäsenfunktio
  whoIs: function()
  {
    //näyttää ikkunan jossa lukee tämän olion id.
    alert(this.id);
  }
});

//ilmentymän luominen
var luokka = new Luokka(4);

//olion metodien kutsuminen
luokka.whoIs();
```

Kuvio 9. Esimerkki Prototypestä

### 4.2.10 JQuery

JQuery on modulaarinen sovelluskehys JavaScriptille, joka on luotu helpottamaan Javascriptin tuottamista tarjoamalla helpot tavat jäsennellä HTML-dokumenttia ja tarkkailla käyttöliittymätapahtumia. JQuery sisältää myös laajan kirjon erilaisia animointirutineeja. [16.]

```
<div id=”animoituva” style=”display:none;”>
  Hei maailma
</div>
<script type=”text/javascript”>
    //Koodi $(function(){}); sisällä suoritetaan sivun latau-
duttua kokonaan
    $(function() {
    /*
    * Animoi liu'uttamalla 1,5s aikana esiin
    * <div> -elementin jonka id on ”animoituva”
    */
    $('div#animoituva').slideDown(1500);
  });
</script>
```

Kuvio 10. Esimerkki JQuerystä

### 4.2.11 TinyMCE

TinyMCE Moxiecode Systems AB:n kehittämä, javascriptillä toteutettu, helppokäyttönen tekstinkäsittelykomponentti web-sivuille, jolla voi tuottaa ja muokata HTML-muotoista sisältöä. Ominaisuuksiltaan TinyMCE vastaa lähes täysin keskivertokäyttäjän tarpeita tekstinkäsittelyssä.

TinyMCE on myös täysin muokattavissa, oletustekstieditoriin voi tuoda uusia toiminnallisuuksia liitännäisten avulla. Mahdollisia liitännäisiä ovat muun muassa tiedosto- ja kuvaselaimet, jotka mahdollistavat esimerkiksi läpinäkyvän tiedostojen siirtämisen palvelimelle ja liittämisen muokattavaan sisältöön. Vakiona TinyMCE:stä löytyy kuitenkin tarpeeksi ominaisuuksia lähes kaikkeen tarvittavaan. [17.]

Yksinkertaisimmillaan TinyMCE ilmentymän luominen tapahtuu määrittelemällä Javascriptissä tinyMCE-luokalle HTML-elementit joihin tekstieditori halutaan sekä käytet-

tävä työkalupaletti. Kun tinyMCE:n komentosarja on suoritettu onnistuneesti, ilmestyy määriteltyjen elementtien tilalle tekstieditori (kuva 11, kuva 12).

```
<script type="text/javascript" src="./tiny_mce.js"></script>
<script type="text/javascript">
tinyMCE.init({
  'mode': "textareas", //mihin elementteihin editori kohdiste-
taan
  'theme': "simple" //käytettävä työkalupaletti
});
</script>
<form method="post" action="somepage">
  <textarea name="content" style="width:100%">
    TinyMCE is a platform independent web based...
  </textarea>
</form>
```

Kuvio 11. Esimerkki TinyMCE -ilmentymän perustamisesta


Kuvio 12. Esimerkki TinyMCE:n käyttöliittymästä

TinyMCE oli luonteva valinta muotoiltavan tekstin syöttämiseen Umbrella Interactivessa, sillä kyseisen toiminnallisuuden kehittäminen ja ylläpito itse olisi syönyt suunnattomia määriä henkilöstöresursseja.

#### 4.2.12 MySQL

MySQL on Oracle Corporationin omistama avoimenlähdekoodin SQL- tietokantapalvelinsovellus. Sovellus tukee useita erilaisia tietokantamoottoreita, kuten MyISAM ja InnoDB, joista jälkimmäisellä on mahdollisuus hyödyntää viite-eheyksien valvontaa. [18; 19]

Umbrella Interactive pystyy käyttämään muitakin tietokantapalvelinsovelluksia tietojensa säilyttämiseen ja ylläpitoon, mutta toistaiseksi MySQL-palvelin ja InnoDB-tietokantamoottorit ovat ainoat käytössä olevat ratkaisut.

## 5 Umbrella Interactive ja kehityshistoria

Umbrella tarjoaa yrityksille toimintajärjestelmän, joka soveltuu johtamis-, toiminta- ja laatujärjestelmäksi, jota 90-100 % yrityksen työntekijöistä voi käyttää. Saman ohjelmiston sisälle voidaan rakentaa yrityksen sisäinen intranet, johtajien työkalut yrityksen luotsaamiseen ja toteuttavan osapuolen laatudokumentaaatio sekä koulutus- ja raportointijärjestelmä. Järjestelmä sisältää monipuoliset käyttäjien ja käyttäjäryhmien hallinnan ja oikeuksien jakamisen, joiden avulla sisällön näkyvyyttä voidaan tarvittaessa rajata tai laatia niin, että se näkyy vain niitä tarvitseville tahoille.

Umbrella Interactive on sovelluksena käynyt läpi erilaisia kehitysvaiheita, jotka käydään seuraavissa luvuissa läpi. Luvuissa kerrotaan lyhyesti kunkin sovellusversion ominaisuuksista ja syistä jotka johtivat uuden version kehittämiseen.

### 5.1 Versio 1.0

Umbrellan ensimmäinen versio laadittiin nopealla aikataululla asiakkaiden toiveiden pohjalta. Ohjelmistoon sisältyivät sivut, dokumentit, koostesivut, lomakkeet ja raportointi.

Ensimmäisen version toteutetus oli vakaa, mutta sisältömäärän kasvaessa useisiin satoihin sivuihin ja dokumentteihin alkoi ohjelman toiminta hidastua merkittävästi. Järjestelmän sisäisten tietorakenteiden tallennus- ja käsittelytavat eivät mahdollistaneet

kattavia jatkokehitysmahdollisuuksia, joten uusien asiakastoiveiden kertyessä syntyi tarve suunnitella uutta versiota, joka vastaisi uusiin vaatimuksiin.

## 5.2 Versiot 2.0 ja 3.0

Umbrella Interactive päätettiin kirjoittaa alusta asti uudelleen syntyneestä tarpeesta saada ohjelmaan lisää toimintanopeutta ja toiminnallisuuksia, joihin vanhempi arkkitehtuuri ei järkevässä ajassa tehtävin muutoksin pystynyt. Olemassaolevilla asiakkailla oli tarve uudelle versiolle, joten päädyttiin uudelleenkirjoittamaan vain olemassaoleva toiminnallisuus. Versio 2.0 tehtiin ulkoasultaan vastaamaan mahdollisimman pitkälti versiota 1.0, ettei vanhoja käyttäjiä etäännytettäisi. Lähes kaikki lähdekoodi tehtiin uudestaan erilaisella, optimaalisemmalla arkkitehtuurilla. Uusi arkkitehtuuri oli hyvin suunniteltu, sillä paikoin versio 2.0 oli jopa tuhansia kertoja nopeampi aikaisempaan verrattuna.

Ohjelmointityö suoritettiin nopealla aikataululla, jotta saataisiin asiakkaille ohjelmisto, jolle voitaisiin tarjota toimivampia tukipalveluita ja jatkokehitysmahdollisuuksia. Versio 2.0 oli lopulta niin ylivoimainen tuote verrattuna aiempaan versioon, että se päätettiin markkinointimielessä korottaa versioksi 3.0.

## 5.3 Versio 4.0

Umbrella Interactive 3 oli vakaa ja oli aika alkaa suunnitella uusia, toivottuja ominaisuuksia järjestelmään. Jälleen kiirehdityn ohjelmointityön tuloksina uusien toiminnallisuuksien lisääminen ei sopinut arkkitehtuuriin ja tietokantaan ilman suuria muutoksia ohjelmiston toimintalogiikkaan. Umbrella 4.0 päätettiin suunnitella alusta uudestaan mahdollisimman joustavaksi laajennusten suhteen samalla pitäen kaiken olemassa olevan toiminnallisuuden.

Versiossa 4.0 kantavana ajatuksena oli toteuttaa käyttöliittymä suurin osin mahdollisimman interaktiiviseksi ja dynaamiseksi, tehden mahdollisimman vähän sivulatauksia käyttäen JavaScript- ja AJAX-tekniikoita. Uusilla tekniikoilla pystyttiin lataamaan sivuille sisältöä vasta tarvittaessa, pienissä erissä, jolloin järjestelmää pystyttäisiin käyttämään nopeasti hitaampienkin internetyhteyksien yli. Versio 4.0 korvasi vanhemman version hyvin nopealla aikataululla julkaisun jälkeen.

