LAHDEN AMMATTIKORKEAKOULU
Lahti University of Applied Sciences

SÄHKÖISTEN MARKKINOINTIVIESTIEN LÄHETYSPALVELU

LAHDEN
AMMATTIKORKEAKOULU
Tekniikan ala
Tietotekniikka
Ohjelmistotekniikka
Opinnäytetyö
Kevät 2014
Niko Salonen

Lahden ammattikorkeakoulu
Tietotekniikan koulutusohjelma

SALONEN, NIKO:

Sähköisten markkinointiviestin lähetyspalvelu

Ohjelmistotekniikann opinnäytetyö, 47 sivua

Kevät 2014

TIIVISTELMÄ

Tässä työssä käsitellään sähköpostin automatisoitua lähetystä. Työssä tutkittiin CodeIgniter-nimistä PHP-pohjaista kehysohjelmistoa ja sen soveltuvuutta tehtävään sekä tutustuttiin myös MVC-mallin toimintaan esimerkkien avulla.

Vaikka sähköpostin kuvittelisi olevan yksinkertainen toteuttaa, on todellisuus kaukana siitä. Kun tarkoituksena on luoda visuaalisesti näyttävä viesti, joka näkyy oikein suosituimmilla sähköpostiohjelmistoilla, ongelmia alkaa esiintyä, sillä jokaisen ohjelmiston tuki HTML- ja CSS-tageille eroaa toisistaan, ja kehittäjän tulee olla tietoinen tästä.

Sähköpostiviestin luomiseen käytetään siis paljon aikaa. On tärkeää pystyä seurata, kuinka moni viestin vastaanottajista viestin on avannut, sillä muuten työpanos on hukkaan heitettyä aikaa ja rahaa. Viestien avausta voi seurata, mutta menetelmät eivät ole täydellisiä. On vastaanottajasta itsestään kiinni, näkyykö hän viestin avanneiden listalla.

Vastaanottajan sähköpostipalvelimen asetuksista riippuen viestit voidaan merkata roskapostiksi hyvinkin herkästi. Siksi on tärkeää, että viestit rakennetaan niin, etteivät vastaanottavat palvelimet pidä niitä roskapostina tai muuten hylkää viestiä.

Asiasanat: PHP, CodeIgniter, Sparks, daemon, web-sovellus, ion_auth, Swift mailer, MVC

Lahti University of Applied Sciences
Degree Programme in Information Technology

SALONEN, NIKO:

Electronic marketing messages as a service

Bachelor's Thesis in software engineering, 47 pages

Spring 2014

ABSTRACT

This Bachelor's Thesis deals with building an automated sender of marketing email as a service. A PHP-based web framework called CodeIgniter and its suitability to the given tasks was studied while also getting to know the MVC-model.

Creating visually ambitious email messages that show properly on all the most popular email clients is challenging. This is because there are no two email clients that support exactly the same HTML and CSS tags and it is very important for the developer to be aware of these differences.

Creating an email might take quite an amount of time. It is important that you can track how many recipients actually open your message so you know if all the effort you have put into the email is for nothing. Luckily you can track the message openings but the methods are not perfect. The recipient can easily opt out from the tracking if he or she wants to.

Email servers can be strict and block messages if the server thinks it might be spam. Therefore it is extremely important to build the messages in a way that pervents this.

Key words: PHP, CodeIgniter, Sparks, daemon, web-application, ion_auth, Swift mailer, MVC

SISÄLLYS

1 JOHDANTO

Sähköposti on yksi yleisimmistä markkinointikanavista internetissä. Ei ole toista yhtä luotettavaa menetelmää, jolla käyttäjä voi tilata omaan postilaatikkoonsa tarjouksia ja uutisia juuri niistä kohteista ja aiheista, jotka häntä itseään kiinnostavat. Systeemin yksinkertaisuudessa on myös varjopuolensa. Ei-toivotut sähköpostit häiritsevät ja täyttävät postilaatikot, eikä niistä edes pääse aina eroon. Pahimmassa tapauksessa minkä tahansa linkin klikkaus tällaisesta viestistä voi johtaa jopa roskapostin määrän lisääntymiseen.

Vaikka sähköpostilla voidaan tavoittaa suuria määriä käyttäjiä, ei teknologian käyttämineen markkinointiin ole niin helppoa. Tekstillä välitetään paljon dataa, mutta aivan kuten ulkomainonnassakin, kuvat ja värit auttavat myymään tuotteita herättämällä lukijan huomion.

Devnet Oy on vuonna 2005 perustettu, monipuoliseen IT-tarjontaan keskittynyt lahtelainen monitoimitalo. Toiminnan pääpainona ovat palvelin-, kotisivu- ja ohjelmistoratkaisut yrityksille. Toimiala kattaa koko IT-alan. Yrityksen tuotteisiin ja palveluihin kuuluvat ohjelmistokehitys, järjestelmäratkaisut, kotisivupalvelut sekä ulkoistuspalvelut. DevNet Oy on yksityishenkilöiden omistama, ja työntekijöitä on tällä hetkellä 31. Konsernin liikevaihto 05/2013 oli 1,25 miljoonaa euroa. (DevNet 2013.)

