HAAGA-HELIA
ammattikorkeakoulu

Web – lomakkeiden määritystyökalu Fivaldi WorkFlow -sovellukseen

Juha Meriläinen

Raportti
Tietojenkäsittelyn koulutusohjelma
2013

HAAGA-HELIA
ammattikorkeakoulu

Tiivistelmä

Tietojenkäsittelyn koulutusohjelma

Tekijä tai tekijät Juha Meriläinen	**Ryhmä tai aloitusvuosi** 2010
Opinnäytetyön nimi Web – lomakkeiden määritystyökalu Fivaldi WorkFlow -sovellukseen	**Sivu- ja liitesivumäärä** 28 + 5
Ohjaaja tai ohjaajat Lili Aunimo	

Opinnäytetyö tehtiin toimeksiantona Oy Finnalli Finland Ab:lle ja sen tarkoituksena oli suunnitella ja toteuttaa drag & drop – tyylinen Web – lomakkeiden määritystyökalu. Työkalu toteutettiin osaksi suurempaa kiinteistönhallinnan sovellusta ja sen tarkoituksena oli mahdollistaa räätälöityjen lomakkeiden luonti käyttäjäkohtaisiin tarpeisiin. Opinnäytetyön tuloksena saatu työkalu myös dokumentoitiin asianmukaisesti.

Opinnäytetyössä esitellään ensin toimeksiantajan Software as a Service – mallin mukaisesti tarjottavan sovelluskokoelman sovellusympäristö, sekä sovellusten perusrakenne. Työkalun tekniset – ja käyttäjävaatimukset tuodaan myös esille.

Tietoperustana esitellään Web – lomakkeiden luontiin kehitettyjä ratkaisuja ja kehyksiä, joista haettiin inspiraatiota ja mallia. Siinä myös kerrotaan, miksi kehitettiin oma ratkaisu valmiiden ratkaisujen sijaan. Pääasialliset ongelmakohdat esitellään ja näiden ratkaisemiseen valitut keinot perustellaan. Osio antaa myös kuvan suunnitteluprosessista, jota tehtiin ennen työkalun kehityksen aloittamista ja sen aikana.

Toiminnallisessa osuudessa kuvataan Web – lomakkeiden määritystyökalun toteutus ja selitetään miten kaikki sen toiminnallisuudet toimivat. Osiossa kuvataan myös työkalun taustalla oleva tietokantarakenne.

Lopuksi pohditaan saavutettuja tuloksia ja annetaan jatkokehitys – ja parannusideoita.

Asiasanat
ohjelmistokehitys, HTML, JavaScript

Abstract

HAAGA-HELIA
University of Applied Sciences

Degree Programme in Business Information Technology

Author(s) Juha Meriläinen	**Group or year of entry** 2010
The title of thesis Web form definition tool for Fivaldi WorkFlow application	**Number of report pages and attachment pages** 28 + 5
Advisor(s) Lili Aunimo	

The purpose of this thesis was to plan and develop a drag & drop style web form definition tool and the study was assigned by Oy Finnvalli Finland Ab. The tool was implemented as a part of a larger property management application and its purpose was to enable the creation of customized forms for client-specific needs. The tool attained as a result of this thesis was also documented properly.

The thesis first introduced the software environment and the basic structure of the applications in the assigner's collection of financial and property management applications, which are delivered using the Software as a Service -model. Technical and user requirements regarding the tool were also presented.

In the theoretical part of the thesis, some solutions and frameworks used for web form creation were looked into. They were used as inspiration and reference for this thesis. In addition, this part explained the need for a customized solution instead of choosing any of the readily available ones. Principal design dilemmas were also presented and the chosen solutions to overcome these were explained. The theoretical part also gave insight into the planning process that preceded and continued throughout the development project of the tool.

The empirical part explained the design of the web form definition tool and it went through how every one of its functionalities works. The underlying database design was described, as well.

Finally, the thesis reflects on the achieved results and it gives suggestions for further development and improvements.

Key words
software development, HTML, JavaScript

Sanasto

Ajax (Asynchronous JavaScript and XML)

Ajax on tekniikka, jolla voi luoda nopeita ja dynaamisia Web – sivuja. Se mahdollistaa Web – sivujen päivityksen asynkronisesti, ilman että koko sivu ladataan uudestaan. (w3schools.com.)

Android

Linux – pohjainen käyttöjärjestelmä, pääasiassa mobiililaitteille.

API (Application Programming Interface)

API eli ohjelmointirajapinta on protokolla, jonka mukaan eri sovellukset keskustelevat keskenään (Wikipedia a).

Backend

Nimitys, jota käytetään palvelinpäässä sijaitsevasta ohjelmistokoodista.

Collision detection

Algoritmi, jolla pyritään toteamaan kahden olion yhteentörmäys (Firth 2011a).

CSS (Cascading Style Sheets)

CSS on mekanismi, jonka avulla voidaan lisätä tyylejä, kuten värejä ja fontteja, Web – sivuille (W3C).

Drag & drop

Osoittimen ele, jossa valitaan jokin kohde, yleensä hiiren painikkeella, ja raahataan se pois alkuperäisestä sijainnista (Wikipedia b).

Frontend

Nimitys, jota käytetään käyttöliittymän ohjelmistokoodista.

Grid

Kaksiulotteinen rakenne, jossa vaaka – ja pystysuorassa poikkeavat akselit muodostavat ruudukon (Wikipedia c).

HTML (HyperText Markup Language)

Kuvauskieli, jolla yleensä muodostetaan Web – sivujen rakenne.

HTTP (HyperText Transfer Protocol)

HTTP on protokolla, jota käytetään tiedonsiirtoon selaimien ja WWW – palvelimien välillä.

JavaScript

Ohjelmointikieli, jolla yleensä lisätään Web – sivuille dynaamista toiminnallisuutta.

JSON

Merkintäkieli, jolla voi kuvata tietorakenteita.

jQuery

JavaScript – kirjasto, joka yksinkertaistaa monia toimintoja helppokäyttöisen rajapinnan avulla (The jQuery Foundation).

jQuery UI

jQuery – kirjaston päälle rakennettu kirjasto, joka mahdollistaa useiden interaktiivisten toimintojen hyödyntämisen käyttöliittymässä.

Oracle Forms

Oracle -tuote, jolla voidaan rakentaa ikkunapohjaisia sovelluksia Oracle – tietokannan päälle.

PL/SQL

Oraclen kehittämä ohjelmointikieli, joka yhdistelee SQL – kieltä ja ohjelmallisia toimintoja, kuten muuttujia ja ehtoja.

PL/SQL - kursori

PL/SQL – ohjelmointikielen rakenne, jolle voidaan asettaa SQL – kysely, jonka jälkeen kyselyn tulosta voi manipuloida rakenteen avulla.

SaaS (Software as a Service)

SaaS tarkoittaa ohjelmiston toimittamista palveluna. Asiakkaat käyttävät samaa tuotantoympäristöä ja maksavat käytön laajuuden mukaan. Toteutus on yleensä selainpohjainen. (Wikipedia d.)

SQL (Structured Query Language)

Kyselykieli, jolla voidaan manipuloida tietokantaa.

Spring

Ohjelmointikehys Java – ohjelmointikielelle.

Wrapper

Ohjelmiston funktio, jonka päätarkoituksena on kutsua toista funktiota.

WYSIWYG (What You See Is What You Get)

WYSIWYG tarkoittaa sitä, että muokattaessa jotain se vastaa ulkonäöllisesti valmista lopputulosta.

XML (Extensible Markup Language)

XML on merkintäkieli, jolla voidaan kuvata ja jäsennellä erilaisia tietoja.

Sisällys

1 Johdanto

Opinnäytetyön tavoitteena oli suunnitella ja toteuttaa Web – lomakkeiden määritystyökalu osaksi toimeksiantaja Oy Finnvalli Finland Ab:n (myöhemmin Finnvalli) uutta Fivaldi WorkFlow (myöhemmin FWF) – sovellusta, jolla hallitaan kiinteistönhallinnan erilaisia tehtäviä ja tapahtumia. FWF – sovelluksen tulevilla käyttäjillä on erilaisia käyttötarpeita, joten ne pyritään täyttämään dynaamisin, käyttäjän itse räätälöimin lomakkein. Välillisenä tavoitteena onkin selvittää lähtökohtia ja erilaisia vaihtoehtoja dynaamisen, JavaScript -pohjaisen käyttöliittymän toteutukseen, jolla käyttäjä pystyy itse luomaan Web - lomakkeita ilman minkäänlaista ohjelmointikokemusta.

Opinnäytetyö käsittää Web – lomakkeiden määritystyökalun suunnittelun, toteutuksen eli ohjelmoinnin, sekä dokumentoinnin. Tuloksen testaus rajattiin opinnäytetyön ulkopuolelle aikataulullisista syistä, sekä sen takia että testausta tehdään myös opinnäytetyön ulkopuolisen tahon, Finnvallin testaustiimin toimesta. Käyttötestausta luonnollisesti tehtiin toteutuksen aikana.

Opinnäytetyössä kerrotaan hieman Finnvallin Fivaldi – palvelukonseptin sovellusten rakenteesta ja tämän pohjalta esitellään FWF -sovellus, jonka osaksi Web – lomakkeiden määritystyökalu toteutetaan. Tämän jälkeen esitellään sovellustoteutuksen lähtökohdat; käyttäjävaatimukset, tekniset vaatimukset, sekä tekijän valmiudet työkalun toteutukseen. Suunnitteluprosessin ja valittujen toteutustapojen perustelujen kautta siirrytään valmiin työkalun toiminnan kuvaukseen. Lopuksi esitellään opinnäytetyön tulokset, pohditaan jatkokehitysmahdollisuuksia ja arvioidaan opinnäytetyöprosessin onnistumista.

1.1 Toimeksiantaja

Oy Finnvalli Finland Ab on suomalainen ohjelmistotalo, jonka päätuotteena on mm. tilitoimistoille, kiinteistönhallintaan ja urakointiin ratkaisuja tarjoava SaaS – pohjainen Fivaldi -palvelukonsepti. Yritys kasvaa nopeasti ja sen liikevaihto vuonna 2011 oli 3,35 miljoonaa euroa. Finnvalli työllistää noin 30 henkilöä mm. myynnin, sovellusasiantuntijan, sovelluskehittäjän ja järjestelmäasiantuntijan tehtävissä. (Oy Finnvalli Finland Ab.)

