VAASAN AMMATTIKORKEAKOULU
VASA YRKESHÖGSKOLA
UNIVERSITY OF APPLIED SCIENCES

Edmin Grbic

MYYNTIKONFIGURAATORIN PIIRTOTYÖKALU

Tekniikka ja liikenne
2012

VAASAN AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

TIIVISTELMÄ

Tekijä	Edmin Grbic
Opinnäytetyön nimi	Myyntikonfiguraattorin piirtotyökalu
Vuosi	2012
Kieli	suomi
Sivumäärä	43 +2 Liitettä
Ohjaaja	Ghodrat Moghadampour

Opinnäytetyö tehtiin vaasalaiselle firmalle nimeltä Wapice Oy, joka on keskittynyt teollisuusyritysten ohjelmistoratkaisuihin ja tietojärjestelmien integrointiin. Opinnäytetyönä tehtiin Summium 5 myyntikonfiguraattorille 2D(kaksiulotteinen)-piirtotyökalu, joka korvaa nykyisen 3D(kolmiulotteinen)-piirtotyökalun projekteissa, joissa vaaditaan nopeampaa toimintaa ja kevyempää ratkaisua. Summuim 5 myyntikonfiguraattori on myyjään tarjous-ja tilaustyökalu, jolla asiakkaille massaräätelöidään tuotteita asiakkaan vaatimusten ja valintojen perusteella.

2D-piirtotyökalu on sovellus, joka lukee Summiumista tulevan XML-tiedoston ja sen perusteella listaa kuvat ja niiden tiedot sivutyöpalkille. Sovelluksen tarkoitus on käyttää webpohjaista toteutusta, jossa kuvia raahataan sivupalkilta piirtoalustalle. Kuvia voi olla monta ja jokaista kuvaa voi raahata piirtoalustalle vain tietyn XML-tiedoston rajoittaman määrän verran. Sovelluksen myös pitää toimia kokonaan kosketusnäytöllä.

Valmis sovellus toteutti kaikki vaatimukset, jopa ylimääräisen vaatimuksen. Nämä vaatimukset ovat ohjelman integraatio Summiin, vedä ja pudota ominaisuus, kosketusnäyttöominaisuus, kuvien tiedot sivupalkilla kuvien vieressä, kuvien kopioinnin rajaus XML-tiedoston mukaisesti, kuvien koordinaatisto raahauksen aikana, kuvia ei voi raahata toistensa läpi vain ne törmäävät toisiinsa, kuvia pystyy pyörrittämään, zoom toiminta, piirtoalustan lopputulos tallenetaan png-kuvana ja viimeisenä ylimääräisenä vaatimuksena oli kuvien yhdistämis toiminta, jossa kuvat meneveät kiinni toisiinsa jos kuva tulee lähelle toista kuvaa.

Avainsanat	jQuery, KineticJS, HTML 5, HTML5 canvas,Vaadin

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Tietotekniikan koulutusohjelma

salesconhigurator

ABSTRACT

Author	Edmin Grbic
Title	Skeching tool for salesconfigurator
Year	2012
Language	Finnish
Pages	43 +2 Appendencies
Name of Supervisor	Ghodrat Moghadampour

This thesis was done for a company named Wapice Oy, which is focused on manufacturing software solutions and system integration. The goal of the thesis was to create a 2D (two-dimensional) sketching tool for Summium 5 sales configurator, which will replace current 3D (three-dimensional) sketching tool in processes that need tools, which are light and work fast. Summium 5 is a salesperson's offer and order tool for configuring products through customers requirements and sales selections.

2D sketching tool is a application that reads the XML file that comes from Summium and uses those information's to list images and their information to a side panel. The goal of the thesis is to use a web based application where images are dragged from the sidebar to the drawing surface. There can be many images on the sidebar and each image can be dragged to the drawing surface set amount of times which is defined by a XML file. The application needs to works perfectly on a touch screen.

The finished program fulfilled all the requirement set, event the bonus requirement. These requirements are program integration to Summium, drag and drop functions, touch screen capabilities, side panel holds the image and image information, images can be cloned the amount of times that they appear in XML file, image coordinates are shown when image is dragged, image collision checking and preventing passing through each other, image rotation, zoom the view, finished product can be saved as an image and the additional requirement was to merge the images when they come close to each other.

Keywords jQuery, KineticJS, HTML 5, HTML5 canvas,Vaadin

KÄYTETYT LYHENTEET

2D	2-dimensional, kaksiulotteinen
3D	3-dimmensional, kolmiulotteinen
JS	JavaScript, Netscape Communications Corporationin kehittämä web-ympäristössä käytettävä komentosarjakieli.
Vaadin	Vaadin on Suomessa kehitetty palvelinpohjainen Java-kehys, jota käytetään rikkaiden WWW- sivujen luomiseen.
jQuery	jQuery on kaikille selaimille tarkoitettu ilmainen JavaScript-kirjasto.
KineticJS	KineteticJs on HTML5 canvas komponentin javaScript kirjasto.
HTML5	HTMl5 on HTML-kuvauskielestä tehty 5 version.
DIV	on html-elementti joka toimii muiden html-elementtien alustana ja jakaa HTML-sivun sisällöt osioihin.
PNG	Portable Network Graphics on bittikarttaformaatti, joka kehitettiin GIF-formaatin korvaajaksi. Tässä projektissa ei ilmene muuten kuin kuvan tyyppinä.

SISÄLLYS

1 JOHDANTO

Summium 5 on uusin versio Summium-myyntikonfiguraattoreista. Summium ohjelmisto tukee massaräätälöivän yrityksen myyntiprosessia. Yritysten koko myynti- ja jälleenmyyntiorganisaatio voi tukeutua keskitettyyn järjestelmään, jossa on kaikki myynnin, tuotannon ja logistiikan tarvitsema tieto myytävistä tuotteista. Summium tarjoaa myyjälle työkalun, jolla hän voi helposti tarjota asiakkailleen juuri hänelle sopivaa tuotetta ja kannattavaa tuotetta. Valintojen perusteella Summium muodostaa tuotteelle hintatiedon, kaupallisen ja teknisen dokumentaation sekä valmistukseen tarvitseman tuoterakenteen reaaliajassa. /6/

3D eli kolmiulotteinen grafiikka on tietokonegrafiikka, joka on mallinnettu kolmen ulottuvuuden suhteen. 3D-piirtotyökalua ollaan käytetty tähän asti kaikissa Summium projekteissa ja tälle tarvittiin kevyempi ja nopeampi korvaaja.

2D-grafiikka on kaksiulotteinen tietokonegrafiikka, joka on mallinnettu kahden ulottuvuuden suhteen. 2D-grafiikka on kevyempi käsitellä ja sen ajamiseen ei koneelta vaadita paljon tehoja.

Nykyään monet sovellukset tarjoavat kosketusnäyttöominaisuuksia, joita ei 3D-piirtotyökalussa löydy. Kosketusnäyttöominaisuudet korvaavat hiiren ja näppäimistön jolloin voi vain käyttää omaa sormea ja kosketusnäyttöä eri tarkoituksiin ja toimintoihin.

Opinnäytetyön tarkoituksena on luoda HTML5 2D-webpohjainen piirtotyökalu tai toisella ohjelmointimenetelmällä samanlainen sovellus, joka toteuttaa vaaditut määritelmät. Sovellus on tarkoitettu soveltaa Summium 5-myynti-konfiguraattoriin, joka korvaa nykyisen 3D-sovelluksen vähemmän vaativissa projekteissa tai projekteissa, jotka vaativat kosketusnäyttöominaisuuksia. Sovellus toteutettiin 3:lla eri ohjelmointitavalla, mutta vain yhdellä tavalla tehtiin loppuun asti. Kaksi muuta toteutustapaa tehtiin vain niin pitkälle, että sovelluksia pystyttiin vertailemaan keskenään.

2 WAPICE OY

Wapice Oy on perustettu vuonna 1999 Vaasassa. Wapice työllistää yli 200 henkilöä 5:llä eri paikkakunnalla Suomessa. Nämä toimipaikat ovat Vaasa, Tampere, Hyvinkää, Seinäjoki ja Oulu. Vaasassa sijaitsee 2 toimipaikkaa, toinen on Palosaarella ja toinen Yrittäjäkadulla. Suurimaksi osaksi Wapice Oy:n omistavat sen työntekijät ja johto. /6/

Wapice Oy on keskittynyt teollisuusyritysten ohjelmistoratkaisuihin ja tietojärjestelmien integrointiin. Wapice tarjoaa yrityksille turvallisen ja tehokkaan tavan ulkoistaa tuotekehitystä ja kehittää yritysten liiketoimintaa. Wapice Oy räätälöi ratkaisuja tuotekehityksestä myynnin ohjelmistoihin sekä toiminnan- ja tuotannonohjauksesta jälkimarkkinointiin. /6/

Wapice Oy tarjoaa asiakkailleen ohjelmistoasiantuntijoitaan ja heidän tehokkaan verkoston. Laajan verkoston takia Wapice pystyy löytämään asiakkaille sopivia ratkaisuja. Asiakas voi huoletta siirtää koko vastuun Wapicelle ja keskittyä omaan ydinliiketoimintaan. /6/

Wapice tarjoaa asiakkailleen ohjelmistohankintaa ja palvelua aina sulautetuista järjestelmistä liiketoimintaratkaisuihin asti. Suurin osa Wapicen asiakkaista on globaalisti toimivia energia- ja koneenvalmistusteollisuuden yrityksiä. /6/

Wapicen ohjelmistoprojektin tavoite on tukea teollisuusyritysten liiketoimintaa. Wapicen asiantuntijat toimivat yhdessä asiakkaidensa kanssa ahkerasti, jotta asiakas saisi juuri hänelle sopivimman ratkaisun. Pitkä yhteistyö monien teollisuuden yritysten kanssa on tuonnut Wapicen työntekijöille vahvan ymmärryksen teollisuuden toiminnasta ja tämä auttaa heitä ratkaisemaan monimutkaisetkin vaatimukset tai ongelmat. /6/

3 KEHITYSTYÖKALUT

Tässä luvussa käydään läpi sovelluksessa käytettyä ohjelmistoympäristöä. Tämä luku käsittelee myös kaikki ohjelmointikielet ja kirjastot joita kolmessa eri toteutustavassa käytettiin.

