Ilari Raijas

Selainpohjaisen sähköpostijärjestelmän toteutus Zend Framework -sovelluskehyksellä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

24.4.2016

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Ilari Raijas Selainpohjaisen sähköpostijärjestelmän toteutus Zend Framework -sovelluskehyksellä 43 sivua + 1 liite 24.4.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Toimitusjohtaja Arne Hook Lehtori Simo Silander

Insinöörityön tarkoituksena oli toteuttaa SoSe Oy:n markkinointityökaluun uusi sähköpostijärjestelmä. SoSe Zeed -nimisen työkalun vanha sähköpostijärjestelmä oli tarkoitus korvata uudella, monipuolisemmalla järjestelmällä. Sähköpostijärjestelmällä pyrittiin takaamaan työkalun käyttäjien ja heidän kontaktiensa välinen esteetön kommunikointi. Järjestelmän haluttiin mukautuvan ympäröivään sovellukseen täydellisesti, ja tarjoavan käyttäjälle mahdollisimman helppokäyttöisen käyttöliittymän.

Työssä esiteltiin järjestelmän vaatimukset ja projektissa käytetyt tekniikat. Näistä tekniikoista syvennyttiin Zend Framework -sovelluskehykseen ja tutkittiin sen soveltuvuutta suurien verkkosovellusten kehittämiseen. Työn toteutusosiossa tarkasteltiin sähköpostijärjestelmille tyypillisten toimintojen, kuten viestien lähettämisen ja vastaanottamisen, toteuttamista SoSe Zeedin sähköpostijärjestelmään.

Työssä todettiin, että sovelluskehysten käyttäminen suurissa verkkosovelluksissa on melkein aina välttämätöntä. Sähköpostijärjestelmän toteuttamisessa käytetty Mandrill-sähköpostipalvelu osoittautui erinomaiseksi välineeksi lähettää automatisoituja ja liiketoiminnallisia sähköpostiviestejä. Toteutettu sähköpostijärjestelmä otettiin odotusten mukaisesti työkalussa käyttöön, ja sitä kehitetään edelleenkin lisäämällä siihen uusia toiminnallisuuksia.

Avainsanat	Zend Framework, PHP, sähköpostijärjestelmä, sähköposti, sovelluskehys

Author(s) Title Number of Pages Date	Ilari Raijas Implementing Web-based Email Client on Zend Framework 43 pages + 1 appendix 24 April 2016
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software engineering
Instructor(s)	Arne Hook, CEO Simo Silander, Senior Lecturer

The purpose of this thesis was to implement a new email client for SoSe Ltd's marketing tool. SoSe Zeed - the old email client of the marketing tool was supposed to be replaced by the new, more versatile client. The aim of the new email client was to enable unobstructed communication between the user and their contacts. The email client was required to adapt to the surrounding application flawlessly and provide user-friendly interface.

This paper includes an introduction to the specifications of the email client and to the techniques used in the project. As to the techniques, Zend Framework was inspected more profoundly and examined how well it performs with large-scale web-applications. In the implementation part of the study the methods used for implementing the functionalities of standard email clients were inspected in detail. These functionalities include procedures such as sending and receiving email.

It was perceived that using frameworks in large-scale web-applications is almost always a necessity. Mandrill - a third party email API used in the implementation of the email client was discovered to be an excellent choice for sending automated and transactional emails. The implemented email client was deployed in the marketing tool as expected, and the email client is currently being further developed to add more functionalities to it.

Keywords	Zend Framework, PHP, email client, email, framework

Sisällys

Lyhenteet

Ajax	Asynchronous JavaScript and XML. Joukko web-teknologioita, joiden avulla on mahdollista tehdä asynkronisia HTTP-pyyntöjä.
CRUD	Create, Read, Update and Delete. Tietokantojen yleisimmät toiminnot.
CSS	Cascading Style Sheet. Verkkosivun ulkoasun määrittelevä tyyliohje.
DOM	Document Object Model. Malli, joka kuvaa dokumentin olioista koostuvana puurakenteena.
HTML	HyperText Markup Language. Merkintäkieli, jonka avulla määritellään verkkosivun rakenne.
HTTP	Hypertext Transfer Protocol. Protokolla tiedonsiirtoon, jota selaimet ja www-palvelimet käyttävät. Protokolla koostuu pyynnöistä ja vastauksista.
JSON	JavaScript Object Notation. JavaScript-syntaksinen tiedostomuoto tiedonvälitykseen.
MVC	Model-View-Controller. Arkkitehtuurityyli, jonka periaatteen mukaan sovelluksen käyttöliittymä ja sovellusaluetieto erotetaan toisistaan.
PHP	PHP: Hypertext Preprocessor. Ohjelmointikieli, jota käytetään pääosin palvelinpuolen ohjelmoinnissa.
URL	Uniform Resource Locator. Verkko-osoite, joka ilmoittaa resurssin sijainnin siihen käsiksi pääsemiseksi.
WYSIWYG	What You See Is What You Get. Tekstinkäsittelyohjelma, joka esittää kirjoitettavan sisällön samannäköisenä kuin lopputuloksen.

1 Johdanto

Insinöörityö tehtiin SoSe – Social Media Seeding Oy:lle osana tuotekehitysprosessia. SoSe Oy on viraalimainontaan erikoistunut yritys, joka tarjoaa asiakkailleen työkaluja ja palveluita verkkomarkkinoinnin toteuttamiseen. Yrityksen suosituin tuote on SoSe Zeed -markkinointityökalu, minkä avulla asiakkaat pystyvät vaivattomasti rakentamaan ja julkaisemaan markkinointikampanjoita verkossa.

Asiakkaiden lisääntyneiden tarpeiden seurauksena yritys halusi laajentaa SoSe Zeed -työkalua, ja tehdä siitä entistä käyttäjäystävällisemmän. Projektin ajankohtana tärkein kehittämiskohde oli vanhan sähköpostijärjestelmän korvaaminen uudella, monipuolisemmalla järjestelmällä. Uudistusten toteuttaminen helpottaisi asiakkaiden ja heidän kontaktiensa välistä kommunikointia entuudestaan. Insinöörityön aihe, selainpohjaisen sähköpostijärjestelmän toteutus oli seurausta tarpeelle kehittää SoSe Zeed -työkalua monipuolisemmaksi.

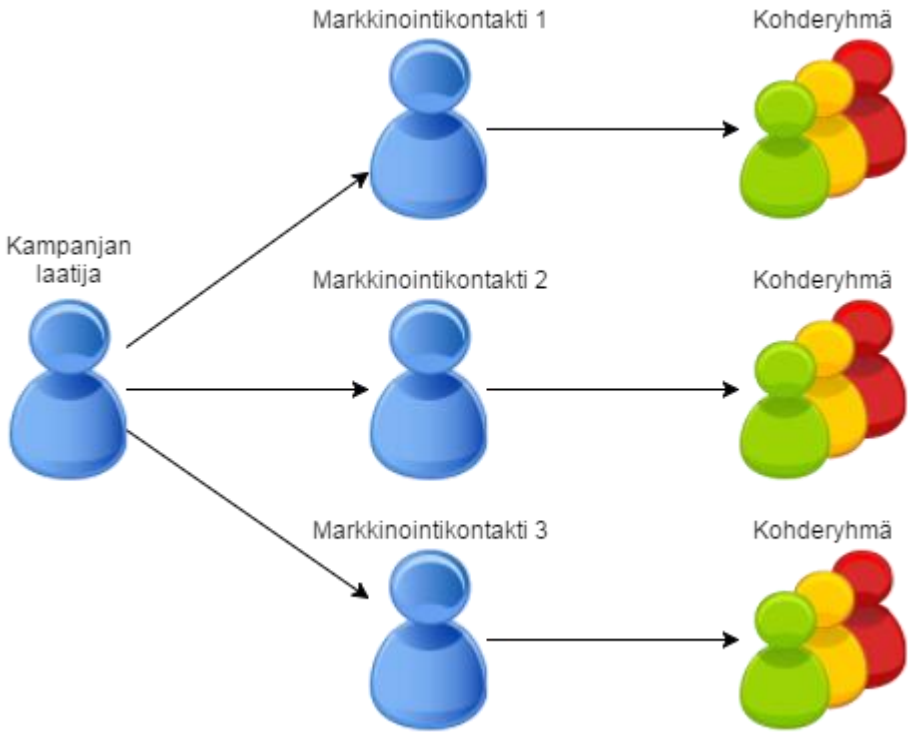
Insinöörityön tavoitteena on selvittää, kuinka PHP-sovelluskehys Zend Framework soveltuu verkkosovellusten kehittämiseen ja miten sen avulla pystytään implementoimaan verkkosovellusten kehittämisessä tärkeäksi todettuja toimintoja. Lisäksi raportissa seurataan selainpohjaisen sähköpostijärjestelmän toteutuksen eri vaiheita ja analysoidaan toteutuksessa käytettyjä tekniikoita.

Sähköpostijärjestelmä kehitetään SoSe Zeed -työkalun tarpeiden mukaisesti, ja sen omia sisäisiä toimintoja hyväksikäyttäen. Tästä syystä sähköpostijärjestelmälle määritellyt toiminnallisuudet eroavat hieman tavallisten sähköpostijärjestelmien toiminnallisuuksista.

2 SoSe Zeed -työkalu

Sähköpostijärjestelmän toteutuksessa tehtyjen valintojen ymmärtämiseksi oleellista on käsittää, mikä tarkoitus SoSe Zeed -markkinointityökalulla on. Työkalu on alun perin kehitetty helpottamaan markkinointikampanjoiden laatimista ja niiden julkaisua verkossa. Kampanja voi olla esimerkiksi yrityksen laatima mielipidetutkimus, jota levitetään sosiaalisessa mediassa. Työkalun avulla käyttäjä pystyy tekemään ja julkaisemaan uusia kampanjoita, ja seuraamaan niistä syntyvää statistiikkaa. Lisäksi työkalu tarjoaa kampanjasivu (landing page) -editorin, jonka avulla käyttäjä pystyy vaivattomasti tekemään kyselyitä, testejä ja muita sivuja, joista halutaan kerätä vierailijoiden antamaa syötettä. SoSe Zeed -työkalu kerää näitä vierailijoiden tuottamia syötteitä ja esittää niitä statistiikoissa.

Viraalimarkkinoinnin merkittävin, ehkäpä myös hankalin osuus on kohdeyleisön tavoittaminen. Kohdeyleisö löydetään yleensä sosiaalisen median kautta, missä tieto leviää nopeasti ihmisten kesken. Onnistuessaan kampanja leviää nopeasti, tavoittaa paljon ihmisiä ja kerää paljon näkyvyyttä. Yksi tärkeimmistä onnistuneen kampanjan piirteistä on sen levitys (seeding) oikeiden tahojen kautta. Kuva 1 esittää kampanjan leviämistä markkinointikontaktien (seeding point) kautta kohderyhmille.



Kuva 1. Kampanjan leviäminen markkinointikontaktien avulla.

Markkinointikontaktit

Markkinointikontaktit (seeding points) ovat yhteyshenkilöitä, jotka levittävät kampanjaa eteenpäin. Markkinointikontakteja voivat olla esimerkiksi blogit tai foorumit, jotka käsittelevät kampanjan aihepiiriä. Markkinoinnissa kampanjoiden tekijät ovat yleensä kiinnostuneita siitä, mitkä markkinointikontaktit tuovat eniten vierailijoita kampanjasivulle. SoSe Zeedin yksi ominaisuus on tämä tiedon kerääminen. Työkalu kerää ylös tietoa siitä, kuinka paljon kampanjasivulle syntyy liikennettä kustakin markkinointikontaktista. Tässä insinöörityössä markkinointikontakteihin viitataan vastedes myös lyhyemmin pelkkinä kontakteina.

SoSe Zeed -työkalun avulla käyttäjä pystyy itse lisäämään kontakteja, ja asettamaan niitä kampanjoille. Kontakti määritellään antamalla sille nimi, sähköpostiosoite, kohderyhmät ja kanava, johon se kuuluu. Esimerkkinä kampanjan laatija, joka haluaa toteuttaa urheiluun ja vapaa-ajanviettoon liittyvän kyselyn: Kampanjan laatija tuntee suositun blogin, jossa käsitellään maastohiihtoa, ja haluaa lisätä kyseisen blogin kampanjansa markkinointikontaktiksi. Työkalussa kampanjan laatija määrittelee kontaktille nimen, sähköpostiosoitteen ja kohderyhmät, joita kontakti tavoittelee. Lisäksi hän lisää kontaktin sitä vastaavalle kanavalle, jotta pystytään tunnistamaan samantyyppiset kontaktit. Esimerkin tapauksessa kampanjan laatija lisää kontaktin blogit-kanavalle. Kyseinen kontakti löytyy sitten blogit-kanavan alaisuudesta, kun kampanjalle asetetaan kontakteja. Markkinointikontakteille voi määritellä paljon enemmänkin tietoa, mutta mainitut ominaisuudet ovat tärkeimmät.

Sähköpostijärjestelmä on toteutettu käyttäen hyväksi kampanjoihin lisättyjä markkinointikontakteja. Sähköpostijärjestelmän osalta kontaktit ovat toimijoita, joille halutaan lähettää kampanjasta sähköpostia, ja jotka vastaanottavat niitä.

Kampanjat

Kampanja on verkkosivu, josta halutaan kerätä sinne saapuvien vierailijoiden tuottamaa syötettä. Kampanjan laatijan kannalta tärkeää on tietää, mistä markkinointikontaktista vierailija on saapunut. Tällä tiedolla pystytään analysoimaan, mitkä kontaktit tuovat eniten vierailijoita, eli mihin kontakteihin kannattaa panostaa kampanjan laajan levityksen aikaansaamiseksi. Kampanjan levitystä varten sille täytyy valita kontakteja, jotka kuulu-

vat kampanjan kohderyhmille. Kontaktien avulla pyritään saamaan kampanjalle mahdollisimman laaja näkyvyys sen kohderyhmien keskuudessa. SoSe Zeed -työkalu mahdollistaa kampanjaan lisättyjen markkinointikontaktien kanssa kommunikoimisen sähköpostien välityksellä. Työkalun sähköpostijärjestelmä tarjosi aiemmin vain sähköpostien lähettämisen kontakteille, mutta ei mahdollistanut vuorovaikutteisten viestiketjujen muodostamista kontaktien ja käyttäjän välillä. Kontaktien ja kampanjan laatijan välinen vuorovaikutus on ensiarvoisen tärkeää, joten sähköpostijärjestelmä päätettiin uusia kokonaan.

Sähköpostijärjestelmä on suunniteltu kampanjoiden ja kontaktien ehdoilla. SoSe Zeed -työkalun käyttäjällä voi olla samaan aikaan useita kampanjoita, joihin on määriteltyinä samoja tai eri markkinointikontakteja. Jotta sähköpostilaatikko ei täyttyisi kaikista käyttäjän viesteistä, on sähköpostijärjestelmä toteutettu kampanjakohtaisesti. Sähköpostijärjestelmä vastaa aina yhteen kampanjaan liittyvistä sähköposteista. Tämä selkeyttää kontaktien kanssa kommunikoimista, kun työkalun käyttäjällä on useita kampanjoita aktiivisena. Sähköpostiviestejä voidaan lähettää vain kampanjaan määritellyille kontakteille, mikä estää viestien lähettämisen järjestelmän ulkopuolisille tahoille. Lähetettävä sähköpostiviesti saa tunnisteet, joilla se linkitetään kampanjaan ja kontaktiin. Näin pystytään erottelemaan kampanjoiden viestit toisistaan ja toteuttamaan sähköpostijärjestelmä kampanjakohtaisena.

Markkinointikontaktien esittelyssä mainittiin työkalun käyttäjä, joka halusi toteuttaa urheiluun ja vapaa-ajanviettoon liittyvän kampanjan. Käyttäjä tunsi entuudestaan blogin, jonka hän sitten lisäsi markkinointikontaktina kampanjalle. Tämän yksittäisen kontaktin lisäksi hän lisäsi kampanjalle paljon muitakin urheiluun ja vapaa-ajanviettoon liittyviä kontakteja. Käyttäjä haluaa tehdä seuraavaksi kampanjan, joka keskittyy pelkästään maastohiihtoon. Luonnollisesti hän valitsee kampanjan markkinointikontakteiksi aiheeseen liittyviä kontakteja. Tässä vaiheessa käyttäjä lisää jo aiemmassa kampanjassa käytetyn maastohiihtoblogin uuden kampanjan kontaktiksi. Nyt kun kyseisen kontaktin kanssa kommunikoidaan sähköpostiviestien välityksellä, viestit eritellään niitä koskeville kampanjoille.

