Avoimeen lähdekoodiin perustuva kuvien taggausjärjestelmä

Pasi Lampinen

Tampereen yliopisto
Informaatiotieteiden yksikkö
Tietojenkäsittelyoppi
Pro gradu -tutkielma
Ohjaaja: Erkki Mäkinen
Kesäkuu 2014

Tampereen yliopisto
Informaatiotieteiden yksikkö
Tietojenkäsittelyoppi
Pasi Lampinen: Avoimeen lähdekoodiin perustuva kuvien taggausjärjestelmä
Pro gradu -tutkielma, 46 sivua, 1 liitesivu
Kesäkuu 2014

Tämä pro gradu -tutkielma käsittelee Sisustussuunitelmat.fi-palvelua esimerkkinä kuvien taggausjärjestelmistä, järjestelmän teknistä toteutusta ja sen yksityiskohtia. Tutkielma on yleinen katsaus järjestelmän toiminnallisuuksissa käytettyihin tekniikoihin, ja erityisesti se esittelee avoimen lähdekoodin hyödyntämistä projektissa ja arvioi sen vaikutuksia järjestelmän lopputuloksen kannalta.

Avainsanat ja -sanonnat: kuvien taggaus, avoin lähdekoodi, sisällönhallintajärjestelmä, Internet-markkinointi

Sisällys

1. JOHDANTO

Kuvat ovat olennainen osa Internetiä. Tiedonsiirtonopeuksien kasvaessa tiedostokokojen merkitys laskee. Siten kuvien määrä verkossa kasvaa ja kuvien laatu paranee. Nykyaikaiset web-sivustot voivatkin vahvasti tai jopa kokonaan rakentua kuvien ympärille. Kuvat ovat tehokas tapa välittää käyttäjille informaatiota ja sisältöä verrattuna pelkästään tekstimuotoiseen tiedon esitystapaan.

Kuvien kasvanut määrä ja merkitys web-sivustoilla on tuonut tekniikoita sisällön esittämiseen kuvien yhteydessä. Kuvissa olevalle sisällölle on hyvä saada tarkkoja ja vaihtoehtoisia esitystapoja. Kuvataggaus on yksi tapa tarkentaa kuvassa olevaa informaatiota. Kuvataggaus tarkoittaa tiettyjen kohtien merkkaamista kuvasta ja sisällön tuomista tälle kohdalle. Tässä voidaan tuoda esiin esimerkiksi henkilöitä, paikkoja, tuotteita ja muistiinpanoja.

Kuvien käyttö sisältönä tuo myös ongelmia semanttisen verkon toimivuudelle. Kuinka voidaan arvoida, mitä kuvat sisältävät? Kuvat koostuvat pikseleistä, joilla on tietty väriarvo ja jotka ovat tietyssä järjestyksessä, mutta pikseleistä muodostuvaa sisältöä on vaikea arvioida koneellisesti. Kuvien kuvaustekstit toimivat vaihtelevasti, eivätkä ne ole riittäviä esittämään kehityvässä web-ympäristössä tarvittavaa tietoa kuvassa olevasta informaatiosta.

Tämä tutkielma käsittelee Sisustussuunnittelu.fi-nimisen kuvataggauspalvelun suunnittelua ja toteutusta. Palvelu toteutettiin osaksi PlanMyRoom-sisustussuunnittelupalvelua. Palvelun taustaa käsitellään tarkemmin luvusssa 2.

Kuvataggauspalvelu Sisustusuunnittelu.fi:n toteutus ja toiminta voidaan jakaa kolmeen eri osa-alueeseen. Ensimmäisenä on palvelun front-end, joka on loppukäyttäjien käyttöliittymä palvelulle. Tämän avulla käyttäjä voi selata itseään kiinnostavia sisustuskuvia, joihin on merkitty tageilla tuotteita. Tämä osa-alue esitellään tutkielman luvussa 3. Toinen osa palvelusta on back-end ja sen tietokanta. Kyseinen osa-alue käsittää palvelun käyttäjähallinnan ja asetukset, jotka on rakennettu erillisen web-sisällönhallintajärjestelmän avulla. Sisällönhallintajärjestelmän tietokannan yhteyteen on toteutettu kuvataggaus-palvelun tietokanta. Tätä osaa palvelusta käsitellään tutkielman luvussa 4. Kolmas osa-alue on tuotteiden ja tagien hallintajärjestelmä. Tätä osaa käyttävät palvelun asiakkaat, ja siinä he lisäävät omiin havainnekuviinsa niissä esiintyviä tuotteita ja tagaavat ne kuviin oikeille paikoilleen. Luvussa 5 esitellään tämä osa-alue tarkemmin.

Kuvataggausjärjestelmä toteutettiin vahvasti ketterillä menetelmillä. Valmista konseptia ei ollut, vaan tärkeää oli saada nopeasti toimivaa toiminnallisuutta, joka mahdollisti testaamisen ja jatkokehityksen. Palvelun toteutuksen ja suunnittelun apuna käytettiin paljon avoimen lähdekoodin materiaalia, mikä mahdollisti alustan luomisen nopeasti. Tällä koodilla toteutettuja toiminnallisuuksia muokattiin ja kehitettiin vastaamaan testaamalla saatuja kehitystoiveita. Avoimella lähdekoodilla toteutettuja toiminnallisuuksia ei kuitenkaan oltu varta vasten koodattu

tätä palvelua silmällä pitäen. Tästä syystä koodissa joudutaan tekemään kompromisseja ja muokkaamaan tätä mahdollisimman sopivaksi kyseiselle järjestelmälle. Avointa lähdekoodia apuna käyttäen säästettiin aikaa, mutta kuitenkin valmiin toteutuksen laadulliset ominaisuudet kärsivät tästä. Tutkielman luvussa 6 arvoidaan avoimen lähdekoodin käytön vaikutusta toteutuksen laadullisiin ominaisuuksiin. Luku 7 sisältää yhteenvedon palvelun totetutuksesta ja avoimen lähdekoodin käytön vaikutuksista palvelun tietojärjestelmällisiin ominaisuuksiin sekä arvioita kuvataggauksen tulevaisuudesta ja Sisustussuunitelmat.fi-palvelun jatkokehitysmahdollisuuksista.

2. SISUSTUSSUUNNITELMAT.FI-PALVELU

2.1. PLANMYROOM-PALVELU

Sisustussuunnitelmat.fi suunniteltiin osaksi PlanMyRoom-palvelua. PlanMyRoom on vuonna 2009 perustettu palvelu, joka tarjoaa kuluttajille sisustussuunnittelua verkossa. Palvelun omisti alun perin PlanMyRoom Finland Oy. Tämän palvelun kautta käyttäjä pystyy tilaamaan ammattilaisen suunnitteleman sisutuksen itselleen. PlanMyRoomin tavoitteena on siirtää sisustussuunnittelun tilausprosessi kokonaan web-ympäristöön, jolloin varsinaisia kotikäyntejä ei tarvita.

PlanMyRoom-palvelussa asiakas valitsee aluksi haluamansa suunnittelijan. Valittavia suunnittelijoita on useita ja heidän tyylistään tarjotaan esimerkkikuvia ja kuvaustekstiä. Seuraavaksi käyttäjä syöttää kohdehuoneen tiedot, lataa kuvia huoneestaan ja antaa toiveita tulevasta suunitelmasta. Tämän jälkeen asiakas siirtyy maksamaan tilauksen. Lopputuotoksena käyttäjä saa sovitun toimitusajan kuluttua havainnekuvat huoneensa valmiista sisustusehdotelmasta. Sisustusehdotelma sisältää suunnittelijan kommentit ja listan kalusteista, joita suunnitelmassa on käytetty.

PlanMyRoom Finland Oy myytiin toukokuussa 2012 Alma Media Oyj:lle. PlanMyRoom ja Sisustussuunnitelmat.fi-palvelut siirtyivät Alma Media Oyj:lle.

Markkinointitoimisto Deeper on vuonna 2009 perustettu, Tampereella toimiva markkinointitoimisto, ja se oli mukana toteuttamassa PlanMyRoom-palvelua. Tämän tutkielman kirjoittaja työskenteli Markkinointitoimisto Deeperissä nimikeellä web-designer sekä PlanMyRoom- että Sisustussuunnitelmat.fi-palveluiden toteutuksen parissa.

2.2. SISUSTUSSUUNNITELMAT.FI

2.2.1. Tausta ja periaate

Tämän tutkielma käsittelee Sisustussuunnitelmat.fi-palvelua. Palvelu kehitettiin PlanMyRoom-palvelun rinnalle, mutta kuitenkin täysin itsenäiseksi kokonaisuudekseen. Tarkoituksena oli tarjota markkinointikanava PlanMyRoom-palvelulle sekä kalusteiden ja sisustustuotteiden valmistajille.

Palvelussa oli tarkoituksena käyttää PlanMyRoomin kautta tehtyjä havainnekuvia ja merkitä niihin eri valmistajien tuotteita tagien avulla (ks. Liite 1, kohta 7). Palvelu suunniteltiin myös tarjoamaan mahdollisuudet esitellä tuotteilla merkittyjä kuvia erilaisissa ympäristöissä kuten ulkopuolisilla web-sivuilla tai isoilla kosketusnäytöillä. Tuotteiden esittäminen oikeassa

kontekstissa, valmiissa sistussuunnitelmassa, tarjoaa kuluttajalle huomattavasti paremman mielikuvan tuotteesta ja helpottaa ostopäätöksen tekoa [Park *et al.*, 2005].

Sisustussuunnitelmat.fi:n periaatteena on linkittää tuotteita web-sivulla oleviin kuviin. Tuotteita voidaan merkata kuviin tagien avulla. Tagit ovat yleensä merkitty ympyröillä, jotka ovat kuvan päällä, niihin liittyvän tuotteen läheisyydessä. Aktivoimalla tageja käyttäjä saa tuotteesta lisätietoa. Aktivointi tapahtuu web-käyttöliittymän kautta joko kohdistamalla tai klikkaamalla kursorilla tuotemerkintäympyrää. Lisätieto tarjotaan käyttäjälle apuikkunoiden, eli tooltipien (ks. Liite 1, kohta 8) avulla.

Tuotemerkintöjen apuikkunat koostuvat tekstistä, linkeistä ja kuvista. Apuikkunassa on tuotteen nimi ja kuvausteksti. Kuvausteksti voi sisältää sanallisen kuvauksen tuotteesta ja lisätietoja kuten materiaalin, mitat ja värivaihtoehdot. Apuikkunassa voi olla kuva myös tuotteesta, josta käyttäjä voi nähdä tuotteen yksityiskohdat helpommin. Tagista aukeavassa tuotemerkinnässä on myös linkki tuotteen valmistajan tai valmistajan verkkokaupan sivuille, josta tuotteesta voi saada tarkempaa tietoa ja mahdollisesti ostaa tuote.

2.2.2. Kehitysvaiheet ja tekniset osa-alueet

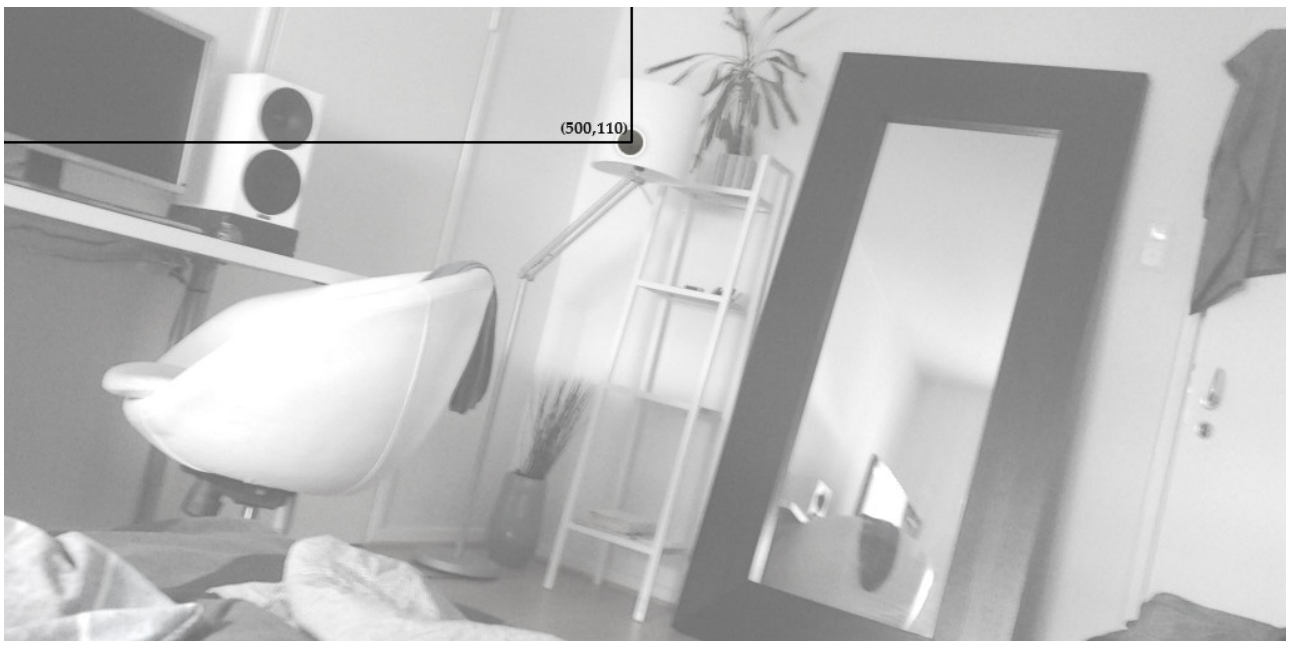
Sisustussuunnitelmat.fi:n kehitys voidaan katsoa noudattaneen ketterän ohjelmistokehityksen menetelmiä. Täysin valmista suunnitelmaa ei ollut, ja palvelun hyödyllisyyttä ja kiinnostavuutta oli vaikea arvioida, joten palvelua kehitettiin pienissä, toiminnallisuutta sisältävissä palasissa. Tällä tavalla pystyttiin vähentämään tuotannollisia riskejä ja suunnittelemaan palvelua asiakkaan kommenttien perusteella. Mikäli palvelun vaatimuksia haluttiin muuttaa nopeasti, oli se ketterän ohjelmistokehityksen avulla mahdollista. [Cockburn and Highsmith, 2001].

Ensimmäinen versio koostui staattisesta HTML-sivusta, jolla oli suunnitelmakuva ja yhteistyökumppaneiden logoja. Kuvaan oli muutamalla tagilla merkattu tuotteita. Tagit ja tooltipit toteutettiin staattisen HTML-koodin ja CSS-määrittelyiden avulla. Tooltipien näyttämiseen (ja piilottamiseen) tagien aktivoinnin seurauksena käytettiin JavaScript-koodia, joka muutti selaimessa näytettävän HTML-dokumentin rakennetta. Tällä versiolla pystyttiin testaamaan idean ja käyttöliittymän toimivuutta.

Toisessa versiossa kuviin liitettävien tagien ja tuotteiden tiedot tallennettiin tiedostoihin. Tiedostoista tiedot tuontiin käytettiin JavaScript- ja PHP-kieliä. PHP:lla kuvaan tuotava informaatio muutettiin JSON-muotoiseksi (ks. Liite 1, kohta 6). JSON-muotoinen tieto haettiin JavaScriptilla ja ja muotoiltiin HTML-kieleksi sisustuskuvan yhteyteen. JavaScriptin lisäkirjasto jQuery (ks. Liite 1, kohta 2) tarjosi valmiita funktioita tiedon hakuun ja käsittelyyn.

Seuraavassa versiossa tuotteiden taggausta varten suunniteltiin käyttöliittymä. Tässä vaiheessa valmiita, tuotteilla tagitettuja suunnitelmakuvia oli viisi ja käyttöliittymää tagien aktivoimiseen oli hiottu paremmaksi. Uusien suunnitelmakuvien tekeminen ja tuotteiden

merkitseminen kuviin oli kuitenkin osoittautunut hitaaksi. Tämän tekeminen vaati pikselein määritellyt koordinaatit kullekin tagille. Koordinaateilla tarkoitetaan tagin etäisyyttä sisustuskuvan vasemmasta reunasta ja yläreunasta pikseleinä ilmoitettuna (ks. kuva 1). Tähän asti näiden määrittäminen täytyi tehdä manuaalisesti, erillisen kuvankäsittelyohjelman avulla. Järjestelmään täytyi kehittää käyttöliittymä, jonka avulla tuotteiden merkintä kuviin, eli koordinaattien etsiminen tuotetageille, onnistui helpommin. Tuotteiden lisäys ja merkitseminen kuviin oli helpointa tehdä suoraan suunnitelmakuvaa lisätessä klikkaamalla suunnitelmakuvassa esiintyville tuotteille paikat. Tämän toteutuksen pohjana käytettiin Photo Tag -pluginia. Tämä vastaa tageihin liittyvistä käyttöliittymätoiminnoista. Photo Tag -pluginiin pohjautuvaa käyttöliittymää ja toiminnallisuutta esitellään luvussa 3.



Kuva 1: Tuotetagien koordinaatit pikseleinä.

Palvelulle oli tässä vaiheessa kehitettävä navigaatio ja tehtävä siitä web-sivusto, jotta suunnitelmakuvien välillä pystyi liikkumaan. Paras tapa on käyttää web-pohjaista sisällönhallintajärjestelmää [McKeever, 2003]. Sisällönhallintajärjestelmät helpottavat tiedon käsittelyä ja esittämistä web-ympäristössä. Ne sisältävät valmiita työkaluja sisällön luontiin ja kattavat käyttäjähallintaominaisuudet. Sisällönhallintajärjestelmäksi valittiin CMS Made Simple (ks. Liite 1, kohta 1), kehittäjien osaamisen perusteella. Sisällönhallintajärjestelmät rakentuvat poikkeuksetta tietokannan varaan, joten tässä vaiheessa oli luonnollista tuoda tuotekuvien ja tagien

tiedot myös tietokantaan. CMS Made Simplen hallintajärjestelmää ja tietokantaa käsitellään luvussa 4.