Tässä työssä toteutetaan DevNet Oy:lle sähköisten markkinointiviestien postitusjärjestelmä, jota yritys tulee tarjoamaan kuukausimaksullisena tuotteena asiakkailleen. Vastaavia palveluita on jo markkinoilla, mutta DevNet haluaa luoda omansa, jota siten voi tarjota osana omaa palvelukokonaisuuttaan.

Opinnäytetyö alkoi keväällä 2011. Työn tarkoituksena oli selvittää, millaisia vaatimuksia ja rajoituksia HTML-muotoiluja sisältävät sähköpostiviestit asettavat. Toinen tutkimuksen kohde oli selvittää, voidaanko järjestelmä toteuttaa käyttäen PHP-pohjaista kehysohjelmistoa ja millaisia hyötyjä tai haittoja valinnasta seuraa. DevNet Oy on käyttänyt useissa aiemmissa projekteissaan EllisLabin ylläpitämää CodeIgniter-kehysohjelmistoa. Tästä syystä kyseinen framework valittiin myös tähän työhön tutkimuksen aiheeksi. Työn toteutus tulee pohjautumaan tutkimuksessa tehtyihin havainnointeihin.

Toisessa luvussa käsitellään järjestelmän asettamia vaatimuksia ja rajoituksia, sekä pohditaan, kuinka varautua ongelmiin. Kolmannessa luvussa tutustaan CodeIgniteriin. Neljännessä luvussa esitellään itse toteutukseen liittyvät suunnitelmat. Viidennessä luvussa käydään läpi toteutusta. Kuudennessa luvussa esitellään kuinka viestin lähetys toimii valmiilla ohjelmistolla. Seitsemäs luku sisältää yhteenvedon, tulevaisuuden näkymät ja johtopäätökset.

2 SÄHKÖPOSTI

2.1 Sähköpostin lyhyehkö historia

Sähköposti on käytännössä ollut olemassa niin pitkään, kuin tietokoneet ovat olleet verkossa. Ennen Internetiä viestit kulkivat ARPANETissä.

Sähköpostit alkoivat yleistyä 1980-luvulla yrityksissä, joissa oli iso keskustietokone. Sähköpostijärjestelmä tarjosi nykyisten postiohjelmien perustoiminnot, mutta käyttöliittymät olivat kankeita. (Järvinen, 2000.)

Vasta Internetin yleistyminen nosti sähköpostin suosion. Toisin kuin aikaisemmin, Internet teki sähköpostittelusta nopeaa ja edullista. Kuka tahansa saattoi hankkia Internetyhteyden kotiinsa ja alkaa lähettää sähköpostia. Etenkin yritykset korvasivat vanhat, kalliit järjestelmänsä hyvin nopeasti Internet-postilla. (Järvinen 2000.)

2.2 Sähköpostin hyödyt

Sähköposti tarjoaa monia etuja käyttäjälle. Sähköposti ei sido käyttäjäänsä aikaan eikä paikkaan. Etenkin nykypäivän mobiililaitteet mahdollistavat viestien lähetyksen ja vastaanottamisen missä tahansa.

Sähköposteja on helpompi hallita kuin paperiviestejä. Viestien säilöminen ei tarvitse fyysisiä mappeja, ja viestien organisointi on vaivatonta. Myös tietokoneella tehtävät haut nopeuttavat halutun viestin löytämistä erilaisiin paperipohjaisiin arkistoihin verrattuna.

Sähköposti kulkee valonnopeudella. Suuria määriä dataa saadaan siirettyä valokaapeleita pitkin paikasta toiseen kirjaimellisesti sekunneissa. Sähköposti on ilmaista. Internet-yhteydellä on tietysti kustannuksensa, mutta viestin lähetys ei erikseen maksa.

2.3 Sähköpostin haitat

Sähköpostin suurin haitta on roskaposti. Vuonna 2011 varovaiset arviot asettivat roskapostin määrän kaikesta sähköpostiliikenteestä 88 - 90 prosenttiin. (MAAWG 2011.)

Joissakin maissa roskapostin lähetys on laitonta, mutta koska se ei ole kaikissa maissa laitonta, löytyy aina joku, joka viestejä lähettää. Lisäksi nykyään suuren osan postituksesta tekevät viirusten saastuttamat zombie-verkot, joissa roskapostittaja säätelee kokonaisen tietokonearmeijan toimia. Tällaisen viruksen voi saada tietokoneelle vaikkapa sähköpostin liitetiedostosta.

2.4 Sähköposti markkinointikanavana

Sähköposti tarjoaa lähes yhtä henkilökohtaisen markkinointikokemuksen kuin tavallinenkin kirje. Sähköpostin helppous on myönteinen asia niin lähettäjälle kuin myös vastaanottajallekin. Vastanotettua viestiä ei tarvitse edes avata, jos lähettäjä tai viestin otsikko ei herätä kiinnostusta. Näin ollen vastaanottaja voi suorittaa viestien suodatusta välittömästi, jo ennen kuin viestejä on avattu. Myös postituslistoilta poistuminen on ominaisuus, jota kirjeposti ei tarjoa niin helposti.