# 6 Toteutettu lomakejärjestelmä

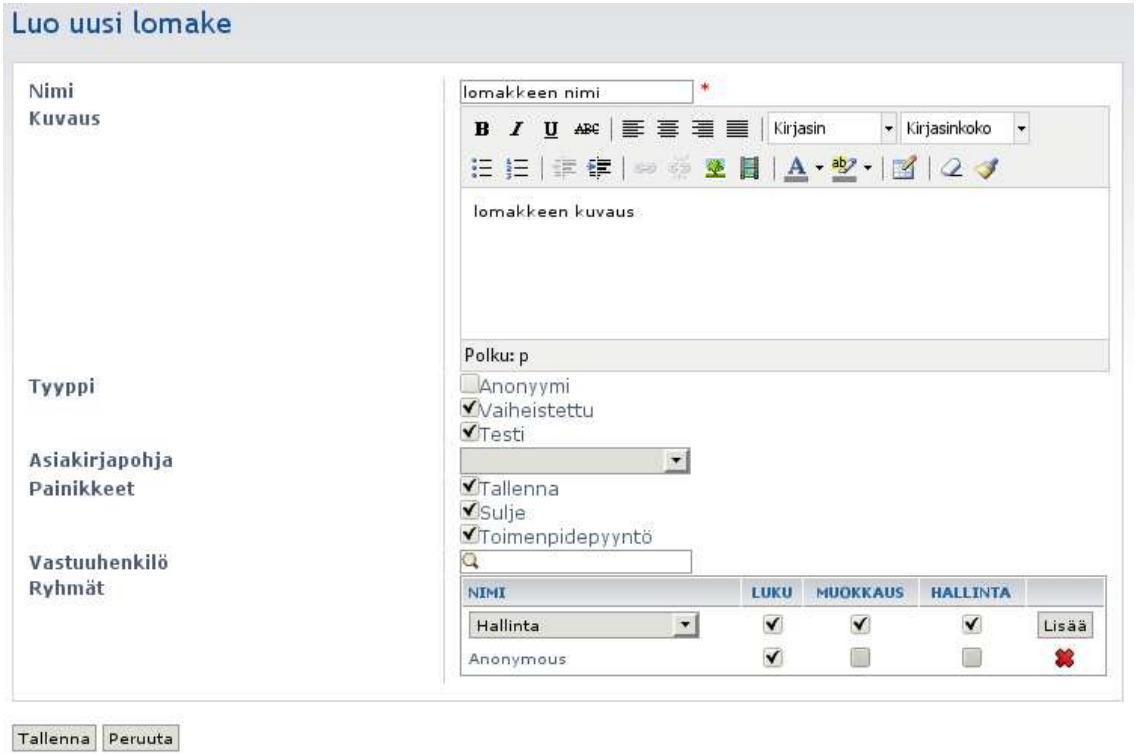
Umbrellan lomaketoiminnallisuudella mahdollistetaan joustava tietojen keräys ja raportointi järjestelmän sisällä. Lomakkeilla voidaan toteuttaa esimerkiksi asiakaspalautteen kerääminen, tilausprosessin seuranta, työilmapiirikysely ja pitää perehdytysjakson päättävä pisteytettävä koe.

Lomakkeille asetettuja kriteerejä ovat joustava lomakepohjan muokattavuus ilman jo syötettyjen tietojen katoamista, mahdollisuus jättää lomake keskeneräiseksi myöhempää työskentelyä varten, mahdollisuus lähettää lomake toiselle käyttäjälle jatkokäsiteltäväksi ja mahdollisuus merkitä lomake käsitellyksi arkistointia varten. Arkistoitua lomaketta kutsutaan suljetuksi, eikä sitä voi muokata. Lomakepohjaan määritellyillä hallintaryhmien käyttäjillä on mahdollisuus avata suljettuja lomakkeita halutessaan uudelleen muokattaviksi.

Lomaketta voidaan siirtää käyttäjältä toiselle toimenpidepyyntöjen avulla. Lähettäessä voidaan vastaanottava käyttäjä valita käyttäjä lomakkeeseen liitetyistä ryhmistä. Lähetetty lomake ilmestyy uuden käyttäjän työjonoon ja hänelle lähetetään tapahtumasta ilmoitus sähköpostitse, mikäli käyttäjä on hyväksynyt Umbrellan viestien vastaanottamisen.

## 6.1 Lomakkeen luonti ja asetukset

Lomakkeen luonti tapahtuu lomakkeen luonti -näkymässä. Lomakkeelle pakollisia tietoja on luotaessa vain nimi. Kuvaukseen lomakkeen laatija voi halutessaan kirjoittaa täyttöohjeita tai muuta hyödyllistä informaatiota tuleville lomakkeen täyttäjille (kuva 13).




Kuvio 13. Lomakkeen luonti- ja asetusnäkymä

Lomakkeen tyyppi määrää, miten lomake tallentaa ja käsittelee tietoa. Tyyppejä on kolme ja niitä voi yhdistellä mielivaltaisesti. Tyyppiä ei lomakkeen luonnin jälkeen enää pysty muuttamaan tiedon eheyden vuoksi.

Anonyymi-tyyppinen lomake ei tallenna täytettyyn lomakkeeseen tietoa alkuperäisestä täyttäjästä ja täten sitä voidaan käyttää esimerkiksi kyselyissä työpaikan työilmapiirin laadusta tai muista arkaluontoisista aiheista, jossa kritiikin esittäminen omalla nimellään voi olla haitallista tai ei muutoin ole haluttavaa. Toisin kuin muita lomaketyyppejä, anonyymi-tyyppistä lomaketta ei voida tallentaa käyttäjälle keskeneräisenä.

Vaiheistettu-tyyppinen lomake jaetaan perättäisesti täytettäviin vaiheisiin, joille voidaan määritellä haluttu vastuuhenkilö, jolle lomake lähetetään sen saavuttaessa kyseisen vaiheen. Lomakkeen ei tarvitse olla vaiheistettu-tyyppinen voidakseen sisältää vaiheita, mutta tällöin vaiheiden mahdollistamia vastuuhenkilöitä ja toiminnallisuuksia ei käytetä. Vaiheistuksella voidaan toteuttaa tiedonsyöttöprosesseja joiden kulku eri täyttäjien välillä on ennalta sovittu.

Testilomakkeella voidaan yksiselitteisiä vastauksia vastaanottaville kentille määrittää oikeat vastaukset ja pistemäärät. Tällaisia kenttiä ovat kokonaisluku, desimaaliluku sekä omat kentät ja -listat. Täytettyjä ja suljettuja lomakkeita voidaan verrata annettuja arvoja vasten ja pisteyttää vastaukset.

Asiakirjapohja-kohdassa voidaan olemassaolevan lomakkeen vaiheet ja kentät kopioida luotavan lomakkeen pohjaksi. Valittavina lomakkeina ovat vain ne joihin tämänhetkisellä käyttäjällä on vähintään luku-oikeudet.

Painikkeet-valinnoilla voidaan säädellä, mitä toimintoja käyttäjä voi käyttää täyttäessään lomaketta. Tallenna-painikkeella täyttäjä voi tallentaa lomakkeen itselleen keskeneräisenä, jolloin täyttämistä voi myöhemmin jatkaa. Sulje-painike mahdollistaa lomakkeen sulkemisen, joka tarkoitaa sitä, että lomakkeen kenttien tiedot validoidaan, tallennetaan ja lomake merkitään suljetuksi, jolloin muokkaaminen ei ole enää mahdollista. Toimenpidepyyntö-painikkeella käyttäjä voi oman osuutensa valmistuttua lähettää lomakkeen toiselle käyttäjälle jatkokäsiteltäväksi. Kun lomake on lähetetty toiselle käyttäjälle, muut kuin kyseinen käyttäjä eivät enää pääse muokkaamaan lomaketta. Käytännössä Tallenna-painike on vähintään sallittava että lomakkeen täyttö ja tallentaminen on mahdollista.

Vastuuhenkilö-sarakkeeseen voidaan järjestelmästä valita henkilö, jolle lähetetään sähköpostitse tiedonanto jokaisesta aloitetusta lomakkeesta.