Seuraavia palvelun vaiheita ohjasi liiketoiminta- ja ansaintamallien suunnittelu. Kuka vastaisi palvelun ylläpidosta ja sisällön tuottamisesta? Palvelun kehityksessä päädyttiin jakamaan käyttäjien roolit seuraavasti: pääylläpitäjät, asiakkaat ja loppukäyttäjät. Asiakkaita varten luotiin hallintaosio tuotteiden syöttöön ja niiden merkitsemiseen tuotekuviin. Tämä vaati samalla yksinkertaisen asiakkuudenhallintajärjestelmän luontia. Asiakastoimintojen kehitys esitellään luvussa 5.

3. LOPPUKÄYTTÄJIEN KÄYTTÖLIITTYMÄ

Loppukäyttäjien käyttöliittymä koostui kuvista, joihin oli upotettu informaatiota tageina. Informaatiota näytettiin käyttäjälle tageja aktivoimalla. Tämän tyyppinen käyttäjän selaimessa tapahtuva HTML-rakenteen muokkaaminen lennosta toteutetaan yleensä JavaScript-koodilla. Tässä projektissa päädyttiin lopulta valitsemaan valmis jQuery-plugin käyttöliittymän rungoksi.

3.1. PLUGINIT

Käyttöliittymän runkona käytetty plugin eli lisäosa oli nimeltään jQuery Photo Tag. Tämän lähdekoodilla on MIT- ja GPL-lisenssit, joten sitä voi MIT-lisenssin puitteissa vapaasti käyttää myös kaupallisiin suljetun lähdekoodin ohjelmistoihin [MIT] [GPL]. Kyseinen plugin on alunperin tarkoitettu kuvissa olevien henkilöiden merkitsemiseen. Tätä pluginia muokkaamalla ja laajentamalla on luotu Sisustussuunnitelmat.fi:n käyttöliittymä.

Photo Tag -plugin käyttää apunaan jQuery ja jQueryUI JavaScript-kirjastoja. Se on rakennettu hyvin erilaisten asetusmuuttujien avulla, mikä mahdollistaa sen monipuolisen ja joustavan käytön. Tämä tarkoittaa, että lisäosalla on tietyt oletusmuuttujat, jotka vaikuttavat sen toimintaan. Lisäosa käyttää parametrinaan oliota, joilla voidaan yliajaa halutut oletusmuuttujat omilla asetuksilla pluginin latauksen yhteydessä. Esimerkiksi muutokset elementtien koossa, käyttäjälle näkyvissä teksteissä, pluginin kutsumissa URL-osoitteissa tai vaikka CSS-luokissa voidaan hoitaa muuttamalla asetuksia. Tämä mahdollistaa lisäosan helpomman kehittämisen ja siirtämisen eri ympäristöihin.

Yleisesti pluginilla tarkoitetaan ohjelmistoissa lisäosaa tai liitännäistä, jolla voidaan lisätä toiminnallisuutta. JavaScript on prototyyppipohjainen ohjelmointikieli, eli sen oliomalli perustuu prototyyppeihin [Javascript]. jQuery-kirjastoa käytettäessä pluginit ovat tapa lisätä toiminnallisuutta, prototyyppinä käytetään jQuery-oliota. jQuery-plugineita käytetään yleensä määrittelemään ja tekemään joku tietty toiminnallinen kokonaisuus. Pluginin toiminnallisuutta ja toimintalogiikkaa voi monipuolistaa parametrina annettavilla asetuksilla. Esimerkissä 1 määritellään jQuery-plugin, joka muuttaa kohde-elementin kokoa. Pluginille annetaan parametrina olio, joka sisältää asetukset, joilla muutetaan pluginin toimintaa. Plugin sisältää itsessään oletusasetukset, joita käytetään arvoille, joita pluginin saama parametriolio ei sisällä. Plugin voi myös sisältää useita funktioita. [jQuery Plugins]

Esimerkki 1. jQuery-pluginin toiminta

```
(function( $ ) {
    $.fn.muuta = function( asetukset ) {
          var asetusolio = $.extend({
              // oletusasetukset
              yksikko: "px",
              leveys:  100,
              korkeus: 100
          }, asetukset );

          //muutetaan pluginilla kutsutun elementin leveyttä ja korkeutta
          this.width(asetusolio.leveys+asetusolio.yksikko)
          .height(asetusolio.korkeus+asetusolio.yksikko);

          return this; // saadaan jQuerylle ominainen ketjutettavuus
    };
}( jQuery ));

/*
    "elem" on jQuery-olio, joka sisältää HTML-dokumentin kaikki
    "muutaminua"-luokkämääreelliset elementit
*/
var elem = $(".muutaminua");
/*
    parametrina annettava olio sisältää asetusmuuttujat, jotka laajentavat
    lisäosan omia asetusmuuttujien arvoja
*/
var omatAsetukset = {
    yksikko: "%",
    korkeus: 50
}
// kutsutaan jQuery-pluginia
elementti.muuta( omatAsetukset );
```

3.2. TOIMINTA

Sisustussuunnitelmat.fi:n tag-käyttöliittymästä vastaa Photo Tag -plugin. Sivun lähdekoodissa on kuvia, joille tagit ladataan. Näillä kuvilla täytyy olla kaikilla sama tietty luokkamääre (class), esimerkiksi "phototag". Näillä kuvilla täytyy olla myös yksilöiviä luokkatietoja, joiden perusteella haetaan oikeat tagit oikealle kuvalle. Esimerkistä 2 käy ilmi millaiset luokkamääreet kuvalla täytyy olla, jotta Photo Tag -plugin osaisi käsitellä sitä oikein. Esimerkin ensimmäisellä kuvaelementillä on luokat "phototag" ja "imageid_6". Ensimmäinen luokkamääre ”phototag” toimii pluginille tunnistavana määreenä, jonka avulla pluginia kutsutaan kyseiselle kuvalle. Toinen luokkamääre ”imageid_6” yksilöi kuvan, jotta plugin osaa hakea tietokannasta juuri tälle kuvalle merkityt tuotetagit (ks. esimerkki 3).

HTML-kielessä elementtejä yksilöivänä määreenä toimii yleensä id-määre. Pluginin toteutuksessa tätä ei kuitenkaan valittu yksilöiväksi määreeksi kuvatageja varten. Käyttämällä luokkamääretta id-määreen asemesta pystytään pluginia käyttämään joustavammin ja monipuolisemmin eri ympäristöissä ja tilanteissa. Joustavuus lisääntyy, mikäli pluginia käytetään valmiilla sivustoilla, kuvissa, joilta löytyy jo id-määreet. Elementeillä on mahdollista olla vain yksi id-määre. Mikäli näitä olemassa olevia id-määreitä käytetään sivuston omiin tarkoituksiin, luokkamääreiden käyttö yksilöivänä määreenä ei aiheuta ristiriitoja sivuston oman toiminnan kanssa. Luokkamääreiden käyttö mahdollistaa myös samalle kuvalle useiden eri tagikokonaisuuksien lataamisen. Tämä saattaisi olla tarpeellista, mikäli samaan kuvaan haluttaisiin tuoda usean eri asiakkaan tageja kerrallaan. Tämä olisi mahdollista antamalla kuvalle luokkamääreenä useita eri yksilöiviä ”imageid_”-luokkamääreitä. Tälle varsinaista toiminnallisuutta ei toteutettu, mutta tämän mahdollinen tarve tulevaisuudessa huomioitiin käyttämällä luokkamäärettä yksilöivänä määreenä. Luokkamääreiden käyttö valitsimena elementeille on kuitenkin huomattavasti hitaampaa kuin id-määreiden käyttö. Tätä käsitellään tarkemmin kohdassa 6.4.

Esimerkki 2. Tagattujen kuvien html-esitystavasta

```
<ul>
<li id="huone1" class="huone" >
    <img src="images/suunnitelmat/suunnitelma1.jpg" alt="" class="phototag
imageid_6"/>
</li>
<li id="huone2" class="huone" >
    <img src="images/suunnitelmat/suunnitelma2.jpg" alt="" class="phototag
imageid_7"/>
```

```
</li>
</ul>
```

Esimerkki 3. Yksinkertaistettu esimerkki Photo Tag -pluginin toimintaperiaatteesta tagien haussa JSON-muodossa

```
//photoTag-plugin extends jQuery
(function($) {
    $.fn.photoTag = function( options ){
    // oletus-optionsit
    var defaultOptions = { ... }

    // parametrien haku kuvan class-määreistä (ainakin kuvan id)
    function haeParametrit( kuva ){ ... }

    // rekisteröi eventit elementille, click, mouseover, mouseout
    function rekisteroiEventitElementille( elementti ){ ... }

    // luodaan tagit haetun JSON-datan perusteella
    function luoTagJSONista( JSONTag )    {
              // luodaan tagille div
              var tag =  $("<div class='tag'></div>");
              // lisätään diviin tagin teksti (lisättäisiin myös muuta dataa)
              tag.append(  JSONTag.text );
              // rekisteröidään eventit
               rekisteroiEventitElementille( tag );
              // palautetaan tag
              return tag;
    }

    var lisaaTagitKuvaan( JSON ) {
              // jokaiselle JSONista löytyvälle Tagille suoritetaan funktio
```

```
            $.each(JSON.Tags,function(){
                var tag = luoTagJSONista(this);
                // lisätään tag kuvaan
                $('#'+options.kuvanIdPrefix+JSON.id).append(tag)
            }
    }

    // jokaiselle elementille seuraava funktio
    this.each(function(){
            // haetaan parametrit
            var parametrit = haeParametrit( $(this) );
            // haetaan tagit
            $.getJSON( options.tagienHakuUrl, parametrit,
                // Haun jälkeen
                function( JSONtulos ){
                    lisaaTagitKuvaan( JSONtulos )
                }
             );
    });
    return this;
    };
})(jQuery);


// Sivun latauduttua kutsutaan seuraavaa funktiota
$(document).ready(function(){
    // Kutsutaan photoTag-pluginia elementeillä (kuvilla), joilla phototag-
luokka
    $('.photoTag').photoTag({
            // omat optionsit
            tagienHakuUrl: 'phototag/haeTagit.php',
            kuvanIdPrefix: 'imageid_'
    });
}
```

Seuraavaksi plugin tekee Ajax-kutsun osoitteeseen, joka sille on asetuksissa tagien hakua varten määritelty, parametrinaan kuvan id. Kyseisen osoitteen tulee palauttaa kannasta tuotteiden tiedot JSON-muodossa pluginille esimerkin 4 mukaisesti. Nämä sisältävät tiedon paikasta kuvassa, tuotteen nimen, tiedot, linkin ja mahdollisesti tuotteen kuvatiedoston nimen. Plugin tekee jokaisesta tuotteesta erikseen sekä tagin kuvaan ja listaelementin kuvan alla olevaan tuotelistaukseen. Näiden muotoilu hoidetaan osittain suoraan JavaSriptilla ja loput tulevat CSS-tiedostosta.

Esimerkki 4. JSON-muotoinen tieto tagien haussa

```
{"Image":
    {"id":6,"Tags":[
        {"id":64,"text":"Verhot","description":"Kuviollinen
verhokangas","category":"Tekstiilit","left":595,"top":76,"url":"http:\/\/www.
verhot.fi","image":"verhot.jpg"},
        {"id":65,"text":"Kattovalaisin","description":"Metallinen
kattovalaisin","category":"Valaisimet","left":406,"top":72,"url":"http:\/\/ww
w.valaisin.fi","image":"valaisin.jpg"}
    ]}
}
```

3.3. TAPAHTUMAT

Lisättäessä eri elementtejä kuvaan Photo Tag -pluginilla näille luodaan myös tapahtumat eli eventit. Plugin sisältää funktiot, joissa määritellään eri käyttöliittymälliset tapahtumat kullekin elementille. Näitä tapahtumia ovat mm. *klikkaus* (click), *kursorin päälle vienti klikkaamatta* (mouseover ja mouseout tai hover ja blur) ja *raahaus* (draggable). Kyseiset funktiot määrittelevät tarkasti, mitä kunkin tapahtuman laukaistaessa milloinkin elementille tehdään. Nämä funktiot ovat hyvin muokattavissa ja säädettävissä. Esimerkiksi tagin klikkaus aiheuttaa tooltipin aukeamisen ja jäämisen päälle, kunnes sitä klikataan uudestaan (ks. esimerkki 5). Kuitenkin pelkkä kursorin vienti tagin päälle aiheuttaa vain tooltipin aukeamisen, mutta tooltip ei jää päälle, vaan katoaa kursorin mentyä pois tagin alueelta.

Esimerkki 5. Tapahtumiin pohjautuva käyttöliittymän ohjelmointi

```
// luodaan tag-elementille klikkaus-tapahtumasta seuraava toiminnallisuus
tag.click( function() {
    // vaihdetaan luokkaa, joka määrittää onko aktiivinen vai suljettu
    $(this).toggleClass("active");

    // mikäli tag muuttui aktiiviseksi
   if ( $(this).hasClass("active") ) {
         //avataan tuoteselite, jolle oma funktionsa
         openToolTip( $(this) )
         // tallennetaan avausten statistiikkaa, trackEvent-funktiolla
         trackEvent( 'click', $(this).attr('id') )
    //mikäli tag suljetaan
    } else
         // sulkemista vastaava funktio
         closeToolTip ( $(this) )
});
```

3.4. KUVANAVIGAATIO

Sivun latauksen yhteydessä luodaan kuvanavigaatio kuvista, joille Photo Tag -plugin luo tagit. Muut kuvat kuin ensimmäinen kuva piilotetaan ja navigaatio mahdollistaa selailun näiden kuvien välillä. JavaScriptilla jQueryn avulla tehdään näkyvillä olevan sivuille nuolet ja yläpuolelle pienoiskuvat, joilla navigointi eri suuntiin onnistuu. Näiden pienoiskuvien täytyy olla HTML-sivun koodissa jo valmiiksi, mutta JavaScriptilla näihin tehdään toiminnallisuus itse navigointiin (ks. esimerkki 6).

Esimerkki 6. Kuvanavigaation HTML-esitysmuoto

```
<div id="kuvaNav">

    <img src="images/suunnitelmat/thumb_suunnitelma1.jpg" id="thumb_huone1"
alt="" />

    <img src="images/suunnitelmat/thumb_suunnitelma2.jpg" id="thumb_huone2"
alt="" />

</div>
```

4. TAUSTAJÄRJESTELMÄ JA TIETOKANTA

Web-sivustot ovat kehittyneet pisteeseen, jossa tarvitaan apuvälineitä tiedon tehokkaaseen hallintaan ja käsittelyyn. Kaikilla laajoilla web-palveluilla ja -sivustoilla on yhtenäisiä vaatimuksia käyttölogiikan ja tietojärjestelmän suhteen. Sisällön täytyy olla useiden henkilöiden päivitettävissä; yksi henkilö ei useinkaan pysty vastaamaan koko palvelun sisällöstä [McKeever, 2003]. Sisältö voi tulla useista eri lähteistä ja yhdistyä palvelussa erilaisiin formaatteihin ja kieliversioihin. Web-sivustot ovat kasvaneet niin laajoiksi kokonaisuuksiksi, että ylläpidettävyyden ja kehittämisen kannalta on järkevää erottaa palvelun sisältö- ja esityskerros toisistaan [Offutt, 2002].

Web-sisällönhallintajärjestelmät ovat järjestelmiä, jotka tarjoavat edellä mainittuja vaatimuksia vastaamaan rakennetun alustan. Yli 30% kaikista web-sivustoista käyttää alustanaan jotain sisällönhallintaan tarkoitettua järjestelmää [W3Techs].

4.1. CMS MADE SIMPLE

Sisustussuunitelmat.fi-palvelussa on käytetty sisällönhallintajärjestelmä CMS Made Simplea. Se on avoimeen lähdekoodiin perustuva web-sisällönhallintajärjestelmä. Sen käyttö on ilmaista ja sen avulla tehty lopputulos voi olla avoimella tai suljetulla lisenssillä. Sisällönhallintajärjestelmän käyttö palvelun alustana tarkoittaa käytännössä sitä, että sivut ja sisältö ovat hallittavissa järjestelmän kautta ja sisällön lisäys ja muokkaus on vaivatonta. Sisällön tuottaminen verkkopalveluihin onkin siirtynyt yhä enemmän varsinaisille kyseisen palvelun aihealueen ja liiketoiminnan osaajille, eikä erillistä web-ylläpitäjää tarvita sisällön tuomisessa verkkoon [McKeever, 2003].

Suurten web-palveluiden tieto ja sisältö on jakautunut usealle tasolle. Osa sisällöstä voi olla tiheään päivittyvää, uutisten kaltaista sisältöä, ja osa taas harvoin päivittyvää, pysyvää sisältöä. Sisältö voi olla myös vain nähtävissä osalle käyttäjistä, esimerkiksi kirjautumisen takana. Sisällöntuottajina voivat toimia myös käyttäjät. Sisällönhallintajärjestelmistä löytyykin usein valmiina ominaisuuksia, kuten käyttäjäroolien hallinta, tukemaan näitä vaatimuksia [McKeever, 2003].