3.1 Ohjelmistoympäristö

Ohjelmistoympäristönä on käytetty, Eclipse (3.7) Indigo, joka on kehitetty avoimen lähdekoodin lisenssillä. Eclipse tukee monia ohjelmistokieliä. Osa on ovat Java, CC, C++, PHP, HTML, JavaScript ja Vaadin. Ohjelmistoympäristöä on kehittänyt IBM vuodesta 1993 vuoteen 2001. Vuodesta 2004 ohjelmistoympäristöä on kehittänyt Eclipse-Foundation. Eclipse on toteutettu pääasiassa Java - ohjelmistokielellä, mutta se ei kuitenkaan käytä Javan kirjastoja, vaan sille on luotu oma kirjasto. Tämän johdosta Eclipse ei ole täysin alustariippumaton.

Eclipse valittiin projektissa sen takia, että se tukee kaikkia 3 tekniikoita, joita projektissa käytettiin.

3.2 HTML 5

HTML, eli Hyper Text Markup Language, on ohjelmistokieli, josta lähes jokainen verkkosivu koostuu. HTML5 on tämän kielen uusin versio, joka luultavasti standarisoidaan vuoteen 2020 mennessä. /4/

HTML5 tukee piirroksia, videoita ja äänitiedostoja paikallisesti, mikä tarkoittaa sitä, ettei kuvioiden, videoiden tai äänitiedostojen toistamiseen tai näyttämiseen tarvita mitään ulkoista selainlaajennusta tai pluginia. /4/

HTML5:n avulla selaimista on tullut monipuolisia työvälineitä. Eräs näistä uudistuksista on sellainen, jossa voidaan raahata työpöydältä tiedosto HTML5-alueelle ja se latautuu sinne. /4/

HTML5 tarjoaa myös monipuolisia kosketusnäyttöominaisuuksia, jotka mahdollistavat monien erityyppisten ohjelmistojen kehitystä. /4/

Canvas-elementti on osa HTML5:sta ja mahdollistaa dynaamisen skriptattavissa olevan käytön 2D-muodoille ja kuville. Canvas-elementti on animaatiotekijöille tai pelien tekijöille unelmakomponentti, koska sillä on helppoa toteuttaa ja ajaa sovelluksia.

Seuraavassa kuviossa esitellään miten HTML5 canvas-elementtiä käytetään HTML-sivussa ja miten canvasille piirretään yksinkertainen 2D-kuvio.

```
<canvas id="canvasID" width="400" height="200" style="border:1px ">
Your browser does not support the HTML5 canvas tag.
</canvas>
<script>
        var c=document.getElementById("canvasID ");
        var ctx=c.getContext("2d");
        ctx.fillStyle="#FF0000";
        ctx.fillRect(0,0,75,50);
</script>
```

Kuvio 1. HTML5 canvas-elementti.

Seuraavassa kuvio 2-osiossa näytetään miten HTML5-koodissa sivulle laitetaan video. Koodissa näkee, että videoiden laittaminen on tehty tosi helpoksi ja yksinkertaiseksi.

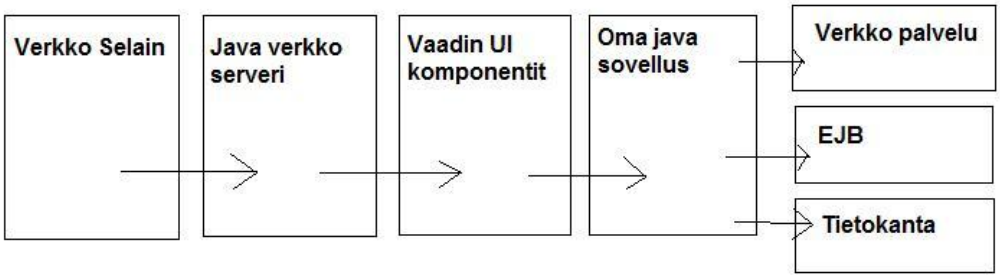
```
<video width="400" height="400" controls="controls">
 <source src="clip.mp4" type="video/mp4">
Your browser does not support the video tag.
</video>
```

Kuvio 2. HTML5 video-elementti

3.3 Vaadin

Ydinosa Vaadin framworkissa on Java-kirjasto, joka on suunniteltu niin, että on helppo luoda ja ylläpitää web-käyttöliittymiä. Vaadinin tärkein ominaisuus piilee siinä, että se sallii ohjelmoijan luoda web-käyttöliittymiä samalla tavalla kuin loisi normaaleja Java-työpöytäjärjestelmiä käyttämällä tavanomaisia työkaluja, kuten AWT, Swing tai SWT, mutta helpommin. /2/

Palvelimen kanssa toimivaan malliin perustuen, Vaadin huolehtii käyttöliittymien hallinnasta ja AJAX-kommunikaatiosta selaimen ja palvelimen välillä. Vaadinia käyttäen ei tarvitse opetella selaintekniikoita, kuten HTML tai JavaScript, vaan kaikki hoituu Java-koodilla. /2/



Kuva 1. Yleinen Vaadin arkkitehtuuri. /2/

Kuvassa 1 on Vaadinin yleinen arkkitehtuuri, joka kuvaa perusapplikaation toiminnan Vaadinilla. Vaadin koostuu palvelimen puolella toimivasta framworkista ja käyttäjäpuoleisesta moottorista, joka toimii selaimessa JS-ohjelmana. Selaimelle luodaan käyttöliittymä, joka lähettää käyttäjän vuorovaikutukset serverille. Koska Vaadin muuttaa kaiken Java-koodin JavaScript-koodiksi, selaimissa ei tarvitse mitään lisäohjelmia jotta Vaadin ohjelmiston voisi käynnistää. /2/

Vaadin erittelee selkeästi käyttöliittymän ulkonäön ja käyttöliittymän logiikan. Molempia voi kehittää erikseen käyttämällä teemoja, joilla määritellään sovelluksen ulkonäkö. Teemat hallitsevat sovelluksen ulkonäköä käyttämällä CSS-tyyppikieltä ja HTML-verkkosivumalleja. Vaadin tarjoaa erinomaisia

perusteemoja jotka usein toteuttavat kaikki sovelluksen ulkonäkövaatimukset, mutta niitä voi myös muuttaa ja muokata niin kuin haluaa./2/

Seuraavassa koodiesimerkissä näytetään miten Vaadin-sovellus käyttää CSS-tiedostoja sovelluksen ulkonäön muokkaamiseen.

```
.mystyle {
   font-style:  italic;
   font-size:   30px;
   font-weight: bolder;
   line-height: 35px;
}
```

Kuvio 3. Vaadin CSS-tyyliasetus.

Vaadinsovelluksessa CSS määrityksen asetus on koodissa helppoa. Kun CSS asetukselle on antanut nimen sitä nimeä voi kutsua tietylle komponentille jos haluaa vaihtaa vain sen komponentin ulkonäköä tai tekstiä. Arvoa voi myös soveltaa koko näkymälle määrittämällä näkymän tyyli ”setStyleName()”-komennolla. Seuraavassa koodiesimerkissä näytetään miten Vaadinissa komponentille asetetaan tyyliasetus.

```
Button btn = new Button();
btn.setStylename(myStyle);
```

Kuvio 4. Vaadin tyyliasetus.

3.4 JavaScript

JavaScript eli JS on Netscape Communicationsin kehittämä ohjelmistokieli, jota käytetään pääasiassa www-sivuilla. JS:llä voidaan tehdä sivuja jotka reagoivat käyttäjän toimenpiteisiin, kuten hiiren klikkaukseen, raahaamiseen tai näppäimistön syöttämiseen. JS:llä voidaan ohjelmoida erilaisia selaintoimintoja, jotka reagoivat käyttäjän toimintoihin. /3/

Ohjelmistokielenä JS on kuitenkin melko yksinkertainen eikä sen luomiseen tarvita mitään erikoista sovellusta, pelkkä tekstieditori riittää. JS on oliopohjainen asiakaspuolen skriptikieli, jonka avulla voidaan www-dokumentteihin lisätä monipuolisia toimintoja. /3/

JS tarjoaa kehittäjälle oman, sisäisen oliomallin, jonka ominaisuuksia ja menetelmiä käyttämällä kehittäjä voi hallita verkkosivua monipuolisesti. /3/

JS on ohjelmistokielenä melko yksinkertainen ja sille on tehty monia kirjastoja, jotka toteuttavat monia toimintoja ja tekevät JS kielestä helpon käsitellä ja käyttää. Seuraavassa koodiesimerkissä näytetään miten JS kirjastoja käytetään. Koodissa viitataan kineticJS kirjastoon ja kirjaston funktiolla luodaan Canvas piirtoalue.

```
<script type="text/javascript" src="js/kineticJS.js"></script>
<script>
window.onload = function () {
   var stage = new Kinetic.Stage({
      container: "container",
      width: 1800,
      height: 850
   });
```

Kuvio 5. JavaScript kirjastoesimerkki.

3.5 jQuery ja jQuery UI

jQuery on nopea ja tiivis JS-kirjasto, joka yksinkertaistaa HTML-dokumenttien toimintoja, tapahtumien käsittelyä, animaatiota ja Ajax-vuorovaikutusta nopeaa webkehitystä varten. jQuery on suunniteltu niin, että se muuttaa JS-koodin käyttötapaa. Kirjaston monipuolisuuden ja helppouden takia siitä on tullut maailman suosituin ja käytetyin JS-kirjasto./5/

jQuery on tehty yhteensopivaksi kirjastoksi, johon on helppo yhdistää muita kirjastoja ja usein se toimii muiden kirjastojen pohjana. Kirjasto tekee JS-ohjelmoinnista tuskattoman ja samalla mahdollistaa monipuolisia JS-sovelluksia. Tälle kirjastolle on myös tehty tuhansia lisäosia joita lisäämällä saadaan JS-koodi tekemään monia eri toimintoja, yksi näistä lisäosista on ”touchpunch”, joka muuttaa kosketusnäyttötoiminnat hiiritoiminnoiksi.

jQuery UI tarjoaa abstraktioita matalan tason vuorovaikutusta ja animaatioita, kehittyneitä efektejä ja korkeatasoisia widgettejä, jotka on rakennettu jQuery-kirjaston päälle. Tätä käyttäen pystytään luomaan korkeatasoisia vuorovaikuttavia websovelluksia. /5/

Käyttämällä jQuery UI kirjastoa pystytään tekemään todella monipuolisia ja nopeasti toimivia sovelluksia.