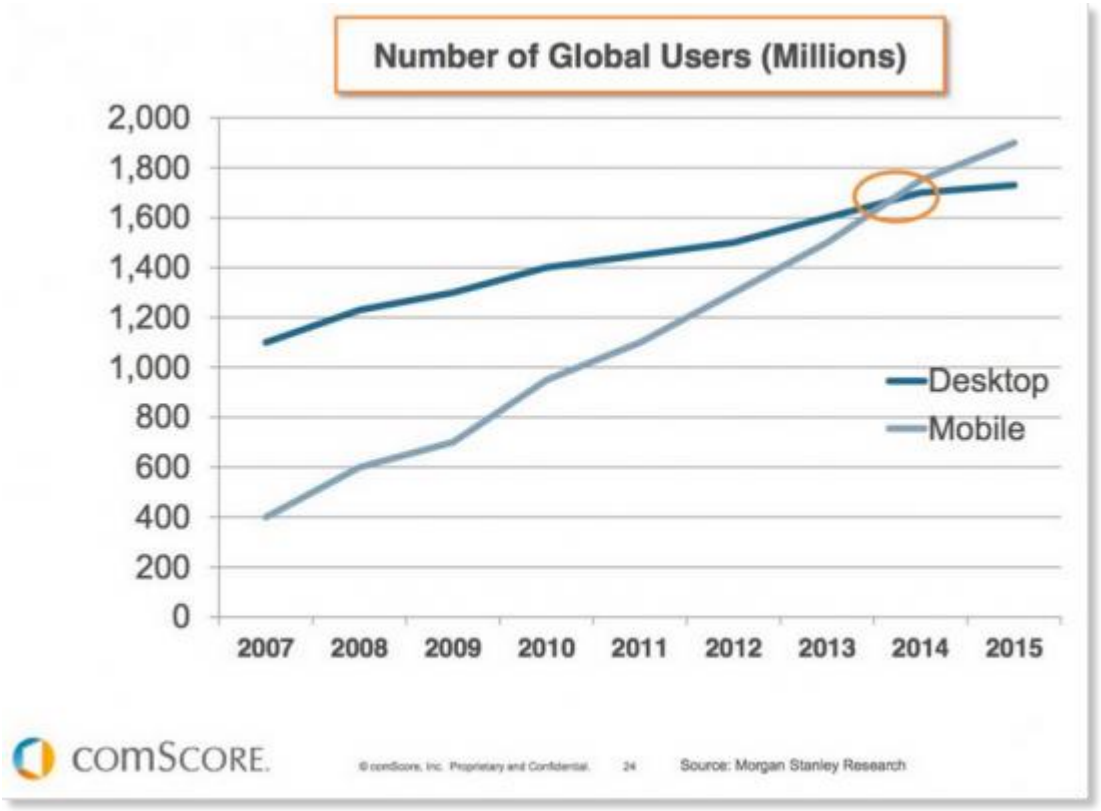
3 SoSe Zeedin teknologiat

SoSe Zeed on kehitetty Zend Framework -sovelluskehyksellä, joka on PHP-pohjainen sovelluskehys suurien verkkosovellusten kehittämiseen. Zend Framework takaa vankan perustan verkkosovellukselle ja tuo paljon ohjelmistotekniikan suunnittelumalleja kehittäjien käyttöön. Zend Framework on monipuolinen sovelluskehys, joka koostuu suuresta määrästä komponentteja. Komponentit ovat luokkia, jotka tuovat erilaisia toiminnallisuuksia kehittäjien käyttöön. Nämä toiminnot ratkaisevat verkkosovellusten ohjelmoinnissa usein kohdattuja ongelmia, ja edesauttavat verkkosovellusten nopeaa rakentamista. Esimerkiksi Zend Frameworkin komentorivityökalu mahdollistaa projektien rungon luomisen yhdellä komennolla. Sovelluskehyksen mukana tarjotaan myös paljon muita toimintoja, joita tarkastellaan syvemmin myöhemmissä luvuissa.

Zend Framework -sovelluskehyksen lisäksi työkalua ylläpitää suuri määrä muita teknologioita. Näitä verkkosovellusten kehittämisessä käytettyjä teknologioita käydään pintapuolisesti läpi tässä luvussa.

3.1 Frontend-teknologiat

Internetiin yhdistävien laitteiden määrä on lisääntynyt räjähdysmäisesti viime vuosien aikana. Näistä laitteista eniten markkinoita ovat valloittaneet mobiililaitteet, kuten älypuhelimet ja tabletit. Kuva 2 osoittaa mobiililaitteiden käyttäjien ohittaneen pöytäkoneiden käyttäjien määrän jo vuoden 2014 aikana.



Kuva 2. Pöytäkoneiden ja mobiililaitteiden käyttäjien määrä verkkosivuilla [1.]

Mobiililaitteiden suosion myötä verkon selaaminen on siirtynyt yhä enemmän perinteisiltä laitteilta, tietokoneilta, mobiililaitteille. Vielä muutama vuosi sitten verkkosivujen suunnitteleminen keskittyi pääosin näille perinteisille, suurikokoisille laitteille. Siirtyminen perinteisistä laitteista mobiililaitteille on tuonut lisää haasteita frontend-kehittäjille. Suurille näytöille suunnitellut verkkosovellukset käyttäytyvät arvaamattomasti mobiililaitteilla. Tästä syystä verkkosovellusten käyttöliittymän suunnittelu ja toteutus on nykyään keskittynyt responsiivisuuteen. Responsiivisuudella pyritään takaamaan verkkosovelluksen käytettävyys kaikenkokoisilla laitteilla. Käytettävyys ja visuaalinen toteutus ovat ominaisuuksia, joita käyttäjät arvostavat vieraillessaan verkkosivuilla. Näitä ominaisuuksia pyritään vaalimaan myös sähköpostijärjestelmän toteutuksessa.

HTML5

HTML (HyperText Markup Language, hypertekstin merkintäkieli) on verkkosivujen rakenteen määrittelemiseen tarkoitettu merkintäkieli. HTML-kieli koostuu elementeistä, jotka määritellään tägeillä. Jokaisella elementillä on alku- ja lopputägi, joiden väliin voi-

daan lisätä muita elementtejä. HTML-dokumentti koostuu useista sisäkkäisistä elementeistä, joista muodostuu puurakenteinen hierarkia. Web-palvelin lukee HTML-tiedoston ja rakentaa siitä verkkosivun, jonka se esittää käyttäjän selaimella. Normaalisti HTML-merkinnällä pyritään kuvaamaan pelkästään sivun rakennetta, eikä sillä oteta kantaa sivun visuaaliseen ilmeeseen. [2.]

HTML5 on uusin, viides versio HTML:stä. Sen tarkoitus on tuoda lisää toiminnallisuuksia ja auttaa luomaan alustariippumattomia verkkosivuja. HTML5 keskittyy tukemaan paremmin moderneja multimedioita ja tekemään verkkosivuista kevyempiä käyttää. Sähköpostijärjestelmän toteutuksessa käytetään hyväksi HTML5:n esittämiä uusia toiminnallisuuksia. [2.]

CSS

Verkkosivujen visuaalisesta ilmeestä vastaavat yleensä CSS (Cascading Style Sheet, porrastetut tyyliarkit) -tyyliohjeet. CSS-tiedostojen avulla mahdollistetaan verkkosivun rakenteen ja visuaalisen esityksen eritteleminen. Toimintojen erotteleminen helpottaa verkkosivujen lähdekoodin lukemista ja sivujen uudelleenkäytettävyyttä. Tämän lisäksi tyylitiedostot mahdollistavat verkkosivun ulkoasun vaihtamisen lennosta, jos esimerkiksi halutaan sivun ulkonäön määräytyvän laitteen mukaan. [3.]

Tyylitiedostot koostuvat säännöistä, jotka määrittelevät elementtien ulkoasun. HTML-dokumentti, joka käyttää tyylitiedostoa, lukee siinä olevat säännöt, ja asettaa ne elementeille. Elementit voivat olla useiden sääntöjen alaisia, mutta päällekkäisissä tilanteissa noudattavat arvojärjestyksessä ensimmäistä sääntöä.

JavaScript ja jQuery

Modernit verkkosivut sisältävät paljon dynaamista toiminnallisuutta, jonka avulla niistä tehdään interaktiivisia sovelluksia. Sähköpostijärjestelmässä käyttöliittymän vuorovaikutteisuutta lisätään JavaScript-ohjelmointikielen avulla. JavaScript on dynaaminen komentosarjakieli eli skriptikieli, jota käytetään yleisesti verkkosivulla tapahtuvien tapahtumien kuuntelemiseen ja sivun rakenteen muokkaamiseen. JavaScriptiä käytetään pääosin selaimen käyttäytymistä ohjaaviin toimintoihin, mutta sen suosio palvelinohjelmoinnissakin on lisääntynyt sovelluskehysten myötä (Node.js). Selainpuolen toimintoja ovat esimerkiksi verkkosivulla tapahtuviin tapahtumiin reagoiminen, DOM-puun (Document

Object Model) muokkaaminen, HTML-elementtien animaatio ja HTTP-pyyntöjen lähettäminen palvelimelle ilman sivun lataamista uudelleen. [4;5.]

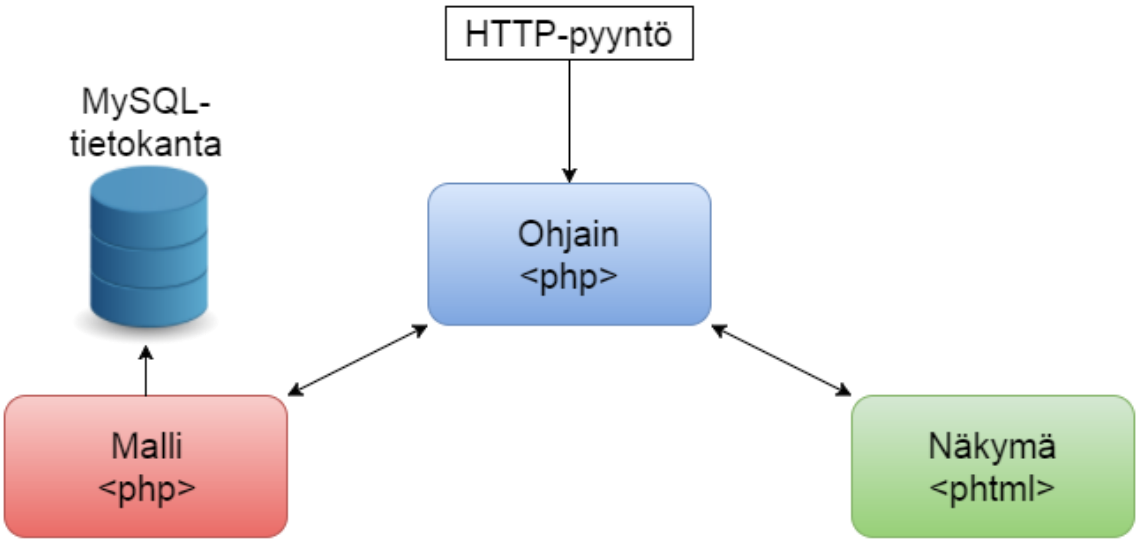
JavaScript on saavuttanut vankan suosion selainpuolen ohjelmointikielien keskuudessa, sillä kaikki nimekkäimmät selaimet tukevat sitä. Tästä on aiheutunut, että sovelluskehyksiä ja kirjastoja tehdään paljon juuri JavaScriptille. Kirjastot voivat olla pieniä lisätoiminnallisuuksia, jotka ehostavat sivun ulkoasua, tai suuria sovelluskehyksiä, jotka tuovat valtavasti lisätoiminnallisuuksia perinteiseen JavaScript-kieleen. Kirjastoja käyttämällä voidaan helposti lisätä tarvittavia toiminnallisuuksia verkkosovellukseen, ja säästetään se aika, mikä menisi ratkaisun toteuttamiseen itse.

Tämän hetken suosituimman JavaScript-kirjaston, jQueryn tavoitteena on helpottaa kehittäjän työtä yksinkertaistamalla JavaScriptin toimintoja. jQuery tuo uuden syntaksin, joka helpottaa dokumentissa liikkumista, DOM-puun muokkaamista ja animaatioiden lisäämistä. jQuery tuo paljon muitakin toimintoja, joista yksi merkittävin on tuki eri selaimille. Tuki takaa sen, että kehittäjien rakentamat verkkosivut toimivat halutulla tavalla kaikilla uusimmilla selaimilla. [6.]

Modernin sähköpostijärjestelmän rakentaminen ei olisi mahdollista käyttämällä pelkkiä lomakkeita. Sähköpostijärjestelmä tekee paljon pyyntöjä palvelimelle, ja lomakkeiden käyttö olisi hyvin epäkäytännöllistä. Jotta sähköpostijärjestelmästä saataisiin mahdollisimman käyttäjäystävällinen, suoritettiin palvelimelta tehtävät haut pääosin Ajaxilla (asynchronous JavaScript and XML) [7]. Ajax on joukko web-teknologioita, jotka mahdollistavat asynkronisten HTTP-pyyntöjen tekemisen ilman, että verkkosivua ladataan uudelleen. Ajaxin avulla sähköpostijärjestelmä pystyy mukautumaan käyttäjän tekemiin valintoihin hakemalla dataa asynkronisesti palvelimelta.

3.2 Backend-teknologiat

Sovelluksen liiketoimintalogiikka on eristetty asiakaspuolesta MVC (Model-View-Controller) -suunnittelumallin mukaisesti. Sähköpostien käsitteleminen ja tietokantaan kohdistuvat operaatiot tehdään mallin kautta. Zend Frameworkissa mallit ovat PHP-luokkia, jotka käsittelevät palvelimella tehtäviä toimintoja. Kuva 3 esittää yleisen MVC-toteutuksen, jota käytetään myös SoSe Zeedissä ja siten myös toteutetussa sähköpostijärjestelmässä.



Kuva 3. MVC-mallin toteutus SoSe Zeed -työkalussa.

PHP

PHP (PHP: Hypertext Preprocessor) on ohjelmointikieli, jota käytetään varsinkin palvelinpuolella. PHP:n käyttäminen on edelleenkin yleistä palvelinpuolella, vaikka varteenotettavia vaihtoehtoja on tullut ajan mittaan paljon lisää. Yhtenä syynä kielen suosiolle on sen saama laaja tuki eri alustoille. PHP-tulkin asentaminen onnistuukin melkein jokaiselle web-palvelimelle ja käyttöjärjestelmälle. Lisäksi PHP tarjoaa kehittäjille suuren määrän sisäänrakennettuja luokkakirjastoja, jotka tarjoavat esimerkiksi tietokanta- ja tiedostonhallintapalveluita. PHP:ta voi kirjoittaa HTML-koodin sekaan, mikä lisää sen soveltuvuutta asiakaspuolella.

Sähköpostijärjestelmän backend-puoli toteutettiin kokonaan PHP:llä. Käytännössä tämä tarkoitti web-palvelimelle tuleviin HTTP-pyyntöihin vastaamista, tietokannan hallitsemista ja ulkoisen sähköpostien lähetyksestä/vastaanotosta huolehtivan ohjelmointirajapinnan hallitsemista.

Zend Frameworkin ansiosta PHP:lla on osansa myös käyttöliittymän puolella. Sovelluskehys tarjoaa backend-puolen lisäksi myös koko sovellukselle yhteisiä toiminnallisuuksia, kuten yhteiset muuttujat, joihin voi viitata näkymästä. Myös ohjaimessa on mahdollista määritellä näkymälle muuttujia, joita voi sitten käyttää näkymän puolella. Zend Frameworkin näkymät ovatkin siis PHTML-tiedostoja, jotka eroavat PHP-tiedostoista vain

tiedostopäätteensä perusteella. Eri tiedostopääte johtuu pelkästään syystä erottaa helposti näkymä muista sovelluksen osista.

3.3 Muut teknologiat

MySQL-tietokanta

SoSe Zeed -työkalussa tietokannan hallitsemisjärjestelmänä käytetään MySQL-relaatiotietokantaa. MySQL on avoimeen lähdekoodiin perustuva tietokantajaohjelmisto, jonka laaja suosio verkkosovellusten kehityksessä perustuu sen helppokäyttöisyyteen ja ylläpidettävyyteen. Relaatiomallin mukaisesti tietokanta koostuu tauluista, jotka muodostuvat reaalimaailmaa jäljentävistä kentistä. Taulut ovat itsenäisiä kokonaisuuksia, jotka viittaavat toisiin tauluihin. Näin pystytään rakentamaan vuorovaikutteinen verkkosovellus hyväksikäyttäen relaatiotietokantojen ominaisuuksia. [8, s. 84.]