2 Ympäristö ja vaatimukset

2.1 Fivaldi Web -sovellukset

Fivaldi – sovellukset on toteutettu osittain HTML-, CSS- ja JavaScript-yhdistelmällä ja osittain Oraclen Forms – sovelluksina. Kummassakin tapauksessa backend -toiminnallisuudet toteuteaan PL/SQL -ohjelmointikielen avulla, Oracle -tietokannassa. Toiminnallisuudet sijaitsevat siis tietokannassa, pois lukien käyttöliittymän toiminnallisuus. JavaScript -pohjaisissa sovelluksissa kutsutaan näitä taustalogiikkaa toteuttavia, tietokannassa sijaitsevia PL/SQL -proseduureja ja -funktioita Ajax – kutsuin, tai HTML -lomakkeen lähetyksen yhteydessä. Kutsuttavat proseduurit ja funktiot ovat ns. wrappereita, joissa tarkistetaan käyttöoikeudet, tarkistetaan istunnon voimassaolo, tarkistetaan mahdolliset parametrit, hoidetaan virheidenkäsittely sekä kutsutaan toista proseduuria tai funktioita, joka suorittaa halutut toiminnallisuudet, kuten esimerkiksi tietokantaan tallennuksen. Wrappereitä voi siis pitää jonkinlaisena rajapintana frontendin ja backendin välillä, vaikka ne sijaitsevatkin todellisuudessa backendissä. Tarvittaessa sisäinen proseduuri tai funktio palauttaa tietoa wrapperille, joka puolestaan palauttaa sen käyttöliittymään. Sovelluksesta ja käyttötarkoituksesta riippuen on palautettava tieto useimmiten HTML-, XML- tai JSON- muotoista, joiden kaikkien muodostamista PL/SQL tukee.

Sovellukset upotetaan Fivaldi – käyttöliittymään niin, että navigointi ja yleiset toiminnallisuudet pysyvät näkyvissä ja vain tietty alue muuttuu käytettävän sovelluksen mukaan (liite 1). Projektissa toteutettu web-lomakkeiden määritystyökalu kuuluu FWF–sovellukseen, joka on edellä mainitun tyyppinen sovellus. Oracle Forms – sovellukset puolestaan aukeavat omaan sovellusikkunaansa.

2.2 Fivaldi WorkFlow

Fivaldi WorkFlow on uusi kehityksessä oleva sovellus Fivaldi – kokonaisjärjestelmään. Se korvaa nykyisen Työmääräin – sovelluksen ja samalla modernisoi ja tuo uusia toiminnallisuuksia käyttäjilleen. Sovelluksen päätoimintoja ovat kiinteistönhallintaan liittyvien tehtävien hallinta ja ohjaaminen, aina säännöllisistä kuntotarkastuksista suurempiin remontteihin.

Vanha Työmääräin – sovellus on toteutettu Oracle Forms:lla, joka korvataan nyt normaalilla selainpohjaisella sovelluksella, ja toteutetaan hyväksikäyttäen HTML:ää, CSS:ää, JavaScriptiä, jQuery:ä ja Ajaxia. Pääasialliset parannukset vanhaan sovellukseen ovat tehtäväkohtaiset HTML -lomakkeet dynaamisin syöttökentin, tehtävien kohteiden rajattomuus, prosessien luominen tehtävien suorittamiselle, rajaton määrä osapuolia tehtävällä, sekä osapuolien laskuttaminen.

Erilaiset tehtävät ovat sovelluksen peruskomponentti, johon muu toiminnallisuus yhdistyy. Tehtävällä on tyyppi, joka kertoo minkälainen tehtävä on kyseessä. Jokaisella tehtävällä voi myös olla kohteita, kenttiä, tapahtumia, osapuolia, rooleja ja prosesseja. Käyttäjä luo ja määrittelee tehtävät haluamallaan tavalla.

Tehtävän kohde kertoo, mihin tehtävä liittyy. Kohde voi olla vaikka tietty huoneisto, rakennuksen osa tai jokin laite. Kohdetyypit valitaan tehtävätyypin perusteella ja ne voivat olla joko pakollisia, yksittäisiä tai niitä voi olla useita. Kohteet määritellään eri kohdetasoille, jolloin ne muodostavat puurakenteen. Ensimmäinen taso voi olla vaikka kunta, siitä seuraava kaupunginosa ja niin edelleen. Tasot ja kohteet ovat täysin käyttäjän määriteltävissä, sovelluksen perustiedoissa.

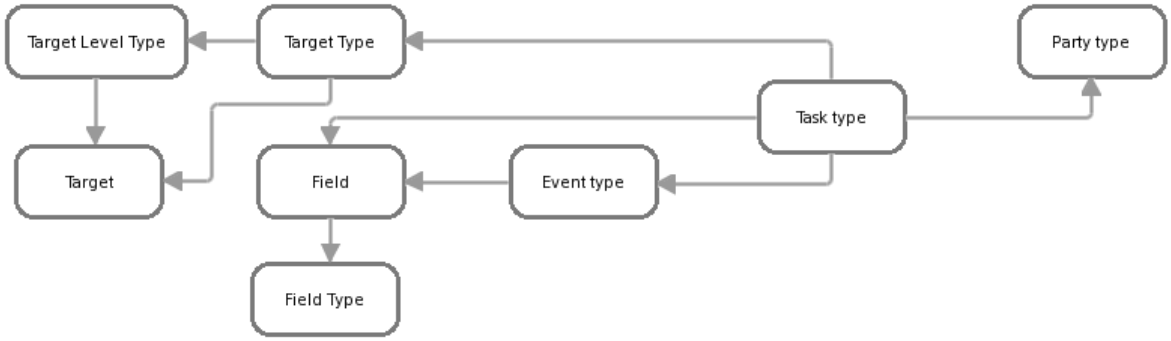
Tehtävään liittyy erityyppisiä lomakekenttiä, jotka valitaan myös tehtävätyypin perusteella. Kuten kohteet, voivat kentät olla myös pakollisia, yksittäisiä, tai useita samantyyppistä. Sovelluksessa on tiettyjä kiinteitä kenttiä, jotka ovat aina olemassa ja käytettävissä. Käyttäjä voi myös luoda ja määritellä itse kenttiä omiin tarpeisiinsa.

Tehtävällä voi olla myös erilaisia tapahtumia. Nekin määräytyvät tehtävätyypin mukaan. Tehtävän tapahtumat kuvaavat tehtävän aikana suoritettavat työt ja muut tapahtumat. Tapahtumat luodaan joko manuaalisesti tai prosessin määritteleminä. Tapahtumilla on tyyppi, joka kuvaa sen tarkoitusta, sekä kuten tehtävillä, voi tapahtumalla olla erilaisia lomakekenttiä.

Tehtävän osapuolet voivat olla joko henkilöitä tai yrityksiä. Osapuolilla on tyyppi, joka ilmaisee miten osapuoli liittyy tehtävään. Osapuolet voivat olla esimerkiksi vastuullisia tietyistä tapahtumista, ja heitä voi myös laskuttaa.

Tehtävän roolit ovat käyttäjäkohtaisia, ja ne määritellään sovelluksen perustiedoissa. Käyttäjällä voi myös olla useampi rooli, eri tapahtumissa. Tehtävien tapahtumien näkymät voivat vaihdella roolikohtaisesti.

Sovelluksen perustietoihin kuuluu aiemmin mainittujen määrittelyiden lisäksi yhtiön perustietojen asetukset, huoltoyhtiön perustietojen asetukset sekä tehtävän prosessin rakentaminen, jossa määritellään miten tehtävä muodostuu. Usein on niin, että yhtiö kuuluu jonkin suuremman huoltoyhtiön alle, jolta perusasetukset ja määrittelyt tulevat, ja joita omat yrityskohtaiset asetukset täydentävät tai korvaavat. Perustietoihin kuuluu myös tehtävä ja tapahtumalomakkeiden määrittely, joka toteutetaan tämän projektin tuloksena saatavalla web -lomakkeiden määritystyökalulla.



Kuva 1. FWF -rakenne

2.3 Tekniset vaatimukset

Määritystyökalu toteutetaan käyttäen vähintään PL/SQL-, SQL-, HTML-, CSS-, JavaScript-, jQuery- sekä Ajax- ohjelmointikieliä ja -kehyksiä. Tarpeen mukaan projektissa voi hyödyntää muita vapaiden lisenssien alla julkaistuja liitännäisiä, ohjelmistokehyksiä tai kirjastoja. Tietokantana on Oracle – tietokanta, ja käyttöliittymän ja tietokannan välinen rajapinta toteutetaan PL/SQL – proseduurein ja -funktioin.

Työkalun on toimittava vähintään Internet Explorer-, Mozilla Firefox-, Safari- ja Google Chrome – selaimilla. Sovelluskehityksessä on noudatettava Finnvallin sisäisiä ohjeita ja ohjelmointikäytäntöjä.

2.4 Käyttäjävaatimukset

Projektin tuloksena syntynyt selainpohjainen työkalu mahdollistaa dynaamisten syöttölomakkeiden määrittelemisen, sekä tallentamisen jatkokäyttöä ja muokkaamista varten. Työkalun tietokantaan tallentamista tiedoista on oltava mahdollista luoda valmiita lomakkeita oikeaa käyttöä varten. Lomakkeen asettelun muokkauksen on toimittava drag & drop – tyylisesti ja sen on noudatettava WYSIWYG – periaatetta. Käyttäjältä ei myöskään vaadita minkäänlaista ohjelmointitaitoa.

Käyttäjän käyttötarpeita työkalulle ovat

- lomakkeen ulkoasutyypin valitseminen
- lomakkeen tyypin valitseminen
- erityyppisien kenttien lisääminen lomakkeelle
- kenttien asettelun muuttaminen
- kenttien koon muuttaminen.
- kenttien poistaminen
- kenttien määritteleminen syöttö- tai näyttö- kentiksi
- lomakkeen tallentaminen tietokantaan
- tallennetun lomakkeen tarkastelu ja muokkaus
- tallennetun lomakkeen poisto

Käyttötarpeita toteuttaessaan on työkalun oltava käytettävyydeltään mahdollisimman yksinkertainen ja looginen.