Kuviossa 6 näytetään miten hyödyllisiä jQuery-ja jQuery UI-kirjastot ovat. Koodiesimerkissä näytetään miten saadaan HTML div-elementin sisältö raahattavaksi ja miten kuvalle määritellään kloonaus ominaisuus raahauksen alkaessa. Lisäksi määritetään kuville yhdistymistoiminta jossa kuva yhdistyy toiseen kuvaan kun se raahataan riittävän lähelle.

```
<!DOCTYPE html>

<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>

<script type="text/javascript" src="js/jquery-1.7.2.js"></script>
<script type="text/javascript" src="js/jquery-ui-1.8.17.custom.min.js"></script>
<script type="text/javascript">
		$('.drag_drop_img').draggable({
				revert : 'invalid',
				helper : 'clone',
				snap:true,
				snapMode: "outer",
				snapTolerance: 20,
				stop : function(event, ui) {

		$(this).draggable('option', 'revert','invalid');
				}
		});
```

Kuvio 6. jQuery ja jQuery UI kirjastojen raahausesimerkki.

Kuviossa 6 näytetyillä JS-kirjastoilla pystyy tekemään paljon erilaisia toimintoja, joita verkkosivulla aina tarvitaan tai vaaditaan. Nämä kirjastot tekevät ohjelmistonsuunnittelijoille JS-ohjelmoinnin helpoksi ja samalla tiivistävät koodia, tehden sovelluksesta pienemmän ja nopeamman.

3.6 KineticJS

KineticJS on HTML5 canvas JS-kirjasto, joka laajentaa 2D-kontekstia mahdollistamalla polkutunnistuksen ja pikselitunnistuksen työpöytä - ja mobiilisovelluksissa. /1/

KineticJS-kirjasto sisältää monia hyviä esimerkkejä ja käytänntöllisiä toimintoja, jotka helpottavat kuvien käsittelyä HTML5-canvasilla. Tämä kirjasto mahdollistaa kuvioiden piirtämistä canvasille ja siten sallii eventien asettelun eri kuvioille. Kuvioita voi joko raahata, suurentaa, pienentää ja kuvioilla voi tehdä jopa erilaisia animaatioita. /1/

KineticJS vaiheet muodostuvat käyttäjän määriteltävistä kerroksista. Jokaisella kerroksella on 2 kontekstia tapahtuma konteksti ja buffer konteksti. Tapahtumakonteksti on näkymä, jonka pystyy näkemään ja bufferi konteksti on piilonäkymä joka kuuntelee tapahtumia. KineticJS koodia käytetään monenlaisissa verkkosivu projekteissa ja sen animaatiotoiminnat ja helposti muutettavat asetukset tekevät siitä monien pienten verkkopeli kehittäjien suosikin.

Seuraavassa koodiesimerkissä nähdään miten kineticJS otetaan käyttöön ja miten raahaaminen canvas-elementillä otetaan käyttöön.

```
<script src="http://www.html5canvastutorials.com/libraries/kinetic-v4.0.0.js"></script>
   <script>
          window.onload = function() {
          var stage = new Kinetic.Stage({
          container: "container",
           width: 578,
           height: 200
           });
      var layer = new Kinetic.Layer();
      var rectX = stage.getWidth() / 2 - 50;
      var rectY = stage.getHeight() / 2 - 25;

      var box = new Kinetic.Rect({
        x: rectX,
        y: rectY,
        width: 100,
        height: 50,
        draggable: true
      });
```

Kuvio 7. KineticJS kuvion raahausasetukset ja piirtäminen.

4 VAATIMUSMÄÄRITTELY

Tässä luvussa esitellään piirottyökalun vaatimukset ja käydään läpi QDF-kaaviot.

4.1 Yleiskuvaus

Opinnäytetyön tarkoituksena on kehittää 2D-piirtotyökalu Summium myyntikonfiguraattorille. Piirtotyökalun tarkoitus on kuvien siirteleminen, vedä- ja pudotatoiminnalla. Kuvien tieto saadaan XML-tiedostosta, joka luodaan kun Summiumissa valitaan halutut kuvat ja niiden määrät. Sovellus lukee XML-tiedoston ja piirtää valitut kuvat sivutyöpalkille, jossa kuvien kloonausmäärä rajataan ja kuvia voi vetää piirtoalustalle. Piirtoalustalla kuvien koordinaatisto näytetään kun kuvia vedetään piirtoalustalla. Piirtoalustalla kuvat eivät saa mennä toistensa läpi vaan ne törmäävät toisiinsa.

Opinnäytetyön lopputuotteena syntyvä sovellus toimii useimmilla web-palvelimilla. Sovellus käynnistetään Summium 5:stä valitsemalla halutut kuvat ja niiden määrä. Summium 5 luo XML-tiedoston, jonka sovellus lukee ja luo kuvalistan sivupalkille. Sovellus toimii 2D-sovelluksena, joka korvaa nykyisen Summium 5 3D-sovelluksen projekteissa joissa ei tarvita 3D-ominaisuuksia. Sovellus sisältää toimintoja, jotka toimivat samalla tavalla kosketusnäytöllä kuin normaalilla tietokoneella.

4.2 Toimintavaatimukset

Määrittelyn perustana on käytetty tehtävänantoa ja QFD-menetelmää. OFD-menetelmän tarkoitus on maksimoida asiakkaan tyytyväisyys keräämällä vaatimukset kolmeen eri kategoriaan.

Ensimmäinen kategoria määrittää mitkä toiminnot ohjelmassa on pakko olla. Toisessa kategoriassa luetellaan sellaiset vaatimukset joita ohjelmassa pitäisi olla. Kolmas kategoria käsittelee vaatimuksia joita sovelluksessa ei tarvitse olla, mutta jos niitä on, ne tekevät sovelluksesta paremman.

Seuraavassa taulukossa näytetään vaatimukset eroteltuna QFD-menetelmän mukaan.

Pakolliset (Must have)	**Pitäisi olla (Should have)**	**Hienoa jos olisi(Nice to have)**
• Kosketusnäyttö-ominaisuus. • Summiumiin integroitava. • Lukee tiedot XML-tiedostosta. • Vedä- ja pudota ominaisuus.	• Kloonaus määrän rajaus. • Kuvat ja niiden tiedot sivupalkille. • Kuvien pyöritys. • Lopputulos tallenetaan kuvana. • Kuvat törmäävät toisiinsa raahauksen aikana.	• Kuvien yhteen-liittyminen. • Piirtoalustan zoom toiminto.

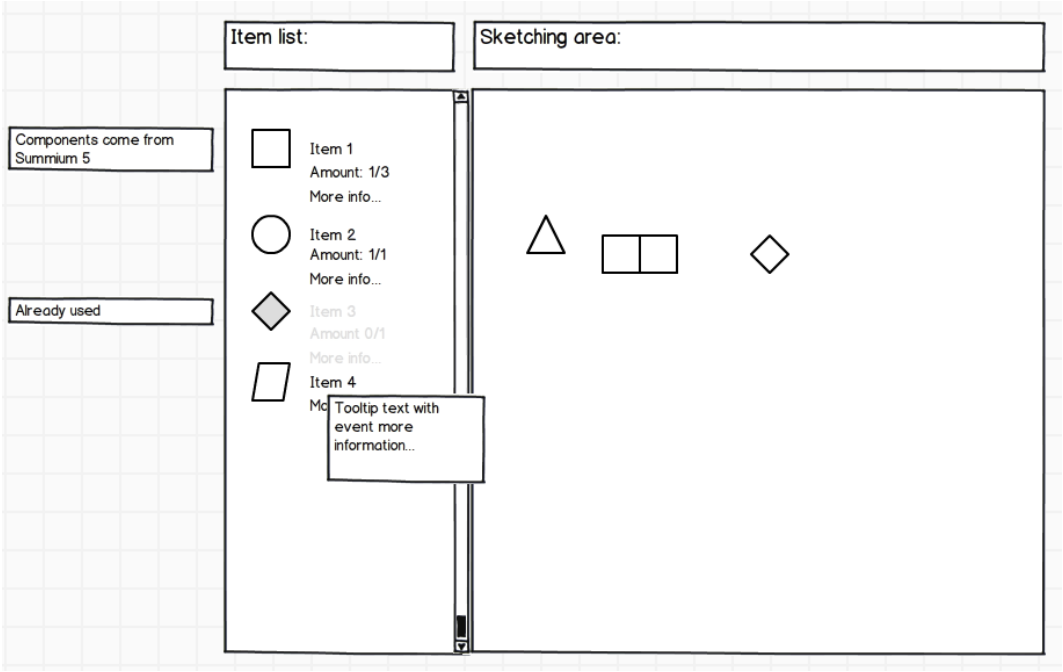
Taulukko 1. Vaatimukset eritelty QFD-menetelmää käyttäen.

Sovelluksen kuvien listauksessa on vaatimuksena, että komponentit tulevat Summiumista. Seuraava listan vaatimus on, kun kuvia on enemmän mitä sivupalkille mahtuu, sivupalkille ilmestyy vierityspalkki ja näin kuvia pystytään laittamaan niin paljon kuin haluaa. Listan viimeinen vaatimus on se, että listalta ei voi kopioida kuvia kuin vain tietyn määrän, jonka jälkeen kuva muuttuu läpinäkyväksi.

4.3 Yhteensopivuusvaatimukset

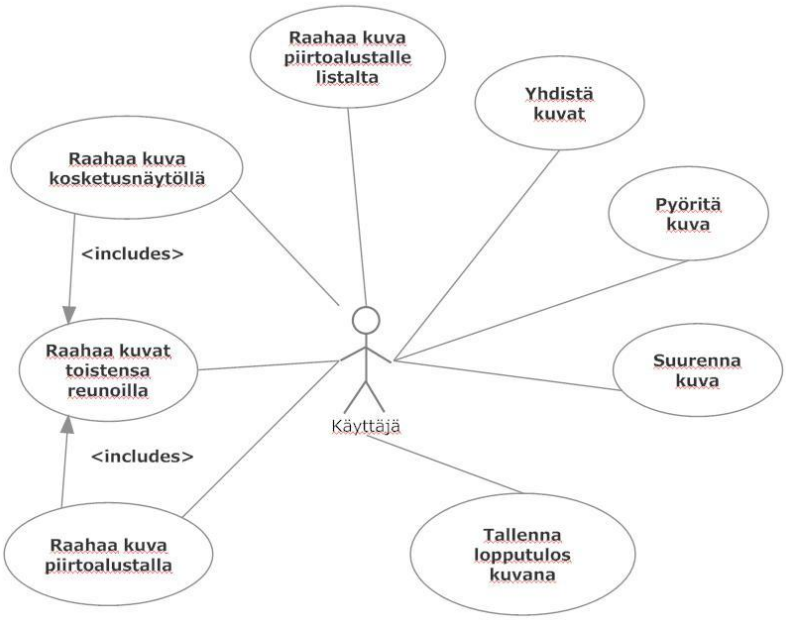
Sovellus integroidaan Vaadin framworkkiin (Summiumiin), ja koska sovellus on webpohjainen, sen liittäminen Summiumiin pitää olla helppoa ja yksinkertaista. Kuvia pitää pystyä lisäämään niin paljon kun haluaa ja kaikki kuvat ovat kansiossa josta kuvat haetaan XML-tiedoston avulla.