CMS Made Simple on PHP-kielellä rakennettu web-sisällönhallintajärjestelmä. Siihen on mahdollista toteuttaa lisätoiminnallisuuksia toteuttamalla omia moduuleita ja plugineita. Järjestelmä on alun perin suunniteltu modulaariseksi ja laajennettavaksi, ja uutta koodia onkin mahdollista tuoda järjestelmään muokkaamatta järjestelmän varsinaisia ydintiedostoja. Tämän avulla järjestelmä on mahdollista päivittää ja ylläpitäminen helpottuu.

Järjestelmässä on sisällön hallintaa varten oma hallintaosionsa. Hallintaosiota käytetään web-käyttöliittymän kautta ja sille on oma URL-osoite. Hallintaa kirjaudutaan käyttäjätunnuskohtaisella

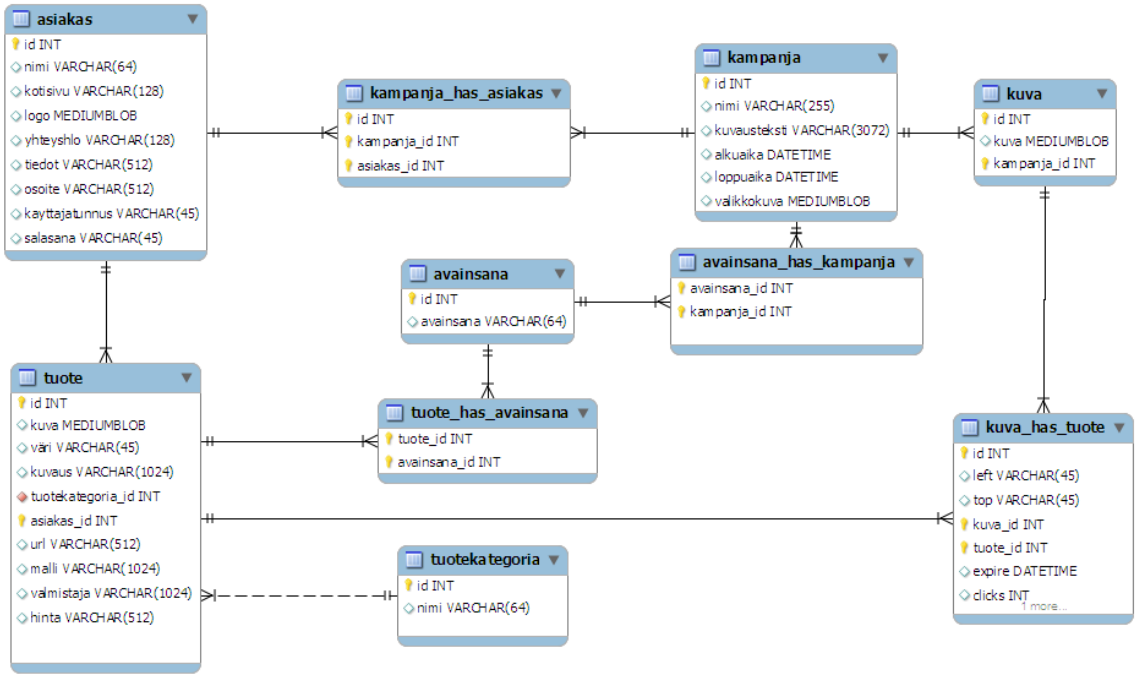
käyttäjätunnuksella ja salasanalla. Hallintaosio on helppokäyttöinen, eikä sisällön päivitys edellytä esimerkiksi HTML-kielen tuntemista.

CMS Made Simplen hallintaosio on toiminnallisuudeltaan hyvin laaja. Siihen on pyritty sisällyttämään iso osa toiminnoista, jotka kuuluvat web-sivustojen ja -palveluiden rakentamiseen ja ylläpitoon. Sivustojen rakennus- ja kehitysvaiheessa hallinnan kautta luodaan sivustolle sivuhierarkia, joka toimii yleensä myös sivuston navigointina. Sivuille rakennetaan hallinnan kautta sisällön esityskerros, eli sivupohjat, jotka määräävät sivujen HTML-rakenteen. Sivupohjat, eli templatet, rakennetaan yhdistämällä HTML-kieltä ja Smarty-mallinnejärjestelmää, jota CMS Made Simple tukee. Smarty on web-sovelluksissa käytetty järjestelmä, jolla erotetaan toimintalogiikka ja esityslogiikka toisistaan [Smarty]. Sivuston ulkoasun määritteleviä CSS-tyylitiedostoja voidaan myös hallita hallintajärjestelmästä käsin, ja ne pystytään yhdistämään haluttuihin sivupohjiin.

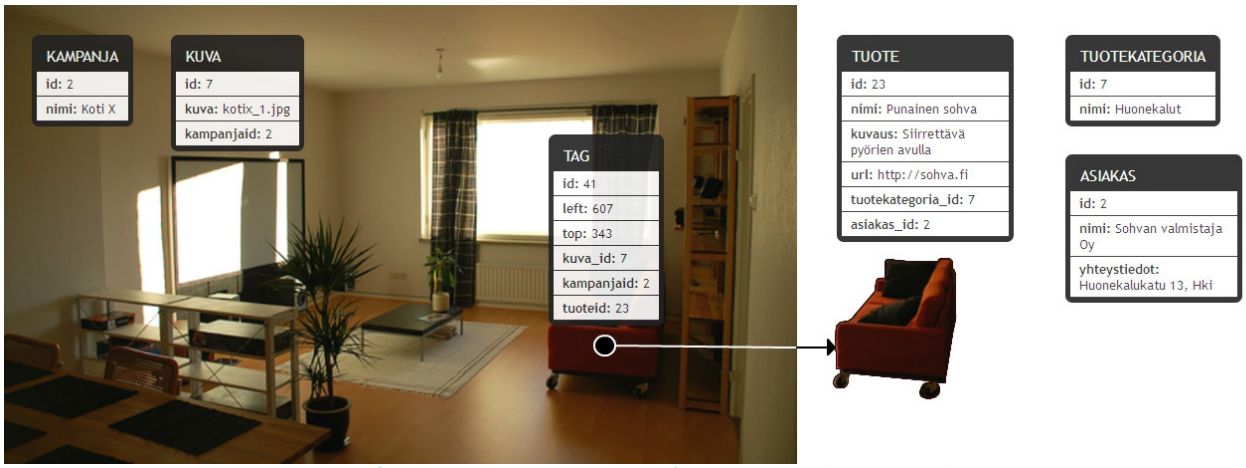
Sisällönhallintajärjestelmän web-käyttöliittymän kautta tehtävät toiminnot on rajattu eri käyttäjäroolien mukaan. Sovelluksen kehitystoimintoihin pääsevät käsiksi vain käyttäjät, jotka kuuluvat nämä toiminnot mahdollistavaan käyttäjäryhmään. Sisällön päivittäjiä varten tarkoitettu käyttöliittymä on suppeampi, eikä heillä ole pääsyä sivuston esityskerrosta tai toiminnallisuutta muuttaviin toiminnallisuuksiin. Tämä helpottaa heidän toimintaansa ja lisää sivuston ylläpidon varmuutta, kun sisällön päivittäjä ei vahingossa pääse muutamaan koko sivuston toiminnallisuutta.

4.2. TIETOKANTA

Web-sisällönhallintajärjestelmät käyttävät tietokantaa varastona sisällölleen. Eniten käytetty tietokanta avoimen lähdekoodin sisällönhallintajärjestelmissä on vapaalla lisenssillä käytettävissä oleva MySQL [MySQL], joka valittiin tämänkin palvelun tietokannaksi. Siihen tallennetaan tagien, tuotteiden, asiakkaiden ja kampanjoiden tiedot. Sisustussuunnitelmat.fi-palvelua varten suunniteltiin relaatiotietokanta, joka on nähtävissä kuvassa 2.



Kuva 2: Palvelun relaatiotietokanta



Kuva 3: Tietokannan ja front-endin havainnekuva

Kuvan 2 tietokanta suunniteltiin vastaamaan havainnekuvassa 3 näkyvää käyttöliittymän ja tuotetietojen välistä suhdetta. Tämä tietokanta toimi teoreettisena pohjana kyseisen palvelukokonaisuuden tietorakenteelle. Tietokantaa ei kuitenkaan implementoitu järjestelmään suoraan noudattaen kuvan 2 mukaista kaaviota. Alustana käytetty web-sisällönhallintajärjestelmä CMS Made Simplen käyttö ja tämän tarjoamien lisäominaisuuksien käyttö muutti lopullisen

tietokannan muodostumista. Tietorakenteen toteutusta nopeuttamaan käytettiin tätä sisällönhallintajärjestelmää varten olevaa valmista moduulia. CMS Made Simplen Catalogue Like Module Maker -moduuli helpotti tietokannan taulujen ja erityisesti niiden hallintapuolen tekemistä sisällönhallintajärjestelmään.

Catalogue Like Module Maker -moduulin avulla pystyi käytännössä luomaan valmiita, asennettavia ja toiseen CMS Made Simpleen siirrettäviä moduuleita. Moduulit on tarkoitettu sellaisen tiedon hallintaan ja esittämiseen, johon sisällönhallintajärjestelmällä ei oletuksena ollut tukea. Yleisesti web-sisällönhallintajärjestelmien käyttötapauksiin kuuluu vain web-sivujen ja tiheään päivittyvän, uutisten kaltaisen sisällön hallinta. Mikäli sisältö palvelussa on vaatimuksiltaan kuitenkin laajempaa, täytyy tälle tuki ja toiminnallisuudet rakentaa yleensä alusta, mikä on kallista ja hidasta.

Käytetyn moduulin avulla säästettiin paljon aikaa, kun suunnitellun tietorakenteen vaatimaa toiminnallisuutta ei tarvinnut toteuttaa alusta asti. Moduulia käytetään sisällönhallintajärjestelmään tehdyn käyttöliittymän kautta. Käyttöliittymässä tietokantatauluja vastaavat rakenteet tehdään nimeämällä ne ja valitsemalla niille halutut kentät ja näille kentille tyypit. Myös tietokantataulujen keskenäisen suhteen pääsee määrittelemään. Tietokantakenttien tyypit määritellään vain käyttöliittymätasolla. Mikäli käyttäjälle halutaan tarjota tekstikenttä, kyseisen kentän tiedon syöttämiseen, hoitaa moduuli varsinaisen tietokantakielen ja luo VARCHAR-tyyppisen kentän. Mikäli jonkin kentän halutaan vaikka olevan kuvatiedosto, moduuli osaa tehdä tälle tarvittavat kuvatiedostoille ominaiset lataus-, tarkistus- ja käsittelytoiminnallisuudet tietokannan luontilauseiden lisäksi. Catalogue Like Module Maker -moduulin muokkaus ja versiointi onnistuu myös, mikäli moduulin vaatimukset muuttuvat myöhemmin.

Moduulilla oli kuitenkin rajoitteensa, eikä sen avulla pystynyt tekemään tietokannasta aivan suunnitellun tietokantakaavion mukaista. Moduuli on tarkoitettu vain yksisuuntaisen, hierarkkisen tiedorakenteen luontiin. Esimerkiksi suhde, jossa on mahdollista olla Asiakas, jolla voi olla kampanjoita, joihin taas kuuluu kuvia ja kuvalla voi olla tietty määrä tuotteita, on täysin mahdollinen moduulilla toteutettavaksi. Kun taas haluamme tuotteiden samalla liittyvän suoraan myös asiakkaaseen, ei tällaista suhdetta ollut mahdollista moduulin avulla rakentaa ja tehdä helposti hallittavaksi ominaisuudeksi valmiiseen moduulin. Kuitenkin tietokanta pystyttiin tekemään lähes suunnitelmaa vastaavaksi, varsinaisena etuna olivat valmiit hallintapaneelit tietotaulujen hallintaan, kuvien lisäykseen ja niiden koon muutoksiin suoraan CMS Made Simplen hallinnan yhteyteen. Tämä tarkoitti sitä, että palvelun hallintaan liittyvää sisältöä, asiakkaita ja kampanjoita pystyi muokkamaan sisällönhallintajärjestelmästä. Moduulilla rakennettiin palvelua varten hallintamoduuli, joka on esitelty kohdassa 4.3.

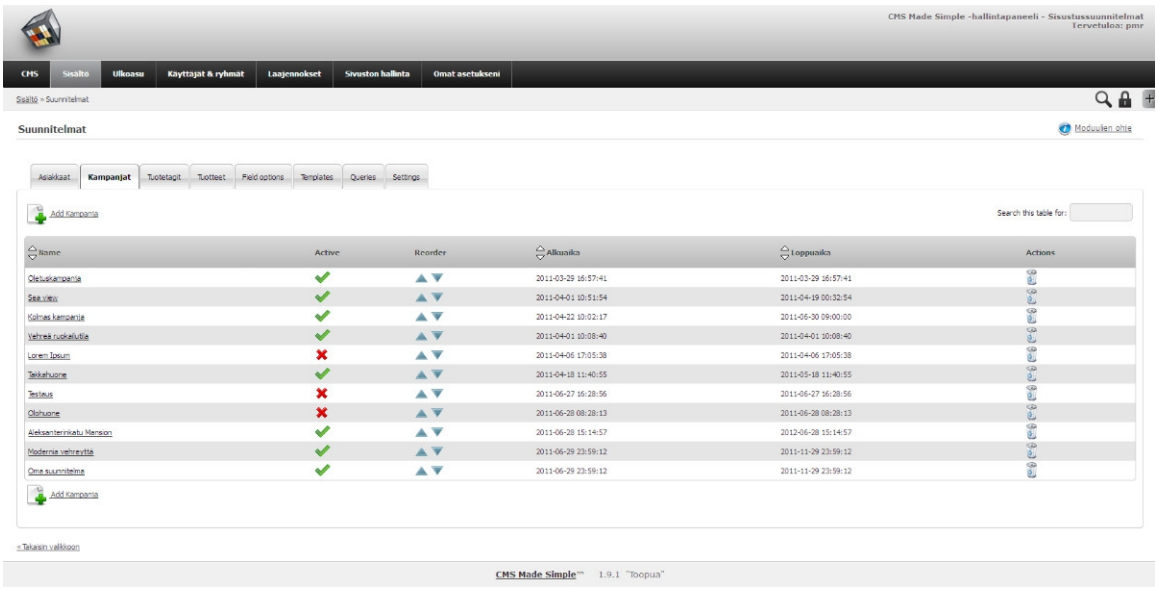
4.3. SUUNNITELMAT-MODUULI

Edellisessä kohdassa kuvattujen toimintojen tuloksena oli Suunnitelmat-moduuli, joka on mahdollista asentaa mihin tahansa CMS Made Simple -asennukseen. Moduuli sisältää asennukseen vaadittavat valmiit funktiot ja tietokannan luontilauseet itsessään. Suunnitelmat-moduuli on tarkoitettu Sisustussuunitelmat.fi:ssa pääylläpitäjälle asiakkaiden, kampanjoiden, tuotteiden ja kampanjoiden avainsanojen ja kategorioiden hallintaan (ks. kuva 4).

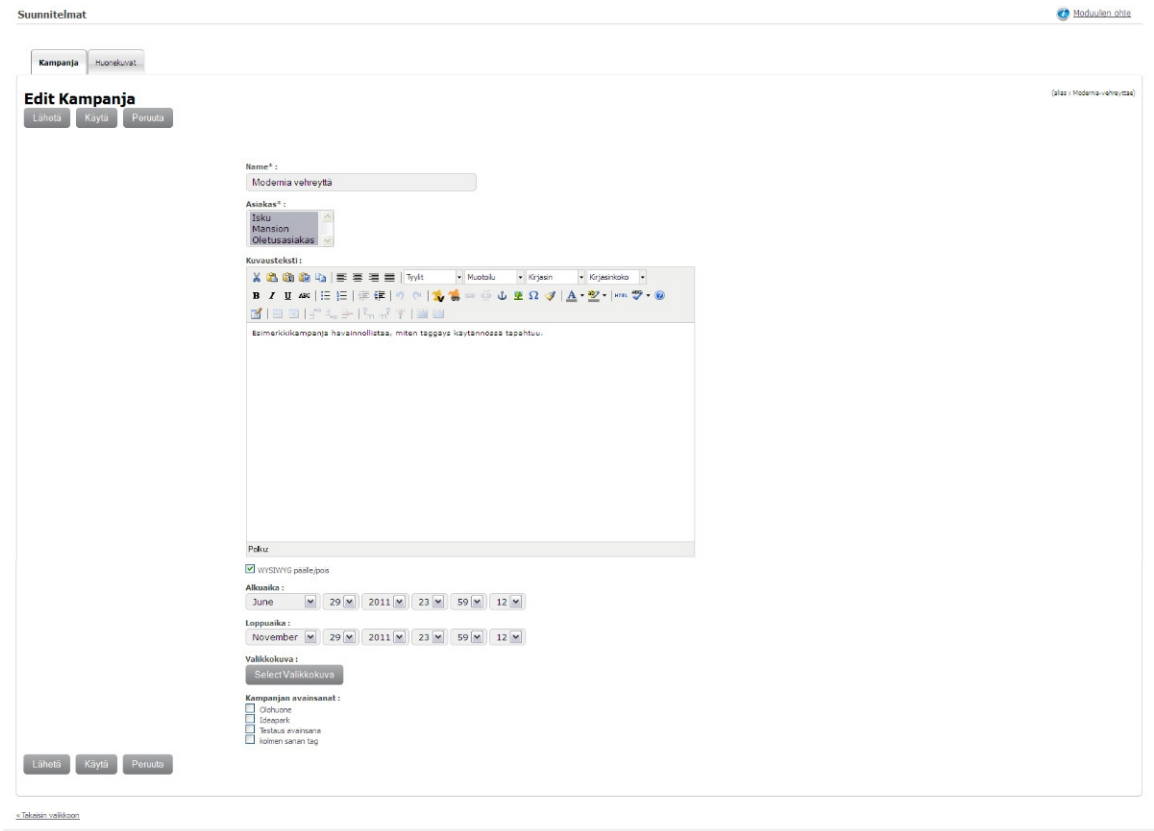
Hallintapuoli mahdollisti näiden tietojen tehokkaan käsittelyn moduulin luoman käyttöliittymän avulla. Kampanjoiden lisäys, poisto, aktiivisuustilan ja -ajan hallitseminen onnistuu kuvassa 4 näkyvän listausnäkymän avulla tai vaihtoehtoisesti kuvassa 5 näkyvän, yksityiskohtaisen kampanjan tarkastelu- ja muokkausnäkymän kautta. Kaikille tietokentille järjestelmästä löytyi vastaavat hallintanäkymät ja toiminnallisuudet.