Markkinointipalveluryritys Acxiomim vuonna 2011 julkaisemasta riippumattomasta tutkimuksesta selviää, että jopa 77 prosenttia yrityksen nykyisistä kuluttaja-asiakkaista ottavat mielellään vastaan sähköpostiviestejä. Osuus on suurempi kuin kirjepostin. Kirjepostia ottaa vastaan mielellään 71 prosenttia asiakkaista. (UTalkMarketing 2011.)

Kuluttajat, joilla ei ollut vielä olemassa olevaa asiakas-suhdetta yritykseen, valitsivat kirjepostin ennen sähköpostia markkinointikanavana. Tässä tilanteessa kirjepostin osuus oli 57 prosenttia ja sähköpostin 52 prosenttia. (UTalkMarketing 2011.)

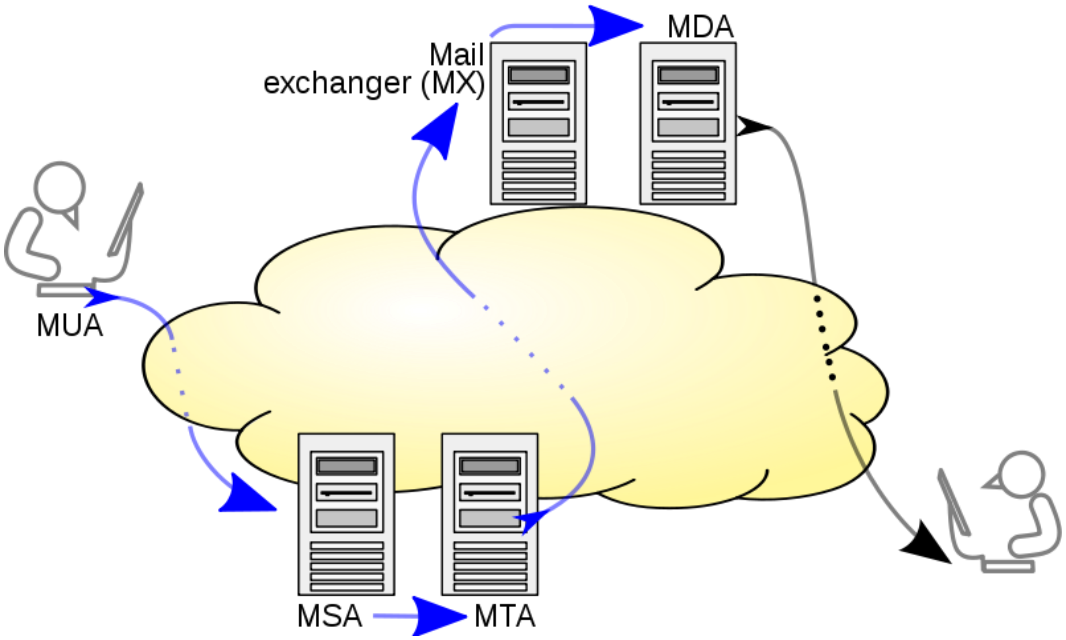
2.5 Sähköpostin lähetys

Sähköposteja lähetyksestä vastaa SMTP-protokolla. SMTP (Simple Mail Transfer Protocol) on TCP-pohjainen protokolla, ja sen toimille on varattu portti 25.

SMTP-protokolla määriteltiin ensimmäisen kerran vuonna 1982. (Wikipedia 2013.)

Nykyaikaiset palvelimet käyttävät laajennettua ESMTP-protokollaa. Kyseinen protokolla määriteltiin ensi kerran vuonna 1995, ja viimeisin määritys on vuodelta 2001. (Wikipedia 2013.)

Erityyppisiä SMTP-palvelimia on markkinoilla useita, ja tunnetuimmat niistä ovat Postfix, Sendmail sekä Microsoftin Exchange. Kaksi ensimmäistä ovat ilmaisia avoimen lähdekoodin projekteja, joskin Postfixistä maksullinen versiokin on saatavilla. Microsoftin Exchange-järjestelmä puolestaan on varsin hintava.



KUVIO 1. Sähköpostiviestin prosessointikuvio. (Wikipedia, 2013)

Kuviosta 1 nähdään sähköpostiviestin prosessointikuvio, eli kuinka lähetetty viesti saavuttaa vastaanottajan SMTP-protokollaa käyttäen. Kun kirjoitettu viesti on valmis lähetettäväksi, kuvan MUA (mail user agent, esimerkiksi Outlook) lähettää viestin MSA:lle (mail submission agent) käyttäen SMTP-protokollaa TCP-verkon yli. Viestin saatuaan MSA välittää sen edelleen MTA:lle (mail transfer agent, esimerkiksi Postfix ja Exim). MTA ja MSA toimivat usein kuviosta poiketen