Työkalussa liikkuva data tallennetaan MySQL-relaatiotietokantaan, jonka avulla mahdollistetaan sivujen dynaaminen rakentaminen ja työkalun olennaiset toiminnallisuudet. Sähköpostijärjestelmä käyttää tietokantaa päällimmäisenä tarkoituksenaan tallentaa lähetetyt ja vastaanotetut viestit, mutta tietokantaa hyödynnetään muihinkin tarkoituksiin. Tietokannasta saadaan myös kampanjoiden ja käyttäjien tiedot, joita käytetään näkymän rakentamiseen ja viestien identifioimiseen.

Mandrill

Mandrill on sähköpostien käsittelemiseen, lähettämiseen ja vastaanottamiseen erikoistunut palvelu, joka helpottaa lähetettävien sähköpostiviestien automatisointia ja personalisointia. Sähköpostien lähettämisen lisäksi Mandrill tarjoaa palvelut viestien vastaanottamiseen ja sähköpostitapahtumiin reagoimiseen. Monipuolinen rajapinta on tarjolla useille alustoille, joita ovat ovat mm. Python, Ruby, Node.js ja PHP. SoSe Zeed toteuttaa näistä alustoista PHP-rajapinnan. [9.]

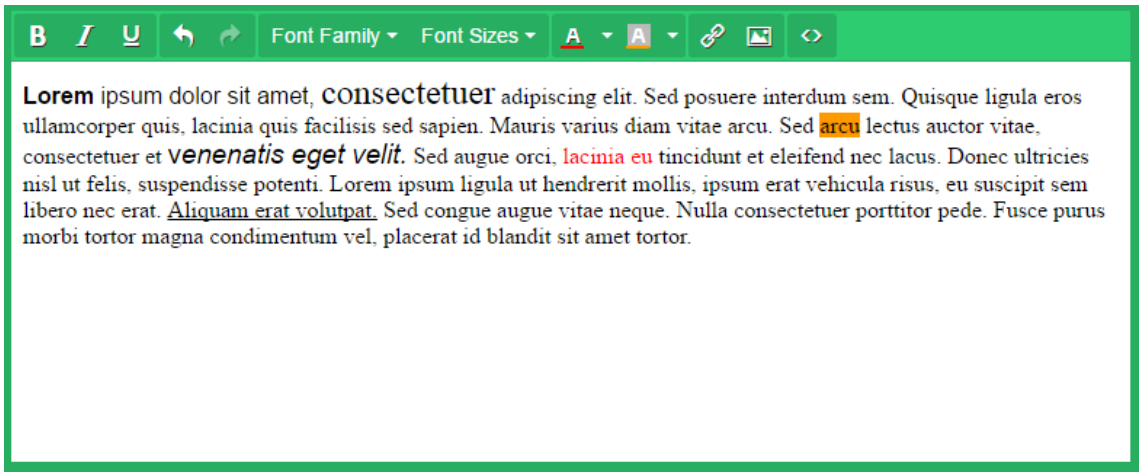
SoSe Zeed käyttää Mandrillia automatisoitujen sähköpostiviestien lähettämiseen. Myös vanha sähköpostijärjestelmä käytti Mandrillia, joten se oli luonnollinen valinta uuden sähköpostijärjestelmän toteuttajaksi. Sähköpostijärjestelmän perimmäinen tarkoitus on lähettää viestiä kampanjoista kontakteille. Näitä kontakteja voi olla allokoituna kampanjalle aina muutamista henkilöistä satoihin kontakteihin asti. Sähköpostiviestien halutaan olevan yksilöllisiä ja henkilökohtaisia, mutta silti tarpeeksi automatisoituja sähköpostijärjestelmän vaivattomuuden takaamiseksi. Sähköpostijärjestelmässä viestille tehdään halutut operaatiot, jonka jälkeen se lähetetään Mandrillin kautta vastaanottajalle. Mandrill seuraa viestiä ja ilmoittaa matkalla tapahtuvista tapahtumista, mikä auttaa kehittäjää, jos matkan varrella tapahtuu jotain arvaamatonta.

TinyMCE

Sähköpostijärjestelmän tekstieditoriksi valittiin TinyMCE WYSIWYG (what you see is what you get) -editori. Valinta tehtiin vertailemalla eri selainpohjaisia avoimen lähdekoodin tekstieditoreita ja valitsemalla niistä sähköpostijärjestelmän kannalta sopivin. TinyMCE on asiakaspuolen JavaScript-kirjasto, joka mahdollista ”textarea” tai muiden HTML-elementtien muuttamisen tekstieditoreiksi. TinyMCE alustetaan sivun latauduttua, ja sille annetaan konfigurointitietoina käyttöönotettavat liitännäiset ja muut käyttäjäkohtaiset asetukset.

TinyMCE:n vahvuuksiin lukeutuu mahdollisuus tekstieditorin laajaan kustoimointiin, ja ulkoasun määrittelemiseen TinyMCE:n omalla Skin Creator -työkalulla. Lisäksi TinyMCE-kirjaston käyttäminen on yksinkertaista, ja se tarjoaa käyttäjälle suuren määrän liitännäisiä.

Sähköpostijärjestelmän kannalta tekstieditorin tärkeimpiä ominaisuuksia ovat tekstinkäsittelyominaisuudet. SoSe Zeedin sähköpostijärjestelmän tekstieditorista haluttiin mahdollisimman yksinkertainen, joten tekstieditorista pyrittiin karsimaan pois kaikki ylimääräiset toiminnallisuudet. Jäljelle jätettiin vain välttämättömimmät tekstinkäsittelyyn liittyvät toiminnot, jotka vaikuttavat tekstin ulkoasuun (kuva 4).



Kuva 4. Tekstieditorin näkymä

4 Zend Framework

Zend Framework on PHP 5:llä kirjoitettu olioperustainen sovelluskehys PHP-verkkosovellusten kehittämiseen. Sovelluskehys koostuu komponenteista, jotka ovat riippuvaisia toisistaan. Riippuvuuksia on kuitenkin pyritty minimoimaan, jotta komponentteja pystyisi käyttämään mahdollisimman paljon erikseen. Esimerkiksi käyttäjä voisi hyödyntää pelkästään Zend_Service_Twitter-komponenttia, ja jättää Zend_Validate-komponentin pois. Täyden tehon sovelluskehyksestä saa kuitenkin, kun käyttää kaikkia komponentteja yhdessä. [10.]

Zend Framework tarjoaa erinomaisen suorituskyvyn omaavan MVC-toteutuksen PHP-sovellusten nopeaan ja ammattimaiseen kehittämiseen. Näitä MVC-komponentteja tutkitaan lähemmin tämän luvun aikana. Tämän lisäksi Zend Framework tarjoaa Zend_Db-rajapinnan tietokantojen hallinnoimiseen ja paljon muita palveluita, kuten Zend_Json- ja Zend_Mime-komponentit.

Zend Frameworkin tarjoama Zend Tools -komentorivityökalu mahdollistaa sovellusten nopean alustamisen, ja kehitystyön pikaisen aloittamisen. Zend Tools -työkalun avulla käyttäjä pystyy komentoriviltä suorittamaan projektin alustuksen, jolloin työkalu tekee projektille rungon Zend Frameworkin käytäntöjen mukaisesti.

4.1 Rakenne

Zend Framework -sovelluskehykselle kehittäessä suositellaan käyttämään heidän määrittelemää projektiarkkitehtuuria. Zend Tools -työkalu luo ennalta määritellyn, Zend Frameworkin mukaisen rungon projektille, joka sisältää kaikki tarvittavat kansiot ja tiedostot uuden projektin aloittamiseksi.

```
newproject
|-- application
|   |-- Bootstrap.php
|   |-- configs
|   |   `-- application.ini
|   |-- controllers
|   |   |-- ErrorController.php
|   |   `-- IndexController.php
|   |-- models
|   `-- views
|       |-- helpers
|       `-- scripts
|           |-- error
|           |   `-- error.phtml
|           `-- index
|               `-- index.phtml
|-- library
|-- public
|   `-- index.php
`-- tests
    |-- application
    |   `-- bootstrap.php
    |-- library
    |   `-- bootstrap.php
    `-- phpunit.xml
```

Kuva 5. Projektin runko [11.]

Kuvasta 5 näkyy, kuinka Zend Tools -työkalu loi projektin rungon ja olennaiset tiedostot. Newproject-hakemisto jakautuu neljään alihakemistoon, jotka ovat application, library, public ja tests. Projektin jakaminen osiin helpottaa kehitystyötä, varsinkin jos sovellusta on kehittämässä useampia henkilöitä.

Public-kansio asetetaan projektin juurihakemistoksi. Tästä seuraa, että vain public-kansion tiedostot ovat julkisia loppukäyttäjille. Sovelluksen kannalta tärkeät suoritettavat tiedostot ovat public-kansion ulkopuolella loppukäyttäjien ulottumattomissa. Public-kansioon laitetaan kaikki tiedostot, joita selain tarvitsee käyttääkseen verkkosovellusta. Näitä tiedostoja ovat esimerkiksi tyylitiedostot, kuvat, JavaScript-tiedostot ja muu staattinen sisältö. Näiden lisäksi public-kansion juuressa sijaitsevat .htaccess- ja index.php-tiedostot.

Palvelimen vastaanottaessa HTTP-pyynnön projektiin palvelin avaa ensimmäisenä .htaccess-tiedoston, joka määrittelee, mihin pyyntö uudelleenohjataan. Jos pyyntö kohdistuu public-kansiossa sijaitsevaan tiedostoon, ohjataan pyyntö sinne. Muuten pyyntö

ohjataan public-kansiossa sijaitsevaan index.php-tiedostoon, jonka tehtävänä on rakentaa koko Zend Framework -sovellus. Tämän jälkeen sovellus suoritetaan siirtämällä kontrolli bootstrap-tiedostolle. [12, s. 8-10; 13, s. 30-33.]

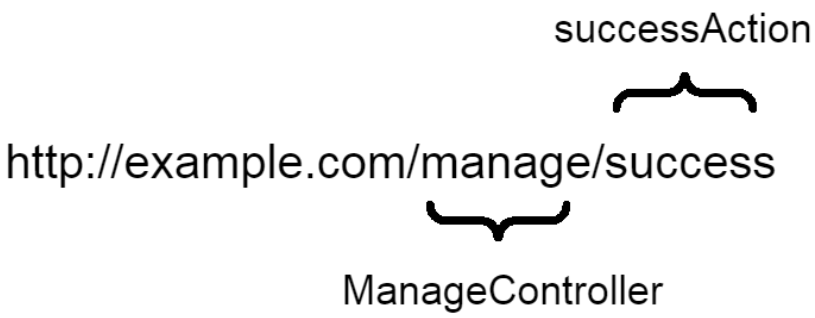
Application-kansio sisältää sovelluksessa käytettävät mallit, ohjaimet ja näkymät. Näiden lisäksi siellä sijaitsee Bootstrap.php-tiedosto, jonka tehtävänä on alustaa kaikki käyttäjän määrittelemät Zend Framework -komponentit. Bootstrap-tiedosto on myös hyvä paikka määritellä reitit, joiden perusteella HTTP-pyynnöt ohjataan oikeille ohjaimille. Sovelluksen asetukset määritellään configs-kansion application.ini-tiedostossa. Tämän tiedoston avulla kehittäjä pystyy määrittelemään sovelluksen asetukset eri ympäristöihin, missä sovellusta käytetään. Kehittäjä voi application.ini-tiedostossa esimerkiksi määritellä tuotanto- ja kehitysympäristöjen asetukset erikseen. [12, s. 10-12.]

Library-kansio luodaan myös Zend Tools -työkalun toimesta. Kansio on tarkoitettu sisältämään kaikki Zend Frameworkin komponentit. Library-kansiota suositellaan myös käyttämään kolmansien osapuolien kirjastojen säilytyspaikkana. [12, s. 10.]

4.2 Ohjain

Ohjaimet ovat PHP-luokkia, jotka vastaanottavat HTTP-pyyntöjä ja ohjaavat niitä metodeille (actions). Metodit vastaanottavat pyyntöjä ja prosessoivat niistä saamansa datan metodin halutun tavan mukaisesti. Prosessointiin kuuluu yleensä datan hakeminen malleista ja sen asettaminen näkymille. [14.]

Kun palvelin saa projektiin kohdistuvan HTTP-pyynnön, sen URL jaetaan osiin. Näistä osista päätellään tavoiteltu ohjain sekä metodi. Kuva 6 esittää, kuinka Zend Framework purkaa URL-osoitteen ohjaimeen ja metodiin. Kuvassa esitetty HTTP-pyyntö ohjataan oletuksena ManageController-ohjaimelle, joka puolestaan ohjaa pyynnön successAction-metodille.



Kuva 6. HTTP-pyynnön ohjaaminen

Kuvan 6 tapaus on yksinkertainen esimerkki tavallisten HTTP-pyyntöjen ohjaamisesta Zend Frameworkilla. URL-osoitteet eivät kuitenkaan aina koostu samanlaisista osista, joten Zend Framework mahdollistaa myös reittien määrittelemisen mukautetusti. Mukautettujen reittien määritteleminen on hyvä tapa kirjoittaa siistiä ja havainnollista koodia. Näitä reittejä käyttämällä tehdään sovelluksen kehittämisestä ja käyttämisestä selkeää sekä muille kehittäjille että loppukäyttäjille. Reittien mukautettua määrittelemistä käsitellään lisää myöhemmin tässä luvussa.

Tapaukset, joissa URL:stä puuttuu kokonaan viimeinen osa eli metodi, ohjataan indexAction-metodille. Esimerkiksi URL ”http://example.com/manage/” suunnattaisiin ManageController-ohjaimen indexAction-metodille. Samoin toimitaan ohjaimen kanssa, jos URL-osoitteesta puuttuu sekä metodi että ohjain. Aiemman esimerkin mukaisesti URL ”http://example.com” ohjattaisiin IndexController-ohjaimen indexAction-metodille. IndexController-ohjain luodaan automaattisesti Zend Tools -työkalun toimesta, kuten kuva 5 osoittaa.

Ohjaimet käsittelevät myös HTTP-pyyntöjen mukana tulevat parametrit. Parametreihin pääsee käsiksi HTTP-pyyntöolion kautta, joka sisältää pyynnön URL-osoitteen, tavoitellun ohjaimen, metodin ja pyynnön mukana tulevat parametrit.

4.3 Näkymä

Ohjaimen suoritettua tehtävänsä metodissa se haluaa esittää loppukäyttäjälle näkymän, joka vastaa tätä metodia. Oikean näkymän löytämiseksi Zend Framework tarjoaa ViewRenderer-komponentin. ViewRenderer on toimintojen avustaja, joka auttaa näkymien hallinnoimisessa. ViewRenderer pitää kirjaa kaikista näkymistä, ja alustaa ne, jotta

käyttäjän ei tarvitse tehdä sitä manuaalisesti. Ilman käyttäjän tekemiä määrityksiä ViewRenderer linkittää aktiivisen ohjaimen ja metodin niitä vastaavaan näkymään. Kuvan 5 avulla voi jo päätellä, miten avustaja osaa käyttää projektin rakennetta hyödykseen näkymien hallinnoimisessa. Aiemmin mainittiin, kuinka HTTP-pyyntö osoitteeseen ”http://example.com/” ohjataan IndexController-ohjaimen indexAction-metodille. ViewRenderer-avustaja tulkitsee mikä ohjain ja metodi ovat kyseessä, ja hakee esitettävän näkymän niiden perusteella. IndexController-ohjaimen ja indexAction-metodin tapauksessa avustaja hakee näkymätiedoston polusta ”views/scripts/index/index.phtml”. [15.]