Kuvassa 2 näytetään millainen sovelluksen on oltava ja mitä tietoja kuvista löytyy sivupalkilla.



Kuva 2. Piirtotyökalun näkymä.

Sovelluksessa on käytetty prosessin määrittelyssä käyttötapakaaviota. Sovelluksessa on 1 käyttäjä, joka käyttää piirtotyökalua niin kuin itse tarvitsee. Kuvassa 3 on käyttötapakaavio, joka näyttää sovelluksen perusrakenteen.



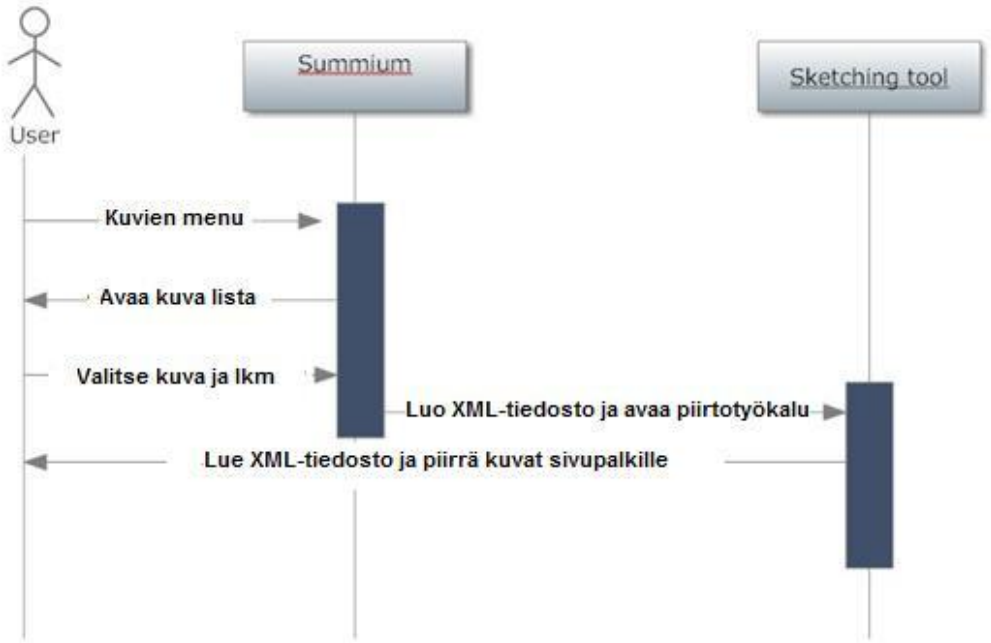
Kuva 3. Käyttötapakaavio.

4.4 Sovelluksen suunnittelu

Ohjelman vaiheet on mietitty sekvenssikaaviota käyttäen. Sekvenssikaaviossa määritellään mitä eri vaiheita ohjelma käy läpi, kun käyttäjä tekee jotain.

Kuvassa 4 on sekvenssikaavio, joka näyttää miten ohjelma toimii kun se on laitettu Summiumiin ja mitä siellä tehdään, jotta sovellus käynnistyisi.

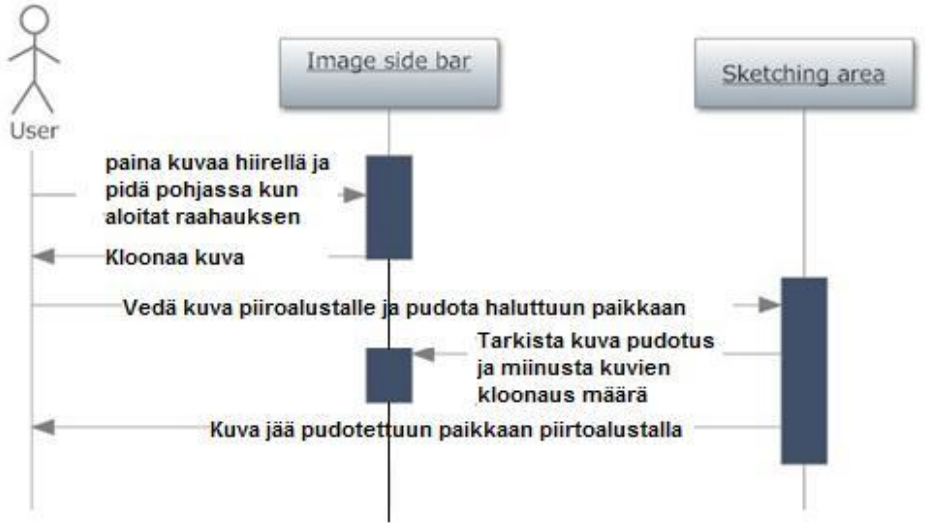
Alussa käyttäjä valitsee kuvat valikosta ja määrittää montako kuvaa käytetään. Kun kaikki kuvat on valittu painetaan nappia jolloin Summium luo XML-tiedoston ja avaa piirtotyökalun, joka lukee XML-tiedoston.



Kuva 4. Käynistyksen vaiheet.

Kuvassa 5 on sekvenssikaavio jossa näytetään mitä välivaiheita ohjelma käy kun kuva kloonataan ja vedetään sivupalkilta piirtoalustalle.

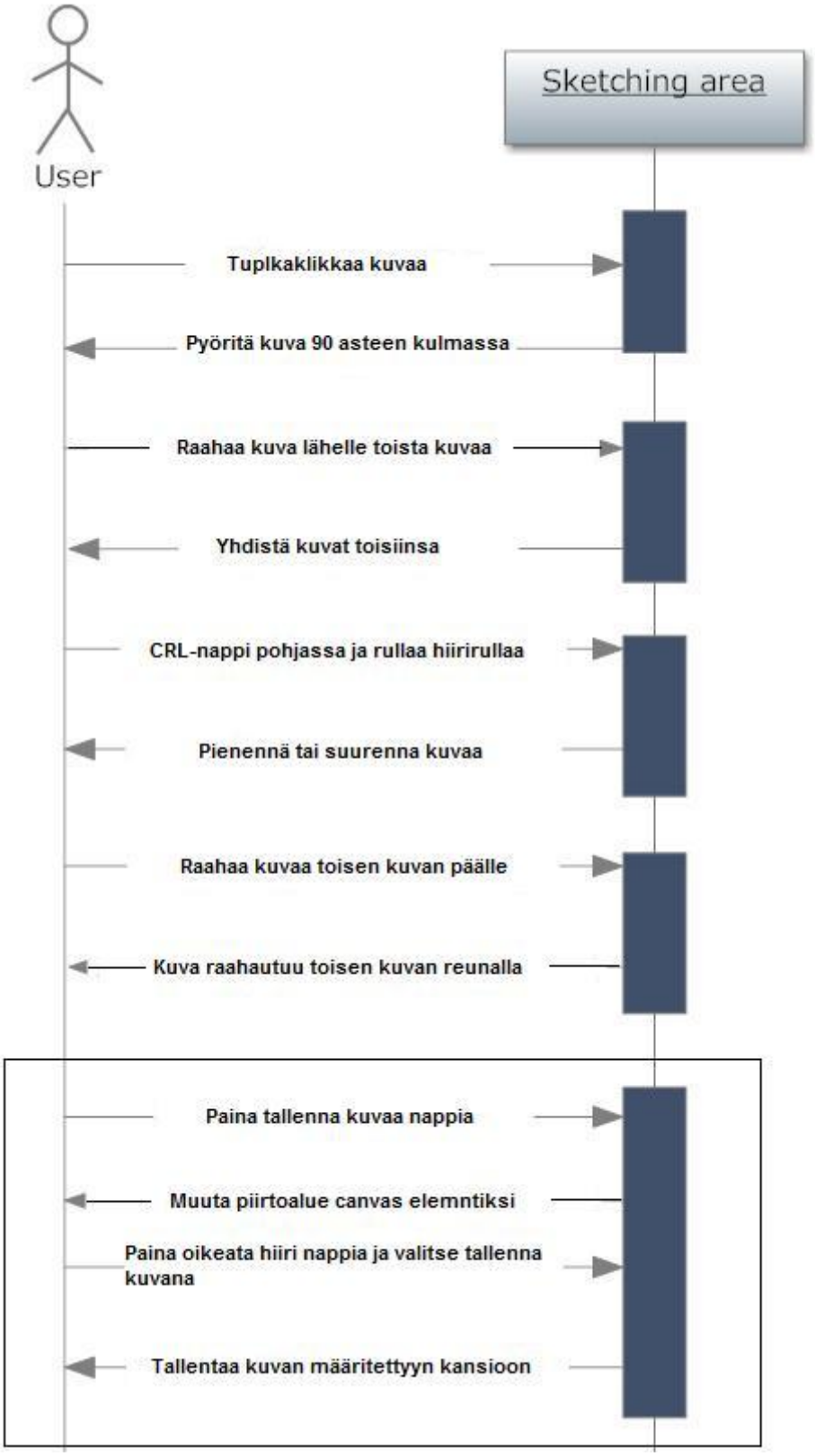
Käyttävä valitsee hiirellä kuvan ja pitää hiirtä pohjalla sillä aikaa kun raahaa kuvan piirtoalustalle. Raahattuaan kuvan piirtoalustalle käyttäjä voi pudottaa kuvan tyhjään paikkaan ei kuitenkaan toisten kuvien päälle.



Kuva 5. Kloonaustoimenpiteen vaiheet.

Kuvassa 6 on sekvenssikaavio, jossa yksinkertaisesti käydään läpi eri toimintojen vaiheet. Kaikki toimenpiteet ovat vaadittuja ja toimivat hyvin ilman että käyttäjältä vaaditaan paljon toimintaa.

Sekvenssikaaviossa on näytetty miten yksinkertaisilla toiminnoilla saadaan eri toimintoja kuville tehtyä.



Kuva 6. Sovelluksen eri toiminnot.

5 TOTEUTUS

Tässä osiossa käydään läpi miten sovellusta aloitettiin tekemään ja mitä vaiheita sovelluksen luonnissa oli.

Sovellus on tehty 3:lla eri tavalla, joista vain 1 toteutustapa vietiin loppuun asti. Loput 2 ohjelmointitapaa tehtiin niin pitkälle, että pystyttiin vertailemaan sovelluksia ja niiden ominaisuuksia keskenään. Tässä esitetään kaikki 3 toteutustapaa, joista vain loppuun asti tehty sovellus esitellään tarkasti.