samalla palvelimella. Nämä kaksi agenttia ovat itse asiassa yhden ohjelmiston kaksi rinnakkaista prosessia, jotka toimivat eriävin parametrein. MTA:n tehtävä on paikantaa viestin vastaanottajan verkkotunnus (kaikki mitä sähköpostiosoitteesta löytyy @-merkin oikealta puolelta). Tämä tapahtuu niin, että MTA tekee DNS-kyselyn Mail exchanger (MX) -palvelimelle. MX palauttaa MTA:lle vastaanottajan tiedot, jos sellaisia löytyy. Kun MTA on vastaanottanut tiedot MX:ltä, ottaa se seuraavaksi varsinaisen yhteyden tähän välityspalvelimeen tarjoten viestiään eteenpäin kuljetettavaksi. Jos välityspalvein hyväksyy yhteyden, välittää se viestin MDA:lle (mail delivery agent) joka on vastuussa viestin paikallisesta jakelusta. Viestin välitys tässä tapauksessa tarkoittaa sitä, että viesti säilötään vastaanottajan sähköpostipalvelimelle odottamaan vastaanottajan sähköpostiohjelmaa sitä hakevaksi. MSA ja MTA tavoin myös MX ja MDA ovat usein yksi ja sama palvelin.

Tässä työssä sähköpostien lähetykseen käytetään DevNet Oy:n omia Postfix-pohjaisia SMTP-palvelimia. Koska palvelimet olivat jo olemassa, ei niiden konfigurointia otettu osaksi tätä työtä.

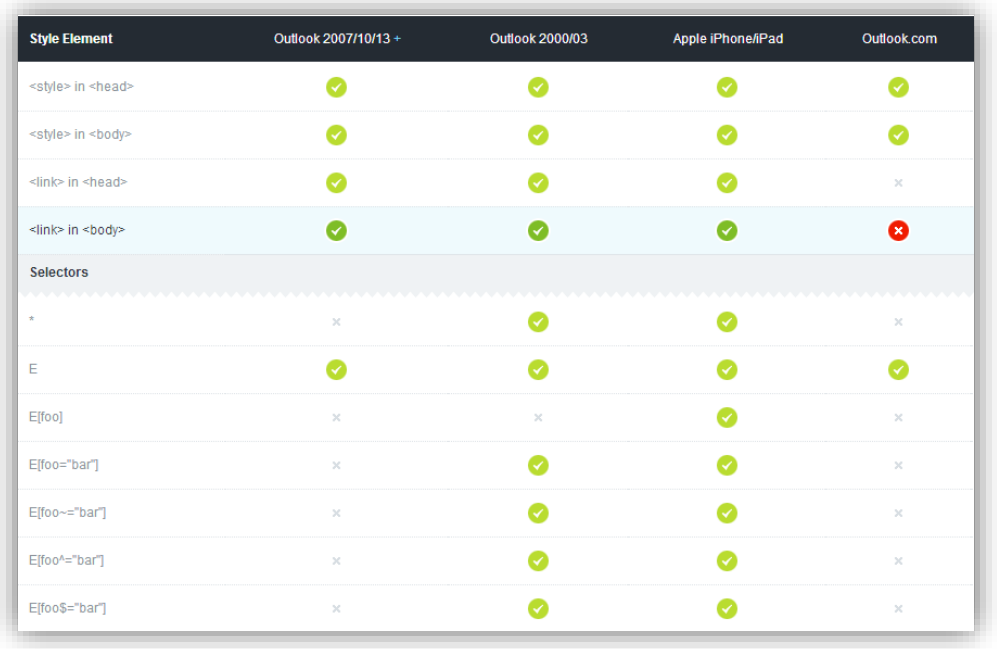
2.6 Sähköposti, HTML ja CSS

Graafisten sähköpostiviestien työstäminen kaikilla suosituimmilla sähköpostiohjelmistoilla samalta näyttäväksi voi olla todella aikaa vievää. Eri ohjelmistojen ja jopa saman ohjelmiston eri versioiden tuki erinäisille CSS-komennoille saattaa poiketa merkittävästi. (CampaignMonitor 2006.)

CampaignMonitor.com–verkkoportaali on koonnut kattavan listan suosituimpien sähköpostiohjelmien CSS-tuista. Kuviosta 2 voidaan esimerkiksi päätellä, että CSS-pohjainen elementin valinta HTML-tagin attribuutin perusteella on hyvin rajoitetusti tuettu Microsoftin Outlook-ohjelmistoilla.

Nykyään Internetsivustoja tehtäessä halutaan välttää sivustorungon rakennus-taulukoiden päälle. Tämä saa sivuston näyttämään siltä, että koodaaja on 10 vuotta aikaansa jäljessä. Sähköpostiviesteissä tämä halveksittu menetelmä puolestaan on ainut tapa rakentaa viestille graafinen ulkoasu, jolla on mahdollisuudet olla hajoamatta täysin, kun se avataan erilaisissa sähköpostiohjelmistoissa. kun taas