Kaikki näkymäskriptit suoritetaan Zend_View-luokan instanssin alaisina, joka määritellään ohjaimessa. Koska suoritettava näkymäskripti on Zend_View instanssin alainen, se pystyy viittaamaan siinä määriteltyihin muuttujiin. Tämän avulla näkymälle voi määritellä muuttujia ohjaimen puolelta, mikä mahdollistaa datan viemisen näkymään vaivattomasti. Seuraavassa esimerkissä ohjain määrittelee näkymälle taulukon, joka sisältää muutaman artistin:

```
$this->view->artists = array(“artist1”, “artist2”, “artist3”);
```

Näkymä pääsee siihen käsiksi seuraavasti:

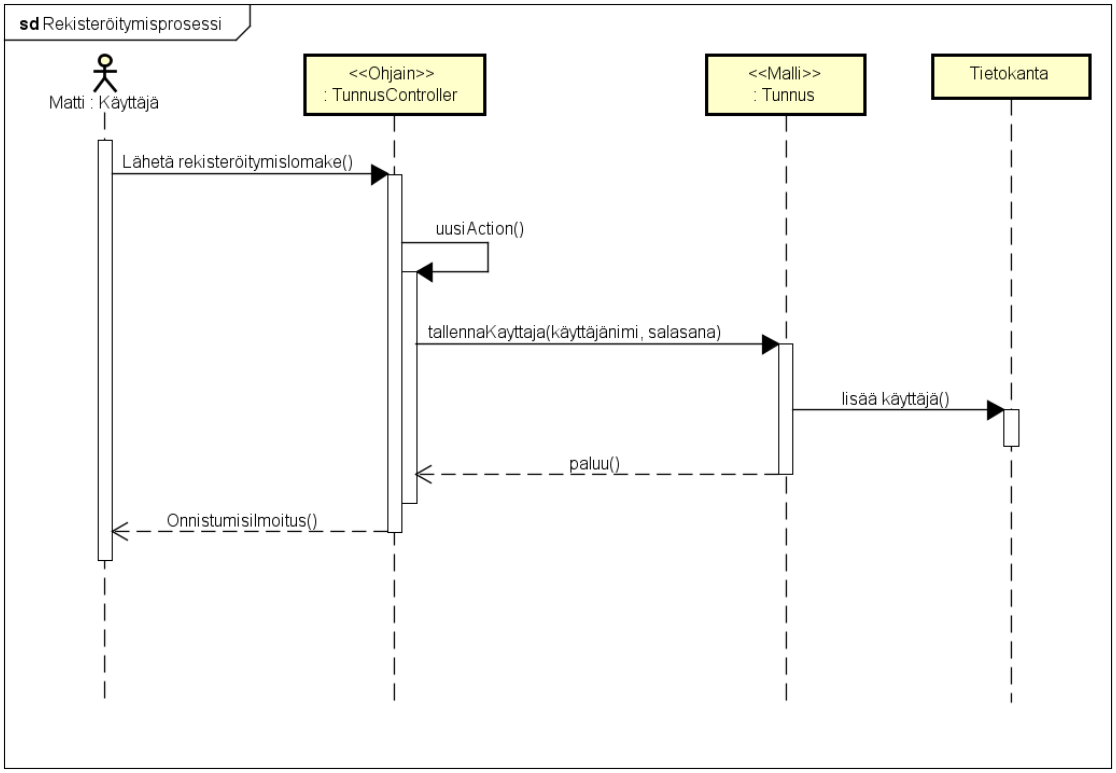
```
$this->artists;
```

4.4 Malli

MVC-arkkitehtuurin mukaisesti sovelluksen liiketoimintalogiikka sijoitetaan malleihin, jotta käyttöliittymät pysyvät erossa sovelluksen loogisista toiminnoista. Useimmiten mallit yhdistetään tietokantoihin, mikä on myös Zend Frameworkin tapa määritellä mallit. Zend Frameworkin MVC-toteutuksen mallit yhdistetään suoraan tietokannan tauluihin. Jokaisella tietokannan taululla on oma malli-luokka, joka käsittelee kyseistä taulua. Mallit sijoitetaan kuvassa 5 näkyvään application/models-kansioon. Malli-luokat perivät Zend_Db_Table_Abstract-luokan, joka tarjoaa olioperustaisen rajapinnan tietokannan tauluille. Rajapinta tarjoaa CRUD (create - luo, read - lue, update - päivitä, delete - poista) -funktioita tietokannan tavanomaiseen käyttämiseen. [12, s. 55-56.]

Zend Framework antaa kehittäjälle vapauden valita käytettävän tietokanta-adapterin. Zend_Db-komponentti tarjoaa tätä varten useita adaptereita eri relaatiotietokantojen ohjausjärjestelmille. Näitä adaptereita ovat esimerkiksi pdo_mysql, pdo_mssql ja mysqli. Sähköpostijärjestelmässä käytetty adapteri on MySQL:n PDO-adapteri. [16.]

Aiemmin läpikäydyssä ohjain-osuudessa seurattiin HTTP-pyynnön reittiä sovelluksessa. Malli on yleensä se luokka, mitä kontrolleri kutsuu tietokantaoperaatioiden tekemiseen. Ohjaimen voi siis asettaa tekemään tietokantaoperaatioita itsekin, mutta suositeltu tapa on kuitenkin sijoittaa tietokantojen hallinnoiminen kokonaan malleihin. Kun ohjain saa HTTP-pyynnön, esimerkiksi käyttäjän täyttämän rekisteröitymislomakkeen, halutaan useimmiten tehdä joitain tietokantaoperaatioita.



Kuva 7. Rekisteröitymisprosessin kulku

Uuden käyttäjän rekisteröitymisprosessi voisi näyttää kuvan 7 kaltaiselta tapaukselta. Käyttäjä Matti täyttää verkkosivun rekisteröitymislomakkeen ja painaa ”rekisteröidy”-painiketta. Lomake lähetetään palvelimelle, missä se otetaan vastaan ja lähetetään edelleen TunnusController-ohjaimelle. Ohjaimessa pyyntö ohjataan uusiAction()-metodille, joka ottaa HTTP-pyynnöstä POST-parametreina saadut käyttäjänimen sekä salasanan

talteen. Käyttäjän tiedot halutaan tallentaa tietokantaan, joten kutsutaan tietokannan tunnukset-taulun vastaavaa mallia - Tunnus-luokkaa. Ohjain kutsuu Tunnus-luokan tallennaKayttaja()-metodia antaen kutsun argumentteina käyttäjän tiedot. Tunnus-luokka on perinyt Zend_Db_Table_Abstract-luokan, joten se voi käyttää perityn rajapinnan insert-metodia tietueen lisäämiseksi tunnukset-tauluun. Tietueen lisääminen tunnukset-tauluun onnistuu, ja mallista palataan ohjaimeen. Mallista palautetaan tieto onnistumisesta, jolloin ohjain laukaisee onnistumisilmoituksen näkymässä.

4.5 Reititys

HTTP-pyynnön tavallinen reititys selitettiin pikaisesti luvussa 4.2. Kuvan 6 tapaus purettiin osiin ja selitettiin, kuinka HTTP-pyynnön URL-osoite jaetaan ohjaimeen ja metodiin. Tapaus oli hyvin yksinkertainen esimerkki tavanomaisesta tilanteesta, jossa pyyntö ohjataan URL-osoitteen mukaan oikealle ohjaimelle ja metodille. Usein kuitenkin tulee vastaan tilanteita, jolloin reititystä halutaan säädellä kehittäjän tapojen mukaisesti. Useimmiten toimintoja halutaan nimetä niitä kuvaavin nimin, eikä se aina ole sopivin tapa määritellä sivun URL-osoite. Näissä tapauksissa kehittäjä pystyy määrittelemään mukautettuja reittejä HTTP-pyyntöjen reitittämiseksi uudelleen.

Ilman mukautettua reittiä voimme myös olettaa, että kuvan 7 tapauksessa lähetetty HTTP-pyyntö menee osoitteeseen ”http://example.com/tunnus/uusi”. Reititys tapahtuu siis loogisesti sovelluksen rakenteen mukaan. Vastaan saattaa kuitenkin tulla tilanteita, joissa kehittäjä haluaa määritellä mukautettuja reittejä pyynnöille. Usein näin käy kun URL-osoitteesta tulee liian vaikeaselkoinen. Mukautettuja reittejä voidaan määritellä Zend_Controller_Router_Route-komponentin avulla. Tämä komponentti määrittelee Zend Frameworkin tavalliset reitit, jotka koostuvat dynaamisista sekä staattisista osuuksista. Sovelluskehys tarjoaa tavallisen Zend_Controller_Router_Route-komponentin lisäksi muita reititys-komponentteja, jotka mahdollistavat reitityksen esimerkiksi pelkkien staattisten arvojen tai säännöllisten lausekkeiden avulla. [17.]

```
$route = new Zend_Controller_Router_Route(
    'tunnus/:kayttajanimi',
    array(
        'controller' => 'tunnus',
        'action'     => 'haekayttaja'
    )
);

$router->addRoute('kayttaja', $route);
```

Koodiesimerkki 1. Mukautetun reitin määrittely

Koodiesimerkki 1 esittää mukautetun reitin määrittelemistä tilanteessa, jossa halutaan hakea käyttäjä HTTP-pyynnön parametrin mukaan. Zend_Controller_Router_Route:n konstruktorille annetaan kaksi argumenttia, joista ensimmäinen määrittelee, mitä URL:ää reitti koskee. Koodiesimerkissä 1 määritelty URL-osoite koostuu staattisesta ja dynaamisesta osasta. Staattinen osa ”tunnus” pysyy aina samana mutta dynaaminen muuttuja ”:kayttajanimi” saa arvonsa URL-osoitteesta. Koodiesimerkissä 1 laadittu reittimäärittely ohjaa kaikki ”http://example.com/tunnus/Matti” kaltaiset pyynnöt TunnusController-ohjaimen haekayttajaAction-metodille.

Toinen Zend_Controller_Router_Route-konstruktorin parametri on taulukko. Se sisältää ohjaimen ja metodin, joita kutsutaan, kun sovellus vastaanottaa HTTP-pyynnön ensimmäisessä parametrissa määriteltyyn URL-osoitteeseen. Kun pyyntö on ohjattu oikeaan ohjaimeen ja metodiin, sen parametreihin päästään käsiksi Zend_Controller_Request_Http-olion kautta. Olio sisältää HTTP-pyynnöille tyypilliset ominaisuudet, kuten GET- ja POST-parametrit. Koodiesimerkin 1 HTTP-pyyntöolio sisältää URL:ssä dynaamisena määritellyn osan, jonka avulla pystytään esimerkiksi rakentamaan käyttäjän omakohtainen näkymä. [17.]

4.6 Muita komponentteja

Aiemmin mainitut Zend Framework -komponentit liittyvät pitkälti sovelluksen rakenteeseen ja työnkulkuun. Ne ovat kuitenkin vain pieni osa sovelluskehyksen tarjoamista toiminnallisuuksista. Tämän osion tarkoitus on mainita muita hyödyllisiä komponentteja, jotka tarjoavat verkkosovelluksille tyypillisiä ominaisuuksia.

Zend_Registry

Zend_Registry-komponentti tarjoaa koko sovellukselle yhtenäisen tietovaraston. Se on käytännössä luokka, joka tarjoaa staattisia metodeja tiedon tallentamiseksi ja sen hakemiseksi. Varastoon voidaan tallettaa olioita ja muuttujia, jotka ovat sitten käytössä kaikkialla sovelluksessa. Muuttujat ja oliot tallennetaan avain-arvo-pareina, jolloin varastosta hakeminen tapahtuu avaimen perusteella. [18.]

Oletuksena Zend_Registry on luokka, josta on vain yksi ilmentymä, ja tämä ilmentymä toimii varastona talletettavalle tiedolle. Vaihtoehtoisesti luokasta voi tehdä sen konstruktorin avulla uusia ilmentymiä, ja näitä ilmentymiä voi käyttää assosiatiivisen taulukon tai olion mukaisesti. [18.]

Zend_Form

Lomakkeiden käsittelyä varten sovelluskehys tarjoaa Zend_Form-komponentin. Tämä muuttaa lomakkeiden käsittelyn olioperustaiseksi ja nopeuttaa lomakkeiden rakentamista. Loppukäyttäjältä syötteitä odottava verkkosovellus on aina vaarassa hyökkäyksille, ja siksi kaikkiin loppukäyttäjien tuottamiin syötteisiin täytyy aina osata varautua ennalta. Zend_Formin avulla lomakkeiden suodattamista ja validoimista ei välttämättä tarvitse edes tehdä itse, sillä Zend Frameworkin sisäänrakennetut toiminnot tarjoavat myös nämä toiminnallisuudet. Lomake tehdään luomalla Zend_Form-luokan ilmentymä, jolle pystytään asettamaan kaikki tavallisen lomakkeen attribuutit olion metodeilla. [13, s. 116-122.]

```
$lomake = new Zend_Form();
$lomake->setAction('success');
$lomake->setMethod('post');
$lomake->setAttrib('id', 'kirjautumisLomake');
$this->view->lomake = $lomake;
```

Koodiesimerkki 2. Kirjautumislomakkeen luominen Zend_Formilla

Koodiesimerkissä 2 Zend_Form-luokasta tehtiin uusi ilmentymä ja siihen lisättiin ominaisuuksia olion metodien avulla. Viimeisellä rivillä lomake annetaan näkymän käyttöön, jolloin siihen voi viitata näkymän puolella seuraavanlaisesti:

```
<?php echo $this->lomake; ?>
```

PHP-tulkin käännettyä koodin rivin tilalla on lomakkeen html-toteutus:

```
<form action=”success” method=”post” id=”kirjautumisLomake”></form>
```

Lomakkeelle asetetaan elementtejä addElement()-metodilla. Tämä metodi saa parametreinaan elementin tyypin ja sille annettavan nimen. Tyypin perusteella elementistä luodaan sen tyyppiä vastaava olio ja se saa nämä oliokohtaiset ominaisuudet käyttöönsä. Elementtityyppejä ovat esimerkiksi tekstikenttä, painike ja alasvetovalikko. Ominaisuuksien asettaminen elementeille tapahtuu samalla tavalla kuin lomakkeillekin.

Zend_Form-komponentti on varsin käytännöllinen toiminto, jos haluaa käsitellä lomakkeita oliopohjaisella tavalla. Oliopohjaisuus ei ole kuitenkaan ainoa syy, minkä vuoksi komponenttia kannattaisi käyttää. Zend_Form käyttää hyväkseen myös muita Zend Frameworkin toiminnallisuuksia, jotka helpottavat lomakkeiden käsittelyä. Näistä toiminnallisuuksista ehkäpä tärkein on validointi.

Zend_Validate

Tiedon luottamuksellisuuden ja eheyden takaamiseksi lomakkeet täytyy aina validoida. Validoinnin kirjoittaminen jokaiselle elementille ei aina ole välttämättä tehokkain ratkaisu. Siksi Zend Framework tarjoaa valmiiksi rakennettuja validointiluokkia koko lomakkeen tai erillisten elementtien tarkistamiseksi.

Kokonaisen lomakkeen pystyy validoimaan yksinkertaisesti kutsumalla Zend_Form-olion isValid()-metodia. Metodi saa parametrina validoitavan tiedon lähteen, joka on GET tai POST. Lomakkeen elementit tarkistetaan sitten suorittamalla jokaisen elementin tyyppikohtainen validaattori. Yhdenkin validaattorin epäonnistuminen tarkoittaa koko lomakkeen epäonnistumista, jolloin lomake voidaan palauttaa käyttäjälle ilmoitusten kanssa. [13, s. 124.]

Lomakkeen jokaisen kentän validoiminen ei ole välttämättä tarpeellista aina. Esimerkiksi sähköpostijärjestelmässä halutaan tarkistaa, että vastaanottajan sähköpostiosoite on ilmoitettu standardien mukaisessa muodossa. Yksittäisen elementin validointi suoritetaan asettamalla elementille validaattori addValidator()-metodilla. Kutsun yhteydessä argu-

menttina annetaan haluttu Zend-validaattori. Zend_Validate-komponentti sisältää suuren määrän erilaisia validaattoreita eri käyttötapauksiin. Sähköpostiesimerkissä elementin validaattorin voisi määritellä seuraavasti: [13, s. 130-133.]

```
$emailElement->addValidator(new Zend_Validate_EmailAddress());
```

Luvussa esitetyt komponentit ovat vain osa Zend Frameworkin laajaa komponenttikirjastoa. Mainitut komponentit antavat lukijalle hyvän yleiskuvan siitä, mitä sovelluskehyksellä voidaan saada aikaan.

5 Sähköpostijärjestelmän määrittely

5.1 Toiminnallinen määrittely

Sähköpostijärjestelmän toiminnallisessa määrittelyssä lähdetään liikkeelle siitä, että järjestelmä integroituu osaksi SoSe Zeed -työkalua. Sähköpostijärjestelmä tulee käyttöön kaikille työkalun käyttäjille, eli kampanjoita laativille henkilöille. Sähköpostijärjestelmästä ei haluta tehdä massiivista järjestelmää, vaan sen halutaan sisältävän vain ja ainoastaan kampanjan mainostamisen kannalta tärkeät ominaisuudet. Sähköpostijärjestelmän täytyy toimia kampanjakohtaisesti, mikä erottaa sen tavallisista, käyttäjäkohtaisista sähköpostijärjestelmistä. Kampanjakohtaisesti siitä syystä, että käyttäjä pystyy luomaan useita kampanjoita, joilla voi olla samoja markkinointikontakteja. Samojen kontaktien kanssa voidaan siis käydä keskustelua eri kampanjoista, eivätkä ne mene sekaisin sähköpostijärjestelmässä.