Ryhmät-sarakkeeseen voidaan lomakkeelle määritellä käyttöoikeudet Umbrellan ryhmittäin. Kullekin ryhmälle voidaan antaa luku-, muokkaus- ja hallinta-oikeuksia mielivaltaisesti. Kukin oikeustaso mahdollistaa erilaisen pääsyn lomakkeelle. Luku-oikeuksilla voidaan täyttää ja lukea käyttäjän omia  lomakkeita, muokkaus-oikeuksin muokata lomakepohjan kenttiä ja vaiheita ja hallinta-oikeuksilla muokata lomakkeen pääsyoikeuksia, nimeä ja muita asetuksia.

## 6.2 Lomakepohjan muokkaus

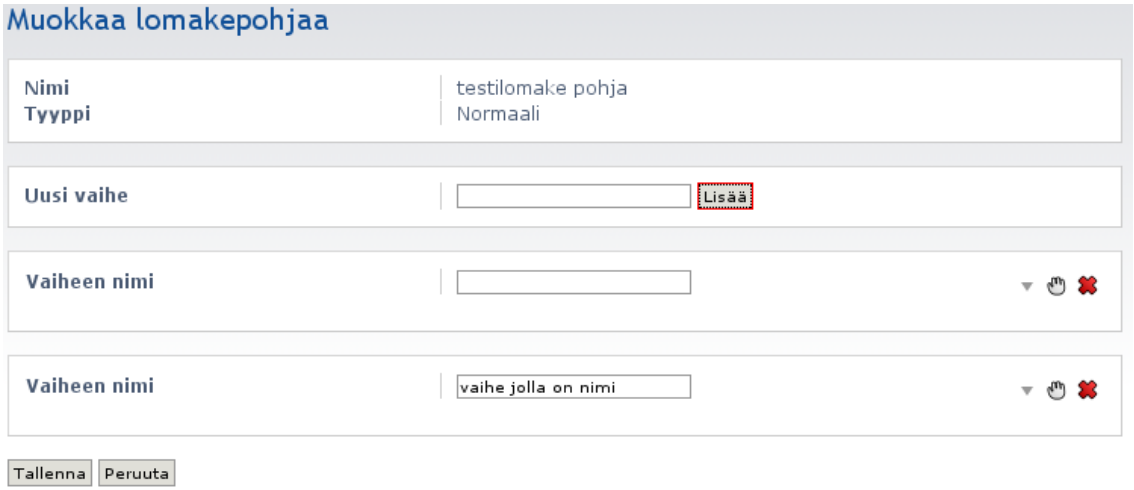
Lomakepohjan muokkauksessa voidaan lomakkeeseen lisätä vaiheita, vaiheisiin kenttiä ja muuttaa näiden asetuksia vastaamaan tiedonkeruun tarpeita.

Taulukko 3. Lomakepohjan muokkauksen kuvakkeet

| Kuvake | Selite |
|---|---|
| | Näyttää / piilottaa supistetun sisällön |
| | Kuvakkeesta raahaamalla voidaan vaiheiden ja kenttien järjestystä muuttaa. |
| | Poistaa vaiheen tai kentän |

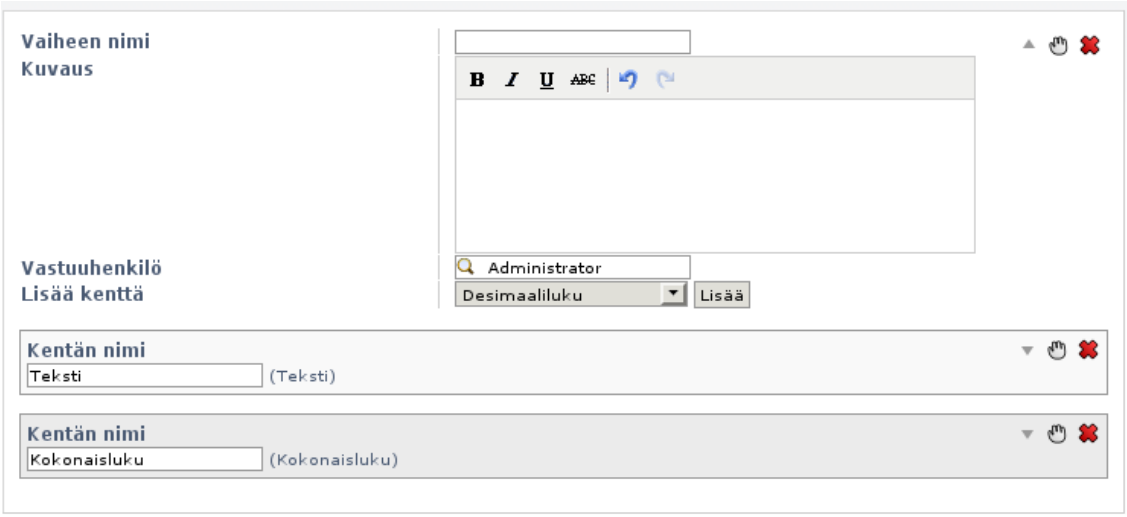
### 6.2.1 Vaiheet

Lomakepohjan muokkauksessa voidaan luoda, poistaa ja siirrellä vaiheita ja kenttiä vaiheiden sisällä. Uusi vaihe voi luodaan täyttämällä vaiheen nimi Uusi vaihe -kohtaan ja painamalla Lisää-painiketta. Tämän jälkeen uusi vaihe ilmestyy vaihelistan alimmaksi (kuva 14.)




Kuvio 14. Vaiheen lisäys

Vaiheen tarkoituksena on selkiyttää lomakkeen rakennetta ja jaotella kenttiä loogisiin kokonaisuuksiin, jotka ovat lomakkeen laatijan itsensä haluamia. Lomake voi koostua mielivaltaisesta määrästä vaiheita ja jokainen vaihe mielivaltaisesta määrästä kenttiä. Vaiheiden järjestystä voi vaihtaa siirtämällä haluttua vaihetta käsi-kuvakkeesta. Kentät ja vaiheen asetukset saa esille klikkaamalla nuoli-kuvaketta (kuva 15).




Kuvio 15. Vaiheen asetukset

Vaiheen nimi ja kuvaus eivät ole välttämättömiä, mutta ovat suositeltavia yleisen selkeyden vuoksi. Nimen perusteella käyttäjä voi muokatessaan erottaa vaiheet toisistaan ja täten hallita niitä paremmin. Kuvausteksti antaa myös lomakkeen täyttäjälle paremman kuvan tiedoista, joita hän tulee täyttämään.

Jokaisessa vaiheessa voidaan määritellä vastuuhenkilö, joka on se käyttäjä, jolle lomake lähetetään kyseiseen vaiheeseen edettyään, mikäli lomake on Vaiheistus-tyyppinen. Normaali-tyyppisessä lomakkeessa kyseisellä sarakkeella ei ole vaikutusta lomakkeen toimintaan. Mikäli Vaiheistus-lomakkeen ensimmäiseen vaiheeseen määritellään vastuuhenkilö, vain tämä käyttäjä voi tällöin aloittaa uuden lomakkeen kyseiseen lomakepohjaan. Yleensä ensimmäisen vaiheen vastuuhenkilö jätetään tyhjäksi, jolloin uuden lomakkeen voi aloittaa kuka hyvänsä käyttäjä, jolta löytyy vähintään luku-oikeus lomakepohjaan.

Lisää kenttä -sarakkeesta vaiheeseen voidaan lisätä kenttiä. Pudotusvalikossa on lajiteltuna järjestelmän peruskenttätyypit, luodut omat kentät sekä omat listat.

### 6.2.2 Kentät

Umbrella 4 tarjoaa useita erilaisia kenttätyyppejä käytettäväksi lomakkeissa, ja tarvittaessa käyttäjät voivat itse laatia lisää uusia kenttätyyppejä ja niiden validointisääntöjä.

Kaikille kentille on nimi- ja kuvauskentät, joihin lomakepohjan laatija voi kirjoittaa täyttäjille ohjeita kyseisen kentän täyttämiseen. Kaikki kenttätyypit, paitsi väliotsikko, voidaan asettaa pakolliseksi täyttää asettamalla rasti Pakollinen kenttä -kohtaan. Teksti, päivämäärä, päivämäärä ja aika, tiedosto, keskustelu ja oma kenttä -tyyppiset kentät eivät sisällä muita asetuksia kuin kentän pakollisuuden, joten niiden asetusnäkymät ovat samanlaiset keskenään (kuva 16).