Suunnitelmat-moduulilla on periaateessa mahdollista hallita myös tuotteita ja tageja, mutta käytännössä siihen sitä ei ole tarkoitettu, edellisen kohdan viimeisessä kappaleessa mainituista puutteista johtuen. Silloin osa id-tietojen yhdistämisistä ja tagin koordinaattitietojen täyttämisestä täytyisi tehdä käsin tekstikenttien avulla, mikä ei olisi tarkoituksenmukaista. Myöskään tagien lisäystä helpottavaa jQuery Photo Tag pluginilla toteutettua käyttöliittymää ei koettu järkeväksi tuoda osaksi sisällönhallintajärjestelmän hallintapuolta. Tagien ja tuotteiden hallintaa varten tehtiin erillinen asiakkaiden/moderaattorien puoli, joka on eristetty CMS Made Simplen hallintapuolesta. Sitä esitellään luvussa 5.

Moduulin etuna on myös sen sisällöstä eristetty esittämiskerroksen suunnitteluominaisuus. Tämän avulla sisällön esittämistä varten voitiin luoda erilaisia pohjia, joilla sisältö voitiin tuoda palvelun käyttäjille. Yleisimpiä näkymiä ovat erilaiset listausnäkymät, joilta navigoidaan yksityiskohtaisille sisältönäkymille. Sisältöpohjia eli templateja moduulille tehdään käyttämällä HTML- ja Smarty-kieliä. Sisustussuunnitelmat.fi-palvelun kohdalla tämä tarkoittaa kategorioiden, asiakkaiden tai avainsanojen perusteella suodatettavia listausnäkymiä, joilla näkyy kampanjakuvia. Yksityikohtaisessa näkymässä näkyy kampanjaan liittyvät kuvat, tagit ja tuotelistaus sisältöpohjassa määritellyllä tavalla.



Kuva 4: Sisällönhallintajärjestelmän hallintapuoli ja Sisustus-moduuli



Kuva 5: Sisustusmoduulin hallinta, kampanjan lisäys

5. TUOTTEIDEN JA TAGIEN HALLINTA

Taustahallintajärjestelmä ja tietokannan luonti CMS Made Simplen avulla mahdollisti Sistustusuunnitelmat.fi:n hallinnan sisällönhallintajärjestelmän web-käyttöliittymän kautta. Tämä ei kuitenkaan tarjonnut käyttöliittymää itse tagien lisäämiselle eli ns. taggaukselle. Tässä on käytännössä kyse tagin paikan määrittämisestä kuvassa, eli koordinaattiarvojen antamisesta tuotetagille ja tagin tietojen syöttämisestä.

Tätä toiminnallisuutta varten toteutettiin erillinen hallintapuoli ja käyttöliittymä. Tämän puolen tarkoitus on tarjota asiakasrajapinnalle mahdollisuus hallita tuotteitaan, lisätä niitä tageina suunnitelmakuviin ja seurata statistiikkaa tuotteiden katseluiden aktiivisuudesta. Hallintapuoli toteutettiin PHP-kielellä ja sen MySQL-tietokantafunktioilla. Tämän apuna käytettiin DB-kirjastoa, joka sisälsi valmiita funktioita tietokannan käsittelyyn ja helpotti kehitystä tehokkailla virheiden etsintä- ja testaustoiminnoillaan.

5.1. HALLINTAPUOLEN KEHITYS ASIAKKAILLE

Tagien ja tuotteiden hallintapuoli on toteutettu PHP-kielellä. Tämä asiakkaille tarkoitettu hallintapuolen pääsivu on tehty CMS Made Simplessä omana sivunaan. Sille toteutetiin oma sivupohja, jonka koodissa kutsutaan palvelimella olevaa, sisältöä ohjaavaa PHP-tiedostoa. Esityskerroksen hallinta pysyi siis edelleen sisällönhallintajärjestelmän sisällä, mutta sisältö tuotiin järjestelmän ulkopuolelta. Sivupohja sisälsi yleisesti kaikilla asiakashallintapuolen sivuilla tarvittavat koodit, kuten sivun ylä- ja alatunnisteen. Ohjaustiedosto luo sivulle hallinta-paneelin joka koostuu valikosta ja sisäänkirjautuneen käyttäjän tiedoista (ks. kuva 6). Luonti tapahtuu sisällyttämällä sivulle (require_once) PHP-tiedostot, jotka luovat kyseiset toiminnot.

Kuva 6: Sivun yläosan hallintapaneeli kirjautuneelle käyttäjälle

Lisäksi ohjaustiedosto sisältää switch-case -valintarakenteen, joka määrittää sivujen sisällön. Sivujen sisältöä ohjataan sivun URL-osoitteessa käytetyllä ohjausparametrilla esimerkin 6 mukaisesti. Tämä siis sisällyttää sivuun aina tietyn, ennalta määrätyn PHP-tiedoston, sen mukaan, mikä action-ohjausparametri on URL-osoitteessa.

Esimerkki 7: Asiakkaiden hallintasivun URL ja ohjausparametri

```
http://www.sisustussuunnitelmat.fi/asiakkaat?action=tuotteet
```

Hallintapuolen käyttö aloitetaan sisäänkirjautumisella. Ohjaustiedosto sisällyttää kirjautumistoiminnallisuudet sisältävän tiedoston, mikäli käyttäjä ei ole kirjautunut sisään tai istunto on vanhentunut. Sisäänkirjautumisen yhteydessä tarkistetaan tietokannasta käyttäjän antaman tunnuksen ja salasanan oikeellisuus, luodaan käyttäjän selaimeen sisäänkirjautuneen istunto ja ohjataan käyttäjä eteenpäin. Mikäli käyttäjä on halunnut jollekin muulle sivulle kuin hallinnan etusivulle ennen kirjautumistaan, ohjataan hänet sinne. Tällainen tapaus on mahdollinen esimerkiksi käyttäjän vaihtaessa sivua hänen istuntonsa vanhennuttua. Tällöin uuden istunnon aloittamiseksi järjestelmä vaatii uuden sisäänkirjautumisen, mutta käyttäjä pääsee kuitenkin haluamalleen sivulle.

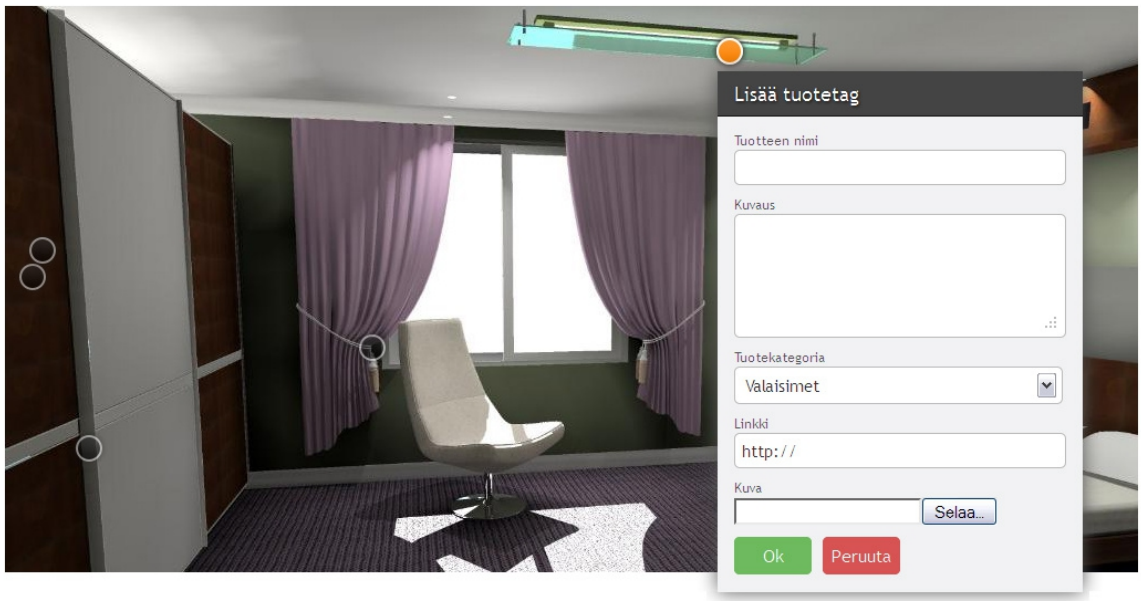
5.2. KAMPANJAT, SUUNNITELMAT JA TAGIT

Asiakkaan suunnitelmakuvat on järjestelmässä lajiteltu kampanjoihin, eli kullakin kuvalla on kampanja. Kampanjat sisältävät rajatun määrän kuvia yhdestä kohteesta, joita loppukäyttäjät voivat selata. Kampanjoiden lisäystä asiakkaan hallintapuolelta ei tehty mahdolliseksi, vaan näiden vastuu jätettiin pääkäyttäjille Sisustus-moduulin avulla CMS Made Simplen hallintapuolelta. Uusien kampanjoiden tilaus- ja ostotoiminnallisuutta jätettiin toteuttamatta järjestelmään. Tämä täytyy hoitaa manuaalisesti, asiakaskohtaisesti, esimerkiksi sähköpostin avulla. Kampanjoiden selaus ja muokkaus onnistuu kuitenkin asiakkaiden hallintapuolella. Asiakkaalle haetaan näytettäväksi kampanjat, joissa kirjautunut käyttäjä eli asiakas on mukana. Kampanjat haetaan tietokannasta ja niiden tiedot ja pienoiskuva listataan näkyville. Pienoiskuvana toimii kampanjan ensimmäisestä kuvasta tehty pienoisversio. Kampanjat on värikoodattu niiden sen hetkisen tilan mukaan. Tiloja ovat seuraavat: *Tuleva*, *Käynnissä oleva*, *Mennyt* ja *Suljettu*. Nämä on toteutettu tarkastelemalla kampanjan alku- ja loppuaikaa sekä sen aktiivisuustilaa. Näiden tarkistusten perusteella kampanjoille on annettu luokka, joille taas CSS-tyylimääritysten avulla annetaan tiloja vastaavat väriarvot.

Kampanjaa klikkaamalla pääsee kyseisen kampanjan muokkaukseen. Tässä luodaan käyttäjälle sivu, johon on lisätty Sisustussuunitelmat.fi-palvelun käyttöliittymän pohja, eli jQuery Photo Tag -plugin ja kuvat, jotka kyseiseen kampanjaan kuuluvat. Plugin lataa kyseiselle kuvalle mahdollisesti jo asetetut tagit. Photo Tag -pluginille on annettu tällä sivulla pääkäyttäjätoiminnot mahdollistavat parametrit. Pääkäyttäjätoimintoihin kuuluu tagien luonti, poisto, siirto ja niiden

sisällön muokkaus loppukäyttäjän tuotetagien tarkasteluominaisuuksien lisäksi. Pluginille annettava parametriolio sisältää myös URL-osoitteet rajapintaan, joita plugin käyttää tuotetagien lisäys-, poisto- ja siirtotoimenpiteisiin.

Pluginin koodi on tällä sivulla siis normaalia front-end-näkymää laajempi. Se sisältää tapahtumat tagien luonnille ja poistolle. Suunnitelmakuvaa – muualla kuin olemassa olevan tagin kohdalla – klikatessa syntyy väliaikainen tagipallo (ks. kuva 7). Tämän tagipallon yhteyteen luodaan lomake, jolla on mahdollista tietojen syöttämisen uudelle tagille. Lomake sisältää tekstikentät ja -alueen uuden tuotetagin nimelle, kuvaukselle ja linkille. Linkki on osoite, josta tuotteesta voi lukea lisätietoa tai mahdollisesti ostaa tuote. Yleensä tämä vie asiakkaan omalle verkkosivustolle. Lisäksi lomake sisältää pudotusvalikon ennalta määrätyille tuotekategorioille. Lomaketta luodessa tämän hetkiset tuotekategoriavaihtoehdot haetaan järjestelmän tietokannasta Ajax-kutsulla. Ne tulevat lomakkeelle JSON-muodossa, josta plugin osaa muotoilla ne HTML-kielen pudotusvalikkoelementiksi. Sisustus-moduulin pääkäyttäjät voivat luvussa 4 esitellyssä hallintaosiossa muokata tuotteiden kategoriavaihtoehtoja. Lomakkeen lopussa on myös mahdollisuus ladata tuotteelle kuva. Tiedoston syöttökentän toteutuksessa on käytetty PHP-Ajax Image Upload -toiminnallisuutta, jonka avulla tiedosto ladataan palvelimelle heti, kun se on valittu. Perinteinen tiedoston lataukseen tarkoitettu HTML-kielen lomake-elementti lataa kuvan palvelimelle vasta, kun lomake lähetetään. Luotaessa sisältöä tämän kaltaisella dynaamisella lomakkeella on käyttäjälle helpompaa, että kuva ladataan palvelimelle heti käyttäjän valittua kuvan koneeltaan ja se näytetään esikatseluna lomakkeen lopussa, vaikkei itse lomaketta olekaan vielä lähetetty palvelimelle. Lomakkeen lopussa on painikkeet tuotetagin tallennukselle ja sille tehtyjen muutosten peruuttamiselle.



Kuva 7: Tagin ja tuotteen lisäys

Tuotetagi on tietokannassa tietty tuote, tietyssä kuvassa ja tietyssä paikassa. Kuvassa 7 näkyvällä lomakkeella luodaan tietokantaan tuote, joka yhdistetään kuvaan, johon se on lisätty. Asiakas saattaa kuitenkin haluta lisätä saman tuotteen useampaan eri kuvaan tai kampanjaan. Tällöin ei ole tietokannan periaatteen mukaista luoda aina uutta tuotetta. Tuotetagien lisäyslomakkeelle toteutettiinkin mahdollisuus hakea asiakkaan luomia tuotteita ja valinta niitä merkattavaksi tageina useampaan kuvaan ja kampanjaan. Pluginille toteutettiin autocomplete-toiminto, jonka avulla on kyseisen asiakas voi hakea lisäämiään tuotteita tagille. Tuotteen nimi -kentälle on kytketty päälle autocomplete-toiminnallisuus, eli jokaisen kenttään kirjoitetun kirjaimen yhteydessä plugin lähettää Ajax-kutsuna kyselyn sillä hetkellä kentässä olevilla merkeillä. Haku tehdään vasta. kun kentässä on vähintään kolme merkkiä, millä vältytään kahdella ensimmäisellä merkillä tehtäviltä hauilta, jotka eivät luultavasti olisi vielä tarpeeksi tarkkoja. Ajax-kutsu tehdään palvelimella olevaan rajapintaan, joka etsii tuotekannasta tuotteet vertaamalla tuotteen nimeä ja kuvausta käyttäjän kirjoittamaan hakuparametriin. Ajax-kutsu saa vastauksena kaikki sisäänkirjautuneena olevan asiakkaan tuotteet, jotka täsmäävät syötettyyn sanaan. Plugin muotoilee vastauksena saadut tuotteet tekstikentän alle erillisiksi riveiksi, josta ne on mahdollista valita tagille klikkaamalla haluttua tuotetta. Valmiin tuotteen tultua valituksi autocomplete-listasta, lomake täyttyy kyseisen tuotteen tiedoilla, jotta käyttäjä näkee onko kyseinen tuote varmasti kuvan sisältöä vastaava tuote. Lomakkeen kenttiä ei tällöin voi muokata, koska tämä muuttaisi tuotteen tietoja, joka saattaa olla merkattuna toisiin kuviin. Mikäli käyttäjä haluaa tuotteelle eri tiedot kuin haetussa tuotteessa, on hänen luotava uusi tuote.

Lomaketta lähetettäessä tarkistetaan JavaScriptilla jo selaimessa, että uuden tuotteen luonnille pakolliset kentät on täytetty. Pakolliset tekstikentät on määrätty koodissa "required"-luokkamäärellä

esimerkin 8 mukaisesti. Pakollisia kenttiä tuotteelle ovat nimi ja tuotekategoria. Lomakkeen lähetyksen plugin hoitaa jQueryn avulla Ajax-kutsulla, pluginin asetusparametrissa määriteltyyn osoiteeseen. Kutsun parametreina lähetetään lomakkeen tiedot (ks. esimerkki 9).