Olennaisinta kampanjan mainostamisen osalta on pystyä lähettämään sähköpostiviestejä kampanjaan asetetuille markkinointikontakteille. Viestien lähettäminen kontakteille oli toteutettu jo vanhassa sähköpostijärjestelmässä, mutta järjestelmälle tahdottiin lisää toimintoja. Kampanjan levittäminen yksitellen kontakteille olisi hidasta ja työlästä, joten viestejä täytyy pystyä lähettämään halutuille kontakteille samanaikaisesti. Massaviestien lähettäminen on kuitenkin huono tapa, joten tällaisia viestejä halutaan pystyä yksilöimään kontaktikohtaisesti. Massaviestien yksilöiminen tekee markkinointikontakteille lähetetyistä viesteistä henkilökohtaisempia, ja vastedes auttaa vuoropuhelun aloittamista heidän kanssaan.

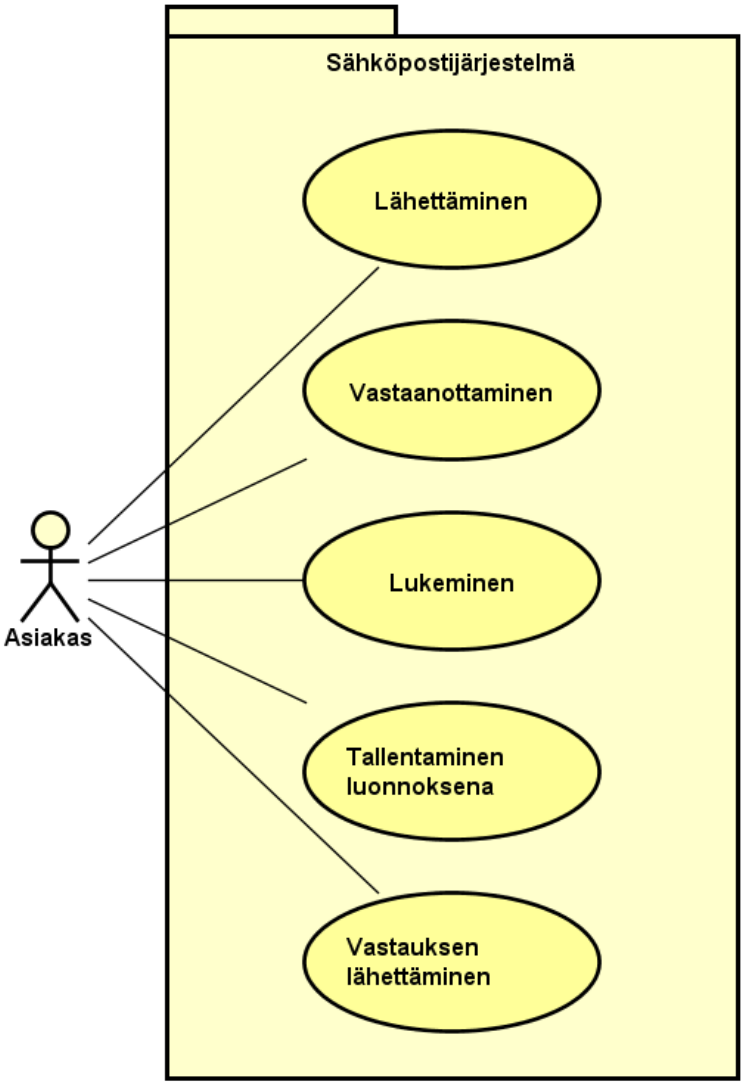
Kampanjan laatijan ja yksittäisen markkinointikontaktin vuorovaikutuksen mahdollistamiseksi halutaan myös vastaanottaa viestejä näiltä kontakteilta. Vastaanotetut viestit yhdistetään kampanjaan, jolloin tiedetään, mihin kampanjaan viesti liittyy. Toisiinsa liittyvistä viesteistä rakennetaan viestiketjuja, jotta sähköpostijärjestelmän käyttäjä pystyy helpommin seuraamaan markkinointikontaktien välillä käytävää keskustelua.

Sähköpostijärjestelmän ensimmäiseen version halutut toiminnallisuudet rajattiin niin, että vain vuorovaikutuksen kannalta välttämättömimmät toiminnallisuudet otettiin mukaan. Ensimmäisessä versiossa haluttiin pystyä lähettämään ja vastaanottamaan viestejä,

sekä vastaamaan markkinointikontakteilta saatuihin viesteihin. Näistä viesteistä rakennettaisiin viestiketjuja, jotka esitetään sähköpostijärjestelmän näkymässä. Viestejä pitää pystyä myös tallentamaan luonnoksina tulevaa käyttöä varten. Sähköpostiviestiä kirjoittaessa tekstieditorin täytyy sisältää vain tärkeimmät tekstin muokkaamisen työkalut. Käyttöliittymän haluttiin siis olevan mahdollisimman yksinkertainen ja selkeä, jättäen kaikki ylimääräinen toiminnallisuus pois. Ensimmäisestä versiosta rajattiin pois muutamia varsin tarpeellisia toiminnallisuuksia, kuten kaikille vastaaminen ja viestin välittäminen. Nämä toiminnot menevät jatkokehityksen puolelle.

5.2 Käyttötapaukset

Toiminnallisten määrittelyjen selkeyttämiseksi sähköpostijärjestelmän käyttötapauskaavio esitetään kuvassa 8. Asiakas on se henkilö, joka käyttää SoSe Zeed -työkalun sähköpostijärjestelmää kampanjoiden mainostamiseksi.



Kuva 8. Käyttötapauskaavio

Ensimmäinen käyttötapaus, lähettäminen, toteutuu, kun kampanjan laatija kirjoittaa uutta viestiä kontakteille. Sähköpostiviestin lähettäminen vaatii onnistuakseen, että pakolliset kentät on täytetty, ja viestille on valittu vastaanottajat. Lähettämisen onnistuttua viesti lisätään lähetyt-kansioon.

Vastaanottaminen tapahtuu, kun markkinointikontakti vastaa kampanjan laatijan lähettämään viestiin. Sähköpostijärjestelmä etsii vastaanotetun viestin otsakkeista tiedon, mihin kampanjaan se liittyy ja ohjaa viestin oikealle kampanjalle. Jos vastaanotetusta viestistä ei löydy kampanjan tietoja, se tallennetaan tietokantaan ilman yhteyksiä mihinkään kampanjaan. Vastaanotettu kampanjaan liittyvä viesti lisätään sitten saapuneet-kansioon lukemattomana.

Sähköpostiviestin avaaminen sen lukemiseksi tapahtuu valitsemalla haluttu viesti sähköpostilaatikosta. Lukunäkymään voi avata lähetettyjä ja vastaanotettuja viestejä, jolloin kaikki valittuun viestiin liittyvät viestit, eli viestiketju, esitetään näkymässä.

Sähköpostiviestejä voidaan myös tallentaa luonnoksina. Viesti tallennetaan, kun käyttäjä painaa viestin muokkausnäkymässä sijaitsevaa tallennus-painiketta. Tallennettava viesti voi olla uusi viesti tai vastaus toiseen viestiin. Tallennettaessa viestin sisältö ja vastaanottajat tallennetaan tietokantaan, minkä jälkeen käyttäjälle esitetään viesti tallennuksen onnistumisesta näkymässä. Viesti lisätään luonnokset-kansioon uutena luonnoksena, mistä käyttäjä pystyy avaamaan viestin sen muokkaamisen jatkamiseksi.

Vastauksen lähettäminen toimii muuten samalla tavalla kuin lähettäminen, paitsi viestin otsakkeisiin lisätään tieto siitä, mihin viestiin lähetettävä viesti on vastaus. Viestiin voidaan vastata, kun painetaan luettavan viestin ”vastaa”-painiketta. Prosessin kulku menee samalla tavalla kuin lähettämisessä: Käyttäjän täyttämät kentät tarkistetaan, ja viesti lisätään lähetetyt-kansioon. Tässä mainitut tilannekuvaukset on esitelty tarkemmin liitteessä 1.

5.3 Tekninen määrittely

Pohjana koko järjestelmälle toimii Zend Framework -sovelluskehys, jonka päälle SoSe Zeed on rakennettu. Sähköpostijärjestelmä integroidaan SoSe Zeed -työkaluun, ja sähköpostijärjestelmä käyttää työkalun palveluita myös omiin tarpeisiinsa. Käyttöliittymä erotetaan sovellusaluetiedosta Zend Frameworkin MVC-toteutuksen avulla.

Vanhasta toteutuksesta uudelleen käytetään näkymän luontiin liittyviä osia. Esimerkiksi ohjaimessa haettavat markkinointikontaktit ja niiden asettaminen näkymälle pysyy käytössä. Myös vanhan toteutuksen mallissa sijaitsevat sähköpostin lähetysskriptit toimivat pohjina uudelle toteutukselle. Käyttöliittymän ulkoasu vaihtuu kokonaan jättäen hyvin vähän alkuperäisestä toteutuksesta mukaan.

Zend Framework on PHP-sovelluskehys, joten järjestelmä on pääosin PHP-pohjainen. Järjestelmän mallit sekä ohjaimet toteutetaan täysin PHP:lla, mutta näkymässä käytetään hyväksi HTML5:n ja PHP:n lisäksi JavaScriptiä. JavaScriptillä toteutetaan suurin osa käyttöliittymän dynaamisista ominaisuuksista.

Sähköpostiviestien lähettämiseen ja vastaanottamiseen käytetään Mandrill-sähköpostipalvelua. Vanhassa sähköpostijärjestelmässä viestien lähettäminen oli toteutettu Mandrill-palvelulla, eikä ollut tarvetta vaihtaa sitä, joten myös uuden järjestelmän toteutuksessa käytettiin tätä palvelua. Mandrill on oiva vaihtoehto SoSe Zeed -työkalun kaltaiselle järjestelmälle, joka vaatii paljon automatisoitujen ja personalisoitujen sähköpostiviestien lähettämistä. Sähköpostiviestien lähettämisen ja vastaanottamisen ulkoistaminen Mandrillin kaltaiselle palvelulle ratkaisee useita ylläpito-ongelmia. Ulkoistaminen säästää huomattavasti kehittäjien aikaa sähköpostien lähettämiseen ja vastaanottamiseen liittyvissä askareissa.

Sähköpostijärjestelmän tekstieditorina käytetään TinyMCE-kirjastoa. Tekstieditoria muokattiin SoSe Zeed -työkalun tyyliin sopivaksi, ja sille asetettiin vaatimusten mukaisesti vain välttämättömimmät toiminnot, kuten kuvassa 4 näkyy.

Sähköpostiviesti liittyy aina kampanjaan ja markkinointikontaktiin. Tietokannassa sähköposteja käsitteleviä tauluja on kaksi: Sähköpostin sisältö, otsakkeet ja kampanjatunnus tallennetaan emails-tauluun. Sähköposti linkitetään aina myös markkinointikontaktiin, joita voi olla yhdellä sähköpostilla useita. Toinen taulu email_seeding_points linkittää sähköpostit kontakteihin ja sisältää vastattavan viestin tunnuksen. Sähköpostijärjestelmän näkymän luonnissa käytetään muistakin tauluista saatavia tietoja, jotka liittyvät kampanjaan ja siihen liitettyihin markkinointikontakteihin.

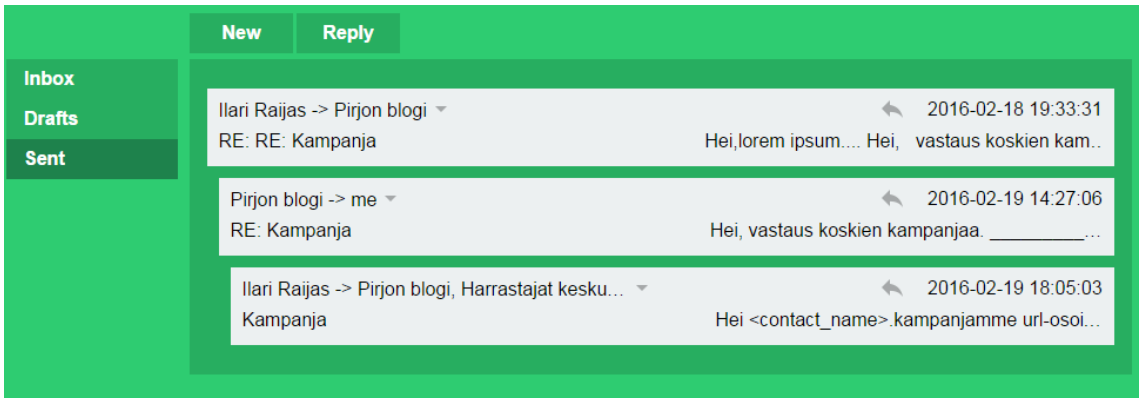
6 Toteutus

6.1 Käyttöliittymä

SoSe Zeed -työkalun sähköpostijärjestelmältä vaadittiin persoonallista, mutta kuitenkin yksinkertaista ulkoasua. Ulkoasun täytyisi mukailla työkalun tyyliä ja käyttää ympäröivää värimaailmaa. Suunnitteluvaiheessa vertailtiin suosittuja sähköpostijärjestelmiä ja valittiin niistä toiminnallisuuksia, joita työkalun sähköpostijärjestelmässä haluttiin käyttää. Kun toiminnallisuudet saatiin päätettyä, niistä tehtiin paperiluonnoksia prototyypin aikaansaamiseksi.

Sähköpostijärjestelmän käyttöliittymä koostuu sähköpostilaatikosta, joka sisältää saapuneet, lähetetyt ja luonnokset -kansiot. Kansioiden välillä navigoidaan sivun vasemmalla laidalla sijaitsevan valikon kautta. Kansioissa esitetään sähköpostiviestit allekkain uusimmasta vanhimpaan järjesteltyinä. Painamalla viestiä sen tiedot haetaan tietokannasta ja viesti avataan uudessa näkymässä. Saapuneet ja lähetetyt viestit avataan lukutilassa ja luonnokset suoraan muokkaustilassa.

Lukutilan näkymässä esitetään koko viestiketju, johon viesti liittyy. Viestiketju muodostuu, kun alkuperäiseen viestiin lähetetään vastauksia ja näihin taas lisää vastauksia. Viestiketjut ovat hierarkkisia, ja näkymässä tätä käytetään hyväksi esittämällä viestit siinä järjestyksessä, kun ne ovat viestiketjussa: Alkuperäinen viesti on alimmaisena ja siihen liittyvät viestit järjestyvät yläpuolelle muodostaen helposti ymmärrettävän viestiketjun. Viestiketjuista saattaa muodostua pitkiä, kymmenien viestien ketjuja. Jotta nämä pitkät viestiketjut eivät venyttäisi näkymää liian suureksi, kaikki viestit esitetään automaattisesti lyyhistyneinä (kuva 9). Lyyhistyneenä viestistä piilotetaan sen sisältö muutamia yksityiskohtia lukuun ottamatta. Viestistä näytetään lähettäjä, vastaanottaja, aihe, aika ja viestin sisällön ensimmäiset merkit. Lisäksi lyyhistyneessä viestissä on painikkeet kaikkien vastaanottajien näyttämiseksi sekä viestiin vastaamiseen.