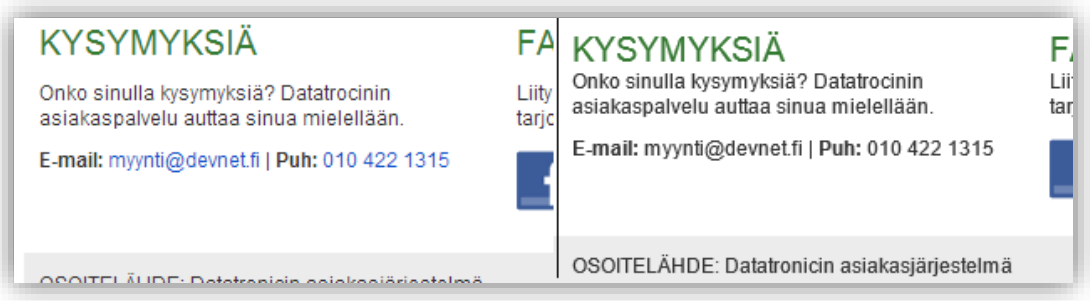
Applen iPhone ja iPad laitteiden Mail-app tukee kaikkia attribuuttivalintoja täysin. Toinen huomattava erikoisuus on, että Microsoftin selainpohjainen Outlook.com ei tue laisinkaan ulkopuolisia tyylitiedostoja, sillä kaikki <link> tagit niin <head> kuin myös <body> tagien sisältä jäävät huomiotta.



Style Element	Outlook 2007/10/13 +	Outlook 2000/03	Apple iPhone/iPad	Outlook.com
<style> in <head>	✓	✓	✓	✓
<style> in <body>	✓	✓	✓	✓
<link> in <head>	✓	✓	✓	×
<link> in <body>	✓	✓	✓	×
Selectors				
*	×	✓	✓	×
E	✓	✓	✓	✓
E[foo]	×	×	✓	×
E[foo="bar"]	×	✓	✓	×
E[foo~="bar"]	×	✓	✓	×
E[foo^="bar"]	×	✓	✓	×
E[foo$="bar"]	×	✓	✓	×

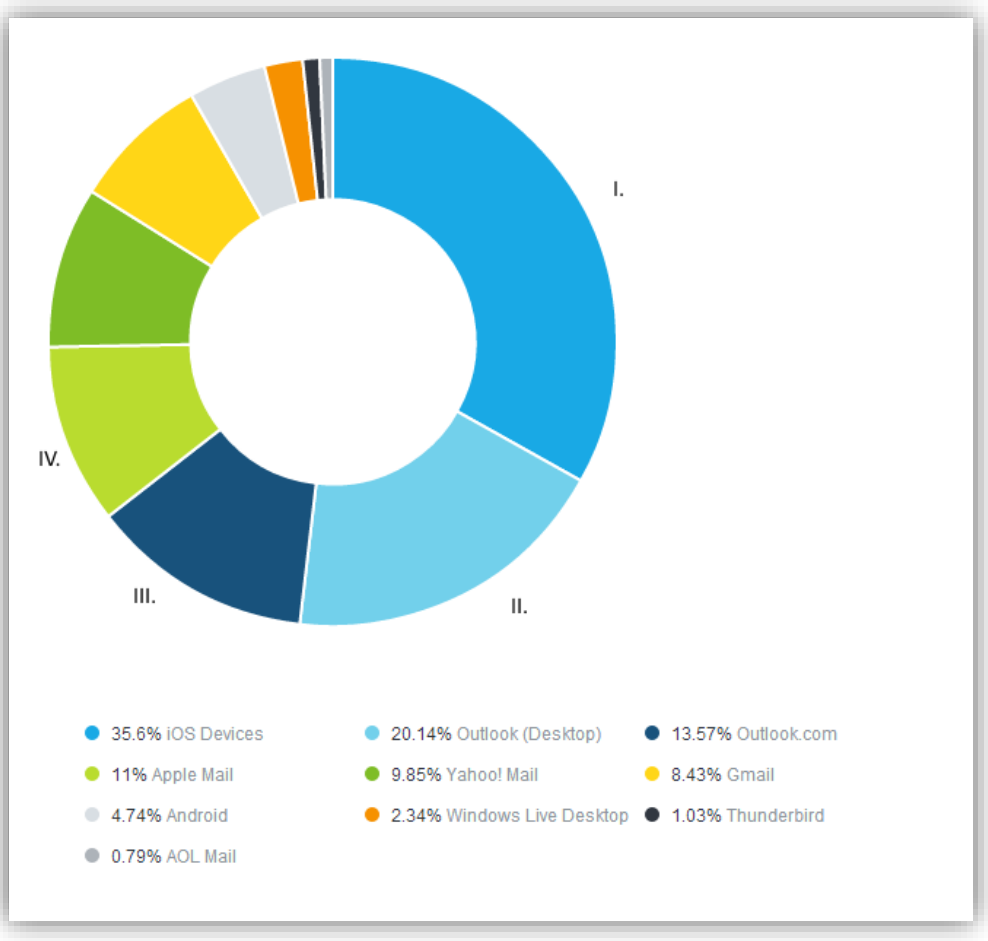
KUVIO 2. Ote eri sähköpostiohjelmistojen CSS muotoilujen tuesta.