Esimerkki 8: Pakollisen tekstikentän luokkamääre

```
<input id="tempInput_name" class="required" type="text" name="name" value="">
```

Esimerkki 9: Uuden tagin luonnin Ajax-kutsu

```
[path_to_lib]/addTag.php?
left=720&top=205&width=20&height=20&name=Keltainen+tuoli&name_id=[jos_valmis_
tuote]&description=Materiaali+puu&category=12&link=http%3A%2F
%2Ftuoli.com&maxSize=9999999999&maxW=200&fullPath=http%3A%2F
%2Fwww.sisustussuunnitelmat.fi%2Fuploads%2Fsuunnitelmat%2Ftuotteet
%2F&relPath=..%2F..%2F..%2Fuploads%2Fsuunnitelmat%2Ftuotteet
%2F&colorR=255&colorG=255&colorB=255&maxH=300&filename=filename&asiakasid=[as
iakkaan_id]&password=[md5_hashed_password]&image_id=19
```

Tuotteen ja tagin luonnissa kutsuttava rajapinta toteutettiin PHP-kielellä. Tämä toimii keskustelukanavana käyttöliittymän ja järjestelmän tietokannan välillä. Kullekin rajapintatoiminnolle oli oma tiedostonsa, mutta nämä jakoivat keskenään toiminnallisuuksia, kuten käyttäjän oikeuksien tarkistus. Rajapinnan tiedostot oli nimetty niiden toimintaa kuvaaviksi englanninkielisiksi termeiksi, esim addTag ja deleteTag. Rajapintatiedostoja kutsuttiin HTTP:n GET ja POST metodeilla ja niille lähetettiin parametrina niiden tarvitsemat tiedot. Rajapinta käyttää tietojen palauttamiseen JSON-formaattia.

Uutta tagia luotaessa rajapinta tarkistaa käyttäjän oikeellisuuden kutsun parametreina olleiden asiakkaan id:n ja salasanan avulla. Tämän jälkeen se siistii datan ja luo niistä uuden tuotteen ja tagin tietokantaan. Mikäli tuote on jo olemassa, tarvitsee kantaan luoda vain uusi tagi. Tämän jälkeen rajapinta palauttaa vasteena lisätyn tagin tiedot JSON-muodossa. Käyttöliittymässä plugin tuhoaa lomakkeen ja väliaikaisen tagin kuvasta ja luo uuden tagin kuvaan, samalla tavalla kuin sivun latauksessa jokainen tagi luodaan. Koska tagi ja tuote luodaan tekemättä sivunlatausta, täytyy uudelle tagille rekisteröidä käyttöliittymätoiminnot, kuten klikkauksen ja kursorin päälle viennin aiheuttamat toiminnot, jotka tageille yleensä luodaan sivun latauksen yhteydessä. Tagin luonnin lisäksi lisätään myös kuvan alla olevaan tuotelistaukseen tiedot tuotteesta.

Tässä näkymässä kaikkien tagien, jotka liittyvät sisäänkirjautuneen käyttäjän lisäämään tuotteeseen, yhteyteen lisätään myös poistopainike. Poistopainikkeet sijaitsevat tagin yläpuolella ja tuotelistauksessa kyseisen tuotteen yläreunassa. Ne tulee näkyviin aina kursorin ollessa kyseisen tuotteen tagin tai tuotelistaelementin kohdalla. Poistonapin/linkin painaminen laukaisee kutsun rajapintaan, joka poistaa tagin. Kutsulle annetaan parametrina kyseisen tagin tunnus ja käyttäjän tiedot, jotta voidaan tarkistaa käyttäjän oikeellisuus. Rajapintakutsu ei varsinaisesti poista tagia, vaan muuttaa pelkästään sen aktiivisuusarvon nollaksi. Tageja, joilla aktiivisuusarvo on nolla, ei näytetä kuvissa. Tämä mahdollistaa kuitenkin mahdollisten virhetilanteiden varalta tagien palauttamisen. Itse tuotetta, johon tagi liittyy, ei poisteta ikinä tagin poistotoiminnallisuuden yhteydessä.

Järjestelmästä löytyy myös toiminnallisuus tagimerkintöjen siirtämiseen kuvissa. Tagin siirrosta käyttöliittymässä vastaa jQuery UI -kirjaston raahaustoiminnallisuus Draggable. Toiminnallisuus mahdollistaa elementtien raahauksen web-käyttöliittymässä. Raahaus tapahtuu kursorin avulla, pitämällä hiiren painiketta pohjassa ja liikuttamalla elementtiä. Draggable sisältää valmiit tapahtumafunktiot, joita on mahdollista käyttää, kun raahaus aloitetaan, raahattavaa elementtiä liikutetaan ja raahattava elementti päästetään irti. Tagin raahauksen loputtua kutsutaan stop-tapahtumafunktiota, jolle on pluginissa määritelty toiminnallisuus, jolla päivitetään tagin sijainti. Tällöin tehdään Ajax-kutsu PHP-rajapintaan, joka päivittää tagin uudet koordinaatit tietokantaan.

5.3. TUOTEREKISTERI

Tuoterekisteri on olennainen osa asiakkaiden hallintaosiota. Uusia tuotteita asiakas voi luoda suoraan tagauksen yhteydessä kohdassa 5.2. esitellyllä tavalla. Tuotteiden lisäykseen ja muokkaukseen on toteutettu myös tuotelistausnäkymä. Tuotelistaus on lista kirjautuneen asiakkaan omista tuotteista. Oletuksena sivulle haetaan kaikki asiakkaan tuotteet. Tuotteita voi myös suodattaa tietyn tuotekategorian mukaan, jolloin näytetään vain valitun kategorian tuotteet. Suodatuskategoria valitaan tuotelistan yläpuolella olevasta tuotelistasta (ks. kuva 8).

Kuva 8: Tuotelistausnäkymä

Tuotelistausta on myös mahdollista lajitella nousevaan ja laskevaan järjestykseen halutun ominaisuuden avulla. Tämä onnistuu klikkaamalla sen sarakkeen sarakeotsikkoa, jonka haluaa toimivan lajitteluperusteena. Tämä taulukon lajittelu-toiminto on toteutettu hyödyntämällä valmista SortTable-JavaScript -kirjastoa. Kyseinen kirjasto muuttaa HTML-kielen taulukkorakenteen dynaamisesti lajiteltavaksi, eli erillistä sivunlatausta ei tarvita. Kirjasto tekee automaattisesti taulukon sarakeotsikoista lajittelun käynnistäviä linkkejä.

Tuotelistaus koostuu jokaisen tuotteen nimestä, kategoriasta, lisätietolinkistä ja sen tilastotiedoista. Statistiikka sisältää tiedot siitä, kuinka monessa tagissa kyseinen tuote on valittuna, kuinka monta kertaa tagia, joka viittaa kyseiseen tuotteeseen on klikattu auki ja kuinka monta kertaa tuotteen lisätietolinkkiä on klikattu.

Tuotelistauksessa jokaisen tuotteen kohdalla oikeassa reunassa sijaitsevat muokkaa- ja poista-toiminnot. Muokkaa-toiminto vie tuotteen tarkastelynäkymään, jossa tuotteen tietoja pääsee muokkaamaan. Sivulle siirryttäessä URL-osoitteessa parametrina on kyseisen tuotteen tunnus, jonka avulla haetaan muokkausnäkymään oikean tuotteen tiedot. Tuotetietoja haettaessa järjestelmä varmistaa muokattavan tuotteen olevan kirjautuneen käyttäjän tuote, jotta kukaan ei pääse muokkamaan muita kuin omia tuotteitaan.

Poista-linkin klikkaukselle on erikseen jQuerylla tehty poistofunktio, joka laukaisee Ajax-kutsun PHP-tiedostoon. Tiedosto saa parametrinaan tuotteen tunnuksen ja jälleen käyttäjän tiedot käyttäjätarkistusta varten. Tarkistusten jälkeen tuote asetetaan tietokannassa epäaktiiviseen tilaan, eli sitä ei varsinaisesti poisteta. Ajax-kutsu palauttaa käyttöliittymälle vasteen onnistuneesta “poistosta”, jonka jälkeen kyseisen tuotteen rivi poistetaan tuotelistasta käyttäen fade-animaatiota.

Tuotteita voidaan lisätä painamalla yläreunassa olevaa "Lisää tuote" -linkkiä. Tästä avautuu näkymä, jolta löytyy lomake tuotteen lisäykseen. Lomake on lähes identtinen kohdassa 5.2. esitellyn

tagin lisäys -lomakkeen kanssa. Tässä tapauksessa lomakkeelle ei tarvitse hakea tietoa Ajax-kutsuilla, vaan ne voidaan hakea suoraan PHP:n avulla tietokannasta ja rakentaa lomake HTML-kielellä. Autocomplete-ominaisuutta tässä lomakkeessa ei ole, eikä sille ole tarvettakaan, koska tällä lomakkeella on tarkoitus luoda vain uusia tuotteita. Lomakkeen tiedot lähetetään normaalisti lomakkeen käsittelijälle. Tässä tapauksessa lomakkeen käsittelijän koodi on samassa tiedostossa kuin itse lomakekin. Mikäli käyttäjä lähettää lomakkeen, tiedot käsitellään ensin ja tallennetaan tietokantaan, jonka jälkeen sivulla näytetään jälleen lomake ja viesti onnistuneesta tallennuksesta. Näin tuotteiden lisäys on nopeaa. Mikäli lomakkeen tiedot ovat puutteelliset, eikä tuotteen luonti onnistu, pysyvät lomakkeessa käyttäjän täyttämät tiedot ja virheilmoitus kertoo, mitä muutoksia käyttäjän tulisi tehdä tietoihin, jotta tuote voitaisiin luoda.

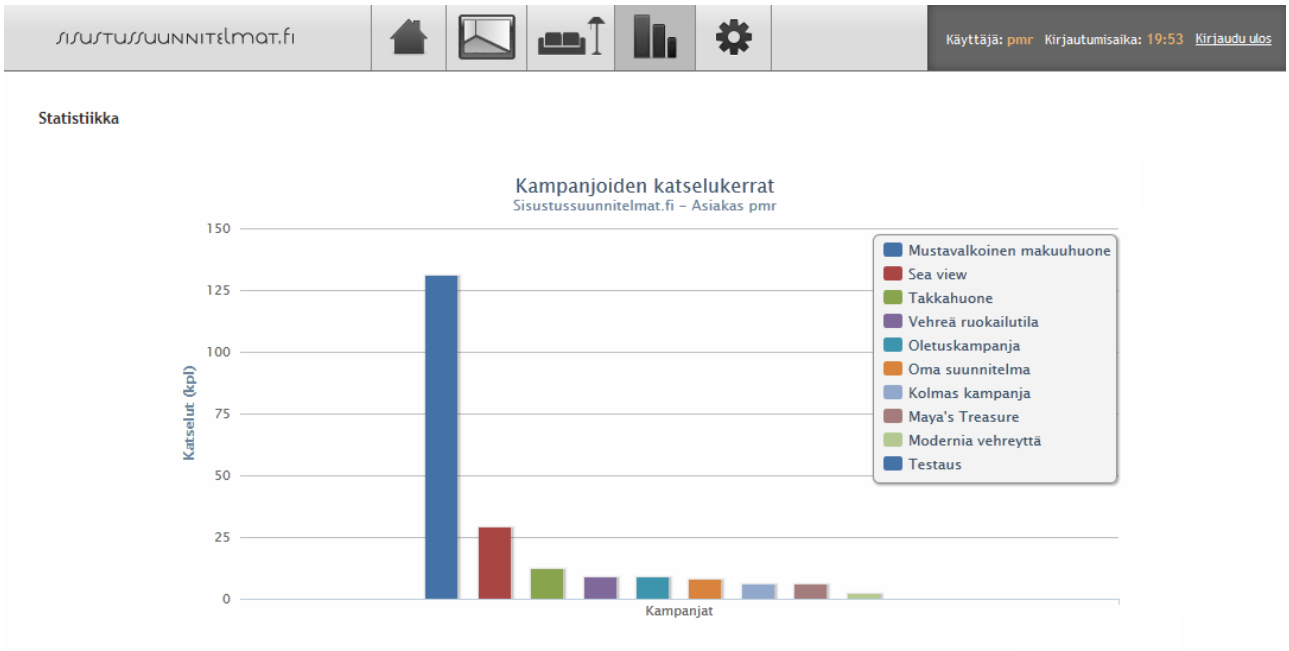
Asiakkaan hallintaosiosta löytyy myös Asetukset-näkymä. Tämän näkymän kautta asiakas voi muuttaa käyttäjätietojaan. Käyttäjätunnuksen, nimi ja osoitetietojen sekä salasanan vaihto onnistuu käyttäjäasetusten kautta. Asiakkaan on mahdollista ladata myös oma logonsa asetusnäkymän kautta. Logo on nähtävissä kaikkien kampanjoiden yhteydessä, joiden kuviin asiakkaan tuotteita on merkittynä.

5.4. STATISTIIKKA

Statistiikka on tärkeä osa Sisustussuunnitelmat.fi:n toimintaa. Seurattavuus ja tulosten analysointi on olennainen osa koko palvelun periaatetta ja merkitystä sen asiakkaille. Tietokantaan tallennetaan loppukäyttäjän, eli suunnitelmakuvien ja tagien selaajan, toimintoja kampanjoita katsellessaan. Tilastojen ja käyttäytymisen perusteella voidaan varsin tehokkaasti arvioida palvelun hyödyllisyyttä ja laskea tuotteiden taggauksen arvoa markkinoinnin ja myynnin näkökulmasta.

Yhdellä asiakkaalla voi olla useita kampanjoita palvelussa. Näiden katselukertoja voidaan seurata palvelun hallintapuolen Statistiikka-osiosta. Kampanjan katselukertoja seuraamalla asiakas näkee suosituimmat kampanjat ja voi arvioida kampanjansa rahallista arvoa (ks. kuva 9)

Järjestelmä tekee aina kampanjasivun ladatessaan Ajax-kutsun tiedostoon, joka lisää kampanjan kävijämäärää tietokannassa. Eri kampanjoiden katselukerrat esitetään tilastona hallintapuolella. Tämän esittämiseen on käytetty JavaScript-kirjastoa *Highcharts.js*. Kyseinen kirjasto on tarkoitettu kaavioiden esittämiseen web-sivustoilla. Järjestelmä hakee tilastot tietokannasta ja muotoilee ne JavaScript-kirjaston vaatimalla tavalla PHP-kielen avulla.



Kuva 9: Yksittäisen asiakkaan kampanojoiden statistiikka

Asiakkaalla on yleensä useita tuotteita, joita kampanjakuvissa esitellään. Sama tuote voi esiintyä useammassa kuvassa eri kampanjoissa ja saman kampanjan sisällä. Järjestelmä tallentaa siksi myös jokaisen tuotteen katselukertojen määrän. Tämä tallennus tehdään aina, kun tuotteen tiedot avataan klikkaamalla, joko tagipallosta tai tuotelistauksesta. Käyttäjä voi katsella tuotetietoja myös pelkästään viemällä kursorin tagipallon päälle, jolloin lisätietoikkuna aukeaa, kunnes kursori viedään pois pallon päältä. Näitä hover-tiloiksi kutsuttuja katseluita ei tallenneta, koska kursoria liikutettaessa kuvan päällä käyttäjä osuu kursorilla myös kohtiin, joita hän ei tarkemmin katsele. Tällöin hover-tapahtumalla tehtyjä tooltipin avauksia ei kannata tallentaa, ellei sitä sido aikaan, jonka kursori on tagipallon päällä. Tätä ei katsottu tarpeelliseksi toteuttaa.

Tuotetta lisätessä tuotteelle on mahdollista asettaa linkki tuotteen esittelysivulle tai verkkokauppaan, josta tuote on mahdollista ostaa. Kun käyttäjä painaa tuotetiedoissa olevaa linkkiä ulospäin valmistajan/verkkokaupan sivuille, tallennetaan kyseinen tapahtuma tietokantaan. Näin ollen pystytään saamaan tietoa katsotuimmista kampanjoista ja tuotteista sekä siitä, kuinka moni katselijoista etenee tuotteen omalle sivulle, josta tuotetta on mahdollista ostaa. Hallintapuolelta on näkyvissä listaukset eri tuotelinkkien avauskerroista.

Tuotteissa oleviin linkkeihin voidaan lisätä parametreina Googlen käyttämiä utm_source-, utm_medium-, utm_campaign- ja utm_content-lisäyksiä (ks. Esimerkki 10). Näiden hyödyntämiseen täytyy linkin kohteena olevan sivuston käyttää Googlen Analytics-seurantaa. Tämä tarkoittaa sitä, että kyseisen sivun HTML-koodista löytyy sivustoa varten tehty oma Google

Analytics -seurantakoodi. Tällöin asiakas voi omalla Google Analytics -tilillään seurata kävijöitä, jotka tulevat Sisustussuunitelmat.fi-sivustolta. Asiakas pystyy erittelemään, mitkä tuotteet ja kampanjat ovat tuoneet eniten liikennettä hänen sivustolleen. Analyticsin avulla asiakas voi myös seurata polkua pidemmälle, ja saada selville, kuinka moni Sisustussuunnitelmat.fi-palvelun kautta tulevista käyttäjistä päätyy ostamaan tuotteen ja arvioida näin palvelun arvoa. Mikäli asiakas linkittää tuotteidensa/verkkokauppansa tuotesivuille muualtakin verkosta, voi hän nähdä tämän palvelun kautta tulleen liikenteen osuuden käyttämällä oikein Google Analyticsia ja utm_-parametreja. [Google Analytics]

Esimerkki 10. Linkki tuotteesta sivustolta ulospäin, Googlen seurantakoodit päälle kytkettynä

```
http://www.sisustusverkkokauppa.fi/
?utm_source=sisustussuunnitelmat
&utm_medium=sisustussuunnitelmat
&utm_campaign=Kampanjan_nimi
&utm_content=Tuotteen_nimi
```

6. KÄYTETYN LÄHDEKOODIN VAIKUTUKSET TUOTTEEN LAATUUN

Projektin toteutuksessa useilla osa-alueilla käytettiin avoimen lähdekoodin materiaalia. Tällä tavalla pystyttiin nopeuttamaan kehitysvaiheita. Kun materiaalia oli monesta eri lähteestä, näiden laatu ja yhteensopivuus vaikuttavat suoraan lopullisen tuotten laatuun. Tässä luvussa perehdytään laadun määritelmään ja arvioidaan avoimen lähdekoodin käytön vaikutuksia yhteen tuotteen laadulliseen kriteeriin, tehokkuuteen.