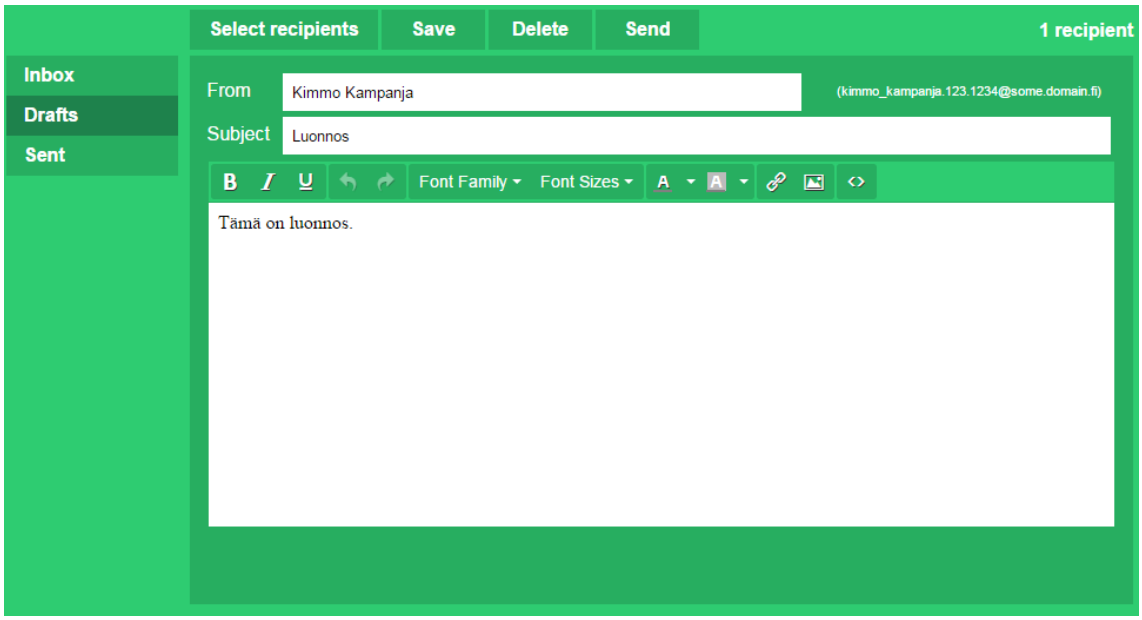


Kuva 9. Viestiketjun viestit lyyhistyneinä

Lyyhistynyttä viestiä painamalla se avautuu kokonaan ja siitä näytetään koko viestin sisältö (kuva 10).

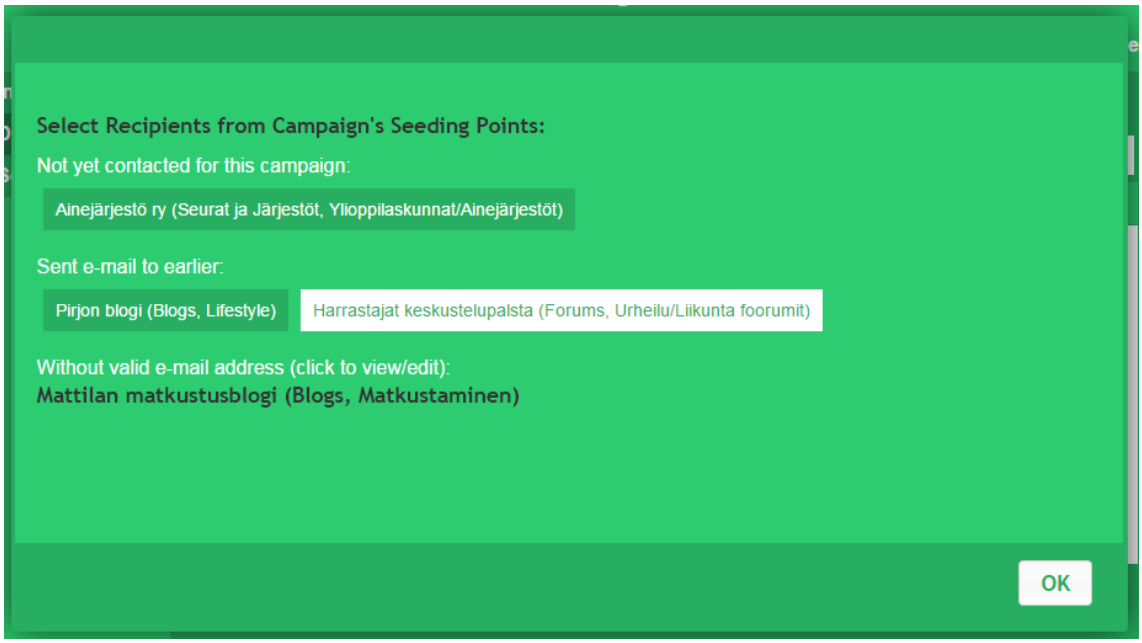
Kuva 10. Avattu viestiketjun viesti

Muokkaustilassa viesti avataan suoraan TinyMCE-tekstieditoriin. Luonnosta avattaessa sille haetaan tietokantaan tallennetut tiedot. Näitä tietoja ovat sähköpostin sisältö ja sille asetetut vastaanottajat, eli markkinointikontaktit. Uuden viestin aloittaminen hakee käyttäjän nimen automaattisesti ”from”-kenttään ja asettaa tekstieditoriin ohjetekstin.



Kuva 11. Luonnoksen avaaminen muokkaustilassa

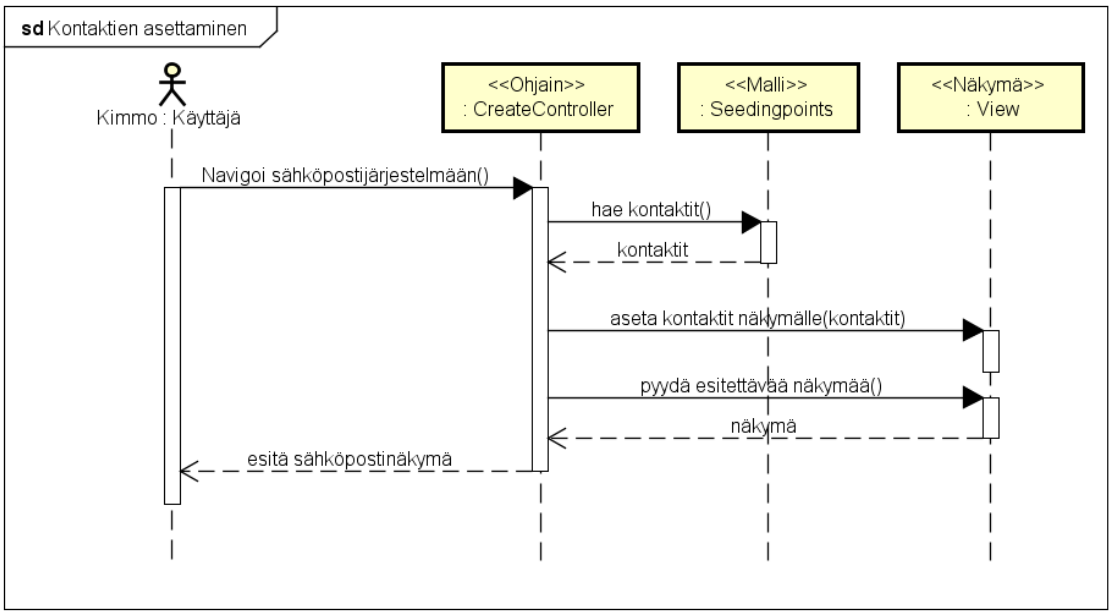
Kuva 11 esittää tilannetta, jossa käyttäjä on avannut luonnoksen. Kuvassa näkyy sähköpostille määriteltävät tiedot ja ylhäällä olevat luonnoksen toimintopainikkeet. Kuvasta huomataan, että käyttöliittymä ei vastaa täysin tavallista sähköpostijärjestelmää. SoSe Zeedin sähköpostijärjestelmässä käyttäjä pystyy itse määrittelemään lähettäjän sähköpostiosoitteen etuosan. Tähän kenttään haetaan automaattisesti viimeisimmässä lähetetyssä sähköpostissa käytetty nimi tai käyttäjän oma nimi, jos aiemmin käytettyä nimeä ei ole. Lähettäjän osoitteen loppuosan muodostaa järjestelmä, ja tätä tutkitaan myöhemmin sähköpostin lähettäminen -osiossa. Sähköpostijärjestelmän tarkoituksen mukaisesti ainoastaan kampanjaan lisätyille kontakteille voidaan lähettää sähköpostia. Tästä syystä käyttöliittymässä ei ole vastaanottaja-kenttää vastaanottajan syöttämiseksi, kuten tavallisissa, käyttäjäkeskeisissä sähköpostijärjestelmissä.



Kuva 12. Vastaanottajien valitseminen

Vastaanottajat valitaan yläpalkin ”Select recipients”-painiketta painamalla. Painikkeesta avautuu modaali, joka esittää kaikki kampanjalle valitut kontaktit (kuva 12). Kontaktit jaetaan kolmeen ryhmään: kontakteihin joihin ei ole otettu vielä yhteyttä kampanjan osalta, kontakteihin joihin on jo otettu yhteyttä ja kontakteihin joiden sähköpostiosoite ei ole kelvollinen.

Kampanjalle valitut kontaktit haetaan, kun käyttäjä siirtyy sähköpostijärjestelmään. Kontaktien haku- ja asetusprosessi on kuvattu kuvassa 13. Kun sovellus saa HTTP-pyynnön sähköpostijärjestelmään, CreateController-ohjain hakee tietokannasta kampanjan markkinointikontaktit. Kontaktit asetetaan sitten View-oliolle assosiatiivisina taulukoina, joihin näkymässä päästään suoraan käsiksi. Kuvan 12 modaali rakennetaan sitten näiden taulukoiden sisältöjen perusteella.



Kuva 13. Kontaktien asettaminen näkymälle

Käyttöliittymän dynaamiset toiminnallisuudet on toteutettu suurimmaksi osaksi JavaScriptillä. Painikkeille asetetut kuuntelijat suorittavat datan haku- ja tallennusoperaatioita sekä muita selaimeen vuorovaikutteisuutta lisääviä toimintoja. Sähköpostilaatikon viestit haetaan ja luonnokset tallennetaan Ajax-pyynnöillä. Ajax-pyynnöt on toteutettu jQuery-kirjaston jQuery XMLHttpRequest-olion avulla. Ajax-pyynnöt mahdollistavat HTTP-pyyntöjen lähettämisen ja HTTP-vastausten käsittelemisen palvelimelta ilman, että sivu latautuu uudelleen lomakkeen tavoin. Saapuneet, luonnokset ja lähetetyt -kansioiden sähköpostiviestit haetaan näillä Ajax-pyynnöillä, jolloin kansioiden näkymät rakennetaan lennosta.

6.2 Sähköpostin lähettäminen

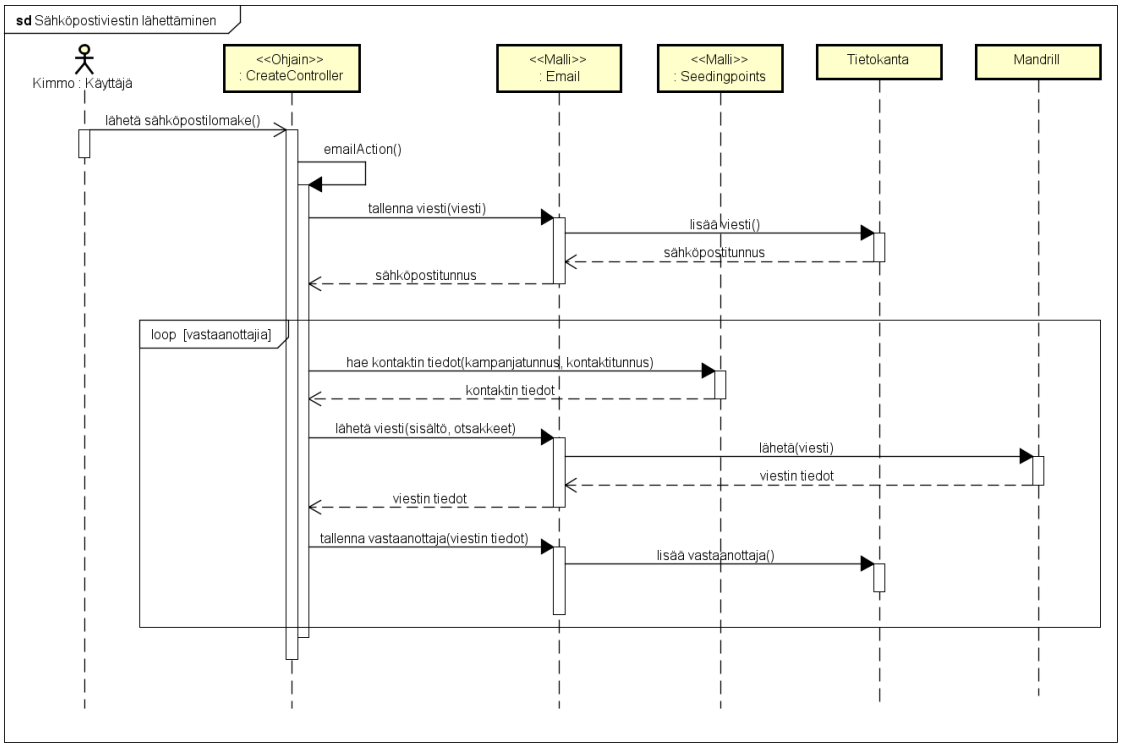
Sähköpostin lähettäminen on prosessi, joka koostuu käyttäjän syöttämien tietojen validoinnista, vastaanottajien tietojen hakemisesta tietokannasta, viestin lähetyksestä Mandrillin kautta ja viestin tallentamisesta tietokantaan. Validointi suoritetaan, kun käyttäjä painaa ”send”-painiketta. Validoinnissa tarkastetaan, että pakolliset kentät ovat täytetty ja syötetyt tiedot ovat vaaditussa muodossa. Jos validointi epäonnistuu, ilmoitetaan käyttäjälle tiedot virheellisestä syötteestä. Kun validointi onnistuu, sähköpostiviestin tiedot lähetetään lomakkeena eteenpäin. Pakollisia kenttiä ovat lähettäjä (from), aihe (subject) ja vastaanottajat (recipients).

Lähettäjän kenttään määritellään nimi, jonka halutaan näkyvän sähköpostin lähettäjänä. Sähköpostijärjestelmä rakentaa automaattisesti lähettäjän sähköpostiosoitteen käyttäjän määrittelemästä nimestä, kampanjan tunnuksesta sekä markkinointikontaktin tunnuksesta. Kuvassa 11 ”From”-kentän oikealla puolella sijaitseva info-teksti näyttää käyttäjälle järjestelmän luoman sähköpostiosoitteen. Sähköpostiosoitteeseen liitetyt tunnukset ovat välttämättömiä, jotta vastaanotettavat viestit pystytään yhdistämään kampanjoihin ja kontakteihin.

Lähetetty lomake ohjataan CreateController-ohjaimen emailAction-metodille, joka ottaa vastaan HTTP-pyynnön post-parametrit. Nämä parametrit sisältävät sähköpostiviestin sisällön ja vastaanottajat. Ohjaimessa viestin tiedot siistitään erikoismerkeistä, jotta tietokantaan ei pystytä injektoimaan koodia. Viestin tiedot tallennetaan tietokantaan, minkä jälkeen viesti lähetetään erikseen jokaiselle määritellylle vastaanottajalle. Vastaanottajat käydään yksitellen läpi toistorakenteessa, jossa vastaanottajan eli markkinointikontaktin nimi, sähköpostiosoite ja muut tiedot haetaan tietokannasta. Tietoja käytetään sähköpostiviestin sisällön personalisointiin ja otsakkeiden täyttämiseen.

Viesti lähetetään vastaanottajalle kutsumalla mallin lähetys-metodia. Kutsulle annetaan argumentteina viestin sisältö ja otsakkeet. Mallin, eli Email-luokan lähetysmetodissa saadaan parametreina lähetettävän viestin tiedot. Viesti lähetetään ulkoisen sähköpostienlähetyspalvelun Mandrillin kautta. Lähetettävä viesti alustetaan luomalla uusi ilmentymä Mandrill-luokasta ja asettamalla käyttäjäkohtaisia tietoja sille. Viesti lähetetään sitten mandrill-olion send()-metodilla, joka lähetyksen onnistuessa palauttaa lähetetyn viestin tiedot. Lähetetyn viestin tiedot palautetaan sitten ohjaimelle, joka viimeistelee lähetysprosessin tallentamalla lähetetyn viestin ja vastaanottajan tiedot tietokantaan.

Kaikki yllä mainitut vaiheet on kuvattu sekvenssikaavion avulla kuvassa 14. Kaavio havainnollistaa prosessin komponenttien väliset suhteet ja lähetysprosessin kulkujärjestyksen.