2.5 Projektin toteuttamisen edellytykset

Tavoitteet FWF – sovellukselle käytiin läpi kokouksessa, yhdessä kaikkien sovelluksen kehitykseen osallistuvien kanssa. Tämän projektin toimeksiantaja, sovelluksen pääpiir-

teiden suunnittelija ja määrittelyiden tekijä esitteli tavoitteet, joiden pohjalta sai kuvan siitä, mitä Web – lomakkeiden määritystyökalulta haettiin.

Kokemusta työskentelystä Finnvallissa ja erilaisten Fivaldi – tuotteiden parissa toteuttajalla oli vajaan 1,5 vuoden verran ennen projektin alkua. PL/SQL – ohjelmointikieli tuli tutuksi nimenomaisissa työtehtävissä, kuten myös ensikosketukset jQuery – kirjastoon sekä syvällisemmin Web – kehitykseen yleensä. Taustalla toteuttajalla oli myös sovelluskehityskokemusta erityyppisten ympäristöjen parissa, muun muassa Spring ja Android – maailmasta. Myös opinnoissa saavutetut perustiedot SQL – kielestä ja tietokannoista ovat syventyneet Finnvallissa työskentelyn aikana, ja ne olivat myös tärkeä lähtökohta projektin toteuttamisessa.

3 Tietoperusta ja toteutussuunnitelma

3.1 Toteutustavan valinta

Ennen ohjelmoinnin aloittamista tehtiin taustaselvitystä mahdollisista toteutustavoista. Ensin kartoitettiin mahdollisia valmiita ratkaisuja lomakkeiden luontiin. Näistä haettiin lähinnä ideoita ja inspiraatiota, koska oli melko ilmeistä, että Fivaldi – ympäristöön tarvittaisiin pitkälti itse mukautettu ratkaisu. Tämän jälkeen pohdittiin toteutuksen pääasiallisia ongelmakohtia, drag & drop – toiminnallisuutta sekä WYSIWYG -ulkoasua, joiden ratkaisuun ei ollut yksiselitteistä vastausta.

Koska FWF – sovellus oli määrä toteuttaa hyödyntäen JavaScript-, jQuery- ja Ajax-ohjelmointikieliä ja – kehyksiä, ja vaatimuksiin niin kirjattiin, oli tärkeää myös suunnitella Web – lomakkeiden määritystyökalu näitä silmälläpitäen.

3.1.1 Valmiit työkalut ja liitännäiset

Web – lomakkeiden helppoa ja nopeaa, ilman ohjelmointitaitoja vaativaa, luontia varten on kehitetty lukuisia eri ratkaisuja, erilaisiin tarpeisiin. Merkittävä osa näistä on tyypiltään sellaisia, joissa lomakkeen luonti, tallennus ja käyttö tapahtuu tarjoajan palvelussa. Palveluntarjoaja ylläpitää lomakkeita ja niistä kerättyjä tietoja, jotka asiakas kerää tarjoajalta omaa käyttöä varten. Esimerkkejä tämäntyyppisistä ratkaisuista on JotForm ja Google Forms. Lomakkeiden ulkoasun muokkauksen taso vaihtelee eri työkalujen välillä rajustikin. Edellä mainittujen JotForm:in ja Google Forms:in välillä on esimerkiksi suuri kontrasti kustomoitavuuden tasossa, joka onkin varmasti näiden kahden välillä ero maksullisen ja ilmaisen välillä. Tällaista ratkaisua ei tietenkään voitu edes ajatella hyödynnettävän tämän opinnäytetyön tapauksessa, koska Finnvallin asiakkaiden tietoja joutuisi luovuttamaan ulkopuoliselle taholle, ja integrointi FWF – sovellukseen ei olisi mahdollista. Maksulliset ja lisenssin vaativat työkalut eivät myöskään tulleet kyseeseen. Näiden ratkaisujen tutkinnan funktio olikin, kuten aiemmin mainittiin, lähinnä saada kuva siitä minkälaisia toteutuksia markkinoilla on kyseiseen tapaukseen.

Opinnäytetyön kannalta mielenkiintoisempia vaihtoehtoja olivat liitännäistyyppiset ratkaisut, jotka voisi mahdollisesti integroida FWF – sovellukseen. Tällainen on esimer-

kiksi jQuery WYSIWYG Web Form builder (Google Code). Ratkaisu on alustariippumaton, ja jQuery / jQuery UI – pohjainen joten sitä harkittiin vakavasti hyödynnettäväksi opinnäytetyössä. Vaikkakin jQuery WYSIWYG Web Form builder olisi ehkä ollut sopivin ja helpoiten integroitava löydetyistä valmiista työkaluista, todettiin että siinä on liikaa ylimääräisiä toimintoja ja FWF – lomakkeille tarpeellisia toimintoja tarvitsisi kuitenkin lisätä. Näistä syistä päädyttiin toteuttamaan oma ratkaisu, jotta saataisiin sellainen työkalu, jossa on vain ne toiminnallisuudet joita tarvitaan, ja integrointi FWF – sovellukseen ja muuhun Fivaldi – ympäristöön on luontevaa. Näin minimoidaan Jakob Nielsenin esittelemää WYSIWYG -editorien ongelmaa, jonka mukaan useimmat käyttäjät eivät löydä kaikkia toimintoja niiden paljoudesta, ja sen takia eivät niitä koskaan käytä (Nielsen 2005).

Yhteistä eri ratkaisuissa, joissa hyödynnettiin drag & drop – tyylistä lomakkeen asettelun muokkausta on se, että niissä on täytynyt ratkaista HTML – elementtien yhteentörmäyksen tunnistus, ja tämän perusteella elementtien uudelleenjärjestäminen loogisella tavalla. Elementtien sijaintitiedot on myös oltava saatavilla tallennusta varten. Opinnäytetyössä toteutettava työkalu ei ollut tästä poikkeus vaan samat ongelmat pitivät paikkansa.

3.1.2 Drag & drop- ja WYSIWYG -käyttöliittymä

Haastavin osuus työkalussa oli lomakekenttien sijaintien määrittelyn toteutus, joka drag & drop -toiminnan, sekä dynaamisten kenttien kokojen takia on melko monimutkainen.

Sovelluksen taustalle luodun tietokantarakenteen (liite 4) perusteella toteutuksen rautalangaksi valittiin grid – tyylinen malli sarakkeineen ja riveineen, jossa elementtien sijainnit pystytään määrittämään sarake – ja rivinumeroin, sekä koot täytettyjen sarakkeiden ja rivien mukaan. Tämä mahdollistaa lomakekenttien sijaintien pysymisen muuttumattomina suhteessa toisiinsa, generoitaessa lomaketta käyttöä varten tietokantaan tallennetun lomakemäärittelyn perusteella. Lomakekenttien liikuttaminen sekä kokojen muutos pysyy myös helpommin hallinnassa, kun se tapahtuu aina sarakkeen tai rivin perusteella, eikä täysin vapaasti lomakealueella.

Koska lomakenttiä, eli HTML – elementtejä piti pystyä raahaamaan lomakealueella, olikin jotenkin tunnistettava niiden yhteentörmäys. Näin estetään elementtien päällekkäisyys ja hallitaan niiden sijaintimuutokset suhteessa liikutettavaan elementtiin sekä toisiinsa. Yhteentörmäyksen tunnistukseen käytetään ns. collision detection – algoritmia, jolla havaitaan kahden olion yhteentörmäys, kun elementtien keskikohdat ovat tarpeeksi lähellä toisiaan (Firth 2011b). Yhteentörmäyksen tapahtuessa täytyy tilanne ratkaista, joka vaatii taas omat algoritminsa, jolla ratkaistaan törmäyksen kohteena olleen elementin liikkumiskohde. Törmäyksen kohteena ollut elementti voi näin liikkuessaan aiheuttaa toisen törmäyksen, joka pitää myös ratkaista. Näin voi aiheutua pitkäkin ketjureaktio, jonka hallitseminen ei ole täysin yksioikoista.

Yhteentörmäävien elementtien käsittelyyn lähdettiin etsimään ratkaisua JavaScript / jQuery – pohjaisista liitännäisistä, koska arvioitiin että varmasti moneen kertaan ratkaistua ongelmaa on turha keksiä uudestaan. Tähän tarkoitukseen löytyi kuitenkin vain kaksi varteenotettavaa liitännäistä. Gridster.js (myöhemmin gridster), sekä JQuery UI Draggable Collision. Ensimmäisen kohdalla drag & drop - vaatimukset täyttyvät täydellisesti ja käyttöönotto vaikutti yksinkertaiselta. Liitännäinen ei kuitenkaan toimi vanhemmilla Internet Explorer – selaimilla. Toinen liitännäinen tarjoaa myös yhteentörmäyksen tunnistuksen yhdessä jQuery UI:n draggable – interaktion kanssa (eruciform). Toisin kuin gridster, se ei kuitenkaan itsessään sisällä toiminnallisuuksia raahattavien elementtien koon muutoksiin, lisäämiseen ja poistamiseen, sekä grid – tyyppiseen rakenteeseen. JQuery UI Draggable Collision vaatii siis hieman enemmän rakenteita ympärilleen, jotka gridsteristä jo löytyivät, joka tietysti teki gridsteristä houkuttelevamman vaihtoehdon.

Kummassakin liitännäisessä oli huonot puolensa, joten aluksi lähdettiin toteuttamaan Web – lomakkeiden määritystyökalua ainoastaan jQuery UI –kirjaston toiminnoilla. Idea tähän näkökulmaan tuli esimerkistä, jossa oli toteutettu drag & drop - käyttöliittymän mallinnus jQuery UI – kirjaston Sortable – interaktiota hyväksikäyttäen (Tamada 2011). Houkuttelevaksi ratkaisun teki se, että ei olisi tarvinnut käyttää erillistä liitännäistä työkalun toteutuksessa, niiden lisäksi, joita FWF – sovelluksessa oli jo käytössä. Ongelmaksi muodostui kuitenkin lomakekenttien liikuttaminen, jossa kenttien järjestämi-

nen onnistuttiin toteuttamaan vain niin, että ne ryhmittyivät aina vasemmalle, jolloin halutun lomakkeen rakentaminen oli hankalaa. Myös kenttien sijaintitietojen ylläpito oli tarpeettoman monimutkaista samasta syystä.

Käytettävyysongelmaan ei löytynyt jQuery UI -kirjastosta ratkaisua, joten lopulta päädyttiin hyödyntämään gridsteriä. Päätettiin, että liitännäistä muokataan toimimaan aikaisemmilla Internet Explorer – versioilla myöhemmin, mikäli projektin aikataulu sen sallii. Vaihtoehtoisesti ominaisuus toteutetaan myöhempänä ajankohtana, projektin ulkopuolella, jos sille katsotaan olevan todellista tarvetta.