Sovellus on 2D-piirtotyökalu, joka toimii Summium 5 myyntikonfiguraattorin työkaluna projekteissa, jossa vaaditaan nopeutta ja kosketusnäyttöominaisuutta. Sovellus korvaa nykyisen 3D-piirtotyökalun projekteissa, joissa ei vaadita 2D-ominaisuutta.

5.1 jQuery toteutus

Tämä on ensimmäinen ohjelmointitapa jota käytettiin sovelluksen luontiin ja ainoa tapa jolla sovellus vietiin loppuun asti.

Ohjelmassa edettiin vaatimus kerrallaan. Alussa oli tärkeintä saada tietää miten kuvaa voi raahata kahden div -elementin välillä. Pitkän tutkimisen jälkeen tultiin siihen tulokseen, että jQuery JS-kirjasto toi tähän ongelmaan parhaimman ratkaisun ja tämä oli clone-komento . Tämä komento kloonaa kuvan ja sen pystyy raahaamaan piirtoalustalle joka on määritetty pudotuslohkoksi. Tämä ominaisuus näkyy kuvioissa 8, 9 ja 10.

Seuraavassa koodiesimerkissä määritetään 4 div lohkoa, joista tärkeimmät ovat lohkot joille annetaan id arvoksi ”drag_drop_list” ja ”drag_drop_container”. Kuten nimistä huomaa lohko, jonka id on ”drag_drop_list”, sisältää kaikki kuvat ja niiden määrätiedot, kun taas drag_drop_container” on piirtoalusta jolle kuvat raahataan ja jolla käsitellään kuvia halutulla tavalla.

```
<div id="frame">
	<table>
		<tr>
			<div id="coord" style="margin-left: 10px;"></div>
		</tr>
		<tr>
			<div id="drag_drop_list"></div>
			<div id="drag_drop_container"></div>
		</tr>
		<tr>
		<div id="extra">
			<button type="button" onclick="rotateall()">Rotate list 	images!
			</button>
			<button type="button" onclick="takePic()">Take a picture!
			</button>
		</div>
		</tr>
	</table>
</div>
```

Kuvio 8. Sivupalkin ja piirtoalustan määritys.

Seuraavassa koodiesimerkissä näytetään miten sivupalkin kuville asetettiin eri ominaisuuksia, joista tärkein ominaisuus oli raahausominaisuus ja clone-ominaisuus.

```
$('.drag_drop_img').draggable({
	revert : 'invalid',
	helper : 'clone',
	snap:true,
	snapMode: "outer",
	snapTolerance: 20,
	stop : function(event, ui) {
	$(this).draggable('option', 'revert','invalid');
	 }
});
```

Kuvio 9. Raahauksen enablointi.

Seuraavassa koodiesimerkissä näytetään miten piirtoalusta määritetään vastaanottajalohkoksi ja miten hoidetaan pudotustoiminta piirtoalustan sisällä. Kuvan asetusta piirtoalustalle rajoitetaan sallimalla vain kuvan pudotuksen, jos kuva on kokonaan raahattu piirtoalustan ylle. Sen jälkeen määritetään mitä tapahtuu kun kuva pudotetaan piirtoalustan yllä. Kuvasta tehdään klooni ja se asetetaan juuri siihen kohtaan mihin se on pudotettu.

```
$('#drag_drop_container').droppable({
        accept : '.drag_drop_img',
        tolerance : 'fit',
        drop : function(event, ui) {
        var newitem = $(ui.draggable).clone();
        $(this).append(newitem);
        $(newitem).addClass("drag_drop_img_clone");
        $(newitem).css('position', 'absolute');
        $(newitem).css('top', ui.position.top);
        $(newitem).css('left', ui.position.left);
        $(newitem).removeClass("ui-draggable drag_drop_img");
```

Kuvio 10. Piirtoalustan asetus.

Saatua kuvat kloonattua ja raahattua sivutyöpalkista piirtoalustalle ryhdyttiin miettimään miten saadaan kuvat piirrettyä piirtoalustalle ja miten lasketaan kuinka monta kloonausta kuvasta saa tehdä. Tämä vaihe oli haasteellinen, mutta ratkesi ajan kuluessa kun huomattiin miten helppoa se on toteuttaa käyttämällä jQuery-ominaisuuksia. Kuvioissa 11, 12 ja 13 esitellään XML-tiedosto ja sitä lukevat koodiosat.

Seuraavassa koodiesimerkissä näytetään millainen on XML-tiedosto ja sen antamat tiedot. Tiedostossa on lueteltu useita muotoja ja kuvien tietoja. XML-tiedostosta tälle toteutustavalle tarvittiin vain nimi, loput tiedot olivat käytössä toisilla toteutustavalla.

```
<lines>
	<line>
		<piece>
			<catalogid>RedBall</catalogid>
			<pieceid>1</pieceid>
			<radius>40</radius>
			<sides>0</sides>
			<color>red</color>
			<height>0</height>
			<width>0</width>
		</piece>
		<piece>
			<catalogid>YellowStar</catalogid>
			<pieceid>2</pieceid>
			<radius>40</radius>
			<sides>6</sides>
			<color>yellow</color>
			<height>0</height>
			<width>0</width>
		</piece>
```

Kuvio 11. XML-tiedosto.

Seuraavassa koodiesimerkissä käytetään jQueryn ajax-komentoa XML- tiedoston käsittelyyn. Ensiksi tiedosto yritetään avata, jonka jälkeen tarkistetaan löytyykö kyseistä tiedostoa, jos löytyy luetaan tiedostosta kuvien nimet ja pistetään ne arraylistaan. Jos tiedostoa ei löydy, tulostetaan virheteksti. Kun kuvien lista on saatu XML-tiedostosta lähetetään lista laskemisfunktioon, joka laskee montako kertaa tietty nimi toistuu XML-tiedostossa.

```
var Images=new Array();
var u = 0;

//Gets the xml file
var request = $.ajax({
        type: "GET",
        url: "ImageNames.xml",
        dataType: "xml",
        });
//If the xml file is not found throws and alert
request.fail(function(jqXHR, textStatus) {
alert( "Xml file was not found check if the file is in the right folder" );
});
//If the file is found gets the names of the images
request.done(function(xml) {
        $(xml).find("piece").each(function(){
                u=u+1;
                var Name =  $(this).find('catalogid').text();
                Images.push(Name);
                });

var getArrays = GetImageNamesAndAmount(Images);
var arrayNames =getArrays.array1;
var arrayAmounts =getArrays.array2;
DragAndDrop(arrayNames,arrayAmounts);
});
```

Kuvio 12. XML- tiedoston avaaminen JS-koodissa.

Seuraavassa koodiesimerkissä näytetään miten funktio vastaanottaa nimilistan ja poistaa toistuvat nimet listalta ja samalla laskee toistuvien nimien määrän. Tämä tapa laskea kuvia helpotti paljon kuvien määrän laskemista.

Kun kuvien nimet on eroteltu ja niiden määrät laskettu, ne palautetaan kahtena eri array-listana, jonka jälkeen nämä listat lähetetään piirtämisfunktioon.

```
function GetImageNamesAndAmount(Images) {
var NameArray =new Array();
var AmountArray =new Array();
//Copy the original array
var copyArray = Images.slice(0);
// Loop all elements
for (var i = 0; i < Images.length; i++) {
        duplicates = 0;
        // Loop all the events that have been copied
        for (var j = 0; j < copyArray.length; j++) {
                if (Images[i] == copyArray[j]) {
                        // Duplicate amount
                        duplicates++;
                        // sets item to undefined
                        delete copyArray[j];
}}
if (duplicates > 0) {
        var nimi;
        var maara;
        nimi = Images[i];
        maara = duplicates;
        NameArray.push(nimi);
        AmountArray.push(maara);
}
}
```

Kuvio 13. Nimilistan toistuvien nimien erottelu ja määrän laskeminen.

Seuraavassa koodiesimerkissä näytetään miten tiedostot asetetaan taulukkoon ja sen jälkeen lähetetään div-elementtiin.

Kuvat rajoitetaan tiettyyn kloonausmäärään ja kloonauksen tapahtuessa teksti, jossa ilmaistaan käytettävien komponenttien määrä, pienenee heti.

```
//Create a table in table a table body in table bodey multiple td elemenst and for each td
//element 2 tr elements.
for (i = 0; i < ImageNames.length; i++) {
        var row = document.createElement("tr");
        var tdID=0;
        for (j = 0; j < 2; j++) {
                var td = document.createElement("td");
                td.setAttribute('id',ImageNames[i]+tdID);
                tdID=tdID+1;
                var pre = document.createElement("pre");
                var info = document.createTextNode("  item: " + i + "\n
                 Amount: "+ImageAmounts[i]+"/"+ImageAmounts[i]);

//Image is only added on the first td of the line. Also each items get the needed attributes
here
                if(j==0){
                var img = document.createElement("IMG");
                img.src = "images/"+ImageNames[i]+".png";
                img.setAttribute('class', 'drag_drop_img');
                img.setAttribute('onmouseover', 'resize(this)');
                img.setAttribute('ondblclick', 'rotateme(this)');
                img.setAttribute('title', ImageNames[i] + '.png');
                img.setAttribute('data-rotate', '0');
                img.setAttribute('amount', '0');
                td.appendChild(img);
                }
//Text is added to the second td of the line
                row.appendChild(td);
        }
pre.appendChild(info);
td.appendChild(pre);
tblB.appendChild(row);
}
```

Kuvio 14. Tiedoston asettelu taulukkoon.

Kun kuvat saatiin piirtämään sivupalkille, aloitettiin miettimään miten kuvat rajataan ja miten saadaan kloonaustoiminta pysäytettyä tietyn kloonausmäärän jälkeen.

Tämä osio oli yksi haastavimmista osioista tässä toteutuksessa. Tämä ongelma ratkesi vasta kun huomasimme, että kuvalle pystyy lisäämään uusia attribuutteja.