Avoimella lähdekoodilla tarkoitetaan lähdekoodin tuottamistapaa, jossa kaikilla on mahdollisuus käyttää koodia vapaasti. Avoimen lähdekoodin ohjelmissa tulee aina olla lähdekoodi mukana tai se tulee olla vapaasti saatavilla. Näiden ohjelmien tulee myös olla vapaasti levitettävissä. Lähdekoodia saa käyttää kuka tahansa haluamaansa, myös kaupalliseen, tarkoitukseen [Open Source Definition]. Projektissa käytettiin MIT- ja GPL-ohjelmistolisensseillä olevaa koodia.

Ohjelmiston laatua arvioidaan useilla eri osa-alueilla. Laatu voidaan määritellä siksi asteeksi, jolla ohjelmisto täyttää sille asetetut vaatimukset. Laadullisiin ominaisuuksiin kuuluu mm. ohjelmiston luotettavuus, käytettävyys ja turvallisuus. Laatuun vaikuttaa myös tuotannolliset menetelmät ja tavat, joihin liittyviä ohjelmiston laatutekijöitä ovat tehokkuus, ylläpidettävyys ja laajennettavuus. [Kitchenham and Pfleeger, 1996]

6.1. KEHITYKSEN NOPEUS

Projektin luonteeseen kuului alusta asti ketterä kehitys ja vaihtuvat vaatimukset valmiiden testituotosten perusteella. Valmiin, avoimeen lähdekoodiin perustuvan toiminnallisuuden käyttö nopeutti kehitystä huomattavasti. Kehitysvaiheissa voitiin edetä nopeammin, kun osa toiminnallisuuksista pystyttiin rakentamaan valmiiden järjestelmien, kehysten ja lisäosien ympärille.

Avoimen lähdekoodin käyttöä voidaan pitää eräänlaisena teknisenä lähestymistapana ohjelmistokehityksessä. Tapoja käyttää avoimen lähdekoodia on useita. Ohjelmistoprojektissa voidaan työvälineenä käyttää avoimen lähdekoodin lisenssillä olevia työkaluja, kuten Eclipse ja Subversion. Yksi tapa käyttää avoimen lähdekoodin lähestymistapaa kehityksessä on käyttää kehitystapoja ja malleja, jotka ovat ominaisia avoimen lähdekoodin yhteisöille. Yleisin tapa avoimen lähdekoodin hyödyntämiseen on kuitenkin avoimen lähdekoodin lisenssillä olevien komponenttien yhdistäminen osaksi omaa projektia. Näiden komponenttien toiminnallisuutta

saatetaan myös joutua laajentamaan vastaamaan kyseisen projektin ohjelmiston vaatimuksia. [Hauge *et al.*, 2010]

Olemassa olevan koodin ja toiminnallisuuden käyttö tulee olla suunnitelmallista ja systemaattista, jotta siitä voi täysin hyötyä. Kustannuksia laskettaessa on otettava huomioon myös aika, joka menee valmiin koodin löytämiseen ja siihen perehtymiseen osana yhdistämisprosessia. Käyttöönotettava koodi tulisi myös arvioida ja varmistua sen laadusta. Koodin laatua voidaan analysoida testaamalla, mittamalla sen kompleksisuutta ja arvioimalla sen aiempia käyttökohteita ja niihin liittyviä statistiikoita [Frakes and Terry, 1996]. Tutkielmassa käsitellyssä projektissa toteutetun kuvataggausjärjestelmässä käytetyt valmiit toiminnallisuudet testattiin aina ennen kuin ne todettiin projektin kannalta hyödylliseksi ja otettiin käyttöön. Kuvataggaukseen liittyvän toiminnallisuuden koodin käyttöä arvioitiin pitkään, mutta valintaa helpotti se, että vaihtoehtoisia avoimen lähdekoodin ratkaisuja ei ollut tarjolla. Järjestelmän arkkitehtuurin taustalla vahvasti oleva sisällönhallintajärjestelmä valittiin aiemman kokemuksen ja nopean kehitysvauhdin takia. Tämän valinnassa ei niinkään arvioitu käyttöönottovaiheessa sen vaikutuksia projektin laatuun ja tulevaisuuteen. Kohdassa 6.3. tarkastellaan sisällönhallintajärjestelmän valinnan vaikutuksia tarkemmin.

Valmiin koodin uudelleenkäyttö yleensä nopeuttaa ohjelmiston kehitystä huomattavasti. Ohjelmistokehityksen nopeutta ja aikaa, jolla ohjelmisto saadaan markkinoille, voidaan pitää yhtenä laadullisena mittarina. Tämä on pääasiassa yritystoimintaan liittyvä vaatimus, jossa tuotteen nopeaa julkaisua ennen kilpailijoita voidaan pitää merkittävänä etuna. Uudelleenkäytettävän koodin voidaan myös olettaa olevan laadukasta, koska sen pitäisi olla useaan kertaan testattu ja siinä olevat virheet on korjattu. Jokainen käyttökerta lisää koodin tarkistusten määrää ja tämän tulisi parantaa koodin luotettavuutta. Voidaan myös puhua tuottavuuden paranemisesta valmiin koodin uudelleenkäytön myötä. Kun vähemmän toiminnallisuuksia tarvitsee luoda alusta, voidaan ohjelmisto kehittää vähemmällä työpanoksella ja keskittää enemmän työpanosta toteutettavien toiminnallisuuksien suunnitteluun. [Lim, 1994]

6.2. TEHOKKUUS JA SUORITUSKYKY

Tehokkuus on yksi ohjelmiston laadun mittari. Arvioitaessa projektissa toteutetun ohjelmiston tehokkuutta ja avoimen lähdekoodien osien vaikutuksia sen suorituskykyyn järjestelmää on tarkasteltava osa-alueittain. Web-sovelluksen selaimessa ajettavan koodin nopeus ja suorituskyky on merkittävässä asemassa tehokkuutta käsiteltäessä. Dynaamisten sovellusten myötä koodin siirtyessä ajettavaksi yhä enemmän asiakkaan laittelle suorituskyky on koetuksella. Omat rajoitteensa sovelluksen tehokkuudelle asettaa palvelinpuolen järjestelmä ja sen arkkitehtuuri. Samalla tulee arvioida tehokkuuteen läheisesti liittyviä ominaisuuksia, kuten ohjelmiston laajennettavuutta ja skaalatuvuutta. Web-sovelluksen kehyksenä käytetty sisällönhallintajärjestelmä

on myös tehokkuuden kannalta tarkasteltava osa-alue. Sisällönhallintajärjestelmän toimiessa vahvasti tietokannan ja esityskerroksen välillä sen arkkitehtuuri, ominaisuudet, tuki eri alustoille ja mahdolliset päivitykset vaikuttavat tuotteen suorituskykyyn nyt ja tulevaisuudessa, tuotteen mahdollisesti laajentuessa ja kasvaessa.

6.2.1. Sisällönhallintajärjestelmä

Kun arvioidaan ohjelmiston tehokkuutta, yksi merkittävä osa-alue on ohjelmiston tietokanta ja arkkitehtuuri. Ohjelmiston alkuvaiheessa suunniteltu arkkitehtuuri nousee ratkaisevaan asemaan, mikäli ohjelmisto ja sen kuormitus kasvaa. Projektin kuvataggausjärjestelmän arkkitehtuurin osana toimii vahvasti sisällönhallintajärjestelmä. Käytetty sisällönhallintajärjestelmä CMS Made Simple toimii PHP-kielellä ja käyttää MySQL-tietokantaa. Näitä samoja alustoja käyttävät tänä päivänä useat suuret verkkopalvelut, joten ne eivät voi olla tehokkuuden ja skaalautuvuuden esteenä.

Kun kuvataggauspalvelua käytetään, loppukäyttäjän selain lähettää palvelimella toimivalle sisällönhallintajärjestelmälle pyynnön näyttää haluttu sivu. Palvelimella sisällönhallintajärjestelmä hakee tietokannasta tälle sivulle kuuluvan sisällön ja sivulle määrätyn esitystavan eli sivupohjan ja muodostaa näistä HTML-muotoista sisältöä. Sisällönhallintajärjestelmä tekee yhtä sivunlatausta kohden lukuisia kyselyitä tietokantaan. Kuvatagit ladataan tietokannasta dynaamisesti kuville vasta sivun latauduttua Javascript-kielen avulla. Mikäli palvelun käyttäjien määrä kasvaa, kasvaa myös järjestelmän sivunlatausten ja tietokantakyselyiden määrä. Palvelin pystyy vastaamaan vain rajalliseen määrään kyselyitä kerrallaan. Lukuisat tietokantakyselyt sivunlatausta kohden hidastavat vasteaikaa ja tuovat laskennallista lisätyötä, joka taas heikentää tehokkuutta. Skaalautuvuus onkin otettava huomioon jo järjestelmän suunnitteluvaiheessa ja teknisiä ratkaisuja valittaessa. Palvelimen tehoja lisäämällä pystytään vastaamaan suurempaan määrään kyselyitä, mutta kustannuksellisesti arkkitehtuurin huolellinen suunnittelu ohjelmistokehityksen alusta asti on parempi vaihtoehto. [Offutt, 2002; Dromey, 1995]

Tänä päivänä sisällönhallintajärjestelmät, käytetty CMS Made Simple mukaan lukien, osaavat käyttää välimuistia tehokkaasti. Välimuistin avulla voidaan jo kertaalleen laskennallisesti tuotettu ja käyttäjälle näytetty sisältö tallentaa ja näyttää seuraaville käyttäjille. Tämän avulla järjestelmät pystyvät vastaamaan kyselyihin nopeammin, kun sisältöä ei tarvitse uudestaan hakea ja prosessoida tietokannasta. Tämä myös säästää tehokkuutta palvelimen puolella, kun tietokantakyselyitä ja laskennallisen tehon käyttöä voidaan vähentää. Näin ollen välimuisti lisää mahdollisten kyselyiden määrää, joihin palvelin pystyy kerrallaan vastaamaan. Järjestelmän välimuistiin tallennettaessa tietoa on otettava huomioon, minkälaista sisältöä ollaan tallentamassa. Tiheällä aikavälillä vaihtuvaa tai dynaamisesti käyttäjäkohtaisesti luotua sisältöä ei kannata tallentaa välimuistiin. Myöskään käyttäjään liittyviä yksityisiä tietoja ei tulisi tallentaa välimuistiin. Tekniikoita välimuistin toteuttamiseksi on useita. Tieto voidaan tallentaa väliaikaisesti palvelimen järjestelmän

muistiin, mikä on nopea tapa tarjota välimuistissa olevaa sisältöä. Vaihtoehtoisesti tietoa voidaan tallentaa välimuistiin tiedostoihin tai tietokantaan. Osa käyttäjän tiedoista voidaan tallentaa väliaikaisesti istuntoon ja evästeisiin. CMS Made Simple käyttää tiedostoihin tallennettavaa tekniikkaa välimuistinsa toteutustapana. [Caching in CMS Made Simple]

Projektissa käytetty sisällönhallintajärjestelmä CMS Made Simple pystyy välimuistin ansiosta skaalautumaan suureenkin käyttöön. Avoimen lähdekoodin sisällönhallintajärjestelmien joukossa se ei välttämättä ole skaalautuvuuden kannalta pidetty parhaana mahdollisena vaihtoehtona. Paremmin skaalautuva ja tulevaisuuden mahdollista laajenemista paremmin tukeva vaihtoehto olisi ollut esimerkiksi avoimeen lähdekoodiin perustuva Drupal. Se on CMS Made Simplen tavoin PHP-pohjainen sisällönhallintajärjestelmä, joka käyttää tietokantanaan MySQL-kantaa, mutta tukee myös muita tietokanta-alustoja [Drupal]. Kuvataggausjärjestelmästä kuitenkin vain pääylläpidon toiminnallisuudet ja loppukäyttäjille näytettävät kuvien näyttötoiminnallisuudet ja sivupohjat olivat sisällönhallintajärjestelmän vastuulla. Näiden toimintojen siirtäminen toiseen sisällönhallintajärjestelmään toisi lisätyötä, mutta olisi mahdollista tehdä. Itse tagien näyttö ja lataaminen eivät ole yhteydessä sisällönhallintajärjestelmään, vaan tälle on toteuttetiin PHP:lla oma rajapinta, joka kommunikoi suoraan MySQL-tietokannan kanssa.

6.2.2. JavaScript ja jQuery

JavaScript on komentosarjakieli, joka web-ympäristössä pyörii yleensä selaimessa. Kun kieltä ajetaan käyttäjän selaimessa, käyttää se käyttäjän päätelaitteen prosessoritehoa. Web-palvelut saattavat tänä päivänä rakentua kokonaan JavaScript-koodin varaan. JavaScriptin etuna on, että se toimii selaimessa kaikissa ympäristöissä, eikä vaadi erillisen ohjelman asennusta. Mobiililaitteiden käytön kasvu tuo web-sovelluksille entisestään lisää käyttäjiä ja erilaisia käyttöympäristöjä. Tällöin myös laitteissa ajettavan JavaScript-koodin suorituskyky ja optimointi korostuu. [Ratanaworabhan *et al*.,2010]

JavaScript ja jQuery ovat molemmat skriptikieliä, joita ajetaan käyttäjän selaimessa. jQuery on JavaScript-kielellä tehty kirjasto. Projektissa asiakaspuolen koodissa käytettiin laajasti jQuery-kirjaston ominaisuuksia siitä löytyvien valmiiden toiminnallisuuksien ja lisäkirjastojen takia. Kuvatagien toiminnallisuudesta vastaava jQuery photoTag -plugin oli kokonaan jQueryn avulla toteutettu, joten oli luontevaa käyttää jQuery-kielen ominaisuuksia kaikissa käyttöliittymätoiminnoissa. Kielen helppokäyttöisyys ja ilmaisuvoima kannustivat myös käyttämään jQuerya tässä projektissa, jossa testatattavaa, toimivaa toiminnallisuutta rakennettiin nopeissa vaiheissa. Projektin toiminnallisuudet olisi ollut mahdollista toteuttaa myös pelkällä JavaScript-kielellä, mutta se olisi vaatinut huomattavasti enemmän aikaa. Tehdyn valinnan vaikutusta lopullisen tuotteen selaimessa ajettavan koodin suorituskykyyn voidaan kuitenkin arvioida.

Käyttöliittymän suorituskyvyllä ja tehokkuudella on web-sovelluksissa iso vaikutus käyttökokemukseen ja niin ollen tuotteen laatuun. [Ratanaworabhan *et al*., 2010; Souders, 2008]

6.2.2.1 Valitsinmetodit

Valitsinmetodit ovat yleinen JavaScript- ja jQuery-kielissä käytetty ominaisuus. Valitsinmetodit ovat metodeita, joilla haetaan haluttuja elementtejä käsiteltäväksi DOM-dokumentista. Nämä ovat käytännössä selaimessa ajettavan koodin tärkeimpiä ominaisuuksia – elementtejä haetaan ja niitä käsitellään. Valitsinmetodit saavat parametrinaan hakuehdon, joka on tyypillisesti jokin CSS-valitsin, esimerkiksi id- tai luokkamääre. Valitsinmetodi taas palauttaa hakuehtoparametrilla löytyvät elementit käsiteltäväksi.

Id-määre tarkoittaa HTML-elementeillä yksilöivää määrettä. Oikein muodostetussa HTML-rakenteessa vain maksimissaan yhdellä elementillä voi olla tietty id-määre, eli elementit eivät voi jakaa id-määrettä. Luokkamääre taas on määre, joka voi olla useammalla HTML-dokumentin elementillä. Samalla elementillä voi olla myös useita luokkamääreitä, kun id-määreitä voi olla vain yksi. Yleensä saman luokkamääreen jakavilla elementeillä on samoja ominaisuuksia ja niille halutaan tehdä samanlaisia operaatioita. Tässä käytetään apuna tarkastetavien skriptikielten valitsinmetodeita.

6.2.2.2 Testitapaus valitsinmetodien tehokkuuden mittamiseksi

Javascriptin ja jQueryn eroja tehokkuudessa voidaan mitata luomalla testitapaus, jossa kokeillaan kielten valitsinmetodin käytön nopeutta eri selainalustoilla. Ensin luodaan HTML-sisältö, jossa haetaan jQuery-kirjasto, jonka jälkeen selaimen on mahdollista käyttää jQueryn ominaisuuksia. Tämän jälkeen luodaan kaksi div-elementtiä, toisella id-määre “hello” ja toisella luokkamääre “bye” (ks. esimerkki 11).

Esimerkki 11. Testitapauksen alustava HTML-sisältö.

```
<script
src="//ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js"></script>
<div id="hello"></div>
<div class="bye"></div>
```

Seuraavaksi testitapaukselle luodaan käyttötapaukset. Näissä käyttötapauksissa käytetään id- ja luokkamääreitä molempien kielten valitsinmetodien parametreina. Tässä muuttujalle $el asetetaan valitsinmetodin palauttama elementti (ks. kuva 10).