Työkalun WYSIWYG-aspekti täyttyisi sillä, että kenttiä lisätessä lomakkeelle, ne ovat määrittelyssä tyypiltään, kooltaan ja ulkoasultaan samanlaisia, kuin käyttöä varten generoidulla lomakkeella. Kenttien koon muutos näkyy työkalussa heti ja kenttien sijainnit pysyvät samoina määrittelyssä ja käytettävässä lomakkeessa.

3.2 Gridster.js

Gridster on jQuery – liitännäinen, joka mahdollista drag & drop -tyylisen useamman sarakkeen grid -ulkoasun. Elementit voivat olla useamman sarakkeen tai rivin kokoisia ja elementtejä voi dynaamisesti lisätä ja poistaa. Elementtejä raahatessa alle jäävä elementti väistää alaspäin, mahdollisesti sen alle jäävä myös samalla tavalla ja niin edelleen. Elementin voi liikuttaa määritellyllä alueella mihin haluaa ja elementtien välissä voi olla tyhjää tilaa. Elementtien kokoja voi myös helposti muuttaa ja muut elementit järjestyvät uudestaan sen mukaan. Tuettuja selaimia ovat Internet Explorer 9+, Firefox, Chrome, Safari ja Opera. Liitännäinen tarvitsee toimiakseen myös jQuery – kirjastoa, joka on ladattava erikseen. (Ducksboard.)

Gridster.js – liitännäiseen kuuluu kaksi tiedostoa, ”jquery.gridster.js” sekä ”jquery.gridster.css”. Tiedostot on lisensoitu MIT – lisenssillä, joka mahdollistaa liitännäisen käytön myös kaupallisissa sovelluksissa, kunhan kopio lisenssistä säilytetään tiedostoissa (Open Source Initiative).

Yksinkertaisimmillaan gridster sisältää ”div” – elementin, jolle annetaan ”gridster” – luokka. Elementin sisään sijoitetaan ”ul” -elementti, jonka alle ”li” – elementtejä, joilla on attribuutteina elementin sijainti ja koko. Nämä attribuutit ovat HTML5 data – attribuutteja, jotka ovat yksi syy miksi gridster ei toimi vanhemmilla Internet Explorer -versioilla.

```
<div class="gridster">
    <ul>
        <li data-row="1" data-col="1" data-sizex="1" data-sizey="1"></li>
        <li data-row="2" data-col="1" data-sizex="1" data-sizey="1"></li>
        <li data-row="3" data-col="1" data-sizex="1" data-sizey="1"></li>

        <li data-row="1" data-col="2" data-sizex="2" data-sizey="1"></li>
        <li data-row="2" data-col="2" data-sizex="2" data-sizey="2"></li>

        <li data-row="1" data-col="4" data-sizex="1" data-sizey="1"></li>
        <li data-row="2" data-col="4" data-sizex="2" data-sizey="1"></li>
        <li data-row="3" data-col="4" data-sizex="1" data-sizey="1"></li>

        <li data-row="1" data-col="5" data-sizex="1" data-sizey="1"></li>
        <li data-row="3" data-col="5" data-sizex="1" data-sizey="1"></li>

        <li data-row="1" data-col="6" data-sizex="1" data-sizey="1"></li>
        <li data-row="2" data-col="6" data-sizex="1" data-sizey="2"></li>
    </ul>
</div>
```

Kuva 2. Gridster perusrakenne

Käyttöön gridster otetaan rekisteröimällä se ”ul” – elementille. Tässä yhteydessä voi toimintaa räätälöidä haluamakseen usein eri parametrein, joista lisätietoa löytyy gridsterin kotisivujen dokumentaatiosta.

```
$(".gridster ul").gridster({
    widget_margins: [10, 10],
    widget_base_dimensions: [200, 200]
});
```

Kuva 3. gridster.js rekisteröinti

Gridsterin API:iin pääsee myös käsiksi asettamalla se muuttujaan. Tätä tarvitaan, jotta voidaan kutsua mm. elementtien poistoa, lisäystä ja koon muutosta. Näitä toiminnallisuuksia esitellään myöhemmin projektin tuloksien yhteydessä, jossa niitä hyödynnetään lomakekenttien sijaintienmäärittelyn muodostamisessa.

```
1  var gridster = $(".gridster ul").gridster().data('gridster');
```

Kuva 4. gridster.js API

4 Opinnäytetyön tulokset

4.1 Työkalun rakenne

Web -lomakkeiden määritystyökalu on osa FWF -sovelluksen perustietoja, jotka käsittävät sovelluksen käyttöön vaikuttavat yleiset, yhtiö sekä huoltoyhtiö – kohtaiset, asetukset. Näihin asetuksiin on siis vain tietyillä pääkäyttäjillä oikeudet. Työkaluun pääsee sovelluksen navigaatiosta, olettaen että käyttäjällä on oikeudet työkalun käyttöön. Navigoidessa työkaluun sovellus tekee Ajax – kutsun PL/SQL – proseduuriin ”tmar_layout_editor.main”. Proseduuri tarkistaa käyttöoikeudet, istunnon voimassaolon, sekä palauttaa sovellusikkunan sisällön HTML -muotoisena ja se lisätään näkyviin (liite 2). Valintalistojen arvoja, otsikoita ja muita tekstejä ei ole kovakoodattu vaan PL/SQL -proseduuri hakee niihin ajantasaiset, yritys ja kieliasetuskohtaiset arvot tietokannasta. PL/SQL kursorille määritetään SQL – kysely, joka palauttaa tallennettujen lomakemäärittelyiden tunnisteen ja kuvauksen, yrityksen ja kielen perusteella (kuva 5).

```
6         cursor c_saved_layout
7         is
8              select a.layout_id layout_id, b.description description
9                from tmar_layout_t a, tmar_layout_desc_t b
10               where      a.yt = v_tunnistus.yt
11                     and a.yt = b.yt
12                     and a.layout_id = b.layout_id
13                     and b.kielikoodi = v_kielikoodi
14           order by a.layout_id;
```

Kuva 5. PL/SQL kursori

Aiemmin määritetty kursori avataan ja tulostetaan HTML – syntaksia sen sisällön perusteella (kuva 6). Näin saadaan muodostettua kokonainen ”div” – elementti, jonka sisällä on ”label” – elementti otsikoimassa valintalistaa, jonka vaihtoehtoina on tietokannasta haettujen, tallennettujen lomakemäärittelyiden tunniste -ja kuvausparit.

```
85      -- Load layout.
86      htp.p (
87            '<div class="top-section layout-load"><label for="s-load">'
88         || tmar_layout_editor_util.k ('load_layout')
89         || '</label></br><select id="s-load"><option selected="selected" value="default">-----</option>');
90
91      open c_saved_layout;
92
93      loop
94         fetch c_saved_layout into v_saved_layout;
95
96         exit when c_saved_layout%notfound;
97         htp.p (
98               '<option value="'
99            || v_saved_layout.layout_id
100           || '">'
101           || v_saved_layout.description
102           || '</option>');
103     end loop;
104
105     close c_saved_layout;
106
107     htp.p ('</select></div>');
```

Kuva 6. HTML:n luonti

Työkalun käyttämät JavaScript ja CSS – tiedostot on jo ladattu sovelluksen avaamisen yhteydessä. Tyylitiedostoja on kaksi kappaletta, yksi työkalun elementtien tyylittelyyn, sekä yksi gridster -tyylitiedosto, jota on hieman muokattu Fivaldin teemaan sopivammaksi. JavaScript -tiedostoja on myös kaksi. Toisessa on käyttöliittymän toiminnallisuudet, pois lukien gridsterin toiminnallisuus, joka on omassa tiedostossaan. Tähän tiedostoon ei projektin puitteissa tarvinnut tehdä muutoksia, mutta jos gridsterin haluaa puhumaan yhteistä kieltä vanhempien Internet Explorer – selainten kanssa, joutuu tiedostoa penkomaan. Gridsterin vaatimat, ja muutenkin hyödynnettävät jQuery sekä jQuery UI – kirjastot ovat käytössä koko FWF -sovelluksessa, joten niitä ei tarvitse erikseen asettaa.

Web -lomakkeiden määritystyökalu koostuu työkalupalkista, jossa on erilaisia toiminto- ja määrittelyä varten, sekä suuremmasta alueesta, jossa käyttäjä voi luoda haluamansa lomakkeen drag & drop -tyylisesti yhdessä työkalupalkin toimintojen kanssa. Tähän on siis rekisteröity käyttöön aiemmin esitelty gridster (liite 3). Työkalupalkki muuttuu käyttäjän valintojen mukaan, näyttäen aina tarvittavat toiminnot tietyssä vaiheessa. Suurin osa toiminnallisuuksista siis piilotetaan sivua ladatessa, ainoat näkyvät toiminnallisuudet ovat tallennettujen lomakemääritysten lataus ja uuden lomakemäärittelyn luonti. Lomakealue on tyhjä kunnes käyttäjä lisää sille kenttiä tai lataa tallennetun lomakemäärityksen, jolle on lisätty kenttiä. Uusi kenttä lisätään aina ensimmäisen rivin ensimmäi-

seen sarakkeeseen ja olemassa olevat kentät siirtyvät sen mukaan. Käyttäjä voi liikutella kenttiä vapaasti, ainoastaan sarakemäärän rajoittamana. Kentät eivät voi myöskään olla päällekkäin. Kun käyttäjä liikuttaa kentän toisen päälle, siirtyy alle jäävä kenttä tai kentät aina alaspäin, pois tieltä. Sama pitää paikkaansa myös liikutuksen seurauksena liikkuvan kentän alle jääville kentille ja niin edespäin. Tämän toiminnallisuuden hoitaa gridster.js – liitännäinen, joka myös ylläpitää tiedot kenttien sijainneista. Kenttien liikuttaminen tapahtuu kentän otsikosta drag & drop – tyylisesti. Työkalu näyttää myös varjon, joka ilmentää sijaintia mihin kenttä sijoittuu, jos käyttäjä päästää irti kentästä. Muualle kuin otsikkoon klikkaamalla käyttäjä valitsee kentän, jolloin siihen voi kohdistaa työkalupalkin kenttäkohtaisia toimintoja. Toimintoja voi suorittaa yhtäaikaisesti kaikille valituille kentille. Kaiken tämän mahdollistaa gridster, sekä jQuery UI – kirjaston ”selectable” – interaktio.

```
24          // Register gridster and selectable. Mousedown binding allows deselecting.
25          $("#containment").gridster({
26                widget_margins: [5, 5],
27                widget_base_dimensions: [glbColWidth - 10, 50],
28                max_size_x: glbColumns,
29                avoid_overlapped_widgets: true,
30                draggable: {
31                      handle: ".handle"
32                }
33          }).bind("mousedown", function(e) {
34                e.metaKey = true;
35          }).selectable({
36                filter: "li",
37                selected: function(event, ui) {
38                      $(".item-controls").children().removeAttr('disabled');
39                },
40                unselected: function(event, ui) {
41                      if ($(".ui-selected").length === 0) {
42                            $(".item-controls").children().attr('disabled', 'disabled');
43                      }
44                }
45          });
46
47          // Get hold of the gridster API.
48          glbGridster = $("#containment").gridster().data('gridster');
```

Kuva 7. Gridster ja selectable käyttöönotto

Gridster sekä selectable rekisteröidään sivun latauksen yhteydessä luodulle ”ul” – elementille (kuva 7). Gridsterille annetut parametrit määräävät sille lisättyjen elementtien marginaalit ja koon. Elementtien koko on riippuvainen käyttäjän selainikkunan koosta.