Toinen internetsivujen hylätty, mutta sähköposteissa hyväksi havaittu käytäntö on lisätä CSS muotoilut HTML-tagien sisään. Tämä johtuu siitä, että esimerkiksi suosittu Googlen Gmail-palvelu poistaa kaikki muotoilut <head> tagien sisältä, eli juuri sieltä, minne muotoilut pitäisi laittaa nykypäivän standardien mukaan. Kuviossa 3 on sama sähköpostiviesti avattu Googlen Gmailissa (vasen) ja Microsoftin Outlook 2013 ohjelmistossa (oikea). Vaikka viestin lähdekoodi on molemmissa viesteissä sama, on eroavaisuuksia fontissa ja asettelussa huomattavissa helposti. Suurinpana erona voidaan huomata Gmailin tapa poistaa värikoodaukset linkeistä.



KUVIO 3. Sama viesti eri sähköpostiohjelmistoilla näytettynä.

On täysin käyttäjän oma päätös, kuinka paljon viestien ulkoasuun halutaan panostaa aikaa. Kuviosta 4 voi nähdä, että esimerkiksi Thunderbird-sähköpostiohjelman (väri: musta) markkinaosuus on vain yksi prosentti. Toisaalta taas Applen iOS (merkattu I) laitteet kattavat suurimmalla osuudella yli kolmanneksen. Kolme suurinta, iOS, Microsoftin Outlook ohjelma sekä Outlook.com verkkosivu (I, II ja III), ovat yli puolet kaikista. Jos vielä neljänneksi suurin, eli Applen työpöytäsähköpostiohjelmisto Mail (IV.) lisätään, katetaan jo kolme neljännestä kaikista sähköpostin avaajista. Tässä tulee miettiä, onko välttämättä vaivan arvoista korjata pieniä kosmeettisia eroavaisuuksia, jos viestit näkyvät virheettä käytetyimmissä ohjelmistoissa. Kuvion taulukko on vuoden 2012 syyskuulta, ja ainoa eroavaisuus nykypäivän tilanteeseen on, että suuret osuudet ovat kasvaneet entisestään ja pienet kuihtuneet pienmmiksi.

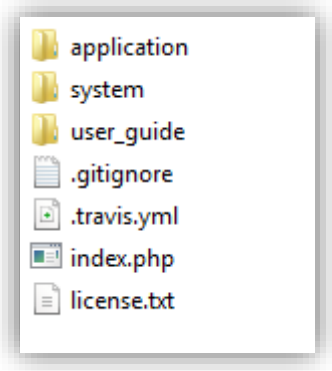


KUVIO 4. Sähköpostiohjelmistojen jakauma syyskuussa 2012

3 CODEIGNITER

CodeIgniter on EllisLab Inc. kehittämä avoimeen lähdekoodiin perustuva PHP-pohjainen web framework. CodeIgniterin tavoitteena on tehdä verkkopohjaisten, dynaamisten sovellusten kehitys helpoksi ja nopeaksi. CodeIgniter tarjoaa lukuisia kirjastoja, jotka palvelevat kehittäjää tehden työläitä, mutta tarpeellisia tehtäviä lähdekoodissa. Näin kirjoitetun lähdekoodin määrä vähenee ja kehitys nopeutuu.

CodeIgniter mainostaa itseään olemalla kevyt, helppo pystyttää ja ylläpitää. Vahvuuksiin kuuluvat myös hyvä suorituskyky sekä soveltuvuus useampaan PHP-versioon. (EllisLab 2013.)



KUVIO 5. CodeIgniter kansionäkymä

Kuviosta 5 nähdään CodeIgniterin kansionäkymä. Näistä tärkeimmät ovat application- ja system-kansio. System-kansio sisältää kaikki järjestelmätiedostot. Tavallisesti kehittäjällä ei ole tarvetta edes koskea tähän kansioon. Varsinainen kehitys tapahtuu application-kansiossa, joka sisältää näkymät, kontrollerit, mallit ja muut rakennuspalikat. Näiden kahden kansion ei tarvitse edes sijaita samassa paikassa. Itse asiassa jotkin jopa suosittelevat erottamista toisistaan ja uudelleen-nimeämistä tietoturvasyistä. Kansioiden nimellä ja lokaatiolla ei ole käytön kannalta väliä. Täytyy vain muistaa laittaa oikea polku kumpaankin kansioon CodeIgniterin asetuksista kuvion 6 tavoin.

```
$system_path = '../system';

$application_folder = '../application';
```

KUVIO 6. System- ja Application-kansioiden polut asetuksissa.