Kuvio 16. Teksti-kentän muokkaus

Lomakepohjan ollessa testi-tyyppinen, voidaan kokonais- ja desimaaliluku kenttätyypeille määritellä oikea vastaus ja oikeasta vastauksesta saatavat pisteet. Lomakkeita pisteyttäessä käyttäjän syöttämää vastausta verrataan oikeaan vastaukseen hyvin kirjaimellisesti, esimerkiksi käyttäjä ei saa pisteitä jos oikeaksi vastaukseksi on määritelty desimaaliluku 3,14 ja käyttäjä on syöttänyt 3,140. Vastaukset tarkistetaan merkki merkiltä jotta esimerkiksi numeraalisissa vastauksissa voidaan vaatia tietty määrä merkitseviä desimaaleja (kuva 17).


Kuvio 17. Kokonaisluku-kentän asetukset

Käyttäjät-kentällä voidaan järjestelmästä löytyvistä ryhmistä muodostaa lista käyttäjistä joista lomakkeen täyttäjät valitsevat joko yhden tai useampia riippuen käytetystä esitystavasta. Mahdollisia esitystapoja ovat pudotusvalikko, valintanapit ja valintalaatikot. Esitystavoista vain valintalaatikot mahdollistavat useamman kuin yhden käyttäjän valitsemisen (kuva 18).


Kuvio 18. Käyttäjät-kentän asetukset

Omat kentät ovat lomakepohjia laativien käyttäjien itse luomia kenttätyyppejä. Kentille voidaan määritellä omat validointiehtonsa joiden mukaan tarkistetaan onko lomakkeen täyttäjien antamat syötteet oikeellisia (kuva 19). Kentän validointisääntöjen määrittely käydään läpi kappaleessa Omat kentät ja listat.


Kuvio 19. Oman kentän ’Lyhyt tekstikenttä’ asetukset

Kuten omat kentät, omat listat ovat lomakepohjia luovien käyttäjien määrittelemiä kenttiä. Omissa kentissä käyttäjä määrittelee avain-arvo listan joista voidaan tarpeen mukaan tehdä lomakkeelle erilaisia kenttiä. Esimerkiksi oma lista -kentästä voidaan tehdä monivalinta valintalaatikoilla ja oikealle vastauskombinaatiolle määritellä saatavat pisteet (kuva 20).


Kuvio 20. Oman listan ' Arviosi asiakaspalvelusta 1-5 (5=kiitettävä)' asetukset

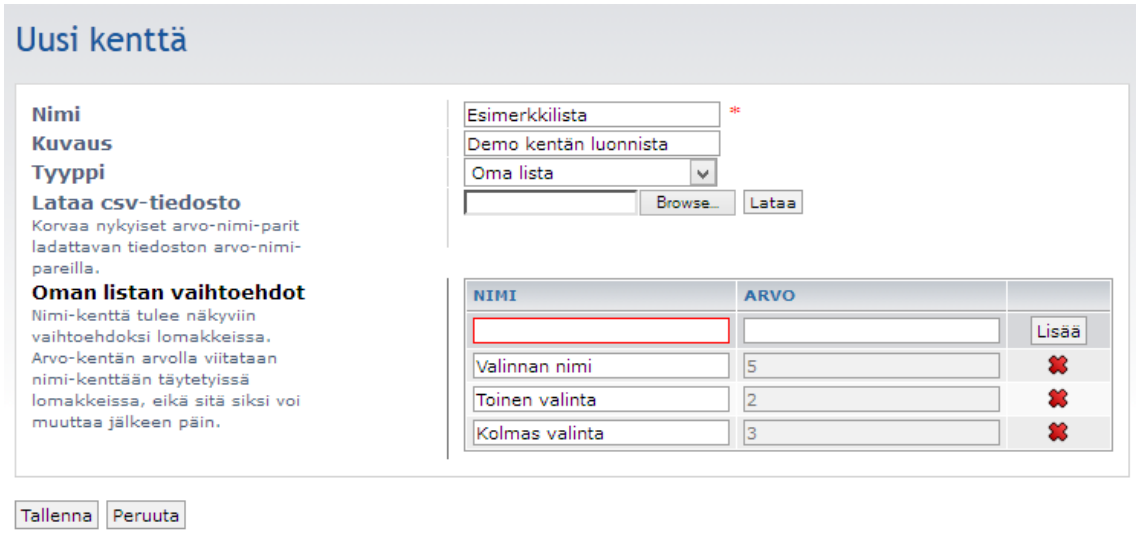
### 6.2.3 Omat kentät- ja listat

Omien kenttien- ja listojen lisäys, poisto ja muokkaus tapahtuu kenttien hallinnointinäkymästä. Näkymässä voidaan tarkastella järjestelmän sisäänrakennettuja kenttätyyppejä sekä luoda eri tarkoitukseen soveltuvia omia kenttätyyppejä (kuva 21).


Kuvio 21. Kenttien hallinta

Uutta kenttää luodessa käyttäjä voi valita, onko kenttä tyypiltään oma lista- tai oma kenttä. Listat toimivat lomakkeella pohjana erilaisille monivalinnoille ja kentät tekstikenttinä, joilla on käyttäjän määrittelemä validointisääntö.

Oma lista -tyyppinen kenttä muodostuu käytännössä assosiatiivisesta taulukosta jossa on nimi ja arvo pareja. Nimi on käyttäjäystävällinen, luonnollista tekstiä sisältävä tietue joka esitetään loppukäyttäjälle tämän laatiessa lomakepohjaa tai täyttäessä lomaketta ja arvoa käytetään annetun vastauksen tallennuksessa sekä raportoinnissa laskettavana arvona (kuva 22).

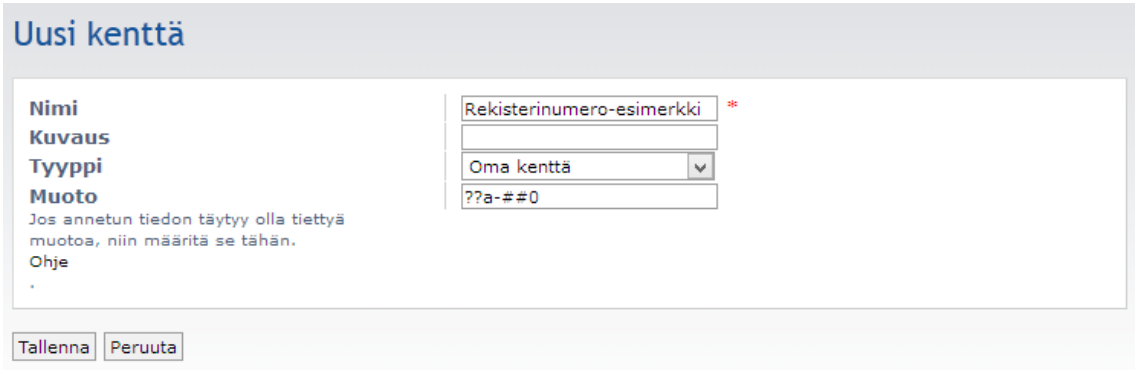



Kuvio 22. Oman listan määrittely

Oma kenttä -tyyppiselle kentälle voidaan määritellä yksinkertaisella syntaksilla vastausmuoto, jonka lomaketta täyttävän käyttäjän pitää antaa kyseiseen kenttään. Vastauksen muotovaatimuksen määrittelevässä kentässä voidaan käyttää alla olevassa taulukossa määriteltyjä symboleita. Esimerkiksi rekisterinumeron muotovaatimus voisi olla ”??a-##0”, jolloin kenttä hyväksyy muun muassa syötteet ”f-1”, ”abc-3” ja ”mi-35” (Taulukko 4, kuva 23).

Taulukko 4. Oman kentän validointisäännön symbolit

| **Symboli** | **Selite** |
|---|---|
| a | Mikä hyvänsä yksittäinen kirjainmerkki |
| ? | Sama kuin 'a', mutta voidaan jättää täyttäessä pois. |
| 0 (nolla) | Mikä hyvänsä numero välillä 0-9. |
| # | Sama kuin '0', mutta voidaan jättää täytettäessä pois. |
| * | Määrittelemätön (0-n) määrä mitä hyvänsä merkkejä. |
| Kaikki muut merkit | Kaikki muut merkit käsitellään literaaleina ja ovat siten vaadittuja jotta kenttä validoituu ja voidaan tallentaa. |




Kuvio 23. Oman kentän muokkaus- ja luontinäkymä

## 6.3 Lomakepohjan lataus ja tallennus

### 6.3.1 Käyttöliittymän tietorakenne

Lomakepohjan muokkauksen vaiheet ja kentät ovat käytännössä web-lomakkeen kenttiä, joiden keskinäisen järjestyksen muuttaminen vaikuttaa suoraan palvelimelle lähetettävän tietorakenteen järjestykseen ja muotoon. Käyttöliittymän ylläpitämä tietorakenne muodostuu moniulotteisesta, hierarkkisesti rakentuvasta taulukosta (kuva 24).