Haetaan siis ”ul” – elementin leveys, jaetaan se asetetulla sarakkeiden määrällä ja vähennetään siitä marginaalien summa. Tällä laskutoimituksella saadaan oikea määrä elementtejä mahtumaan lomakealueelle vierekkäin. Korkeus on kovakoodattu, mutta siihen mahtuu isoinkin kenttä -elementin sisältö, ja korkeutta voi manuaalisesti muuttaa, niin kuin leveyttäkin. Koon muuttamisen yhteydessä hoitaa gridster itse laskutoimitukset, joten niitä ei tarvitse itse ohjelmoida. Parametri ”draggable” kertoo gridsterille että elementtejä voi raahata ympäriinsä ja sen parametri ”handle” kertoo mistä raahaus aktivoituu.

Selectable mahdollistaa elementtien valitsemisen hiiren klikkauksella, jolloin niiden taustaväri muuttuu ja ne saavat luokan ”ui-selected”, jota käytetään hyödyksi kun pitää kohdistaa työkalupalkin toimintoja valittuihin kenttiin. Klikkaamalla kenttää uudestaan voi valinnan poistaa.

4.2 Virheiden käsittely

Virhetilanteen sattuessa, HTTP -pyyntöjen yhteydessä, käsitellään tilanne niin, että käyttäjälle näytetään virheilmoitus, jossa kehotetaan ottamaan yhteyttä Fivaldin tukeen. Mitään tarkempaa tietoa virheestä ei päädy käyttäjälle asti. Tietokantapäässä virhe kirjoitetaan käyttölokiin, josta sen voi tarvittaessa kaivaa esille virheselvitystä varten.

4.3 Selainikkunan koon muutoksen käsittely

Selainikkunan koon muutos aiheuttaa gridsterin sekoamisen. Tätä varten on täytynyt tehdä JavaScript – funktio, jota kutsutaan aina kun huomataan ikkunan koon muuttuneen. Tähän on myös toteutettu viive, ettei funktiota suoriteta jatkuvasti selainikkunan koon muuttuessa, vaan vasta kun ikkuna on ollut paikallaan hetken aikaa muutoksen jälkeen. Funktio kerää talteen kaikki tiedot lisätyistä lomakekentistä. Tämän jälkeen tuhotaan elementti, jolle gridster sisältöineen päivineen on rekisteröity. Elementti luodaan uudestaan, gridster rekisteröidään uudestaan päivitetyin kokomäärittelyin ja kentät luodaan uudestaan talteen otetuista tiedoista.

4.4 Toiminnallisuudet

4.4.1 Uuden lomakemäärittelyn luonti

Käyttäjälle näytetään aluksi vain tallennettujen lomakkeiden valinta muokkausta varten, valintalistana, sekä painike uuden lomakkeen luomista varten. Kun käyttäjä painaa ”Luo uusi” – painiketta piilotetaan näkyvillä olevat valinnat ja näytetään kenttä, johon syötetään lomakemäärittelyn kuvaus, ”Tallenna” – painike, ”Peruuta” – painike, sekä lomakemäärittelyt. Lomakemäärittelyihin kuuluu ulkoasutyypin valinta, tehtävä – tai tapahtumatyypin valinta, kenttien lisäys, sekä sarakkeiden määrän valinta. Kenttien näyttö ja piilotus tapahtuu hyväksikäyttäen jQuery -kirjaston ”selector” – toimintoja, joilla voidaan valita elementtejä esimerkiksi luokan tai tyypin perusteella, sekä ”hide”- ja ”show” -funktioita.

```
54              $(".item-controls").hide();
55              $(".layout-controls").hide();
56              $(".layout-new").hide();
57              $(".layout-load").show();
```

Kuva 7. Elementtien piilotus ja näyttö

4.4.2 Tallennetun lomakemäärittelyn muokkaus

Valittaessa tallennettu lomakemäärittely valintalistasta, näytetään ja piilotetaan samat toiminnallisuudet kuin uudenkin luonnissa, mutta näytetään myös kenttäkohtaiset toiminnallisuudet, jos vain tallennetulla lomakkeella on lisättyjä kenttiä. Valinnan tehtyä tehdään myös Ajax – kutsu, jolle annetaan parametrina kutsuttava proseduuri muodossa ” paketin nimi”,”proseduurin nimi”, HTTP – metodi, parametrit, tässä tapauksessa tallennetun lomakemäärittelyn tunniste, eli ”id”, sekä vastauksen tyyppi. Kutsua seuraava funktio suoritetaan vasta kun kutsu on valmis, jonka jälkeen voidaan käsitellä vastaus.

```
168             params = "p_id=" + id;
169             promise = doAjax("tmar_layout_editor_web.get_layout_web", "GET", params, "json");
170             promise.done(function(data) {
171                 ...
```

Kuva 8. Ajax – kutsun muodostus

Kutsuttu proseduuri tarkistaa käyttäjän oikeudet ja istunnon voimassaolon, sekä kutsuu sisempää proseduuria. Sisemmässä proseduurissa haetaan tietokannasta oikea lomakemäärittely ”id” – kentän perusteella, sekä lomakemäärittelyyn liitetyt kentät. Tiedot kootaan JSON – muotoon ja palautetaan aina kyselyn tehneelle JavaScript – funktiolle asti. Tiedot sisältävät lomakemäärittelyn ulkoasutyypin, tehtävä – tai tapahtumatyypin, kuvauksen, sarakemäärän, sekä jokaisesta kentästä, tunnisteen, kuvauksen, tyypin, tyypin kuvauksen, rivin, sarakkeen, rivien määrän, sarakkeiden määrän, minimi – ja maksimi rivien määrän, minimi – ja maksimi sarakkeiden määrän, sekä tiedon onko kenttä syöttö- vai näyttökenttä. Tietojen perusteella asetetaan työkalupalkin lomakemäärittelyt oikeisiin arvoihin, sekä lisätään kenttätietojen perusteella kentät gridsteriin. Kenttien lisäys gridsteriin tehdään käymällä läpi palautunut JSON-oliolista, jossa kenttien tiedot sijaitsevat.

```
99      function addItem(data) {
100         var id, element, inputField;
101
102         id = "li" + glbIdSeq;
103         glbGridster.add_widget('<li id="' + id + '">' + data.itemContent + '</li>', data.cols, data.rows, data.col, data.row);
104
105         glbIdSeq += 1;
106         element = $("#" + id);
107
108         element.data("fieldId", data.fieldId);
109         element.data("fieldYt", data.fieldYt);
110         element.data("minCols", data.minCols);
111         element.data("maxCols", data.maxCols);
112         element.data("minRows", data.minRows);
113         element.data("maxRows", data.maxRows);
114         element.data("display", data.display);
115         element.data("itemContent", data.itemContent);
```

Kuva 9. Kentän lisäys ja tietojen talteenotto

Jokaisen lomakekentän lisäyksen kohdalla kutsutaan funktiota, jossa mm. lisätään kenttä gridsteriin funktiolla, joka ottaa parametreinä elementin rakenteen, leveyden sarakkeina, korkeuden riveinä, sijainnin sarakkeina ja sijainnin riveinä. Funktiossa ”AddItem(data)” myös asetetaan juuri lisätylle elementille tietoja jQuery – kirjaston ”data” – funktiolla, joka rekisteröi tiedot elementille (kuva 9). Tämä on oleellinen osa työkalun toimintaa, koska tietojen on oltava saatavilla jatkuvasti, niin liikuttamisen, koon muutoksen, kuin tallentamisenkin yhteydessä. Gridster järjestää lisätyt kentät automaattisesti oikeille paikoilleen annettujen parametrien perusteella. Kun kaikki kutsussa palautunut tieto on käsitelty, voi lomakemäärittelyjä ja kenttiä muuttaa haluamakseen, samanlailla kuin uudenkin lomakkeen luonnin yhteydessä.

4.4.3 Ulkoasutyypin valinta

Ulkoasutyypin valinta tehdään valintalistasta. Käytännössä tässä on kaksi vaihtoehtoa, joko tehtävätyyppinen ulkoasu tai tapahtumatyyppinen ulkoasu. Valintaa muuttaessa vaihtuu myös jompikumpi tehtävätyypin tai tapahtumatyypin valinnoista näkyville, riippuen kumman tyyppisen ulkoasun valitsi. Myös valintalista, jossa sijaitsee lomakkeelle lisättävissä olevat kentät, muuttuu koska kentät vaihtelevat tehtävä – ja tapahtumatyypin perusteella.

4.4.4 Tehtävä – tai tapahtumatyypin valinta

Käyttäjän valitessa tehtävä – tai tapahtumatyyppiä lomakemäärittelylle, tai ulkoasutyyppiä muuttaessa, jolloin asetetaan oletusarvo valituksi, suoritetaan aikaisemmin esitellyn tyyppinen Ajax – kutsu, tietysti eri parametrein, PL/SQL – proseduurille. Proseduurissa tarkistetaan käyttöoikeudet, istunnon voimassaolo, sekä kutsutaan sisempää proseduuria, joka hakee tietokannasta kenttien tunnisteet ja kuvaukset, parametreina annettujen ulkoasutyypin sekä tehtävä – tai tapahtumatyypin tunnisteiden perusteella. Tiedot palautetaan tässäkin tapauksessa JSON -muotoisena ja ne lisätään ”Lisää kenttä” -option listin sisällöksi.