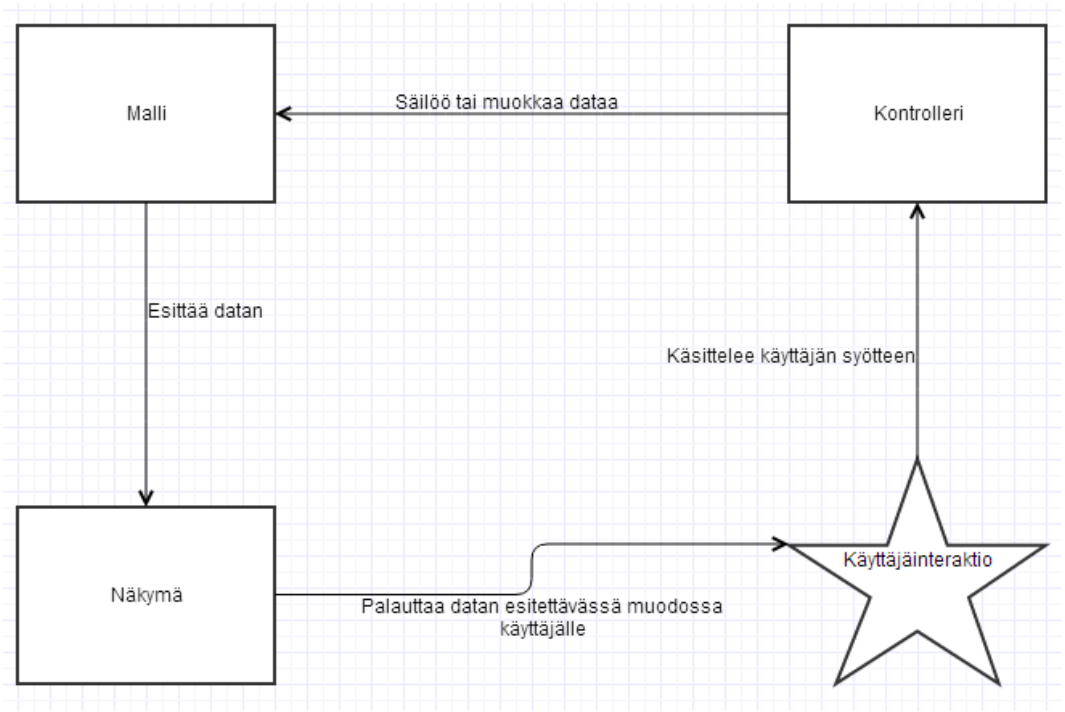
Helppous ei ole liioiteltua, sillä CodeIgniterin saa asennettua yksinkertaisimmillaan vain purkamalla ladattu paketti HTTP-serverin juureen. Ainoat vaadittavat tiedot asennuksessa ovat purettujen kansioiden sijainnit, kirjautumistiedot tietokantaan ja että application-kansion sisältämä public-kansio on julkisesti saatavilla tai ainakin linkattu julkisesti saataville.

Ensimmäinen versio CodeIgniteristä julkaistiin jo vuonna 2006. Tätä opinnäytetyötä kirjoitettaessa viimeisin vakaa versio on 2.1.4. ja tämä versio on myöskin käytössä järjestelmän kehityksessä. (EllisLab 2013.)

3.1 MVC-arkkitehtuuri

MVC-arkkitehtuuri eli model-view-controller (malli-näkymä-kontrolleri)-arkkitehtuuri on ohjelmistoarkkitehtuuri. Tämän arkkitehtuurin pääperiaatteena on pitää käyttöliittymä ja sovellusaluetiedot erillään toisistaan. Kyseinen suunnittelumalli on tänä päivänä suosittu etenkin verkkopohjaisissa sovelluksissa.

Kuviossa 7 kuvataan, kuinka MVC-malli toimii. Käyttäjä on tekemisissä vain näkymän kanssa. Näkymän tehtävä on välittää data kontrollerille, joka manipuloi sitä mallia hyödyntäen. Lopuksi malli välittää muokatun datan takaisin näkymään, ja käyttäjä näkee tuloksen.



KUVIO 7. MVC-toimintamalli

CodeIgniter noudattaa MVC-mallia, vaikkakin aika löyhästi. Näkymä ja kontrolleri vaaditaan, mutta mallin käyttö on vapaaehtoista.