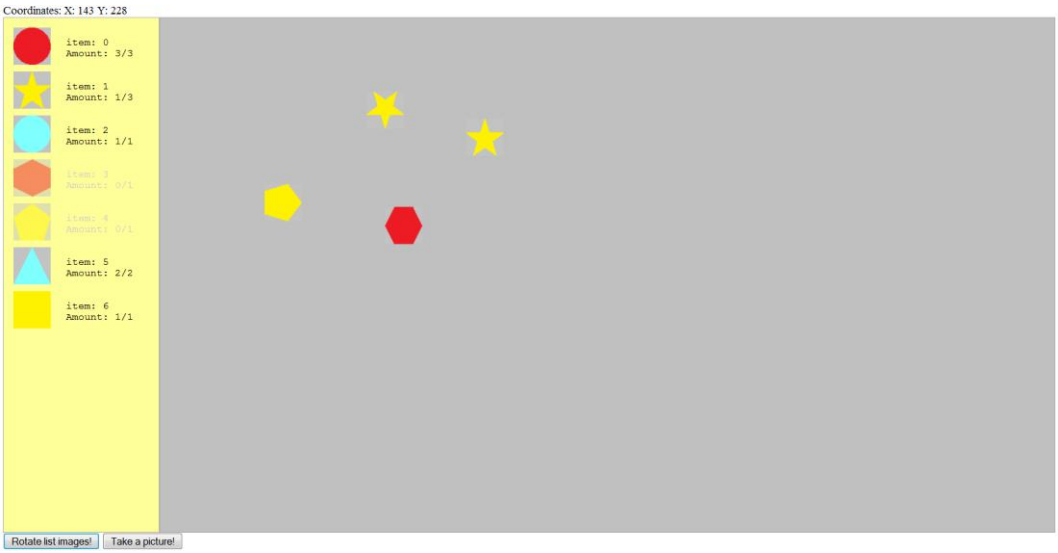
Kuvan raahauksen estäminen oli helppoa käyttämällä jQuery-kirjastoa apuna. Kuvalle pystyi poistamaan raahausominaisuudet ”removeclass”-komennolla. Sen jälkeen jäljelle jäänyt kuva ja teksti muuttuivat läpinäkyväksi.

Seuraavassa koodiesimerkissä näytetään miten tämä ongelma ratkaistiin.

```
for(t=0; t<ImageNames.length; t++){
        if (getName($(ui.draggable).attr('src')) ==ImageNames[t] ){
                var amount = (parseInt($(ui.draggable).attr("amount"))+1);
                $(ui.draggable).attr("amount",amount);
                var pre = document.createElement("pre");
                var td = document.getElementById(ImageNames[t]+"1");
                var tdText = td.childNodes.item(0);
                td.removeChild(tdText);
                var value = ImageAmounts[t]-amount;
                var newtxt = document.createTextNode("  item: " + t + "\n  Amount: "
                +value+"/"+ImageAmounts[t]);
                pre.appendChild(newtxt);
                td.appendChild(pre);
                if (amount==ImageAmounts[t]) {
                        $(ui.draggable).removeClass("ui-draggable drag_drop_img");
                        $(ui.draggable).css('opacity', .5);
                        var font = document.createElement("font");
                        font.style.color="#CCCCCC";
                        font.appendChild(newtxt);
                        pre.appendChild(font);
                        .appendChild(pre);
                }
        }}
```

Kuvio 15. Kloonauksen rajaus ja toteutus.

Seuraavat vaiheet olivat kuvien eri toiminnot. Jokainen näistä toiminnoista vei aikaa testaamiseen ja muuhun koodiin yhdistämiseen. Kun kaikki ominaisuudet oli saatu toimimaan, lopputulos näkyy kuvassa 7.



Kuva 7. Sovelluksen lopputulos.

Kuvassa 7 on ensimmäisen toteutuksen lopputulos. Sovelluksella on sen alakulmassa 2 nappia. 1 nappi on tarkoitettu kosketusnäyttönapiksi jolla voi pyörittää kaikkia listan kuvia. Hiirtä käytettäessä riittää, että tuplaklikkaa kuvaa jolloin se pyörähtää, mutta ei kosketusnäytössä toimi, koska tuplaklikkaus siellä zoomaa näyttöä.

Sovelluksessa on käytetty seuraavia kirjastoja, jotka ovat helpottaneet sovelluksen kehitystä paljon.

```
<script type="text/javascript" src="js/jquery-1.7.2.js"></script>
<script type="text/javascript" src="js/jquery-ui-1.8.17.custom.min.js"></script>
<script type="text/javascript" src="js/JQueryRotate.js"></script>
<script type="text/javascript" src="js/html2canvas.min.js"></script>
<script type="text/javascript" src="js/jquery.plugin.html2canvas.js"></script>
<script type="text/javascript" src="js/jquery-collide.js"></script>
<script type="text/javascript" src="js/jquery.ui.touch-punch.js"></script>
```

Kuvio 16. JS-kirjstot.

Kaikki käytetyt kirjastot helpottivat sovelluksen kehitystä paljon. jQuery-kirjastot olivat sovelluksen ydin, joiden päälle kaikki rakentui. Rotate-kirjasto hoiti kuvien pyörityksen ja samalla koordinaatistokorjauksen, mikä johtuu kuvien pyörittelystä. Htm2Canvas-kirjastot muuttavat piirtoalustan canvas-muotoon, jonka avulla voidaan tallentaa lohko kuvana. Touch-punch-kirjasto muuttaa kaikki sormella tehtävät liikkeet hiiriliikkeiksi, jonka avulla kaikki ominaisuudet toimivat kosketusnäytöllä. Viimeiseksi Colide-kirjasto hoiti kuvien törmäykset ja esti kuvien toistensa läpimenemisen.

Kuva 8. -elementti muutettuna <canvas>-elementiksi.

5.2 KineticJS-ja canvas-toteutus

Canvas-elementtisovellus oli melko pitkälle samanlainen kuin ensimmäinen sovellus. Käyttämällä KineticJS-kirjastoa monet ominaisuudet oli helppo toteuttaa. Ainoa ongelma mitä canvas-elemntille jäi ratkaisematta oli kopioiminen yhdestä canvas-elementistä toiseen. Tämä sovellus käydään vain pintapuolisesti läpi, koska se oli vain vertailukohde, jonka olisi voinut tehdä loppuun asti ajan salliessa ja siinä on monia samanlaisia toimintoja kuin edellisessä toteutuksessa.

Tätä toteutusta ei suoritettu kuvilla vaan kuvioilla, jotka kineticJS piirtää. Tämä päätös toi sellaisia ongelmia, että jokaiselle kuviolle piti laskea reunapisteet, jotta törmäystoiminta toimisi hienosti. Tämä johti siihen tulokseen, että raahaus pätki liian moneen pisteen takia. Jos tämä olisi toteutettu kuvilla, pisteitä ei olisi tarvinnut laskea, jolloin raahaus olisi voinut toimia yhtä hyvin kun ensimmäisessä toteutuksessa.

Kuvioissa 17 ja 18 on vain yksi pieni osa laskuista joita käytettiin jokaista muotoa varten, jotta törmäys tapahtuisi vain kun kuviot osuvat toisiinsa. Tämän canvas-törmäyksen toteuttaminen vei paljon aikaa varsinkin näiden laskujen takia. Laskut mittaavat tietyn kuvion kaikki reunapisteet joiden avulla voi verrata milloin kuvat törmäävät toisiinsa.

```
if (NameList[k].indexOf("Star") != -1) {

        var xP = XList[k];
        var yP = YList[k];
        IntersectPointX.push(Math.round(xP));
        IntersectPointY.push(Math.round(yP));
        var tX = xP;
        var tY = yP - 40;
        var rtX = xP + (60 / Math.sqrt(3));
        var rtY = yP - 20;
        var ltX = xP - (60 / Math.sqrt(3));
        var ltY = yP - 20;
        var rmX = xP + 20;
        var rmY = yP;
        var lmX = xP - 20;
        var lmY = yP;
        var rbX = xP + (60 / Math.sqrt(3));
        var rbY = yP + 20;
        var lbX = xP - (60 / Math.sqrt(3));
        var lbY = yP + 20;
        var bX = xP;
        var bY = yP + 40;
        var trX = xP + 20;
        var trY = yP - 20;
```

Kuvio 17. Tähtikuvion pisteiden laskemista.

```
for (var i = 0; i <= 1; i = i + 0.1) {
        var trlineX = trX + i * (tX - trX);
        var trlineY = trY + i * (tY - trY);
        IntersectPointX.push(Math.round(trlineX));
        IntersectPointY.push(Math.round(trlineY));

        var tllineX = tlX + i * (tX - tlX);
        var tllineY = tlY + i * (tY - tlY);
        IntersectPointX.push(Math.round(tllineX));
        IntersectPointY.push(Math.round(tllineY));

        var trblineX = trX + i * (rtX - trX);
        var trblineY = trY + i * (rtY - trY);
        IntersectPointX.push(Math.round(trblineX));
        IntersectPointY.push(Math.round(trblineY));

        var tlblineX = ltX + i * (tlX - ltX);
        var tlblineY = ltY + i * (tlY - ltY);
        IntersectPointX.push(Math.round(tlblineX));
        IntersectPointY.push(Math.round(tlblineY));

        var mrtlineX = rmX + i * (rtX - rmX);
        var mrtlineY = rmY + i * (rtY - rmY);
        IntersectPointX.push(Math.round(mrtlineX));
        IntersectPointY.push(Math.round(mrtlineY));

        var mltlineX = lmX + i * (ltX - lmX);
        var mltlineY = lmY + i * (ltY - lmY);
        IntersectPointX.push(Math.round(mltlineX));
        IntersectPointY.push(Math.round(mltlineY));

        var mrblineX = rmX + i * (rbX - rmX);
        var mrblineY = rmY + i * (rbY - rmY);
        IntersectPointX.push(Math.round(mrblineX));
        IntersectPointY.push(Math.round(mrblineY));
```