4.4.5 Kentän lisäys

Kentän lisäys gridsteriin tehdään ”Lisää kenttä” – option listasta valitsemalla. Valinta laukaisee jälleen Ajax – kutsun PL/SQL – proseduuriin, jossa tarkistetaan käyttöoikeudet, istunnon voimassaolo, sekä kutsutaan sisempää proseduuria, joka puolestaan tunnuksen perusteella palauttaa tietokannasta valitun kentän tiedot, JSON -muodossa. Kentästä palautetaan samat tiedot kuin tallennetun lomakemäärittelyn latauksen yhteydessä. Kentän sijaintitiedot alustetaan ensimmäisen rivin ensimmäiseen sarakkeeseen, koska niitä ei tietenkään kentältä vielä löydy. Tämän jälkeen kenttä lisätään gridsteriin, samalla funktiolla kuin tallennetun lomakkeen kenttienkin yhteydessä. Jos lisätty kenttä oli lomakkeen ensimmäinen, tuodaan kenttäkohtaiset toiminnallisuudet näkyviin työkalupalkilla.

4.4.6 Kentän poisto

Kentän poisto tapahtuu työkalupalkin kentille tarkoitetusta ”Poista” – painikkeesta. Toiminto hakee kaikki valitut kentät ja poistaa ne gridsteristä. Gridster ei reagoi poistoon järjestelemällä kenttiä uudestaan, vaan poistettu kohta jää tyhjäksi.

4.4.7 Kentän valitseminen

Käyttäjä voi valita kentän hiiren vasemmalla painikkeella, jonka jälkeen siihen voi kohdistaa työkalupalkin kenttäkohtaisia toiminnallisuuksia. Kenttiä voi valita myös useita kerralla, jolloin toiminnat suoritetaan niille kaikille. Kenttien valinta ei kuitenkaan vaikuta mitenkään niiden liikuttamiseen, joka pitää tehdä yksi kerrallaan. Valinta on toteutettu jQuery UI -kirjaston ”selectable” – interaktiolla, joka on rekisteröity jokaiselle gridsterin listaelementille. Kun kenttä valitaan tai valinta poistetaan, tarkastetaan valittujen kenttien määrä. Jos yhtään kenttää ei ole valittuna asetetaan kenttäkohtaisille toiminnoille attribuutti ”disabled”, joka estää toimintojen käytön. Toiminnat asetetaan jälleen käyttöön kun yksi tai useampi kenttä on valittuna.

4.4.8 Kentän koon muuttaminen

Valitun tai valittujen kenttien kokoa voi muuttaa työkalupalkin ”+” ja ”-” – painikkeista, joita on sarakkeiden määrälle ja rivien määrälle omansa. Jokaisella kentällä on omat minimi – ja maksimi – sarakkeensa ja rivimäärät, joita suuremmaksi niitä ei voi kasvattaa. Nämä tiedot on tallennettu elementille lisäyksen yhteydessä ja ne luonnollisesti tarkastetaan kokoa muutettaessa. Jos maksimi -sarake tai -rivi on nolla, ei kentällä ole suuruusrajaa. Minimi -sarakkeen tai -rivin tapauksessa nolla on sama kuin yksi sarake tai rivi, koska kenttää ei voi kutistaa olemattomiin. Kenttä ei tietenkään voi myöskään kasvaa ulos lomakkeelta, joten sarakkeiden määrää kasvattaessa tarkistetaan, ettei se ylitä sarakkeiden kokonaismäärää, sijainnista huolimatta.

```
350     $("#btn-width-minus").click(function() {
351         $(".ui-selected").each(function() {
352             var data = glbGridster.serialize($(this));
353             if (data[0].size_x > $(this).data("minCols")) {
354                 glbGridster.resize_widget($(this), data[0].size_x - 1, data[0].size_y);
355             }
356         });
357     });
```

Kuva 10. Kentän leveyden pienennys

4.4.9 Kentän määrittäminen syöttö – tai näyttökentäksi

Jokaisella lomakkeelle lisätyllä kentällä on oikeassa yläkulmassa valintaruutu, josta voidaan muuttaa syöttö / näyttökenttä -statusta. Syöttökentiksi määritellyt kentät antavat oikeaan käyttöön generoidulla lomakkeella muokata sisältöään. Näyttökentiksi määritellyt puolestaan eivät.

4.4.10 Lomakemäärittelyn tallennus

Lomakemäärittelyn tallennus tapahtuu ”Tallenna” – painikkeesta. Lomakkeelle on täytynyt antaa kuvaus, muuten tallennus keskeytyy ja annetaan virheilmoitus. Tallennus tapahtuu HTTP POST -metodilla, joka toteutetaan Ajaxin avulla PL/SQL-proseduuriin, jossa tarkistetaan käyttöoikeudet, istunnon voimassaolo ja viedään tallennettavat tiedot sisempään proseduuriin, jossa suoritetaan tietojen lisäys tietokantaan.

Tietojen keräys tallennusta varten tehdään käyttämällä tilapäistä HTML form -elementtiä, johon lisätään JavaScriptin avulla input – elementtejä jokaiselle tallennettavalle tiedolle. Tiedot kerätään lomakemäärittelyistä, sekä jokaiselta kentältä, jQuery ”data” – funktiolla. Input – elementit nimetään niin, että eri kenttien tiedot voi erottaa toisistaan. Kun kaikki tieto on lisätty form – elementin sisään, se serialisoidaan, jolloin saadaan muodostettua avain – arvo pareja. Nämä annetaan parametreina HTTP- pyynnölle ja PL/SQL – proseduurissa parit tarkistetaan ja asetetaan insert-lauseisiin. Kun tallennus on valmis, näytetään käyttäjälle viesti tallennuksen onnistumisesta. Työkalu jää siihen tilaan missä se olikin ennen tallennusta, siltä varalta, että käyttäjä haluaa vielä jatkaa lomakkeen muokkaamista.

4.4.11 Lomakemäärittelyn poisto

Lomakemäärittelyn voi poistaa ”Poista” – painikkeesta. Käyttäjältä kysytään haluaako hän varmasti, että määrittely poistetaan. Jos hyväksytään, tehdään HTTP POST – metodin kutsu Ajaxilla PL/SQL – proseduuriin, jolle viedään lomakemäärittelyn tunniste. Proseduuri poistaa kaiken lomakemäärittelyyn liittyvän tiedon.

4.5 Tietokantarakenne

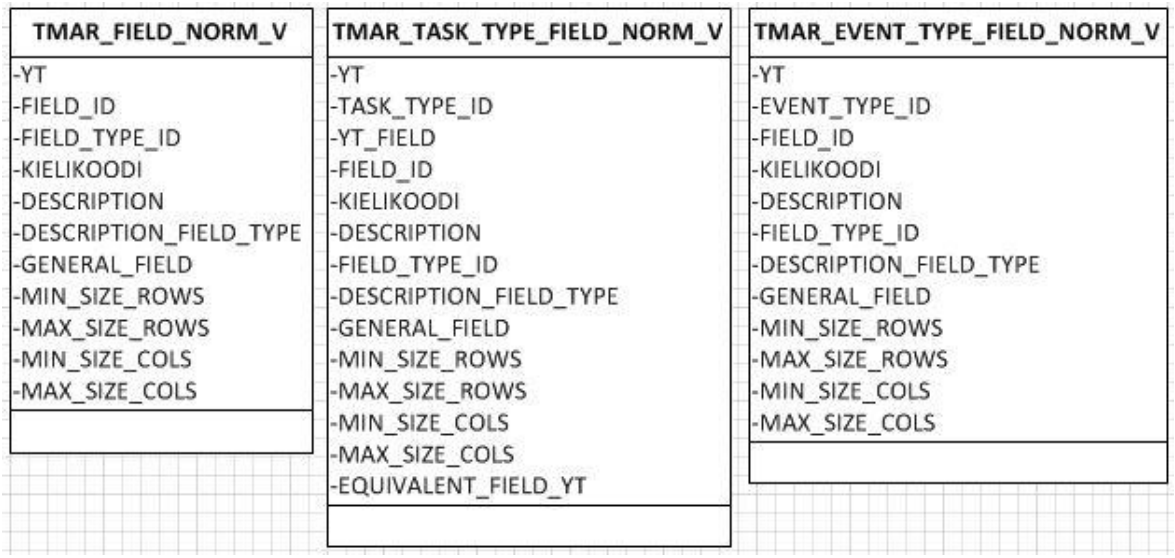
FWF – sovelluksen taustalla on laaja kantarakenne, josta vain osa koskee oleellisesti lomakkeiden määritystyökalua. Tässä luvussa esitellään siis vain näkymät, joista saadaan haettua kenttien tiedot, sekä tallennukseen ja tallennetun lomakkeen lataukseen liittyvät tietokantataulut.

4.5.1 Näkymät

Web – lomakkeiden määritystyökalussa käytetään kolmea eri näkymää. Näkymä on virtuaalinen taulu, jossa yhdistetään tietoa useammasta eri taulusta. Näkymä siis muodostetaan SQL – kyselyistä ja siitä myös haetaan tietoja kohdistamalla siihen kyselyitä.

Näkymästä ”TMAR_FIELD_NORM_V” löytyy kaikki sovelluksen kentät, yritystunnuksen ja kentän tunnisteen perusteella. Tämä on hyödyllinen silloin kun tarvitaan tiedot jostain tietystä kentästä, esimerkiksi kenttää lisättäessä lomakkeelle haetaan kyseisestä näkymästä tietoa, josta kenttätyypin perusteella muodostetaan oikean tyyppinen kenttä lomakkeelle vietäväksi.

Näkymät ”TMAR_TASK_TYPE_FIELD_V” ja
”TMAR_EVENT_TYPE_FIELD_V” puolestaan palauttavat kaikki tehtävä – tai tapahtumakohtaiset kentät. Näitä näkymiä käytetään kun halutaan kenttiä ulkoasutyypin perusteella. Työkalussa ulkoasutyypin ja tehtävä – tai tapahtumatyypin monivalintalistan muuttaminen suorittaa kyselyn, jossa tapauksesta riippuen haetaan jommasta kummasta näkymästä kentät työkaluun valittaviksi.