```
Lomakepohja : {
 ”7": { // vaiheen ID
  "name": "Vaiheen nimi",
    "desc": "Vaiheen kuvaus",
    "user": 0, //vaiheen vastuuhenkilö. 0 = ei kukaan
    "username": "", //vastuuhenkilön nimi.
    "order": 0, //vaiheen järjestysnumero, numerointi alkaa nollasta
    "fields": { //vaiheen kentät
   "40": { //olemassaolevan kentän ID
    "name": "Kokonaisluku -kentän nimi",
        "desc": "kvaus",
        "order": 0, //kentän järjestysnumero vaiheen sisällä
        "field": 2, //kenttätyypin ID, 2 = kokonaisluku
        "settings": {
     "max":"1", //kentän oikean vastauksen pistemäärä
          "correct": ["0"] //kentän oikea vastaus, nolla.
        }
   },
   //uusi kenttä, esiintyy vain tallennusdataa lähettäessä
      "new1": { //vaiheeseen tehty uusi kenttä, id ei ole numeerinen
    "name": "Päivämääräkentän nimi",
        "desc": "kuvaus",
        "order": "1",
        "field": "4"
   }
  },
 //uusi vaihe, esiintyy vain tallennusdataa lähettäessä
 ”new1”: {
  "name": "Vaiheen nimi",
    "desc": "Vaiheen kuvaus",
    "user": 3, //vaiheen vastuuhenkilö. 0 = ei kukaan
    "username": "Teppo Taalasmaa", //vastuuhenkilön nimi.
    "order": 1 //vaiheen järjestysnumero, numerointi alkaa nollasta
  //ei fields -indeksiä; vaiheella ei ole kenttiä
 }
}
```

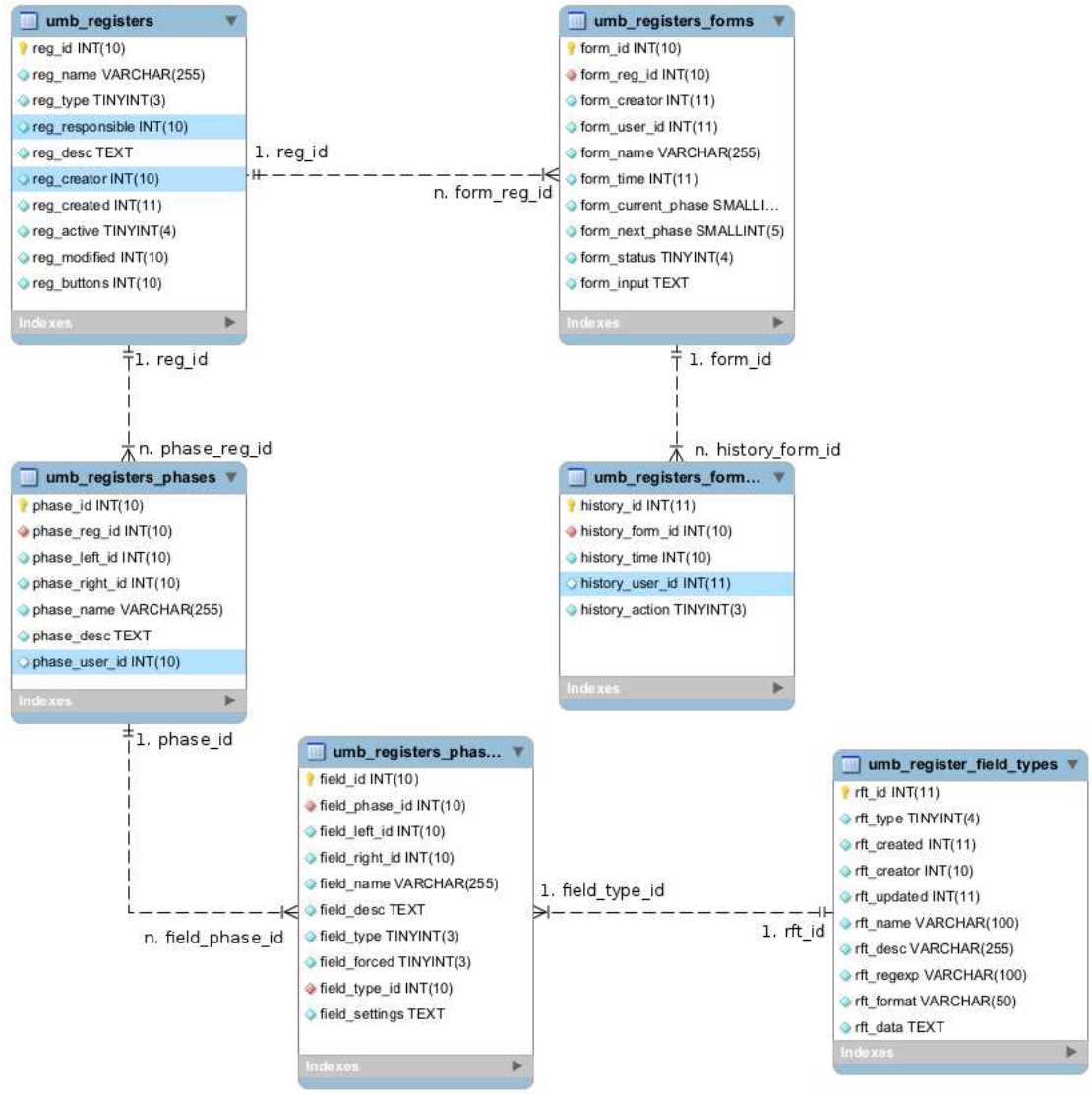
Kuvio 24. Lomakepohjan muokkauksen käyttöliittymän tietorakenne esitettynä JSON-notaationa

Tietorakenne on näkymää ladattaessa ja tallentaessa identtinen. Ulkoasumoottori luo tietorakenteesta JSON-muuttujan jonka lomakepohjan muokkauksesta huolehtiva FieldManager-luokan ilmentymä jäsentää muokkauskäyttöliittymäksi.

Tallennettaessa FieldManager jäsentää läpi kaikki vaiheet ja kentät, jotka ovat käyttäjän jäljiltä olemassa ja muodostaa näistä kuvan 24 mukaisen taulukon ja lähettää sen http(s) POST-lähetyksenä palvelimelle jäsennettäväksi ja tallennettavaksi.

### 6.3.2 Rakenne tietokannassa

Umbrellan lomakekokonaisuus rakentuu kuudesta taulusta tietokannassa . Lomakepohjaa kuvaa rivi umb_registers taulussa, johon voi liittyä mielivaltainen määrä vaiheita. Vaiheet on kuvattu umb_registers_phases-taulun riveillä, joihin puolestaan liittyy umb_registers_phases_fields-taulun kenttiä kuvaavat rivit. Kenttiä vastaavat kenttien mallit löytyvät umb_registers_field_types-taulusta. Täytetyt lomakkeet tallennetaan umb_registers_forms-tauluun ja lomakkeen käsittelyhistoria umb_registers_forms_history-tauluun (kuva 25, taulukko 5, taulukko 6, taulukko 7, taulukko 8).