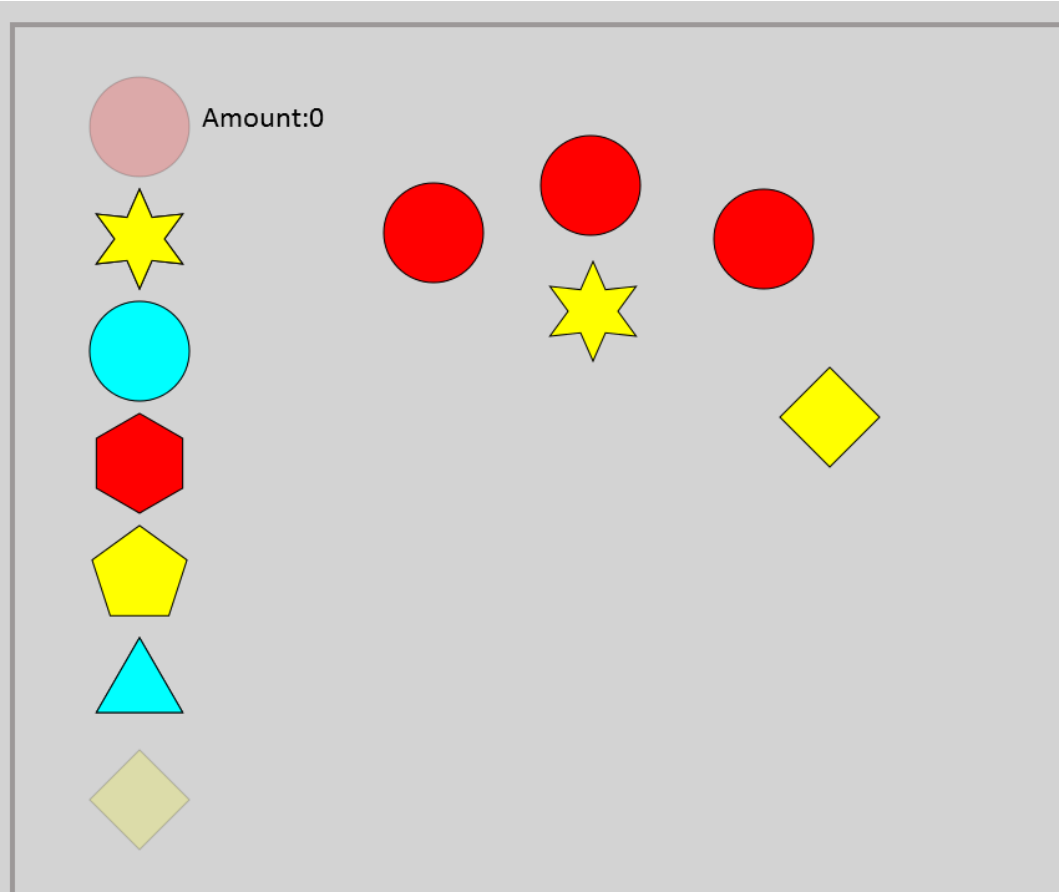
Kuvio 18. Tähtikuvion törmäystarkistus.

```
name.on("dblclick dbltap", function () {
    //layer.remove(name);
    // layer.remove(this);
    this.rotateDeg(45);
    Xpos = this.getX();
    Ypos = this.getY();
    //  alert("Names:" + NameList + " X amount:" + IntersectPointX.length);
    // alert("X: "+IntersectPointX+" Y: "+IntersectPointY);
    layer.draw();
});

 name.on("dragend touchend", function (evt) {
           kikka.clear();
         //resets the X and Y points in the array so that they could be updated
         IntersectPointX.length = 0;
         IntersectPointY.length = 0;

         //Cheks if there is any info in NameList
         if (NameList.length == 0) {
         NameList.push(this.getName());
         XList.push(this.getX());
         YList.push(this.getY());
  }
```

Kuvio 19. Kuvioiden raahausasetukset.

Kuviossa 19 näkee miten helppoa canvasilla on asettaa eri raahaustoiminnat tai kosketusnäyttöominaisuudet.

Kuvien lukeminen XML-tiedostosta hoituu samalla tavalla kuin ensimmäisessä sovelluksessa. Nämä kaksi toteutustapaa ovat melko samankaltaisia ja molemmissa löytyy omia paremmuuksia.



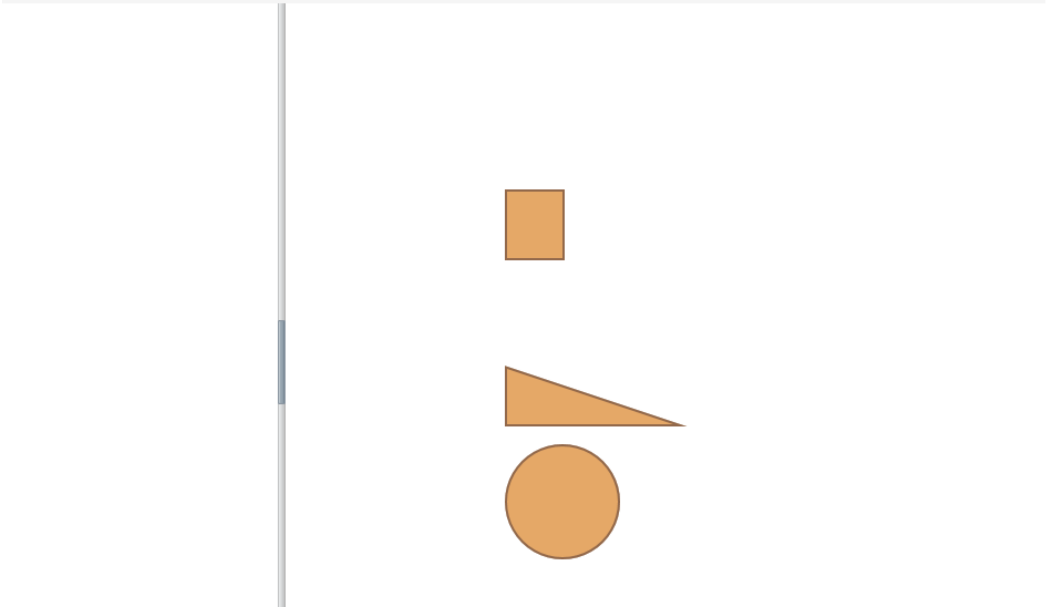
Kuva 9. Toisen toteutuksen lopputulos.

Kuten kuvassa 9 näkyy, tämä sovellus tehtiin melko pitkälle, puuttui vain kuvien erottelu kahteen eri canvas-elementtiin. Kuvien määrä näkyy kun hiiri on sen kohdalla.

5.3 Vaadin canvas winget toteutus

Kolmas ja viimeinen toteutus ei edennyt pitkälle, koska itse widgetti ei ole kehittynyt riittävän pitkälle jotta tällaisia sovelluksia sillä voisi tehdä.

Sovelluksessa saamaan kuviot piirtämään ja raahaamaan, mutta kun törmäys tapahtuu, kuvat jäävät paikoilleen ja senjälkeinen raahaus pätkii.



Kuva 10. Canvas-lopputulos.

Kuvassa 10 näkee mihin vaiheeseen tämä toteutus jäi. Syy miksi se jäi lyhyeen oli se, että heti alussa huomattiin ettei canvas-widgettiä kannata käyttää tällaisen sovelluksen luontiin, koska sen kehitys on vielä kesken. Sovelluksessa kuitenkiin tehtiin vedä, pudota ja törmäystoiminta. Sovellus tehtiin niin pitkälle että sitä pystyttiin vertaamaan muihin sovelluksiin.

Lopuksi näytetään pieni koodiosa, missä HTML-tiedosto liitetään Vaadinkomponenttiin.

```
@Override

        public void init() {

        Window mainWindow = new Window("Skeching_tool Application");
                ExternalResource res=new ExternalResource
                ("VAADIN/widgetsets/resources/resource.html");
                Embedded browser = new Embedded("", res);
                browser.setType(Embedded.TYPE_BROWSER);

                browser.setWidth("1750");
                browser.setHeight("750px");
                mainWindow.addComponent(browser);

                setMainWindow(mainWindow);

        }
```

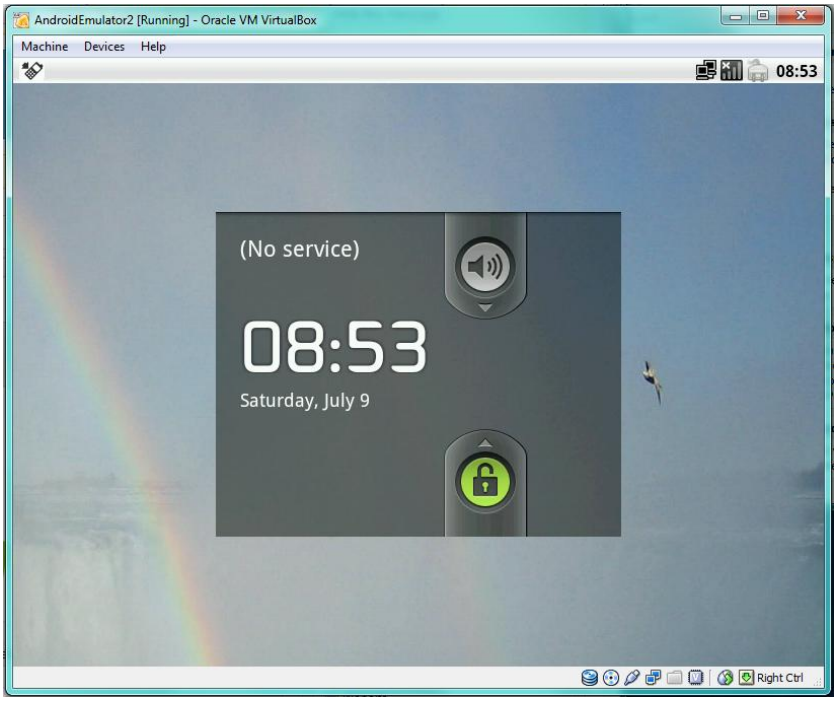
Kuvio 20. HMTL-tiedoston käyttö Vaadinissa.

6 TESTAUS

Kun jokainen sovellus tehtiin niin pitkälle kun piti, ryhdyttiin sovelluksia testaamaan, toimivatko kaikki toiminnot. Kaikki testit tehtiin vaatimuslistan mukaan. Katsottiin mitä vaaditaan ja testattiin toimiiko kyseinen toiminta. Tämä oli monissa toiminnoissa turhaa, koska toiminto oli jo hyvin testattu koodin tekemisen aikana.

Tärkein testivaihe, joka tuli vasta kun sovellus oli kokonaan valmis, oli kosketusnäyttöominaisuudet. Meillä ei löytynyt kosketusnäyttöä joten päätettiin testata kosketusnäyttö ominaisuudet virtual boxiin asennetulla android järjestelmällä.

Kuvassa 11 nähdään ohjelma jolla sovelluksen eri kosketusnäyttö-ominaisuudet testattiin.



Kuva 11. Android järjestelmä asennettuna virtualboxiin.

7 YHTEENVETO

Opinnäytetyönä toteutettu 2D-piirtotyökalu vastasi hyvin kaikkia asetettuja tavoitteita, ja toteutti kaikki vaaditut ominaisuudet, lukuun ottamatta zoom-toimintaa, joka ei toimi täydellisesti. Tätä raporttia kirjoitettaessa kaikki 3 toteutusta on tehty niin pitkälle kuin ne tehdään, ja ensimmäinen toteutus on kokonaan valmis.