Kuva 11. Näkymien kentät

4.5.2 Tallennuksen ja generoinnin tietokantataulut

Lomakemäärittelyn tallennuksen ja latauksen yhteydessä päätaulu on ”TMAR_LAYOUT_T”. Taulu sarakkeina on yritystunnus, lomakkeen tunniste, lomakkeen tyyppi, referenssitunniste eli lomakkeen tyypin mukaisesti tehtävän tai tapahtuman tunnus, sekä lomakkeen sarakkeiden määrä. Tauluun liittyy yritystunnuksen ja lomakkeen tunnisteen perusteella taulu ”TMAR_LAYOUT_DESC_T”, jossa sijaitsee lomakkeen kuvaus eri kielillä.

Lomakemäärittelyn kentät tallennetaan tauluun ”TMAR_LAYOUT_PLACE_T”, jossa on yritys ja lomaketunnisteiden lisäksi kenttien sijainnit, syöttö/näyttökenttä-määritys, sekä kentän tunniste. Taulu yhdistyy kentän tunnisteen perusteella tauluun ”TMAR_LAYOUT_FIELD_T”, josta löytyy kentän tunnus ja yritys. Näillä tiedoilla voi generoitaessa lomaketta käyttöä varten hakea kaikki kentän tiedot ”TMAR_FIELD_NORM_V” -näkymästä (liite 4).

5 Pohdinta ja johtopäätökset

5.1 Projektin tulokset

Oy Finnvalli Finland Ab:n toimeksiannosta opinnäytetyönä suoritetun projektin tarkoituksena oli suunnitella ja toteuttaa uuden Fivaldi WorkFlow – sovelluksen osaksi Web – lomakkeiden määritystyökalu. Työkalun tuli toimia drag & drop - tyylisesti, noudattaa WYSIWYG – periaatetta, sekä toteuttaa vaatimusmäärityksessä mainitut käyttötarpeet ja muutkin vaatimukset. Nämä vaatimukset täytettiin ja ratkaisu siirtyi Finnvallin testausprosessiin, kuten oli projektin tavoitteena.

Web – lomakkeiden määritystyökalun toteutustavan selvityksen ja suunnittelun yhteydessä todettiin, että työkalun tarpeet olivat niin spesifejä ja alustariippuvaisia, että tässä tapauksessa ei voitu hyödyntää joitakin lukuisista lomakkeen luontiin tarkoitetuista valmiista työkaluista. Ratkaisussa päädyttiin hyödyntämään jQuery -pohjaista gridster.js – liitännäistä, joka mahdollistaa lomakekenttien dynaamisen lisäämisen, poistamisen, liikuttamisen ja koon muutokset. Nämä olivat selkeästi toteutuksen monimutkaisimmat osa-alueet ja liian laajat itse toteutettavaksi alusta loppuun, opinnäytetyölle varattujen resurssien puitteissa. Tämän pohjan päälle rakennettiin FWF – sovelluksen tarvitsemat toiminnallisuudet, sekä WYSIWYG -ominaisuudet.

Vaikkakin päädyttiin ratkaisuun, jossa valmis liitännäinen käsittelee HTML – elementtien drag & drop – järjestelemisen vaatimat laskennat, avattiin hieman teoriaa tämän algoritmiikan taustalla, jotta ymmärrettäisiin Web – lomakkeiden määritystyökalun toimintaa mahdollisimman hyvin. Toteutusta suunniteltaessa saatiin myös kattava käsitys markkinoilla olevista ratkaisuista Web – lomakkeiden luontia varten ja ymmärrettiin niiden rajoitteet ja mahdollisuudet niin yleisellä tasolla, kuin tämän opinnäytetyönkin kontekstissa.

Opinnäytetyön tuloksena saadun Web – lomakkeiden määritystyökalun avulla voi luoda lomakemäärityksiä kiinteistönhallinnan tarpeisiin ja näiden pohjalta generoida lomakkeita käyttöä varten FWF – sovelluksessa. Vaikkakin FWF – sovellusta ei vielä opinnäytetyötä tehdessä ollut julkaistu Finnvallin asiakkaiden käyttöön, määritystyökalua

pystyi jo hyödyntämään rinnakkaisena projektina toteutetun lomakkeiden generoinnin toiminnan testauksessa. Tämän opinnäytetyön tuloksien hyödynnettävyys kasvaa edelleen FWF – sovelluksen käyttöönoton myötä, jolloin Web – lomakkeiden määritystyökalu todennäköisesti tuo sovellukselle lisäarvoa.

5.2 Jatkokehitysideoita

Selkeä kehittämiskohde Web – lomakkeiden määritystyökalulle on Internet Explorer 9:ää aikaisempien Internet Explorer -versioiden tuki. Fivaldin muutkin sovellukset tukevat näitä selaimia, joten olisi tärkeää ylläpitää tätä standardia läpi Fivaldin. Tämän toteutus vaatii muutoksia gridsteriin, josta tukea näille selaimille ei löydy.

Tuloksen testaus rajattiin projektin ulkopuolelle, mutta QUnit – yksikkötestauksen implementointi voisi olla hyödyllistä niin testausprosessia varten, kuin ylläpidettävyydenkin kannalta. Tämä vaatii myös todennäköisesti rakenteellisia muutoksia nykyisen toteutuksen käyttöliittymän ohjelmistokoodiin. QUnit olisi ollut hyvä ottaa käyttöön jo kehitysvaiheessa mutta aikaisemman kokemuksen puuttuessa ja sitä myöten aikataulullisista syistä näin ei tehty.

Koska toteutus on jQuery – pohjainen on jatkossa myös huolehdittava sitä, että Web – lomakkeiden määritystyökalun toiminta ei hajoa jQuery -versiopäivitysten yhteydessä. Toteutuksessa käytettiin jQuery -versiota 1.8.3 ja jQuery UI – versiota 1.9.2. Kummastakin kirjastosta on julkaistu uudempia versioita, joten päivitys voisi olla ajankohtainen piakkoin, mutta se koskee samalla muitakin Fivaldi Web -sovelluksia.

5.3 Oman oppimisen arviointi

Merkittävintä ammatillista kehitystä on opinnäyteyön aikana tullut web – sovelluskehityksen osa-alueella ja erityisesti JavaScript- ja jQuery- taitojen saralla. Aikaisempaa Web – kehityskokemusta oli vain muutaman pienehkön projektin ajalta, joten koen lisääntyneen kokemuksen erittäin merkittävänä ja hyödyllisenä jatkoa ajatellen.

Opinnäytetyön aikana korostui myös malttamattomuuteni huolellisen suunnittelun saralla. Liian usein tulee vain aloitettua ohjelmointi vajavaisella suunnittelulla ja päädyttyä

ratkaisuihin pitkällisen ”trial and error” -prosessin kautta. Tässä tapauksessa lähdettiin liian hätäisesti toteuttamaan työkalua jQuery UI:n sortable – interaktion pohjalle ja yritettiin liian pitkään saada sillä toimivaa ratkaisua aikaiseksi. Huolellisemmalla suunnittelulla olisi varmasti todettu ratkaisun toimimattomuus aikaisemmin ja näin säästetty resursseja.

Opinnäytetyöprosessi on myös antanut harjoitusta projektinhallinnasta, jota en aikaisemmissa projekteissani ole suuremmin harjoittanut. Erilaisten dokumenttien ja raporttien kirjoittaminen on myös palauttanut mieleen hieman unohtumaan päässeet kirjallisen ilmaisun taidot.

Lähteet

Ducksboard. gridster.js. Luettavissa: http://gridster.net/. Luettu: 27.4.2013.

eruciform. JQuery UI Draggable Collision. Luettavissa: http://sourceforge.net/projects/jquidragcollide/. Luettu: 17.5.2013.

Firth, P. 2011. Collision detection for dummies. Luettavissa: http://www.wildbunny.co.uk/blog/2011/04/20/collision-detection-for-dummies/. Luettu: 16.5.2013.

Google Code. JQuery WYSIWYG Web Form builder. Luettavissa: https://code.google.com/p/jquery-form-builder-plugin/. Luettu: 15.5.2013.
.

Nielsen, J. 2005. R.I.P. WYSIWYG. Luettavissa: http://www.nngroup.com/articles/rip-wysiwyg/. Luettu: 15.5.2013.

Open Source Initiative. The MIT License. Luettavissa: http://opensource.org/licenses/MIT. Luettu: 27.4.2013.

Oy Finnvalli Finland Ab. Oy Finnvalli Finland Ab. Luettavissa: http://www.finnvalli.fi/Yritys. Luettu: 5.5.2013.

Tamada, S. 2011. Drag and Drop Template Management System with Jquery. Luettavissa: http://www.9lessons.info/2011/03/drag-and-drop-template-management-with.html. Luettu: 15.5.2013.

The jQuery Foundation. What is jQuery?. Luettavissa: http://jquery.com/. Luettu: 17.5.2013.

Wikipedia a. Application Programming Interface. Luettavissa: https://en.wikipedia.org/wiki/Application_programming_interface. Luettu: 17.5.2013.

Wikipedia b. Drag and drop. Luettavissa:
http://en.wikipedia.org/wiki/Drag_and_drop. Luettu: 17.5.2013.

Wikipedia c. Grid (graphic design). Luettavissa:
http://en.wikipedia.org/wiki/Grid_(graphic_design). Luettu: 17.5.2013.