Kuvio 25. Lomakepohjien käyttämät tietokantataulut

Taulukko 5. Umb_registers-taulun rakenne

| **Sarake** | **Tyyppi** | **Selite** |
|---|---|---|
| reg_id | int, taulun avain | Lomakepohjan yksilöivä id-numero |
| reg_name | varchar | Lomakepohjan nimi |
| reg_type | tinyint | Numero joka määrittää lomakepohjan tyypin, käsitellään bittikenttänä.<br>1 = Anonyymilomake<br>2 = Testi<br>4 = Vaiheistettu<br>Esimerkiksi vaiheistettu testilomake: 4+2=6 |
| reg_responsible | int | Viite lomakepohjan vastuuhenkilöön käyttäjätaulussa. |
| reg_desc | Text | Lomakepohjan kuvausteksti |
| reg_creator | Int | Viite lomakepohjan luoneeseen käyttäjään käyttäjätaulussa. |
| reg_created | Int | Aikaleima jolloin lomakepohja luotiin |
| reg_modified | Int | Aikaleima jolloin lomakepohjaa on viimeksi muokattu |
| reg_buttons | Int | Ilmaisee mitä painikkeita lomakkeella käyttäjille esitetään. Käsitellään bittikenttänä.<br>1 = Tallenna<br>2 = Sulje<br>4 = Toimenpidepyyntö<br>Esimerkiksi: lomakkeelle halutaan vain tallenna ja sulje -painikkeet: 1+2 = 3 |
| reg_active | Tinyint | Määrittelee onko kyseinen lomakepohja aktiivinen. Lomakepohjan ollessa aktiivinen siihen voi täyttää uusia lomakkeita. |

Taulukko 6. Umb_registers_phases-taulun rakenne

| **Sarake** | **Tyyppi** | **Kuvaus** |
|---|---|---|
| phase_id | Int, taulun avain | Vaiheen yksilöivä id-numero |
| phase_reg_id | Int | Viite um_registers-tauluun. Ilmaisee lomakepohjaa johon vaihe kuuluu. |
| phase_left_id | Int | Ilmaisee vaiheiden keskinäistä järjestystä lomakepohjassa. |
| phase_right_id | Int | Ilmaisee vaiheiden keskinäistä järjestystä lomakepohjassa. |
| phase_name | Varchar | Vaiheen nimi |
| phase_desc | Text | Vaiheen kuvaus |
| phase_user_id | Int | Viite vaiheen vastuuhenkilöön käyttäjätaulussa. |

Taulukko 7. Umb_registers_phases_fields-taulun rakenne

| **Sarake** | **Tyyppi** | **kuvaus** |
|---|---|---|
| field_id | Int, taulun avain | Vaiheeseen luodun kentän yksilöivä id-numero |
| field_phase_id | Int | Viite umb_registers_phases taulussa olevaan vaiheeseen johon kenttä kuuluu. |
| field_left_id | Int | Kokonaisluku joka ilmaisee kenttien keskinäistä järjestystä vaiheessa |
| field_right_id | Int | Kokonaisluku joka ilmaisee kenttien keskinäistä järjestystä vaiheessa |
| field_name | Varchar | Kentän nimi |
| field_desc | Text | Kentän kuvaus |
| field_type | Tinyint | Kentän esitystapa, mikäli sellainen on. Pätee vain käyttäjät ja omalista -kenttätyypeille |
| field_forced | Tinyint | Ilmaisee onko kenttä pakollinen vai ei. |
| field_type_id | Int | Viite umb_register_field_types taulun kenttien malleihin |
| field_settings | Text | Kenttään tallennetaan tekstimuotoon muunnettu PHP:n taulukko kenttäkohtaisista asetuksista, esimerkiksi käyttäjät-kentän ryhmät. |

Taulukko 8. Umb_register_field_types-taulun rakenne

| **Sarake** | **Tyyppi** | **kuvaus** |
|---|---|---|
| rft_id | Int, taulun avain | Kenttätyypin yksilöivä id-numero |
| rft_type | Tinyint | Kertoo kenttätyypin |
| rft_created | Int | Aikaleima jolloin kenttä on luotu |
| rft_creator | Int | Viite kentän luoneeseen käyttäjään käyttäjä-taulussa. |
| rft_updated | Int | Aikaleima jolloin kenttää on muokattu viimeksi. |
| rft_name | Varchar | Kenttätyypin nimi |
| rft_desc | Varchar | Kenttätyypin kuvaus |
| rft_format | Varchar | Kenttätyypin käyttäjäystävällinen esitys formaatista, esimerkiksi merkkijono ###000 joka tarkoittaa 3-6 numeroa. |
| rft_regexp | Varchar | Kenttätyypin formaatista muodostettu säännöllinen lauseke jonka avulla kenttien syöte validoidaan. |
| rft_data | Text | Oma kenttä -kenttien avain-arvo lista tekstimuotoon muunnettuna PHP:n taulukkona |

Taulujen välillä vallitsee relaatio aina alemmasta tasosta ylempään, esimerkiksi lomakkeen kentästä on relaatio sen sisältävään vaiheeseen sekä kentän malliin. Relaatioiden ansiosta esimerkiksi lomakepohjan poisto voidaan toteuttaa yksinkertaisella poistokyselyllä joka poistaa useasta taulusta kaikki lomakepohjaan liittyvät tiedot.

### 6.3.3 Tallennus

Lomakepohjan käyttöliittymässä luotu lomakepohja lähetetään tallennettaessa selaimesta lomakkeen POSTDATA:na taulukkomuotoisena tietona PHP:lle. Taulukon ensimmäisellä tasolla ovat vaiheet asetuksineen sekä niiden alla kullekin vaiheelle kuuluvat kentät ja määritellyt asetukset. Taulukko muodostuu itsestään oikeaan muotoon muokkauskäyttöliittymän mukaan. Käyttöliittymä on suuri form-elementti, eli lomake, jonka käyttöliittymäkomponenttien, tässä tapauksessa vaiheiden ja kenttien käyttöliittymät. Esimerkki lähetettävästä POSTDATA:sta (kuva 26).

```
[
  //olemassaoleva vaihe
  31 = [
    'name' = 'Vaiheen nimi',
    'desc' = 'vaiheen kuvaus',
    'order' = 1,
    'fields' = [
      312 = [
        'name' = 'Kentän nimi',
        'field' = 3, //tyyppi
        'desc' = 'Kuvaus',
        'data' = [] //asetukset
      ],
      24 = [ .. ],
      //uusi kenttä
      'new3' = [ .. ]
    ]
  ],
  //uusi vaihe
  'new1' = [ .. ]
];
```

Kuvio 26. Esimerkki lomakepohjan tallennettavasta POSTDATA-tiedosta kommentoituna

PHP-käsittelijän vastaanottaessa lähetettyä dataa se jaetaan lomaketta, vaiheita ja kenttiä hallinnoiville luokille, jotka suorittavat vertailua vanhan tallennusdatan kanssa, jolloin voidaan päätellä poistuneet vaiheet ja kentät. Uudet vaiheet ja kentät tunnistetaan niiden poikkeavasta indeksoinnista, esimerkiksi phase_new_1. Lomakepohjaan tallennettavat kentät ja vaiheet numeroidaan left- ja right-numeroinnilla, joka määrittelee niiden keskinäisen järjestyksen.

### 6.3.4 Left- ja right -numerointi

Left- ja right-numeroinnilla on mahdollista rakentaa puumainen hierarkia jolla voidaan tehdä mahdolliseksi esimerkiksi vaiheen sisäiset vaiheet. Umbrellan lomaketoiminnallisuus käyttää left- ja right -numerointia vain yhdessä tasossa, mahdollisuus laajempaan hierarkiaan jätettiin mahdolliseksi potentiaalisen jatkokehityksen varalta.

Numeroinnissa olennaista on se, ettei kahta samaa numeroa esiinny kahdesti eikä yhtäkään numeroa hypätä yli. Numerointi alkaa aina luvusta 1 joka on vasen luku hierarkian ensimmäiselle elementille. Oikeanpuoleinen luku on aina vähintään yhden suurempi kuin vasen. Perättäisten, samassa tasossa olevien elementtien vasen luku on edellisen elementin oikeaa lukua yhden numeron suurempi. Oikea luku suurenee elementin alla olevien elementtien (lasten) määrän mukaan (kuva 27).

Lukujen laskentakaavat

n = edeltäneen samalla tasolla olevan elementin oikea luku

Left = n + 1

Right = left + 1 + lapset * 2

Esimerkki hierarkiasta:

| **Left** | **Right** | **Nimi** |
|---|---|---|
| 1 | 8 | A |
| 2 | 7 | B |
| 3 | 4 | C |
| 5 | 6 | D |
| 9 | 10 | E |
| 11 | 14 | F |
| 12 | 13 | G |

Esimerkin numeroinnilla rakentuu seuraavanlainen hierarkia, suluissa vasen ja oikea luku.