Javascriptin valitsinmetodeina toimivat getElementByID ja querySelector. GetElementById-metodi hakee nimensä mukaisesti vain id-määreellisiä elementtejä, ja se ottaakin parametrinaan vain haettavan id-määreen nimen. Toinen valitsinmetodi querySelector on käyttötavaltaan monipuolisempi. Se pystyy hakemaan paramterinaan saamansa CSS-valitsinta vastaavat elementit. Parametrina oleva CSS-valitsin voi kohdistua joko id- tai luokkamääreeseen.

jQueryn valitsinmetodina toimii $(CSS-valitsin). Metodissa dollarimerkki toimii aliaksena jQuery-oliolle. Haettaessa id-määrettä CSS-valitsimena #-merkillä alkava "#hello". Haettaessa luokkamääreellä alkavia elementtejä, CSS-valitsimena on pisteellä(.) alkava ".bye".

jQuery ID Selector	`var $el = $('#hello');`
JavaScript ID Selector	`var $el = document.querySelector('#hello');`
jQuery Class Selector	`var $el = $('.bye');`
JavaScript Class Selector	`var $el = document.querySelector('.bye');`
GetElementById	`var $el = document.getElementById('hello')`

Kuva 10: Testitapauksen määrittelyt eri valitsinmetodeille

Tuloksena saatu Taulukko 1 sisältää ensimmäisessä sarakkeessa yleisimmät käytety selaimet ja niiden versiot (sarakeotsikko *UserAgent*). Seuraavat sarakkeet sisältävät lukuarvona tiedon siitä, kuinka monta operaatiota sekunnissa kyseinen selain pystyi sarakeotsikossa määrättyä valitsinmetodia suorittamaan. Sarakeotsikoissa olevat valitsinmetodien nimiä vastaavien metodien koodit ovat siis nähtävissä kuvassa 10. [jsperf]

UserAgent	GetElementById	JavaScript Class Selector	JavaScript ID Selector	jQuery Class Selector	jQuery ID Selector
Android 2.3.7	593,877	27,754	51,088	9,397	44,765
Chrome 17.0.963	9,771,506	209,067	346,535	136,222	866,185
Chrome 18.0.1025	8,748,569	326,550	468,159	274,745	937,292
Chrome 23.0.1271	13,888,750	807,935	3,966,487	288,391	1,215,937
Chrome 26.0.1410	17,501,776	766,096	5,305,886	316,889	1,183,905
Chrome 28.0.1478	15,143,378	505,029	3,604,789	243,076	1,039,332
Chrome 28.0.1500	19,374,704	600,005	4,774,852	205,249	1,045,471
Chrome 29.0.1547	18,578,458	857,393	4,786,339	265,646	1,367,005
Chrome 30.0.1599	13,716,327	1,275,193	5,932,674	235,420	972,639
Chrome 31.0.1607	16,608,922	1,639,809	9,281,207	310,903	1,336,212
Chrome 31.0.1650	17,723,411	1,409,954	6,961,374	233,677	1,022,412
Chrome 32.0.1700	17,038,034	1,214,363	8,194,905	256,806	1,022,315
Chrome 33.0.1704	6,460,748	610,584	3,248,487	63,955	392,142
Chrome 33.0.1750	17,010,368	1,371,374	8,518,782	238,456	863,223
Chrome 34.0.1809	22,855,000	2,760,219	9,196,075	246,645	1,535,386
Chrome 34.0.1820	19,087,680	1,719,818	8,136,680	235,929	1,334,106
Chrome Mobile 28.0.1500	3,032,515	49,783	556,958	19,003	56,415
Chrome Mobile 32.0.1700	2,708,030	174,668	1,403,575	16,526	43,646
Chrome Mobile iOS 32.0.1700	454,569	166,609	308,850	18,421	36,697
Chromium 30.0.1599	11,709,028	1,251,544	5,559,485	136,307	584,878
Chromium 31.0.1650	30,794,706	2,553,168	14,312,168	289,112	1,553,432
Chromium 32.0.1700	10,488,671	1,097,825	5,594,591	166,745	614,940
Firefox 21.0	10,456,871	301,361	1,192,881	50,333	81,305
Firefox 23.0	12,359,752	307,680	924,031	88,927	244,028
Firefox 24.0	5,888,358	284,745	683,233	63,268	162,101

JavaScript jQuery

Taulukko 1: JavaScript- ja jQuery-kielien valitsinmetodien operaatioiden määrä sekunnissa eri selaimilla

Tuloksista nähdään, että JavaScript on valitsinmetodeiltaan huomattavasti nopeampi kuin jQuery. Javascript-kielen metodi getElementById osoittautui nopeimmaksi. Se pystyi testiaineiston perusteella suorittamaan keskimäärin 17 kertaa enemmän operaatioita sekunnissa kuin jQuery-kielen id-määreellisten elementtien hakumetodi. Javascript-kielen toinen id-määreellisten

elementtien hakutapaus, jossa käytettiin querySelector-metodia, oli 5 kertaa tehokkaampi kuin jQueryn vastaava hakutapaus.

Luokkamääreellisten elementtien haussa Javascript-koodi on myöskin nopeampi. QuerySelector-metodi pystyi tekemään keskimäärin 6 kertaa enemmän operaatioita sekunnissa kuin jQueryn vastaava valitsinmetodi $(CSS-valitsin).

6.2.2.3 Tulosten arviointi suhteessa lopulliseen ohjelmaan

Näiden tulosten perusteella voimme arvioida Javascript-kielen olevan suorituskyvyltään huomattavasti tehokkaampi valitsinmetodien käytössä. Jqueryn etuna on kuitenkin parempi ilmaisuvoima sekä laaja kirjaston hyödyllisiä funktiota ja aktiivisen yhteisön tekemiä plugineita. Usein menetetty teho on kuitenkin merkityksestön suhteessa ominaisuuksien tuomiin etuihin kehityksen nopeuden kannalta. Projektissa käytetyn pluginin lopullinen koodi ja sen kutsuminen sisälsi arviolta 200 id-määreisiin perustuvaa valitsinmetodi-kutsua. Näiden jakautuminen voidaan luokitella koodin perusteella seuraavalla tavalla:

- 10% käytetään **aina kun kuvia vähintään yksi**, riippumaton kuvien määrästä
- 40% käytetään yhtä monta kertaa kuin **kuvia** on
- 50% käytetään yhtä monta kertaa kuin (**kuvien * tagien määrä**).

Kaikkia valitisinmetodeita ei luonnollisesti käytetä samaan aikaan peräkkäin missään tapauksessa. Voidaan kuitenkin esittää teoreettinen käyttötapaus, jossa kaikkia valitsinmetodeita käytettäisiin peräkkäin. Oletetaan, että jokaisella kuvalla olevan keskimäärin 5 tagia ja käytössä olisi testitapauksen hitain selainympäristö (ks. taulukko 1). Kyseessä on siis pahimman tapauksen suoritusajan laskeminen [Wilhelm *et al.*, 2008]. Näin ollen lasketaan, kuinka monta kuvaa tarvittaisiin yhtä aikaa käsiteltäväksi, jotta saataisiin yhden sekunnin ero suoritusnopeudessa. Saadaan 200 * (0.1 + (0.4 * kuvat) + (0.5 * kuvat * 5) > 454559 / ((454559 – 36697) / 36697), josta ratkaisemalla saadaan kuvat > (454559/((454559-36697)/36697)) / 580 = 68,8.

Näin ollen kuvia täytyisi olla vähintään 69 kappaletta, jotta saavutettaisiin yhden sekunnin ero kutsuttujen valitsinmetodien tehossa käytettäessä JavaScriptiä. Näin monelle kuvalle tagien lataaminen yhtäaikaa voidaan olettaa olevan epärealistinen tapaus. Valitsinmetodeita ei myöskään missään tapauksessa kutsuta kerralla aivan peräkkäin tavalla, johon laskuesimerkki perustui. Lisäksi, käyttämällä mitä tahansa testausesimerkin nopeampaa selainympäristöä, sekunnin eroon vaadittavien kuvien määrä kasvaa moninkertaiseksi. Voidaankin siis tehdä johtopäätös, että pelkästään valitsinmetodien kannalta katsottuna JavaScriptin käytöllä ei olisi saavutettu merkittävää hyötyä realistisissa käyttötapauksissa.

Valitsimia käytettäessä jQuery on siis huomattavasti hitaampi kuin JavaScript. Käytettäessä jQuery-kirjaston toiminnallisuuksia elementeille kielen valitsimien käyttö on kuitenkin

välttämätöntä. Käsiteltävästä elementistä on jQuery-olio, joka saadaan vain käyttämällä jQuery-kielen valitsinta. Projektin koodista löytyy kuitenkin useita käyttötapauksia, joissa käytettiin elementin valitsemiseen jQuerya, mutta sille ei tehty mitään, joka varsinaisesti tarvitsi jQuery-kielen ominaisuuksia. Tällaisissa tapauksissa korvaamalla jQuery-valitsimet natiivin JavaScriptin valitsimilla olisi mahdollista optimoida ohjelman selaimessa ajettavan koodin tehokkuutta. Myöskin käytettäessä jQuery-kielen valitsimia on tehokkuuden kannalta merkitystä, kuinka valitsimia käyttää. Valitsinlauseita käytettäessä valitsinehto kannattaa aina aloittaa id-määrellä, mikäli se on vain mahdollista hakuehdon kannalta. Jos hakuehdolla haetaan useita elementtejä tietyllä luokkamääreellä, kannattaa kuitenkin elementtien lähimmän yhteisen vanhempielementin id-määrettä käyttää haussa. Tällä pystytään parantamaan DOM-haun tehokkuutaa. Myöskin valitsinehtojen elementtien täsmällisellä määrittelyllä voidaan optimoida käytettävän valitsinmetodin tehokkuutta. [jQuery Optimize Selectors]. jQuery-kieli käyttää elementtien valinnassa Sizzle-Javascript-kirjastoa [Sizzle].

6.2.3. Sovelluksen lataus

Käsiteltäessä web-sovelluksen tehokkuutta ja suorituskykyä on huomioitava myös palvelun latausaika ja siitä johtuvaa viive. Lataus- ja vasteaikaa voidaan nopeuttaa varmistamalla palvelun palvelinten ja arkkitehtuurin tehokas toteutus. Sisällön, esimerkiksi kuvien, tallentamisessa käytetyn erillisen sisällönjakeluverkon (Content Delivery Network) avulla voidaan nopeuttaa palvelun latausaikoja. Sisällönjakeluverkot koostuvat useista palvelimista, jotka tarjoavat tiedostoja sieltä, mistä ne ovat nopeiten saatavilla. Selaimet myös rajoittavat rinnakkaisten yhteyksien määrää samaan verkkotunnukseen sivuston sisältöjä ladattaessa. Jos sivuston sisältö ladataan useasta eri verkkotunnuksesta, selain pystyy lataamaan enemmän osia kerralla ja teoriassa latausaika pienenee. Selaimet tulkitsevat myös aliverkkotunnukset ”eri” verkkotunnuksiksi, joten sivustolle ladattavien tiedostojen eriyttäminen omaan aliverkkotunnukseen pelkästään mahdollistaa selaimen lataavan enemmän tietoa kerralla. [Souders, 2008]

Selaimessa ajettavan koodin määrän kasvaessa sivujen latausaikakin kasvaa. JavaScript-koodia ladattaessa käyttäjän latausaikaa nopeuttaa koodin tiivistäminen ja yhdistäminen [Burtscher *et al.*, 2010]. Tällä tarkoitetaan, että JavaScript-tiedostoja yhdistetään ja ne tiivistetään tiivistysalgoritmilla. Tuloksena saatava tiedosto on huomattavasti alkuperäisiä JavaScript-tiedostoja pienempi. Web-palvelut myös lataavat sivua ladattaessa käyttäjän selaimeen paljon JavaScript-koodia, jota ei tarvita heti. Souders [2008] tutki kymmentä suurta sivustoa, joilla on paljon selaimessa ajettavaa koodia. Ladatun koodin funktioista vain 26% tarvittiin heti sivun ladattua ja 74% vasta myöhemmin. Tällöin onkin sivunlatauksen nopeuttamiseksi mahdollista jakaa koodi osiin, jossa vain välttämättömät funktiot ladataan sivun latauksen yhteydessä ja muut funktiot vasta myöhemmin. Tutkielman aiheena olevan projektin kuvataggaukseen liittyvän JavaScript-

kooditiedostojen koko on 47,2 kilobittiä ja tiivistettynä 24,9 kilobittiä. Tiivistämällä tiedostot pienenevät siis 47%. Nämä laskutoimitukset eivät sisällä jQuery- ja jQuery UI -kirjastoja. Nämä kirjastot ladattiin käyttäjille tiivistettyinä tiedostoina käyttäen jQueryn ja Googlen tarjoamia CDN-palveluita. Tällöin saatiin jaettua ladattavaa sisältöä yhä useammalle verkkotunnukselle, mikä nopeuttaa latausaikoja ja pienentää omien palvelinten kuormaa.

Kuvataggauksessa käytetty jQuery photoTag -plugin on toteutettu muokattavaksi asetusten avulla. Sitä kutsuttaessa voidaan antaa parametrina olio, joka sisältää arvoja, joilla voidaan ohjata kuvatagitoiminnallisuutta. Sovelluksen parametrisointi mahdollistaa sen uudelleenkäytetävyyden ja siirrettvyyden eri tilanteisiin ja ympäristöihin [Dromey, 1995]. Ohjelman käyttöliittymän toimintalogiikkaa, ulkoasua ja kehotteita pystyy muokkaamaan eri asetuksilla. Muuttamalla muutamaa parametria voidaan esimerksiksi kuvatag-käyttöliittymän vuorovaikutuslogiiikka ja ulkoasu tehdä sopivaksi suurille kosketusnäytöille, joilla on erilaisia toiminnallisia vaatimuksia kuin perinteisellä tietokoneen selaimella. Kosketusnäytöillä ei ole varsinaista hover-tilaa. Tämän tilan ollessa käyttöliittymän yksi olennaisimmista tapahtumista käyttöliittymässä kuvatagin aukaisuun täytyy kosketusnäytöille olla oma, vaihtoehtoinen logiikkansa. Kosketusnäyttöjä varten kuvissa olevien tagimerkintöjen kokoa voidaan myös kasvattaa, jotta ne ovat paremmin käyttäjän sormella painettavissa.

Kääntöpuolena parametrisointi kasvattaa koodin määrää ja näin ollen ladattavien tiedostojen kokoa hidastaen sen latausta ja ajoa. Kun ohjelmaan tehdään useita vaihtoehtoisia suorituspolkuja toimintalogiikalle, jotka riippuvat sen saamista parametreista, ohjelman monimutkaisuus kasvaa ja sen ylläpidettävyys ja ymmärrettävyys vaikeutuu. Näiden laadullisten ominaisuuksien takaamiseksi onkin tärkeää, että moduuli parametrisoidaan vain tarpeellisia käyttötapauksia varten [Dromey, 1995]. Lukuisten parametrien käytön myötä ohjelman kasvaessa käyttäjän latausaikaa nopeuttamaan ohjelmasta voidaan tehdä erillinen optimoitu tuotantoversio, joka tiivistämisen lisäksi karsitaan niistä ominaisuuksista ja parametrien mahdollistamasta logiikasta, jotka kyseiselle käyttötapaukselle ovat turhia. Käyttäjän laitteelle ladattava koodi voitaisiin optimoinnin kannalta myös jakaa heti sivun latauksen jälkeen tarvittaviin funktioihin ja myöhemmin tarvittaviin funktioihin, jotka ladattaisiin vasta sivun ladattua, jolloin sivun lataus nopeutuu [Souders, 2008].

7. YHTEENVETO

Tutkielmassa esitetyn järjestelmän toteutusta voidaan pitää onnistuneena projektina. Siinä pystyttiin hyödyntämään avoimen lähdekoodin materiaalia tehokkaana osana ketterää kehitysprojektia. Valmista ja toimivaa toiminnallisuutta syntyi paljon ja lopputuloksena on toimiva järjestelmä. Tuloksena on myös jonkinlainen teoreettinen malli siitä, kuinka tagien lisääminen kuviin järjestelmänä toimisi. Projektin järjestelmä voidaan katsoa olevan liian sidottu kyseiseen käyttötapaukseen, jossa kuviin lisätään huonekaluja. Myöskin lopullisen järjestelmän käyttäjäroolitus ja toimintalogiikka saattavat olla rajoittavia tekijiöitä, mikäli järjestelmästä haluttaisiin eriyttää yleinen malli, joka olisi jalostettavissa mihin tahansa muuhun käyttötapaukseen ja -ympäristöön.

Käytetty avoin lähdekoodi oli projektin onnistumisen kannalta olennaisessa osassa. On mahdoton arvioida, olisiko projekti ikinä edennyt haluttuun suuntaan ilman valmista koodia. Jotta tutkielmassa arvioidut vaikutukset tehokkuuteen voisivat olla koetuksella, täytyisi järjestelmän kasvaa todella suureksi ja suosituksi. Ilman valmista toiminnallisuutta se ei ainakaan olisi mahdollista. Toki on selvää, että mitä myöhemmässä vaiheessa ohjelmistoprojektin elinkaarta aletaan tekemään esimerkiksi tehokkuutta parantavia toimenpiteitä, sitä kalliimpaa ja vaikeampaa se on. Tehokkuuteen vaikuttaa ohjelman arkkitehtuuri, koodin kompleksisuus ja käytetyt algoritmit joiden muuttaminen jälkikäteen on työlästä ja osittain myös mahdotonta.