Työ oli mielenkiintoisaa ja haastavaa, koska se toteutettiin käyttämällä JS:ä, jota koulussa ei paljon ole tullut opiskeltua. JS:n opiskeluun menikin hieman aikaa, mutta pian kaikki sujui hienosti. Työ oli siinä mielessä myöskin mielenkiintoinen, että saimme toteuttaa sen 3:lla eri tekniikalla ja näin saada itsellemme kokemusta kaikista 3:a tavasta joista yhdestäkään ei ollut aikaisempaa kokemusta.

Suoritettuamme nämä 3 tapaa huomattiin selvästi mikä tapa on parasta tälläisissä projekteissa käyttää. Se tapa on toteutus 1 eli jQuery tapa. KineticJS ja jQuery toteutukset ovat melko samoja ja kineticJS toiminnat olivat helpompia ymmärtää. Kuitenkin jQuery vei voiton tämän tyyppisessä sovelluksessa vaikka KinetcJS canvas-elementti olisi paljon parempi sovelluksissa joissa käytetään animaatiota. Kolmas toteutustapa oli huonoin. Canvas-widgetin kehitys on vielä kesken ja sen käyttö ei sovi tämäntyyppisiin toteutuksiin. Vaadin-sovellus on helpoin soveltaa Summiumiin, koska se on samalla ohjelmointikielellä tehty sovellus.

LÄHDELUETTELO

/1/ Growell E. KineticJS johdanto. Viitattu 1.8.2012. https://github.com/ericdrowell/KineticJS/wiki

/2/ Grönroos M. Vaadin Book overview. Viitattu 2.8.2012. https://vaadin.com/book/-/page/intro.html

/3/ Koulutus ja konsultointipalvelu. JavaScript-opas. Viitattu 8.8.2012. http://www.2kmediat.com/jscript/johdanto.asp

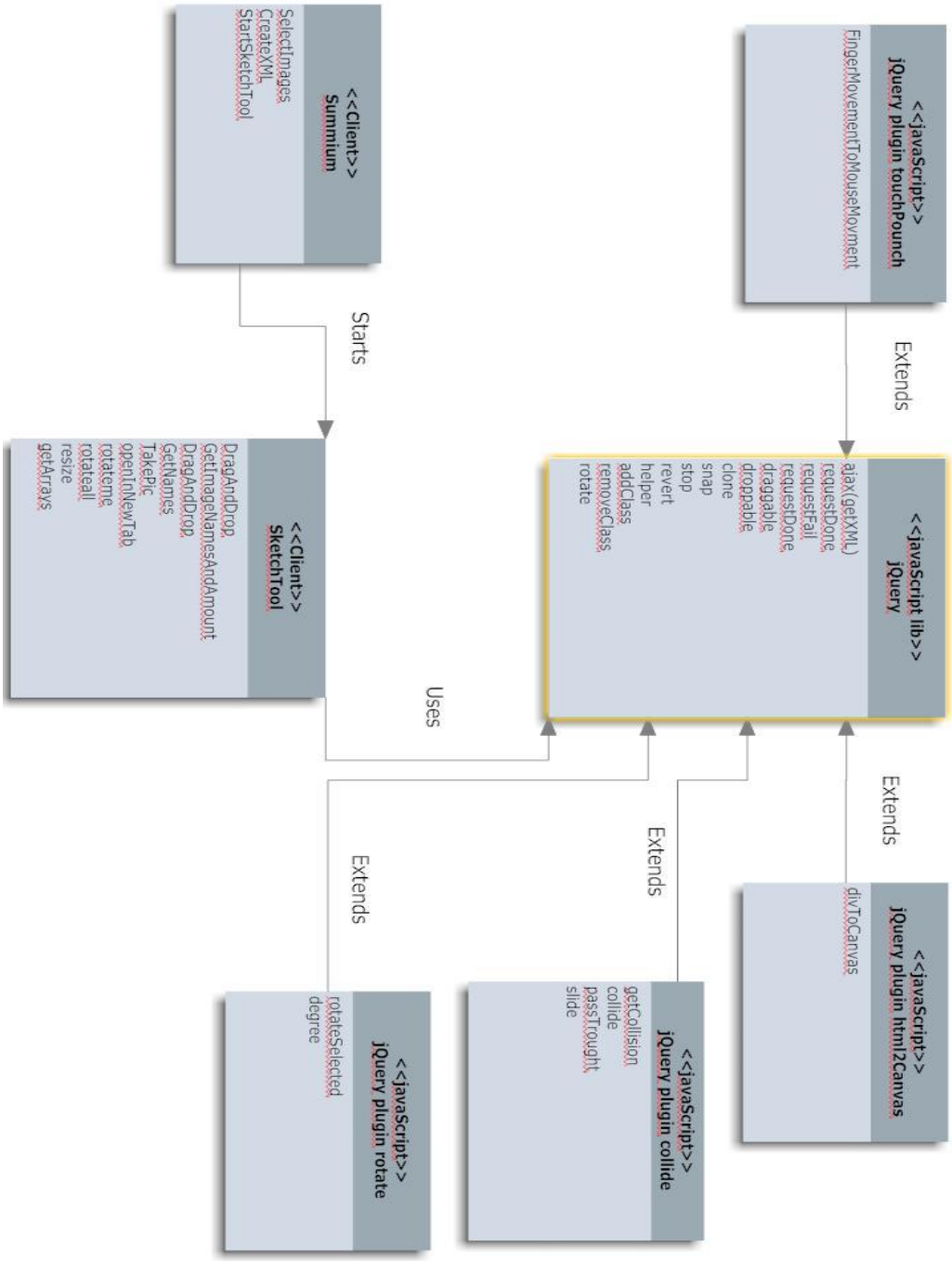
/4/ Paavilainen J. HTML5 mitä kaikkea tarvitsee tietää. Viitattu 2.08.2012. http://www.snoobi.fi/blogi/html5-mita+kaikkien+tarvitsee+tietaa

/5/ Resig J.2010 How jQuery works. Viitattu 1.8.2012 http://docs.jquery.com/How_jQuery_Works

/6/ Wapice Esite 2010. Viitattu 3.8.2012. http://w3.wapice.com/files/wapice-esite-2010_fi_web.pdf

LIITELUETTELO

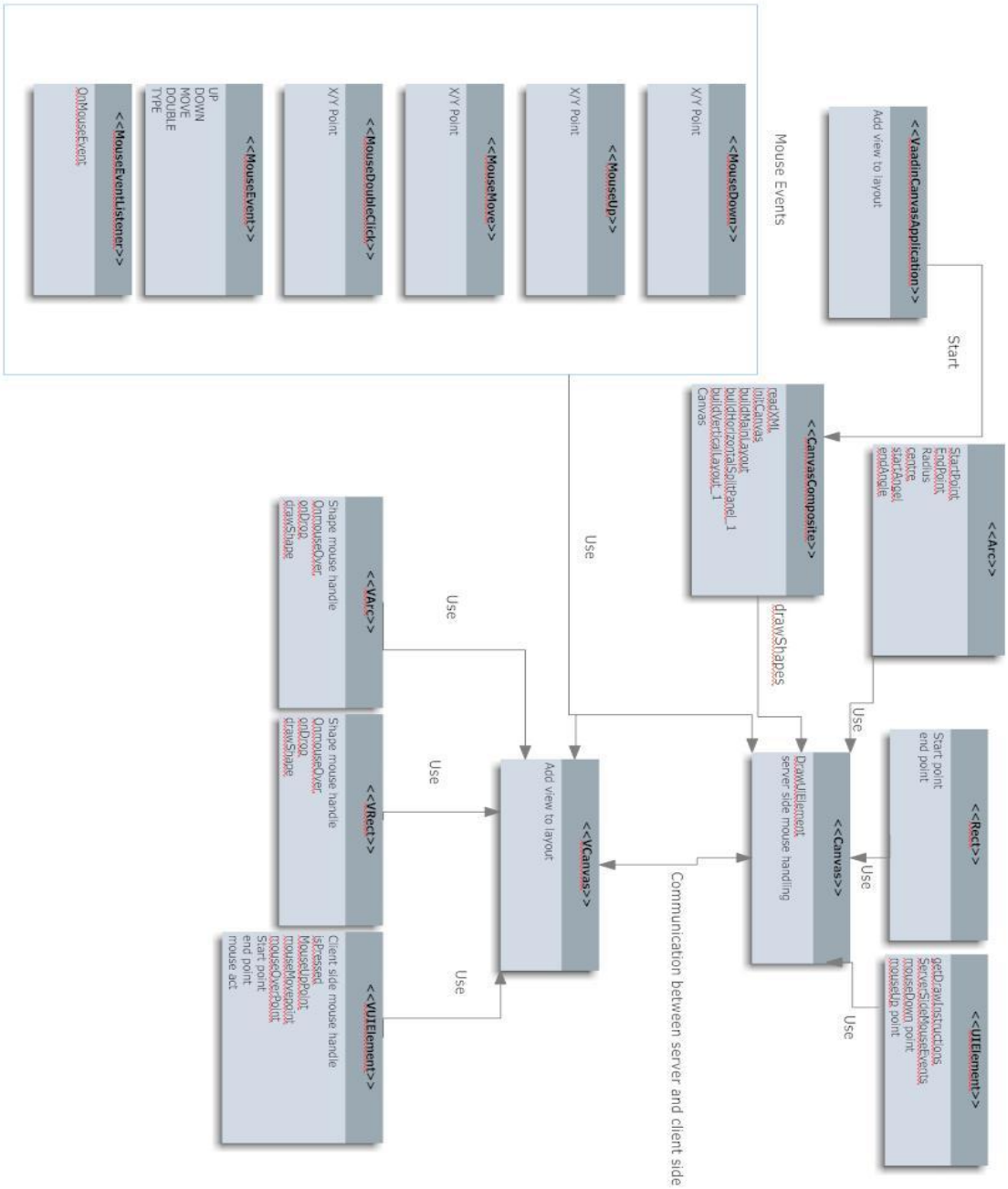
LIITE1



Kuva 12. JS-luokkakaavio.

Kuvassa 12 on JS toteutuksen luokkakaavio. Luokkakaaviossa näytetään mitä luokkia toteutuksessa on. JS toteutuksessa ei ole monta luokkaa ainoastaan voidaan jakaa eri JS kirjastot ja itse toteutus eri luokkiin.

LIITE2



Kuva 13. Vaadin-projektin luokkakaavio

Kuvassa 13 on vaadin toteutuksen luokkakaavio, jossa näytetään mitä eri luokkia vaadin projektissa käytettiin. Suurin osa kaikista luokista kuului Vaadin windget luokkiin joita muokkaamalla ja lisäämällä onnistuttiin pääsemään tiettyyn vaiheeseen asti.