A (1,8)
└ B (2,7)
└ C (3,4)
D (5,6)
E (9,10)
F (11,14)
└ G (12,13)

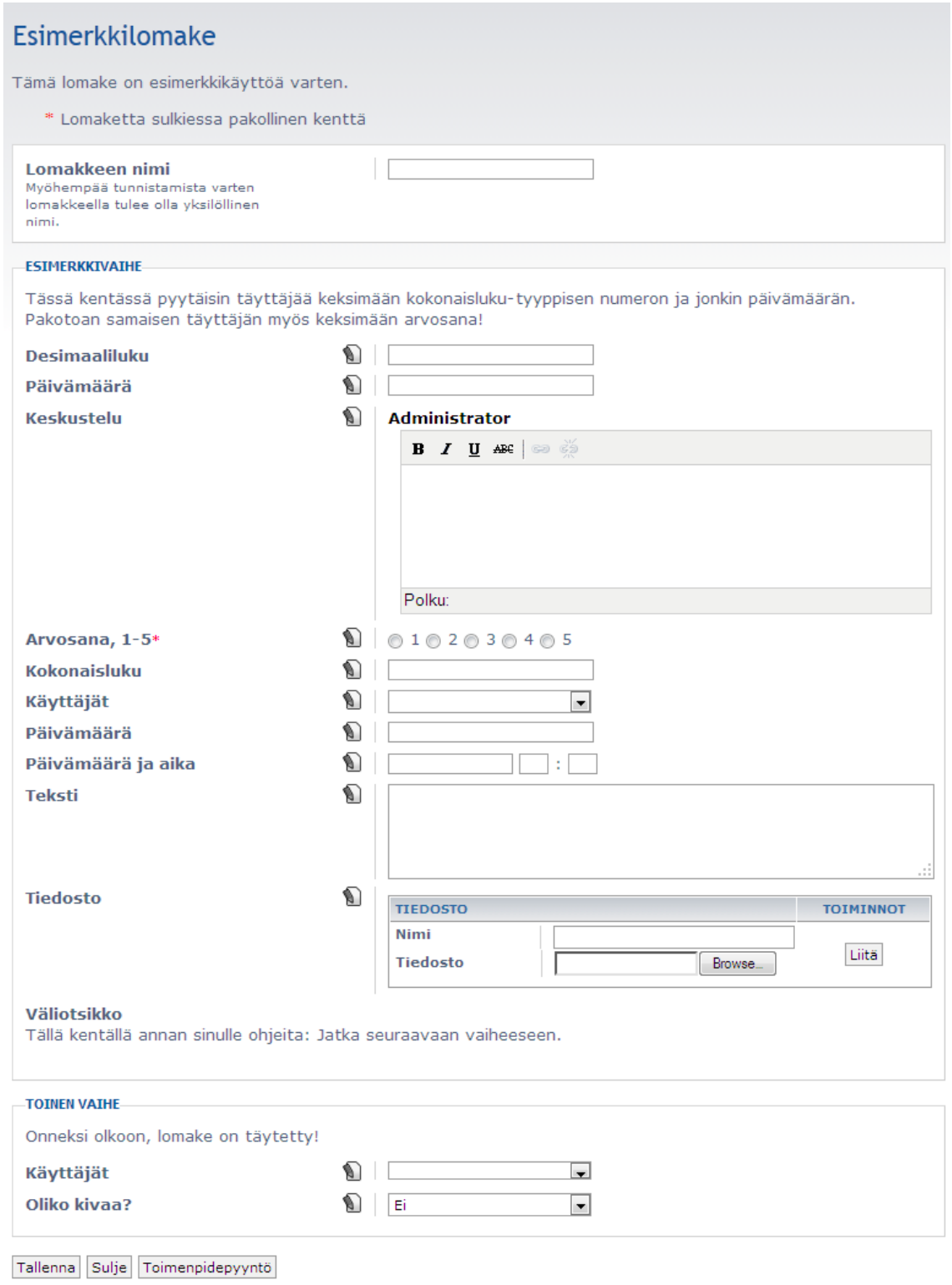
Kuvio 27. Esimerkki left- ja right-numeroinnilla rakennetusta hierarkiasta

Tietokannassa elementit voivat sijaita mielivaltaisessa järjestyksessä, sillä ladattaessa ne voidaan järjestää vasemman luvun mukaan nousevassa järjestyksessä jolloin vaiheet ja kentät saadaan ladattua käyttäjän määrittelemässä järjestyksessä. Käyttöliittymä ei lähetä vasen/oikea -lukuja vaan ne generoidaan tallennettaessa palvelimella lähetettyjen vaiheiden ja kenttien mukaan.

## 6.4 Lomakkeen täyttö

Täyttötilassa käyttäjälle näytetään web-lomake, joka on muodostettu määritellyn lomakepohjan mukaan. Jokaiselle uudelle lomakkeelle on annettava yksilöllinen nimi jotta sen myöhempi tunnistus on mahdollista. Mikäli annetun niminen lomake on jo täytetty lomakepohjaan, näytetään asiaankuuluva virhe tallennuksen yhteydessä.

Käyttäjä täyttää näkyvien vaiheiden kentät ja valitsee pohjalta toiminnon Tallenna, Sulje tai Toimenpidepyyntö (kuva 28).

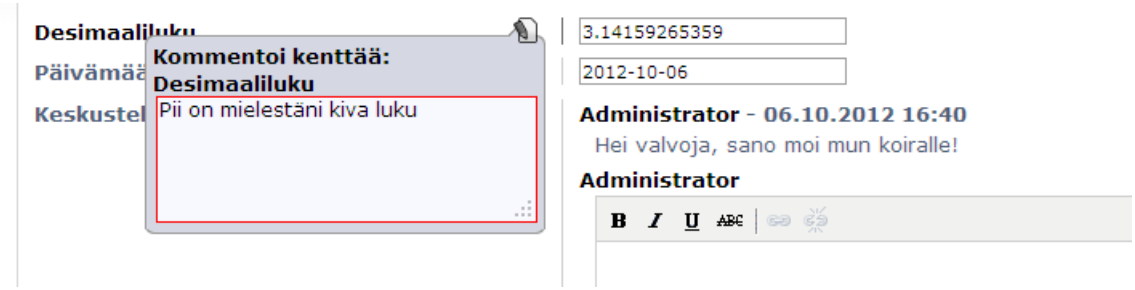



Kuvio 28. Lomakkeen täyttötila

## 6.5 Lomakkeen lataus ja tallennus

Kun käyttäjä on täyttötilassa päässyt tilanteeseen, jossa hän haluaa tallentaa työnsä, hän painaa tallennuksen toteuttavaa painikettä lomakkeen pohjalla. Tallennettaessa lomake lähettää normaalin web-lomakkeen form-elementin tavoin tiedot palvelimelle käsiteltäväksi.

Sen lisäksi, että käyttäjä voi täyttää validoitavia tietoja lomakkeen kenttiin, voi käyttäjä myös syöttää lisätietoa mihin hyvänsä kenttään käyttäen kunkin kentän vierestä löytyvää kommentointipainiketta. Painiketta napsauttamalla käyttäjälle avataan pieni puhekuplaa muistuttava dialogi johon annettua vastausta voi perustella. [Kuva 29.]




Kuvio 29. Kentän kommentointi

### 6.5.1 Kentät

Kentät täyttötilassa ovat muutamaa poikkeusta lukuunottamatta tavallisia input- ja select -html-elementtejä. Poikkeuksen muodostavat tiedosto- sekä keskustelu -tyyppiset kentät.

Keskustelukenttä on toiminnoiltaan rajattu TinyMCE -instanssi, jonka avulla käyttäjä voi tarvittaessa korostaa osia viestistään. Kentässä on estetty useamman kuin yhden viestin kirjoittaminen per lomakkeen muokkauskerta. Tällä on pyritty ohjaamaan käyttäjää välttämään turhien viestien kirjoittamista ja pitämään keskusteluketjun pituutta aisoissa.

Tiedostokenttä muodostuu tiedostoja hallinnoivasta Javascript-luokasta sekä iframe-elementistä, jonka avulla tiedostot lähetetään ilman, että itse lomakesivulla tehtäisiin sivulatausta. Lähetetyt tiedostot siirretään palvelinpuolella väliaikaishakemistoon ja käyttöliittymälle palautetaan tiedostoon viittaava tiivistesumma. Kun lomaketta tallenne-

taan, siirretään tiedostot pois väliaikaishakemistosta ja lomakkeen tallennettavaan tietueeseen merkitään viittaukset kyseisiin tiedostoihin.

### 6.5.2 Tallennus

Lomakkeen tallennus tapahtuu lähettämällä lomakkeen kentistä muodostuva taulukko palvelimelle käsiteltäväksi. Palvelimen aloittaessa lähetetyn lomakedatan jäsentämisen, tarkistetaan, onko lähetetyllä lomakkeella ID-numeroa ja onko lähetyksen tehneellä käyttäjällä täyttöoikeutta kyseisellä ID-numerolla varustetulle lomakkeelle. Mikäli ID-numeroa ei lähetetty, oletetaan, että kyseessä on täysin uusi lomake. Jokaisella kentällä on omalla vaiheen ja kentän ID-numerolla indeksoitu , jonka mukaan palvelinpuolen tallennus ja validointi suoritetaan (kuva 30).