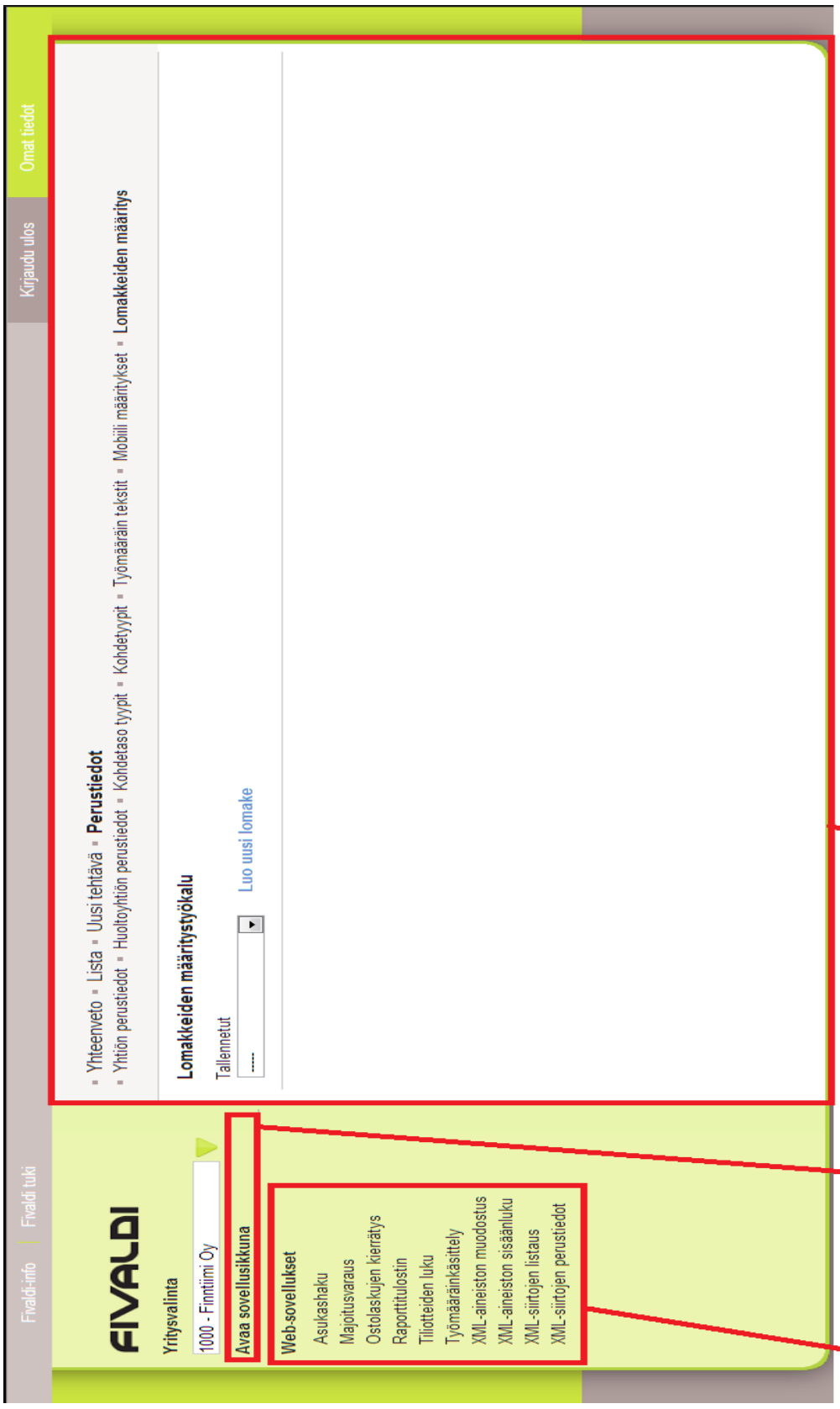
Wikipedia d. Software as a Service. Luettavissa:
http://fi.wikipedia.org/wiki/Software_as_a_Service. Luettu: 17.5.2013.

W3C. What is CSS?. Luettavissa: http://www.w3.org/Style/CSS/. Luettu: 17.5.2013.

w3schools.com. AJAX Introduction. Luettavissa:
http://www.w3schools.com/ajax/ajax_intro.asp. Luettu: 17.5.2013.

Liitteet

Liite 1. Fivaldi käyttöliittymä

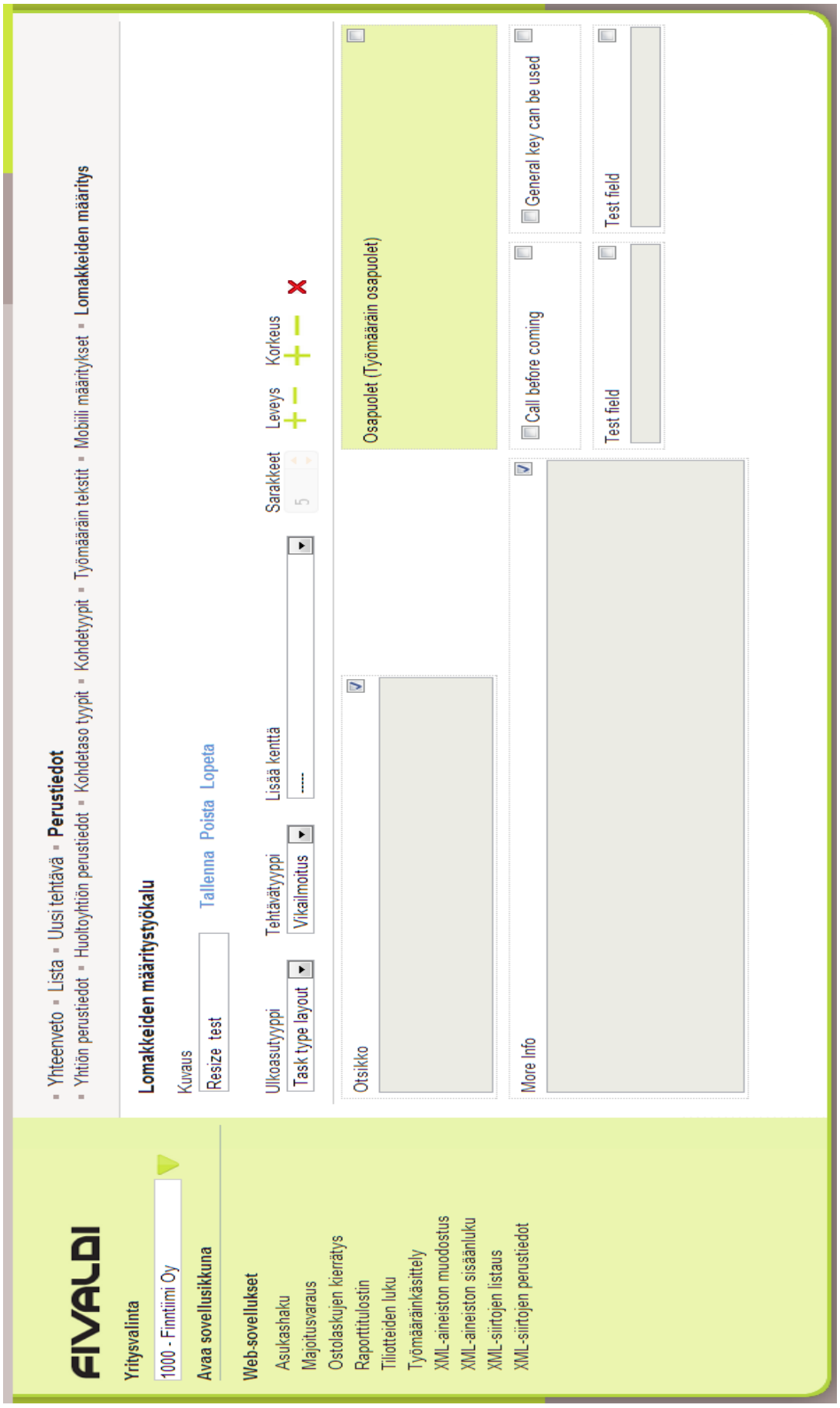


Liite 2. HTML – rakenne

```
<div id="tmar_palsta">
    <div id="top">
        <h1>Lomakkeiden määritystyökalu</h1>
        <div class="success message">
            <h3>Ok</h3>
            <p>Toiminto suoritettu.</p>
        </div>
        <div class="error message">
            <h3>Virhe!</h3>
            <p>Ota yhteys asiakastukeen.</p>
        </div>
        <div id="save-load-controls">
            <div class="top-section layout-load">
                <label for="s-load">Tallennetut</label></br>
                <select id="s-load">
                    <option selected="selected" value="default">-----</option>
                    <option value="2">Edit event sketch 2</option>
                    <option value="7">TEST</option>
                </select>
            </div>
            <div class="top-section layout-load">
                </br><button id="btn-new">Luo uusi lomake</button>
            </div>
            <div class="top-section layout-new">
                <label for="i-layout-descr">Kuvaus</label></br>
                <input type="text" id="i-layout-descr" />
            </div>
            <div class="top-section layout-new"></br>
                <button id="btn-save">Tallenna</button>
                <button id="btn-delete">Poista</button>
                <button id="btn-stop">Lopeta</button>
            </div>
            <div class="clearer"></div>
        </div>
```

```
35          <div id="layout-controls">
36              <div class="top-section layout-controls">
37                  <label for="s-layout">Ulkoasutyyppi</label></br>
38                  <select id="s-layout">
39                      <option value="TASK">Task type layout</option>
40                      <option value="EVENT">Event type layout</option>
41                  </select>
42              </div>
43              <div id="d-task" class="top-section layout-controls">
44                  <label for="s-task">Tehtävätyyppi</label></br>
45                  <select id="s-task">
46                      <option value="1" selected="selected">Vikailmoitus</option>
47                      <option value="2">Huoltotehtävä</option>
48                  </select>
49              </div>
50              <div id="d-event" class="top-section layout-controls">
51                  <label for="s-event">Tapahtumatyyppi</label></br>
52                  <select id="s-event">
53                      <option value="1" selected="selected">Huolto</option>
54                  </select>
55              </div>
56              <div id="d-field" class="top-section layout-controls">
57                  <label for="s-field">Lisää kenttä</label></br>
58                  <select id="s-field"></select>
59              </div>
60              <div id="d-cols" class="top-section layout-controls">
61                  <label for="i-cols">Sarakkeet</label></br>
62                  <input id="i-cols" name="value"  />
63              </div>
64              <div id="d-width" class="top-section item-controls">
65                  <label for="btn-width-plus">Leveys</label></br>
66                  <button id="btn-width-plus">+</button>
67                  <button id="btn-width-minus">-</button>
68              </div>
69              <div id="d-height" class="top-section item-controls">
70                  <label for="btn-height-plus">Korkeus</label></br>
71                  <button id="btn-height-plus">+</button>
72                  <button id="btn-height-minus">-</button>
73              </div>
74              <div class="top-section item-controls"></br>
75                  <button id="btn-delete-item">Poista</button>
76              </div>
77              <div class="clearer"></div>
78          </div>
79      </div>
80      <div class="gridster">
81          <ul id="containment">
82          </ul>
83      </div>
84      <form id="save-form" action="javascript:void(0);"></form>
85  </div>
```

Liite 3. Työkalun ulkoasu



Liite 4. Lomakemäärittelyn relaatiokaavio