Kuva 14. Sähköpostin lähettäminen

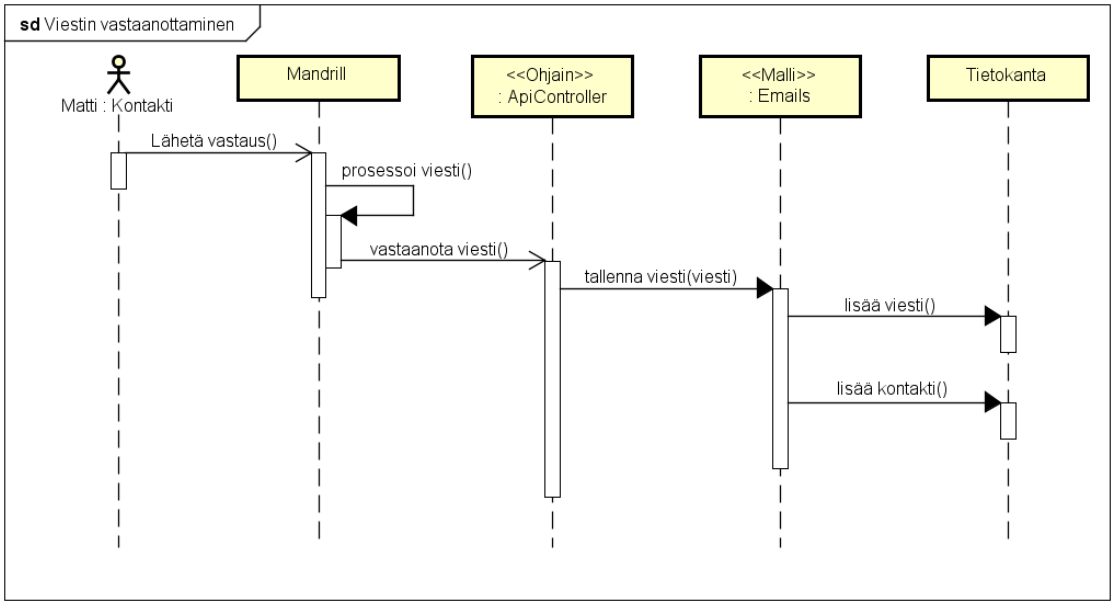
6.3 Sähköpostin vastaanottaminen

Vastaanottaminen tapahtuu, kun markkinointikontakti vastaa kampanjan laatijan lähettämään sähköpostiviestiin. Aiemmassa luvussa mainittiin, että lähetettäessä sähköpostia järjestelmä rakentaa lähettäjän sähköpostiosoitteen. Järjestelmä lisää osoitteeseen kampanjan tunnuksen, jotta vastaanotettaessa viesti se voidaan yhdistää kampanjaan. Osoitteessa määritelty toinen tunnus kuuluu markkinointikontaktille. Sen avulla järjestelmä pystyy yhdistämään sähköpostin lähettäjän markkinointikontaktiin. Näiden tietojen avulla pystytään siis pitämään huoli siitä, että sähköpostijärjestelmä toimii kampanjakohtaisesti, ja eri kampanjoiden sähköpostit ohjataan aina oikeille tahoille.

Sähköpostiviestien vastaanottaminen on määritelty Mandrill-palvelun tehtäväksi. Mandrill vastaanottaa ja prosessoi viestit, minkä jälkeen se lähettää sähköpostiviestin tiedot HTTP-pyyntönä SoSe Zeedille. Työkalu vastaanottaa Mandrillin tekemän pyynnön Api-Controller-ohjaimessa, ja välittää pyynnöstä saadut parametrit edelleen Email-malliluokan vastaanotto-metodille. Kuvan 15 sekvenssikaavio esittää vastaanottoprosessin eri vaiheet.

Mandrill on prosessoinut viestin niin, että sähköpostin otsakkeet, ja sisältö on jäsennelty valmiiksi. Vastaanotto-metodi saa nämä tiedot parametrina assosiatiivisen taulukon muodossa. Taulukosta etsitään ensimmäisenä vastaanottajan sähköpostiosoite, joka on tässä tapauksessa kampanjan laatija. Osoitteesta saadaan kampanjan ja markkinointi-kontaktin tunnukset, joita käytetään, kun viesti tallennetaan tietokantaan. Muut taulukossa olevat tiedot eivät vaadi esikäsittelyä, vaan niitä käytetään sellaisenaan. Otsakkeista otetaan lähettäjän tiedot, viestin tunnus ja vastatun viestin tunnus. Tunnuksien avulla sähköpostiviestit linkitetään toisiinsa, ja niistä pystytään rakentamaan viestiketjuja.

Sähköpostiviestistä saadut kampanja-, markkinointitunnus, otsaketiedot ja viestin sisältö tallennetaan sitten tietokantaan, josta sähköpostijärjestelmä hakee vastaanotetut viestit näkymään.



Kuva 15. Viestin vastaanottaminen

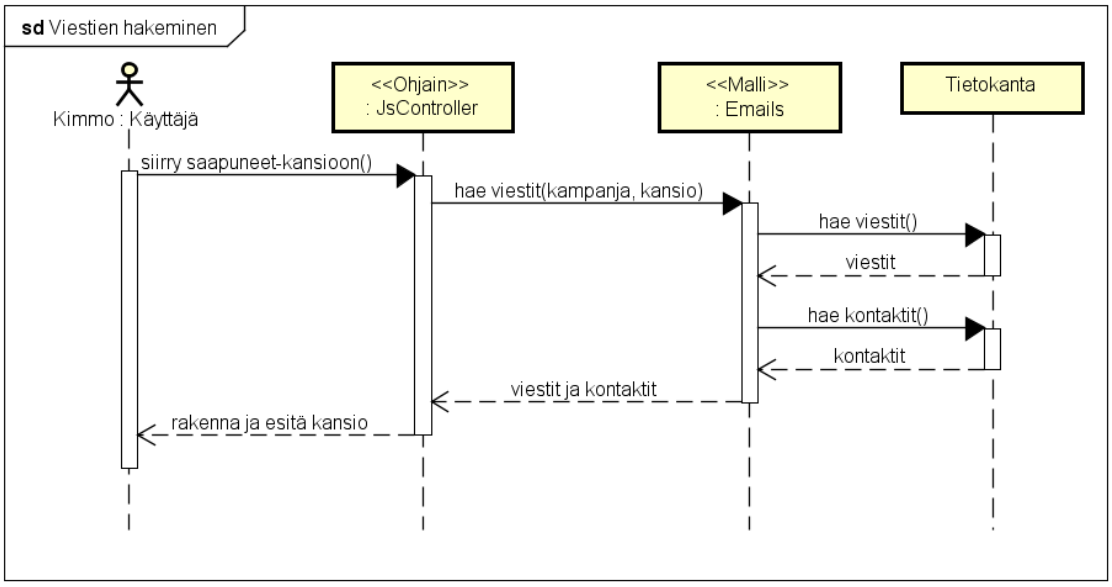
6.4 Sähköpostien hakeminen kansioon

Sähköpostijärjestelmän aloitusnäkymänä toimii saapuneet-kansio. Kun käyttäjä navigoi sähköpostilaatikkoon, järjestelmä listaa kaikki saapuneet viestit näkymään. Tietokantaan tallennettujen viestien hakeminen tehdään Ajax-pyynnöillä. Ajax-pyynnöt koostuvat URL-osoitteesta, johon pyyntö kohdistuu ja pyynnölle annettavista parametreista. URL-

osoite haetaan Zend_Registry-tietovarastosta, joka sisältää ympäristökohtaisen URL:n. Kehitys- ja tuotantoympäristöt sijaitsevat eri osoitteissa, jolloin HTTP-pyynnöt täytyy kohdistaa oikeisiin ympäristöihin. Asettamalla URL-osoite tietovarastoon, siihen voidaan viitata aina samalla tavalla ympäristöstä riippumatta. Tällöin tuotantoon siirron yhteydessä ei tarvitse muuttaa kuin tietovarastoon määriteltävä URL-osoite.

Viestien hakemiseksi Ajax-pyynnöille asetetaan parametreiksi kampanjan tunnus ja haettava kansio. Pyyntö ohjataan jälleen ohjaimen kautta, jossa sen parametrit otetaan vastaan ja lähetetään eteenpäin mallille. Malli saa parametrit vastaan ja käyttää niitä hakeakseen tietokannasta oikean kampanjan ja kansion viestit. Viesteille haetaan vielä niihin liittyvien markkinointikontaktien tiedot tietokannasta.

Kun jokaisen viestin kontaktitiedot on haettu, viestit lähetetään paluuarvona takaisin ohjaimelle. Ohjain muuttaa saadun PHP-taulukon JSON (JavaScript Object Notation) -muotoon, jotta sitä voi käyttää näkymässä JavaScriptillä. JSON-muotoinen tietorakenne palautetaan näkymälle, jolloin kutsutaan alkuperäisen Ajax-pyynnön callback-funktiota. Funktio saa parametrina vastaan tietokannasta haetut viestit, ja välittää ne edelleen kansion luontifunktiolle. Luontifunktion tarkoitus on rakentaa HTML-esitys haetuista viesteistä ja asettaa tämä esitys näkymään.



Kuva 16. Viestien hakeminen

Viestien hakeminen ja rakentaminen on kuvattu kuvassa 16. Käyttäjä aloittaa hakuprosessin painamalla valikosta kansiota, jonka sisällön hän haluaa näkyville. Ajax-pyynnöt ovat luonnoltaan asynkronisia, joten pyynnöt eivät häiritse selaimen toimintoja. Haun päätyttyä näkymä rakentaa saaduista viesteistä taulun, jonka se esittää käyttäjälle.

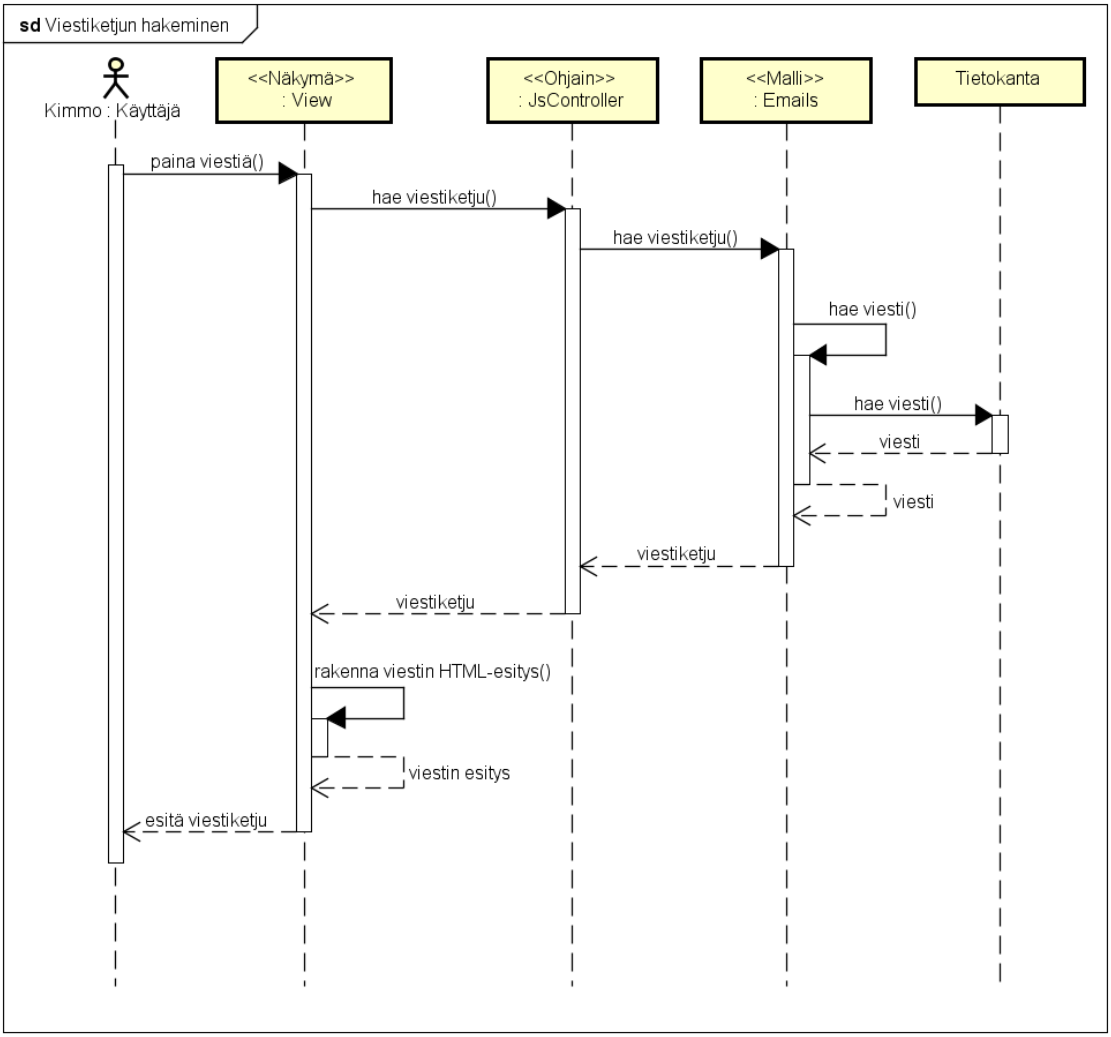
6.5 Sähköpostiviestin hakeminen ja avaaminen

Aivan kuten sähköpostiviestien hakeminen kansioon, viestin avaamiseenkin käytetään Ajax-pyyntöä. Viestin avaaminen eroaa koko kansion hakemisesta kuitenkin siten, että valitun viestin koko viestiketju haetaan esitettäväksi. Sähköpostiviesti avataan painamalla kansiossa listattua viestiä. Painalluksesta aiheutuu Ajax-pyyntö, jolle asetetaan haluttu URL-osoite ja parametrit. URL-osoitteella pyyntö kohdistetaan oikealle ohjaimelle ja metodille, ja parametrina pyynnölle asetetaan valitun viestin tunnus, joka saadaan viestin HTML-elementistä.

Hakuprosessi etenee samalla tavalla kuin koko kansion viestejä haettaessa: ohjain ottaa pyynnön parametrit vastaan ja kutsuu edelleen mallin haku-metodia. Mallissa viestin hakeminen ei kuitenkaan ole enää niin yksinkertaista, sillä valitun viestin koko viestiketju täytyy hakea. Viesteille on määritelty ”email_parent”-kenttä, joka ilmoittaa hierarkiassa ylempänä olevan viestin. ”Email_parent”-kenttä osoittaa siis sen viestin, johon kyseinen viesti on vastaus. Jos viesti ei ole vastaus toiseen viestiin, ”email_parent”-kentän arvo on 0. Kenttää seuraamalla pystytään hakemaan kaikki viestiketjun viestit käänteisessä järjestyksessä.

Viestiketjun rakentaminen tapahtuu hakemalla yksitellen viestiketjun viestit tietokannasta. Haut tehdään rekursiivisessa funktiossa, joka palauttaa käsiteltävänä olevan viestin. Funktiota kutsutaan niin kauan, kunnes käsiteltävän viestin ”email_parent”-kenttä on 0. Lopulta, kun vastaan tulee alkuperäinen viesti, eli viesti jonka ”email_parent”-kenttä on 0, aletaan kutsujonoa purkamaan. Funktiosta palautettu taulukko koostuu siten viesteistä, jotka sisältävät lisää viestejä. Tämä sisäkkäisistä viesteistä koostuva taulukko palautetaan ohjaimelle, joka taas palauttaa sen JSON-muodossa näkymälle. Ajax-pyynnön callback-funktion parametrina saadaan jälleen tietorakenne, joka välitetään viestiketjun luontifunktiolle.

Luontifunktio rakentaa viestiketjun HTML-esityksen hieman eri tavalla kuin kansion listausta rakennettaessa. Siinä missä kansion listaus rakennettiin normaalissa toistorakenteessa, viestiketju rakennetaan taas hyväksikäyttäen rekursiota. Rekursio toimii viestiketjun rakentamisen kannalta hyvin, sillä viestit pystytään esittämään helposti toistensa alaisina. Viestien hierarkia näkyy kuvissa 9 ja 10: Ulommaisin, ylhäällä sijaitseva viesti on kansiossa valittu viesti. Sen sisällä on aiemmat viestit, joihin valittu viesti liittyy. Luontifunktiossa viestit käydään jälleen läpi, kunnes ”email_parent”-kenttä osoittaa käsiteltävän viestin olevan alkuperäinen viesti. Vuorossa olevasta viestistä rakennetaan HTML-esitys, joka palautetaan kutsujalle. Palautettujen viestien HTML-esitykset muodostavat yhdessä viestiketjun, joka sitten asetetaan näkymään.