```
{
  form_name: ”Täytetyn lomakkeen nimi”,
  data: { //kenttien data
    123: { //indeksinä vaiheen ID
      47: { //indeksinä kentän ID
        //kentän data, sisällön tyyppi vaihtelee kentittäin
        //tiedosto & oma lista -kentissä taulukko, muuten merkkijono.
        data: ””,
        //kentän kommentti
        comment: ”kommentti”
      },
      66: { //indeksinä kentän ID
        data: [3,5,2], //oma listan valitut indeksit
        //kentän kommentti
        comment: ”kommentti”
      }
    }
    99: { ... //seuraavan vaiheen data
    }
  }
}
```

Kuvio 30. Lomakkeen tallennuksen POST-data esitettynä JSON-notaatiolla

Lomakepohja tarkistaa, onko lähetetyssä taulukossa syötettyä dataa kullekin vaiheelle ja kyseisen datan löytyessä antaa sen vaiheelle. Kukin vaihe tarkistaa saadun datan kenttiään vasten ja antaa kullekin kentälle tarkoitetun datan kyseisille kentille. Kun kenttä vastaanottaa datan, se sijoittaa sen sisäiseen muuttujaansa ja jää odottamaan validointia. Kun kaikki data on syötetty kentille, aloitetaan validointivaihe, mikäli valittu tallennusoperaatio on joko sulkeminen tai toiselle käyttäjälle lähetys.

Validointivaiheessa kentät iteroidaan läpi ja kukin kenttä palauttaa totuusarvon joka ilmaisee täyttääkö kenttä asetetut muotovaatimukset tai ei. Mikäli yksikin kenttä palauttaa tiedon, jolla ilmaistaan kentän syötteen olevan kelvoton, palautetaan lomake annetuilla vastauksilla takaisin käyttäjän näkyville ja ilmoitetaan missä kentissä on kelvoton syöte. Mikäli kaikki kentät validoituivat ongelmitta, suoritetaan tallennus. Tallennus käytännössä siirtää käyttäjän selaimesta lähetettyä vastaavan sarjallistetun taulukon tietokantaan.

### 6.5.3 Lataus

Ladattaessa lomaketta tietokannasta toimitaan käytännössä samalla tavalla kuin lomaketta tallennettaessa, sillä erotuksella että lomakkeen data ei tule selaimesta vaan tietokannasta eikä sitä validoida.

# 7 Kokemukset, rajoitukset ja jatkokehitys

Käyttäjät ovat pääasiallisesti mieltäneet lomakemoduulin käytön varsin vaivattomaksi ja yksinkertaiseksi. Yksittäiset käyttäjät ovat moittineet yksilöllisesti nimettävien lomakkeiden logiikkaa, sillä loppukäyttäjät eivät voi tietää, minkä nimisiä lomakkeita lomakepohjassa jo on, joten he joutuvat joissakin tapauksissa nimeämään lomakkeen yrityksen ja erehdyksen kautta, kunnes vapaa nimi löytyy.

Umbrella Interactive 4:n lomakemoduuli ei anna siirtää kenttiä vaiheesta toiseen, mikä kasaa ylimääräistä suunnitteluvastuuta lomakepohjan laatijalle. Mikäli lomakepohjaan on täytetty lomakkeita ja havaitaan, että tietyt kentät eivät ole loogisesti oikeassa vaiheessa, joudutaan nämä kentät poistamaan ja luomaan uusia jolloin kyseisiin kenttiin tallennettu data menetetään.

Umbrella Interactiven versiossa 5 lomakemoduuli koki suuren muutoksen tarjoamalla interaktiivisemman lomakepohjan sekä lomakkeen täyttötilan. Lomakepohjat sallivat uudessa versiossa kenttien siirtelyn vaiheiden välillä ja järjestelmään esiteltiin kokonaan uusia kenttätyyppejä, kuten laskukaavakenttä, jolla lomake voi suorittaa erilaisia laskukaavoja käyttämällä kenttien vastauksia. Lomakkeen täyttö puolestaan sai paljon kaivattuna lisäominaisuutena muun muassa täyttämisen aikana tapahtuvan validoinnin, tämän avulla käyttäjän ei enää tarvinnut yrittää tallennusta jotta saisi lomakkeen tallennettua oikein.

# 8 Yhteenveto

Tässä työssä käytiin läpi Umbrella Interactive 4 -version lomakemoduulin käyttämät tekniikat, tarvittavat kolmansien osapuolten toimittavat sovellukset, vastaavat kilpailevat sovellukset sekä moduulin toiminnallisuudet ja tallennustavat. Valitut tekniikat ja ohjelmistot palvelevat tarkoitustaan ja mahdollistavat ohjelman jatkokehityksen tulevaisuudessa.

Toteutetun lomakemoduulin lopputulos oli tarkoitukseen sopiva. Lomakepohjien muokkausnäkymä keräsi kiitosta aikaisempien versioiden käyttäjiltä, sillä jokainen kentän tai vaiheen siirto ei tarvinnut enää tallennusta ja sivulatausta. Kaiken kaikkiaan lomakemoduulin käyttökokemus muuttui merkittävästi paremmaksi.

## Lähteet


1 Solinum Oy:n web-sivu. Verkkodokumentti. <http://www.solinum.fi/index.php?sivu=tuotteet>. Luettu 21.10.2013.

2 Coffee Cup Software:n Web Form Builder -sovelluksen esittely. Verkkodokumentti. Coffee Cup Software. <http://www.coffeecup.com/web-form-builder/>. Luettu 21.10.2013.

3 Network Working Group, UTF-8, a transformation format of ISO 10646. Verkkodokumentti. Network Working Group. <http://www.ietf.org/rfc/rfc3629.txt>. Luettu 21.10.2013.

4 What is PHP? Verkkodokumentti. The PHP Group. <http://fi.php.net/manual/en/intro-whatis.php>. Luettu 21.10.2013.

5 W3C: Extensible Markup Language (XML) 1.0 (Fifth Edition). Verkkodokumentti. The World Wide Web Consortium. <http://www.w3.org/TR/xml/>. Luettu 21.10.2013.

6 W3schools.com: XML 10 kohdan tiivistelmä. Verkkodokumentti. Refsnes Data. <http://www.w3c.tut.fi/translations/xml/xmlin10pts/>. Luettu 21.10.2013.

7 W3schools.com: XML Elements. Verkkodokumentti. Refsnes Data. <http://www.w3schools.com/xml/xml_elements.asp>. Luettu 21.10.2013.

8 W3schools.com: XML Attributes. Verkkodokumentti. Refsnes Data. <http://www.w3schools.com/xml/xml_attributes.asp>. Luettu 21.10.2013.

9 W3C: XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition). Verkkodokumentti. The World Wide Web Consortium. <http://www.w3.org/TR/xhtml1/>. Luettu 21.10.2013.

10 W3C: Cascading Style Sheets Level 2 Revision 1 (CSS 2.1) Specification. Verkkodokumentti. The World Wide Web Consortium. <http://www.w3.org/TR/CSS2/>. Luettu 21.10.2013.

11 Mozilla Developer Network: Javascript. Verkkodokumentti. Mozilla Developer Network. <https://developer.mozilla.org/en/JavaScript>. Luettu 21.10.2013.

12 W3C: Scripts. Verkkodokumentti. The World Wide Web Consortium. <http://www.w3.org/TR/REC-html40/interact/scripts.html>. Luettu 21.10.2013.

13 Introducing JSON. Verkkodokumentti. <http://www.json.org/>. Luettu 21.10.2013.

14 Network Working Group: The application/json Media Type for JavaScript Object Notation (JSON). Verkkodokumentti. Network Working Group. <http://www.ietf.org/rfc/rfc4627.txt?number=4627>. Luettu 21.10.2013.

15 Prototype. Verkkodokumentti. Prototype Core Team. <http://prototypejs.org/>. Luettu 21.10.2013.

16 Jquery. Verkkodokumentti. The jQuery Foundation. <http://jquery.com/>. Luettu 21.10.2013.

17 TinyMCE. Verkkodokumentti. Moxiecode Systems AB. <http://tinymce.moxiecode.com/using.php>. Luettu 21.10.2013.

18 About MySQL. Verkkodokumentti. Oracle Corporation. <http://www.mysql.com/about/>. Luettu 21.10.2013.

19 MySQL, The InnoDB Storage Engine. Verkkodokumentti. Oracle Corporation. <http://dev.mysql.com/doc/refman/5.0/en/innodb-storage-engine.html>. Luettu 21.10.2