7.1. KUVIEN TAGGAUKSEN TULEVAISUUS

Tagien lisääminen kuviin on tapa laajentaa webin toiminnallisuutta. Niiden avulla voidaan lisätä ja yhdistää tietoa kuvissa. Kuvien määrä, koko ja niiden sisältämä informaatio tulee kasvamaan osana Internetiä. Jotta tageistä voisi tulla yleisesti käytetty tapa lisätä tietoa kuviin, täytyisi sille muodostua oma formaattinsa. Tällainen yleinen, yksiselitteinen muoto helpottaisi hakukoneiden ymmärtämistä kuvatagien sisällöstä ja paikasta. Mikäli tapoja esittää tageja kuvissa on useita erilaisia, hakukoneiden kehitys kuvatagien lukemisen mahdollistamiseksi todennäköisesti hidastuu ja vaikeutuu.

Kuvataggauksen kehityksen kannalta on oleellista, että kuvien taggaus tehdään mahdollisimman helpoksi. Tagien merkitsemisen kuviin täytyy olla mahdollista kaikille Internetin käyttäjille. Tagien lisääminen kuviin tulisi toimia selaimessa ilman asennettavia lisäosia, juuri halutun kuvan yhteydessä. Käyttäjien kuviin lisäämästä tiedosta olisi mahdollista muodostua Wikipedian kaltainen tietoyhteisö, jossa tieto kehittyisi käyttäjien mukana ja loisi paremmin hahmotettavaa ja rakennettua Intenetiä. Tämä edellyttää kuitenkin kuvatagien helppokäyttöisyyttä ja yhtenäisyyttä.

Kuvien taggauksesta saatava data on hyödyllistä. Asioiden, kuten esineiden, henkilöiden ja paikkojen merkitseminen kuviin tarkentaa Internetiä. Luomalla linkki kuvan ja kuvassa juuri tietyssä kohdassa olevan asian välille tuo tarkempaa ja monipuolisempaa tietoa webin käyttäjille. Kuvien tarkentuessa ja kasvaessa ei välttämättä olekaan tarve näyttää tiedon etsijälle koko kuvaa, vaan on mahdollista näyttää vain osa tai osia kuvasta, kunhan on olemassa tietoa siitä mitä kuvassa on. Kyseistä tietoa on myös mahdollista hyödyntää kaupallisesti. Kaupan siirtyessä yhä voimakkaammin verkkoon, kuvat ovat olennaisessa osassa käyttäjien ostopäätöksen teossa. Tuotteita ei enää välttämättä päästä koskemaan fyysisesti, joten kuvat ovat käyttäjien lähin kontakti tuotteeseen. Tuotteiden näkeminen havainnollistavissa kuvissa helpottaa tuotteiden hahmottamista. Tuotteita myyvän tahon on mahdollista markkinoida ja myydä tuotteitaan suoraan kuvien yhteydessä, jossa tuotteet todennäköisesti herättävät verkossa eniten mielenkiintoa.

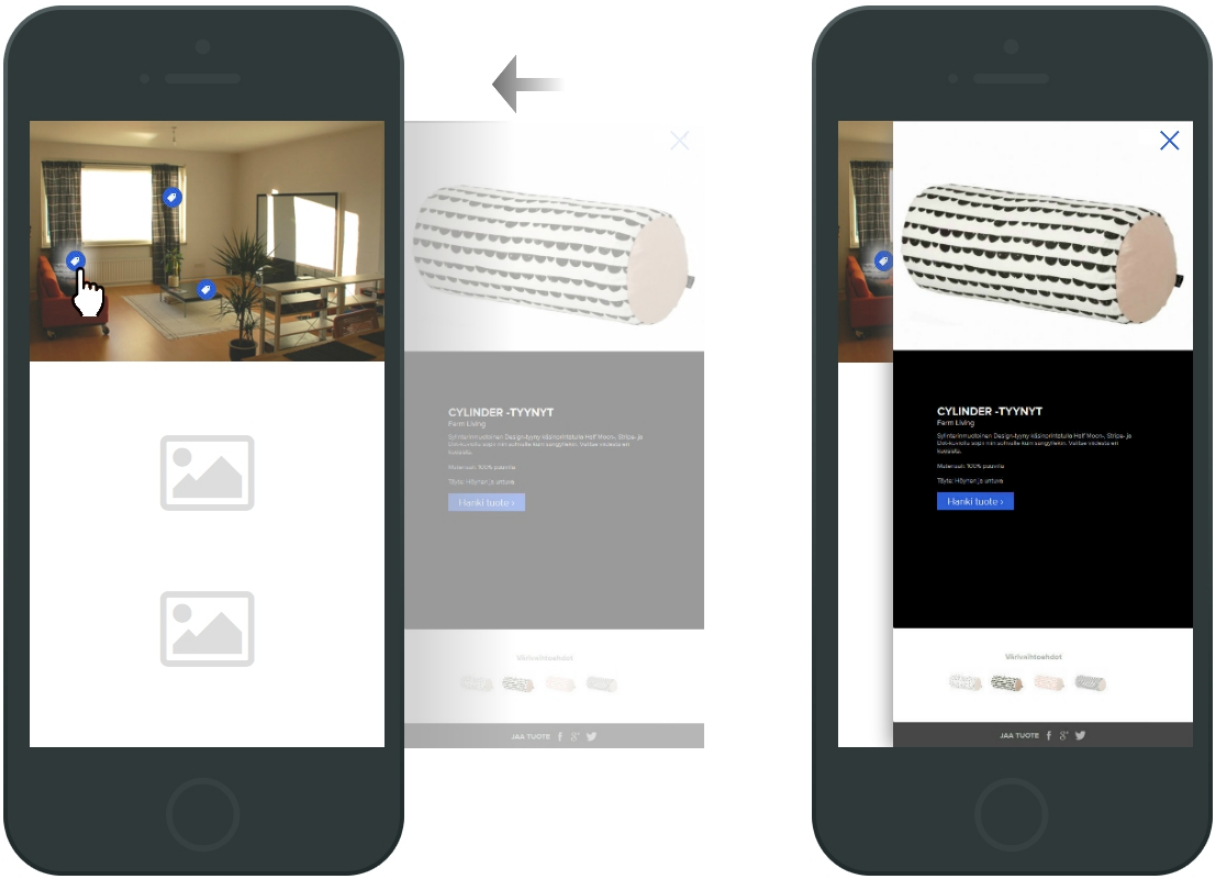
7.2. JATKOKEHITYS

Tutkielmassa esiteltyä järjestelmää olisi mahdollista jatkokehittää monella tavalla. Mikäli järjestelmä keskittyisi vain sisustuskuviin ja niihin liittyviin tuotteisiin, olisi tärkeä keskittyä tuotteiden ja kuvien yhdistämisen helppouteen ja laiteriippumattomuuteen. Toteutettu järjestelmä perustui siihen, että tuotteet lisättiin omaan rekisteriin ja niitä linkitettiin kuviin. Oman tuotetietokannan ylläpitäminen on työlästä, mikäli tuotteiden määrä kasvaa suureksi, eikä ylläpidosta vastaa itse tuotteiden valmistaja tai myyjä. Yksi ratkaisu tähän voisi olla jonkin avoimen tuotetietokannan käyttö. Tällainen tietokanta on Freebase, joka on käyttäjien ylläpitämä avoin tietokanta [Freebase]. Avoimen tietokannan käyttö mahdollistaisi kyselyvaiheessa tiedon jalostamisen ja rikastamisen muunkin avoimessa kannassa olevan tiedon avulla.

Toinen mahdollisuus olisi käyttää suoraan tuotteiden valmistajien ja jälleenmyyjien verkkokauppojen tuotekantoja. Valitettavasti tällä hetkellä yleistä mallia tälle ei ole olemassa, koska verkkokauppa-alustoja on monia ja niiden toteutukset ja kantarakenteet vaihtelevat. Suoraan erilaisten verkkokauppojen tuotekantoihin ei kyselyitä kannattaisi alkaa tekemään, vaan tuotteita tulisi hakea jonkin rajapinnan kautta. Siksi olisikin hyvä, että verkkokaupat tarjoaisivat suljetun tai avoimen rajapinnan tuotekyselyihin, joka mahdollistaisi tuotteiden esittämisen muuallakin kuin vain verkkokaupan yhteydessä. Yksi suosituimmista ja eniten kasvavista verkkokauppa-alustoista on avoimeen lähdekoodiin perustuva ohjelmisto Magento. Magento tarjoaa SOAP- ja REST-rajapinnan kautta mahdollisuuden päästä käsiksi verkkokaupan tuotteisiin. [Magento] Tällaisen rajapinnan kautta tuotekannan käyttö suoraan verkkokaupan tietokannasta helpottaisi ylläpitoa, kun tuotteen tietojen ylläpito jäisi varsinaiselle myyjälle, eikä sitä tarvitsisi pitää omassa kannassa. Se mahdollistaisi myös projektissa toteutetun järjestelmän tuotetietojen rikastamisen verkkokaupan tiedoilla. Kuvaan merkityn tuotteen yhteyteen voisi reaaliaikaisesti päivittää tuotteen hintaa, mahdollisia tarjouksia ja varastosaldoa. Tätä toteutustapaa hieman laajentamalla tuote voisi

periaatteessa liittyä useaan verkkokauppaan, josta se on mahdollista saada. Käyttäjälle voidaan hakea näistä esimerkiksi halvin hinta, lähin varasto tai paras saatavuus, mikäli kyseinen tuote on loppu toisesta kaupasta.

Kuvatagien toimivuus kaikilla laitteilla olisi myös jatkokehityksen kohde. Alkuperäinen toimintalogiikka, jossa tageja näytetään tooltip-muodossa kuvissa perustui siihen, että tagista avautuva tooltip voi mennä kuvan reunoista yli. Tämä asettaa oman haasteensa varsinkin mobiililaitteille, joissa näytön koko on pieni ja kuvat ovat usein täyden ruudun levyisiä. Tagikäyttöliittymän responsiivinen toteutus varmistaisi toimivuuden eri kokoisilla päätelaitteilla. Mobiililaitteiden yleistyessä ja muuttuessa jopa pääasiallisiksi Internetin selausvälineiksi voisi järjestelmää jatkokehittäessä arvioida vaihtoehtoisia esitystapoja kuvien yhteydessä esitettäville tageille ja niistä avautuville lisätietoikkunoille. Yksi vaihtoehtoinen tapa mobiililaitteille voisi olla off canvas -malli, jossa tieto tuodaan näkyville ruudun ulkopuolelta, yleensä vasemmalta tai oikealta. Pieninäyttöisillä laitteilla, jossa vaakasuunnassa tilaa on rajoitetusti, websivut kasvavat pystysuunnassa pitkiksi. Tällöin osa toiminnallisuudesta, kuten navigointi saattaa olla kaukana, mikäli ne löytyvät vain sivun ylä- tai alalaidasta. Off-canvas -mallisella toteutustavalla tärkeitä toiminnallisuuksia on mahdollista tuoda yhden klikkauksen päähän. Tällöin sisältöä voidaan tuoda esiin nykyisen sisällön päälle [Wroblewski, 2012]. Tagia painamalla tuotetiedot voitaisiin siis dynaamisesti hakea sivun reunasta näkyviin tulevalle kerrokselle, joka olisi kuvan ja muun websivun sisällön päällä havainnekuvassa 11 esitetyllä tavalla.



Kuva 11: Off Canvas -mallin mukainen esitystapa mobiililaitteille

Viiteluettelo

[Burtscher *et al.*, 2010] Martin Burtscher, Benjamin Livshits, Gaurav Sinha and Benjamin G. Zorn, JSZap: Compressing JavaScript Code. In: *Proceedings of the 2010 USENIX Conference on Web Application Development* (2010), 39–50.

[Caching in CMS Made Simple] Caching in CMS Made Simple, Available at http://docs.cmsmadesimple.org/general-information/caching-in-cmsms. Checked 4.6.2014.

[Cockburn and Highsmith, 2001] Alistair Cockburn and Jim Highsmith, Agile software development: The people factor. *IEEE Computer* **34** (2001), 131–133.

[Dromey, 1995] R.G. Dromey, A model for software product quality. *IEEE Transactions of Software Engineering,* **21**, 2 (1995), 146–162.

[Drupal] Drupal.org, Available at https://drupal.org/. Checked 4.6.2014.

[Frakes and Terry, 1996] William Frakes and Carol Terry, Software reuse: metrics and models. *ACM Computing Surveys* **28** (1996), 415–435.

[Freebase] Freebase, Available at http://www.freebase.com/. Checked 5.6.2014.

[Google Analytics] Google Analytics, Available at http://www.google.com/analytics/. Checked 22.4.2014.

[GPL] The GNU General Public License v3.0 - GNU Project - Free Software Foundation. Available at http://www.gnu.org/licenses/gpl-3.0.html. Checked 9.4.2014.

[Hauge *et al.*, 2010] Øyvind Hauge, Claudia Ayala and Reidar Conradi, Adoption of open source software in software-intensive organizations – A systematic literature review. *Information and Software Technology* **52** (2010), 1133–1154

[Javascript] Javascript, Available at http://fi.wikipedia.org/wiki/JavaScript. Checked 22.4.2014.

[jsperf] jQuery vs JavaScript Performance Comparison, Available at http://jsperf.com/jquery-vs-javascript-performance-comparison/22. Checked 22.4.2014.

[jQuery Optimize Selectors] Optimize Selectors, Available at http://learn.jquery.com/performance/optimize-selectors/ . Checked 1.5.2014.

[jQuery Plugins] Plugins, Available at http://learn.jquery.com/plugins. Checked 22.4.2014.

[Kitchenham and Pfleeger, 1996] B. Kitchenham and S.L Pfleeger, Software quality: the elusive target. *IEEE Software* **13** (1996), 12–21.

[Lim, 1994] Wayne C. Lim, Effects of reuse on quality, productivity, and economics. *IEEE Software* **11** (1994), 23–30

[Magento] Magento Ecommerce Software & Ecommerce Platform Solutions, Available at http://magento.com/. Checked 5.6.2014.

[McKeever, 2003] Susan McKeever, Understanding Web content management systems: evolution, lifecycle and market. *Industrial Management & Data Systems* **103** (2003), 686-692.

[MIT] The MIT License (MIT) Open Source Initiative. Available at http://opensource.org/licenses/MIT. Checked 9.4.2014.

[MySQL] MySQL Open Source Database, Available at http://www.mysql.com/. Checked 17.4.2014.

[Offutt, 2002] Jeff Offutt, Quality attributes of web software applications. *IEEE Software* **19** (2002), 25–32

[Open Source Definition] The Open Source Definition, Available at http://opensource.org/docs/definition.php. Checked 28.4.2014.

[Park *et al.*, 2005] Jihye Park, Sharron J. Lennon and Leslie Stoel, On-line product presentation: effects on mood, perceived risk, and purchase intention. *Psychology and Marketing* **22** (2005), 695–719.

[Ratanaworabhan *et al.*,2010] Paruj Ratanaworabhan, Benjamin Livshits and Benjamin G. Zorn, JSMeter: Comparing the behavior of JavaScript benchmarks with real Web applications. In: *Proceedings of the USENIX Conference on Web Application Development* (2010), 27–38.

[Sizzle] Sizzle JavaScript Selector Library, Available at http://sizzlejs.com/. Checked at 1.5.2014.

[Smarty] Smarty, Available at http://fi.wikipedia.org/wiki/Smarty. Checked 21.3.2014.

[Souders, 2008] Steve Souders, High Performance Web Sites. *Queue – Scalable Web Services* **6**, 6 (2008), 30-37.

[W3Techs] Usage Statistics and Market Share of Content Management Systems for Websites, April 2014. Available at http://w3techs.com/technologies/overview/content_management/all. Checked 17.4.2014.

[Wilhelm *et al.*, 2008] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, Frank Mueller, Isabelle Puaut, Peter Puschner, Jan Staschulat and Per Stenström: The worst-case execution-time problem–overview of methods and survey of tools. *ACM Transactions on Embedded Computing Systems* **7** (2008).

[Wroblewski, 2012] Luke Wroblewski, Multi-Device Layout Patterns. Available at http://www.lukew.com/ff/entry.asp?1514. Checked 5.6.2014.

Määritelmät, termit ja lyhenteet

1. **CMS Made Simple** on avoimeen lähdekoodiin perustuva yksinkertainen ja tehokas sisällönhallintajärjestelmä. Sisältää paljon valmiita moduuleita ja plugineita ja aktiivisen kehittäjäyhteisön.
2. **jQuery** on avoimen lähdekoodin JavaScript-kirjasto. Tarjoaa hyvän pohjan js-sovellusten luontiin, paljon tehokkaita apufunktioita. jQuery on Internetin käytetyin JavaScript-apukirjasto. http://jquery.com/, https://developer.mozilla.org/en-US/docs/Web/JavaScript
3. **jQueryUI** on jQuery:n päälle rakennettu kirjasto, joka tarjoaa valmiita käyttöliittymäfunktioita, esim: raahaus, liuku, ja automaattinen tekstikenttien täyttö (drag, slide ja autocomplete). http://jqueryui.com/
4. **Ajax** on lyhenne sanoista Asynchronous JavaScript and XML. Se on yhdistelmä useita tekniikoita, jotka mahdollistavat www-sivun dynaamiset toiminnallisuudet ilman sivunlatausta ja siirtymistä sivulta toiselle.
5. **PHP** on yleisesti käytetty ohjelmointikieli web-ympäristöissä. Käytetään dynaamisten web-sivujen luontiin.
6. **JSON** on yksinkertainen tiedonsiirtomuoto, jota käytetään JavaScript-sovelluksissa.
7. **Tag** on merkintä kuvassa. Sisustussuunitelmat.fi:n tapauksessa tagit on esitetty palloina.
8. **Tooltip** on vihje/apuikkuna, joka sisältää lisäinformaatiota. Koostuu yleensä laatikosta, jossa tieto on ja kärjestä, joka osoittaa kohteeseen, josta tieto on.