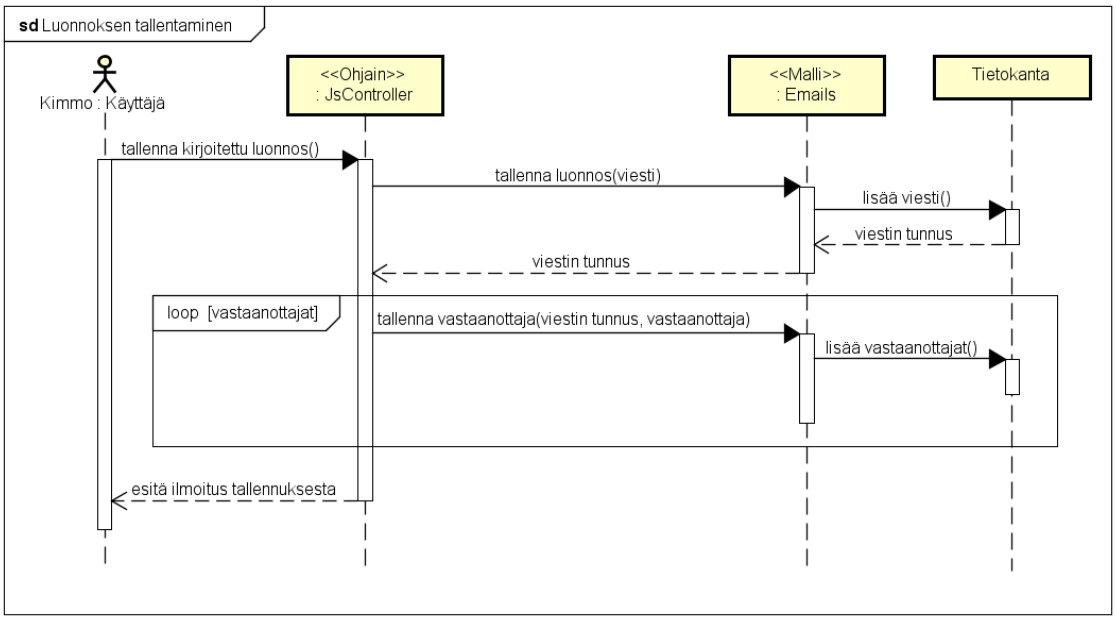
Kuva 17. Viestiketjun hakeminen

Kuvan 17 kaaviossa on esitetty mainitut viestiketjun hakuprosessin vaiheet. Käyttäjä aloittaa prosessin painamalla kansiossa listattua viestiä, jolloin järjestelmä lähettää Ajax-pyynnön ohjaimelle. Viestiketju rakennetaan mallissa, josta se palautetaan ohjaimen kautta näkymään. Näkymässä viestiketju käsitellään ja esitetään käyttäjälle.

6.6 Luonnosten tallentaminen

Tulevaa käyttöä varten kirjoitettu sähköpostiviesti täytyy pystyä tallentamaan. Luonnoksen tallennus tapahtuu pitkälti samalla tavalla kuin lähetetyn viestin tallennus. Tallentamistavat eroavat siinä, että luonnos tallennetaan Ajax-pyynnön kautta, kun taas viesti lähetetään lomakkeella. Viestin tallennus tietokantaan ei kuitenkaan eroa muulla tavalla, sillä rakenteeltaan luonnokset ovat täysin samanlaisia kuin muutkin viestit.

Tallennettaessa viestiä sille asetetaan ominaisuus, joka kertoo sen tyypin. Viesti voi olla tyypiltään lähetetty, vastaanotettu tai luonnos. Tämän ominaisuuden avulla pystytään tunnistamaan viestit, kun haluttu kansio avataan. Esimerkiksi luonnokset-kansiota avattaessa näkymään listataan vain ne viestit, jotka ovat tyypiltään luonnoksia.



Kuva 18. Luonnoksen tallentaminen

Kuvan 18 kaavio muistuttaa pitkälti viestin lähettämisen kaaviota (kuva 14). Luonnoksen tallentaminen on kuitenkin lyhyempi prosessi, sillä siinä toteutuu pelkästään viestin ja

vastaanottajien tallennus tietokantaan. Viesti tallennetaan ensin tietokantaan kuvan 18 mukaisesti, ja ohjaimelle palautetaan viestin tunnus. Vastaanottajat tallennetaan sitten yksitellen antamalla kutsuttavan metodin parametreina viestin tunnus ja vastaanottajan tiedot. Vastaanottajien tallentamisen jälkeen käyttäjälle esitetään ilmoitus tallennuksen onnistumisesta. Kun käyttäjä avaa luonnokset-kansion, tallennettu luonnos on listattuna kansion näkymässä, ja sen pystyy uudelleenavaamaan muokkausta tai lähetystä varten. Kun käyttäjä lähettää luonnoksen, sen tyyppi vaihtuu lähetetyksi. Lähetyksen jälkeen viesti siirtyy siis luonnokset-kansiosta lähetetyt-kansioon.

7 Yhteenveto

Insinöörityön tavoitteena oli toteuttaa uusi sähköpostijärjestelmä SoSe Zeed -markkinointityökalulle. Sähköpostijärjestelmän oli tarkoitus korvata vanha järjestelmä, joka oli toiminnallisuuksiltaan asiakkaiden tarpeille riittämätön. Sähköpostijärjestelmä kehitettiin osaksi SoSe Zeed -työkalua, joka on toteutettu Zend Framework -sovelluskehyksen päälle. Zend Framework -sovelluskehyksen lisäksi sähköpostijärjestelmän toteutuksessa käytettiin muita kolmansien osapuolten kirjastoja ja toimintoja, joilla mahdollistettiin nopea ja ketterä kehitystyö.

Insinöörityö aloitettiin esittämällä SoSe Zeed -markkinointityökalu, ja tutustumalla sen toimintoihin. Työkalun toimintojen ja merkityksen ymmärtäminen oli välttämätöntä työssä käytettävien termien ymmärtämiseksi. Työssä esiteltiin projektissa käytettyjä tekniikoita ja syvennyttiin Zend Framework -sovelluskehyksen toiminnallisuuksiin siltä osin, kuin ne olivat projektin kannalta tärkeitä.

Projekti aloitettiin toiminnallisilla määrityksillä, joilla esiteltiin järjestelmältä halutut toiminnallisuudet. Toiminnallisuudet rajattiin tarkasti, jotta järjestelmä ei paisuisi liian suureksi ja sähköpostijärjestelmän saisi nopeasti tuotantoon. Toiminnallisuuksien määrittelemisen jälkeen projektille tehtiin tekniset määrittelyt, joilla määriteltiin sähköpostijärjestelmässä käytettävät tekniikat. Sähköpostijärjestelmän toteutus tehtiin hyväksikäyttäen Zend Framework -sovelluskehyksen toimintoja, joista useat komponentit osoittautuivat varsin käytännöllisiksi. Projektin aikana sovelluskehyksen tarpeellisuus nousi esille, kun suurikokoiseen sovellukseen alettiin tekemään muutoksia ja lisäämään toiminnallisuuksia. Zend Frameworkin tarjoama MVC-toteutus jakoi projektin loogisiin kokonaisuuksiin, minkä ansiosta siihen oli helppo kehittää uusia toimintoja. Sovelluskehys tarjosi suuren määrän muitakin komponentteja, jotka helpottivat esimerkiksi kehitysympäristössä työskentelyä ja tietokannan hallitsemista.

Sovelluskehyksen käyttäminen vapauttaa kehittäjän useiden jo valmiiksi ratkaistujen ongelmien pähkäilemisestä ja antaa keskittyä itse sovelluksen toteuttamiseen. Sovelluskehykset nopeuttavat kehitystä, auttavat ylläpitämään sovelluksia ja tekevät sovelluksista modulaarisimpia. Nämä ominaisuudet nousevat enemmän esille sovelluksen kasvaessa suureksi, ja siksi moderni verkkosovellusten kehitys suosiikin sovelluskehysten käyttämistä.

Sähköpostijärjestelmän toteuttaminen oli hyvin monipuolinen projekti, jonka aikana opittiin verkkosovellusten ohjelmoinnin käytäntöjä ja tekniikoita. Varsinkin laajojen verkkosovellusten kehittämiseen tarkoitetut käytännöt ja eri ympäristöissä toimiminen toivat lisää ymmärrystä verkkosovellusten kehittämiseen. Projektissa perehdyttiin sähköpostien rakenteeseen ja opittiin ymmärtämään tarkemmin, mistä sähköpostijärjestelmät rakentuvat. Sähköpostien lähettämisestä ja vastaanottamisesta vastaava Mandrill-sähköpostipalvelu osoittautui varsin käytännölliseksi. Sähköpostijärjestelmän testaaminen helpottui huomattavasti Mandrillin tarjoamien toimintojen avulla, ja sen käyttäminen oli varsin vaivatonta.

Sähköpostijärjestelmä rajattiin sisältämään vain välttämättömimmät toiminnallisuudet. Tästä aiheutui se, että useat sähköpostijärjestelmille tutuista toiminnallisuuksista jätettiin ensimmäisestä julkaisusta pois. Nämä toiminnot kuten viestin välittäminen ja kaikille vastaaminen katsottiin menevän jatkokehityksen puolelle. Projektin toteutushetkellä niiden lisääminen sähköpostijärjestelmään ei kuitenkaan ollut vielä olennaista.

Lähteet

1 Mobile Marketing Statistics compilation. 2015. Verkkodokumentti. ComScore. <http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics/>. Luettu 27.2.2016

2 HTML. Verkkodokumentti. Wikipedia. <https://en.wikipedia.org/wiki/HTML>. Luettu 29.2.2015

3 Cascading Style Sheets. Verkkodokumentti. Wikipedia. <https://en.wikipedia.org/wiki/Cascading_Style_Sheets>. Luettu 29.2.2015

4 JavaScript basics. Verkkodokumentti. The Mozilla Developer Network. <https://developer.mozilla.org/en-US/Learn/Getting_started_with_the_web/JavaScript_basics>. Luettu 29.2.2015

5 JavaScript. Verkkodokumentti. Wikipedia. <https://en.wikipedia.org/wiki/JavaScript>. Luettu 29.2.2016

6 jQuery. Verkkodokumentti. Wikipedia. <https://en.wikipedia.org/wiki/JQuery>. Luettu 29.2.2016

7 Ajax getting started. Verkkodokumentti. The Mozilla Developer Network. < https://developer.mozilla.org/en-US/docs/AJAX/Getting_Started>. Luettu 29.2.2016

8 Rochkind, Marc J. 2013. Expert PHP and MySQL. E-kirja. New York: Springer.

9 TRANSACTIONAL EMAIL FOR MAILCHIMP. Verkkodokumentti. Mandrill. <https://www.mandrill.com/>. Luettu 15.3.2016

10 Overview. Verkkodokumentti. Zend Technologies Ltd. <http://framework.zend.com/manual/1.8/en/introduction.html>. Luettu 15.3.2016

11 Zend_Application Quick Start. Verkkodokumentti. Zend Technologies Ltd. <http://framework.zend.com/manual/1.8/en/zend.application.quickstart.html#zend.application.quick-start.zend-tool>. Luettu 17.3.2016

12 Lyman, Forrest. 2009. Pro Zend Framework Techniques. E-kirja. New York: Apress.

13 Padilla, Armando. 2009. Beginning Zend Framework. E-kirja. New York: Apress.

14 Zend_Controller Quick Start. Verkkodokumentti. Zend Technologies Ltd. <http://framework.zend.com/manual/1.8/en/zend.controller.html>. Luettu 21.3.2016

15 Action Helpers. Verkkodokumentti. Zend Technologies Ltd. <http://framework.zend.com/manual/1.8/en/zend.controller.actionhelpers.html#zend.controller.actionhelpers.viewrenderer>. Luettu 21.3.2016

16 Zend_Db_Adapter. Verkkodokumentti. Zend Technologies Ltd. <http://framework.zend.com/manual/1.8/en/zend.db.html>. Luettu 27.3.2016

17 The Standard Router. Verkkodokumentti. Zend Technologies Ltd. <http://framework.zend.com/manual/1.8/en/zend.controller.router.html>. Luettu 30.3.2016

18 Using the Registry. Verkkodokumentti. Zend Technologies Ltd. <http://framework.zend.com/manual/1.8/en/zend.registry.html>. Luettu 30.3.2016

Tilannekuvaukset

Taulukko 1. Sähköpostiviestin lukeminen

Nimi	Lukeminen
Suorittajat	Asiakas
Esiehdot	Sähköpostilaatikko ei ole tyhjä.
Jälkiehdot	Viestiketju on avautunut näkymään
Kuvaus	1. Asiakas avaa sähköpostilaatikon. 2. Järjestelmä näyttää avatun laatikon perusteella vastaanotetut tai lähetyt sähköpostiviestit. 3. Asiakas painaa hiirellä viestiä, jonka haluaa avata luettavaksi. 4. Järjestelmä avaa viestiketjun lukunäkymään.
Poikkeukset	Sähköpostilaatikossa ei ole viestejä: Järjestelmä ilmoittaa, ettei laatikossa ole viestejä.

Taulukko 2. Sähköpostiviestin lähettäminen

Nimi	Lähettäminen
Suorittajat	Asiakas
Esiehdot	Sähköpostiviesti on kirjoitettu, vastaanottajat lisätty ja lähettäjän sähköpostiosoite on oikeassa muodossa.
Jälkiehdot	Viesti lähetetään ja se näkyy lähetetyt-kansiossa.
Kuvaus	1. Asiakas painaa "new"-painiketta aloittaakseen uuden sähköpostiviestin kirjoittamisen. 2. Asiakas kirjoittaa viestin, lähettäjän sähköpostiosoitteen ja lisää vastaanottajat. 3. Asiakas painaa "send"-painiketta. 4. Viesti lähetetään vastaanottajille ja näytetään lähetetyt-kansiossa.
Poikkeukset	Lähettäjän osoite puuttuu: Virheilmoitus, sähköpostia ei lähetetä. Vastaanottajia ei ole lisätty: Virheilmoitus, sähköpostia ei lähetetä.

Taulukko 3. Sähköpostiviestin tallentaminen luonnoksena

Nimi	Tallentaminen luonnoksena
Suorittajat	Asiakas
Esiehdot	Asiakas on avannut viestin muokkausnäkymässä.
Jälkiehdot	Asiakkaalle esitetään ilmoitus tallennuksen onnistumisesta ja viesti näkyy luonnokset-kansiossa.
Kuvaus	1. Asiakas aloittaa uuden viestin kirjoittamisen "new"-painiketta painamalla, tai avaa vanhan viestin muokkaamistilassa. 2. Asiakas painaa "save"-painiketta 3. Viesti tallennuksen onnistumisesta näkyy asiakkaalle. 4. Viesti näkyy luonnokset-kansiossa.
Poikkeukset	Kaikki kentät tyhjiä: Virheilmoitus, sähköpostiviestiä ei tallenneta.

Taulukko 4. Sähköpostiviestin vastaanottaminen

Nimi	Vastaanottaminen
Suorittajat	Asiakas
Esiehdot	Vastaanotettavan viestin lähettäjä on asiakkaan kontaktilistalla. Kontakti on lisätty kampanjaan, johon viesti on saapumassa.
Jälkiehdot	Viesti lisätään saapuneet-kansioon ja merkitään lukemattomaksi.
Kuvaus	1. Kontakti lähettää sähköpostiviestin asiakkaalle. 2. Järjestelmä vastaanottaa viestin ja lisää sen oikealle kampanjalle. 3. Viesti näkyy saapuneet-kansiossa lukemattomana.
Poikkeukset	Vastaanottaja-kentästä puuttuu tietoja: Viesti tallennetaan, mutta sitä ei liitetä mihinkään kampanjaan.

Taulukko 5. Sähköpostiviestin lähettäminen vastauksena

Nimi	Vastauksen lähettäminen
Suorittajat	Asiakas
Esiehdot	Lähetettävä sähköpostiviesti on vastaus johonkin toiseen viestiin. Sähköpostiviesti on laadittu, vastaanottajat lisätty ja lähettäjän sähköpostiosoite on oikeassa muodossa.
Jälkiehdot	Sähköpostiviesti näkyy lähetetyt-kansiossa.
Kuvaus	1. Asiakas avaa vastaanotetun tai lähetetyn viestin muokkausnäkymässä. 2. Asiakas kirjoittaa viestin ja lisää otsakkeet. 3. Asiakas painaa "send"-painiketta. 4. Viesti lähetetään vastaanottajille ja näytetään lähetetyt-kansiossa.
Poikkeukset	Lähettäjän osoite puuttuu: Virheilmoitus, sähköpostia ei lähetetä. Vastaanottajia ei ole lisätty: Virheilmoitus, sähköpostia ei lähetetä