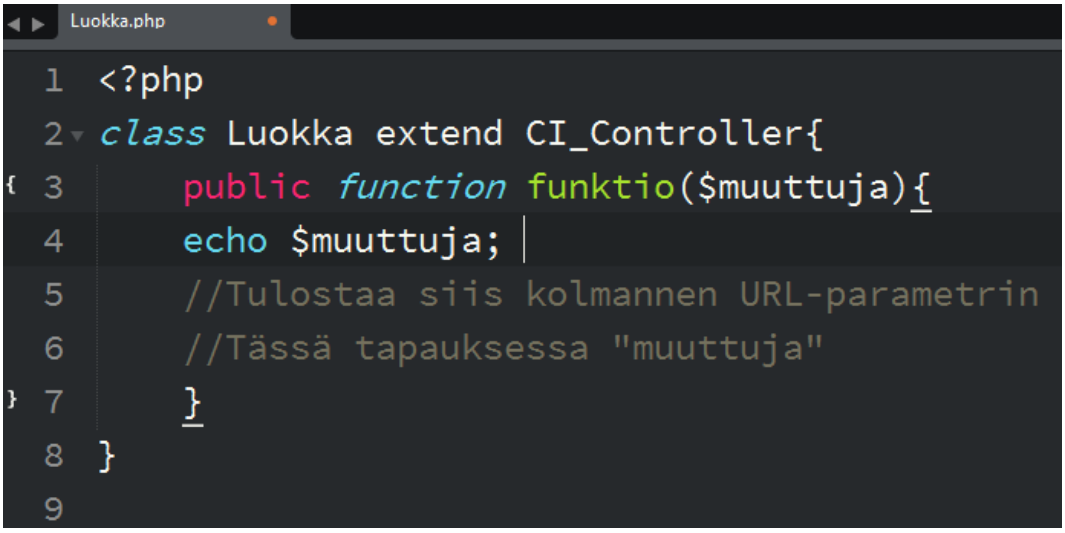
3.2 Kontrolleri

Kontrollerit ovat CodeIgniter-sovelluksen sydän. Kontrollerin päätehtävä on kontrolloida palvelimelle saapuvia HTTP-kyselyitä. Rakenteellisesti kontrolleri on vain luokka-tiedosto, joka on nimetty, niin kuin luokka halutaan esittää verkkosivuston osoiterivillä.

esimerkki.fi/luokka/funktio/muuttuja

KUVIO 8. Kontrollerin rakenne osoiterivillä.

Kuvion 8 kaltainen verkko-osoite on toteutettavissa kuvion 9 lähdekoodilla.



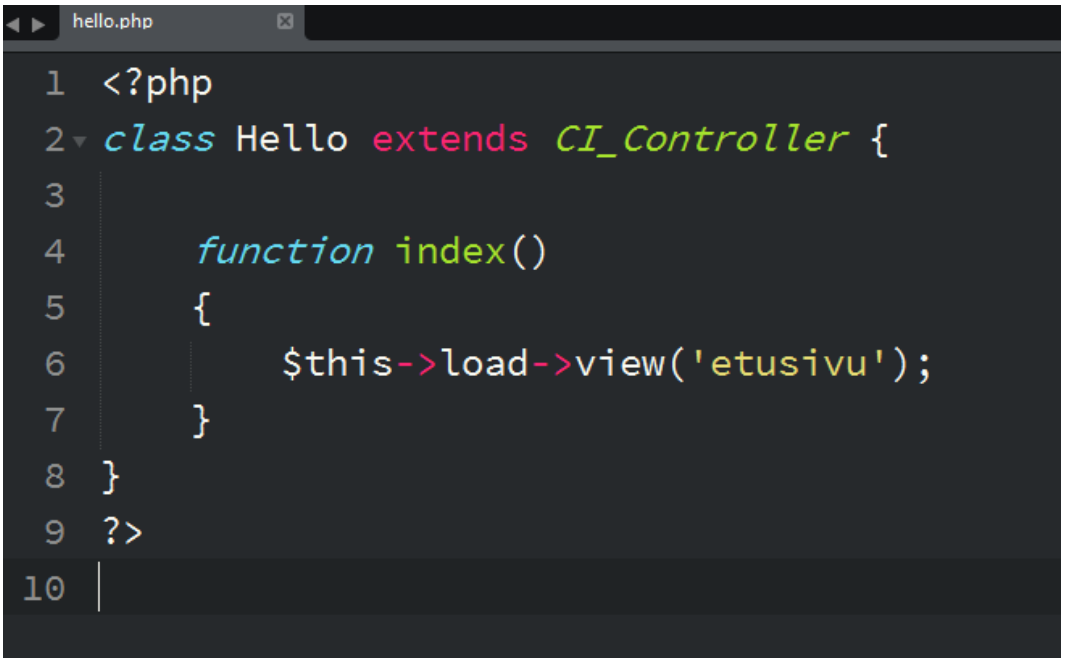
KUVIO 9. Kontrollerin rakenne lähdekoodissa

On tärkeää huomata, että tiedoston ja luokan nimien tulee täsmätä toisiinsa. Toinen huomionarvoinen asia on, että kolmas URI-osio eli ”muuttuja” on funktion parametri. Näitä parametreja voi olla useampia. Tällöin ne erotellaan /-viivalla toisistaan.

3.3 Näkymä

Näkymä on yksinkertaisesti vain www-sivu tai sen osa, kuten ylätunniste tai sivuvalikko. Näkymiä voi myös sijoittaa toistensa sisään, jos rakennettavan verkkosivuston rakenne sitä vaatii. (EllisLab 2013.)

MVC-mallista johtuen verkkoselain ei koskaan kutsu näkymää suoraan. Näkymiä kutsutaan vain ja ainoastaan kontrollereiden välityksellä. Näkymän kutsuminen tapahtuu kuvan 10 mukaisesti. Kuvasta nähdään, että kun selain saapuu esimerkkisivun /hello/-kontrollerin juureen, haetaan etusivu.php niminen näkymä näytettäväksi. Näkymien .php-päätettä ei ole tarpeen lisätä näkymän nimeen.



KUVIO 10. Näkymän lataus kontrollerissa.

MVC-malli mahdollistaa sen, että ennen kuin kontrolleri näyttää näkymän käyttäjälle, voi se sitä ennen suorittaa lukuisia tehtäviä. Yksi tällainen tehtävä on dynaamisen datan lisäys näkymään. Dynaaminen data käsitellään näytettävään muotoon, jo ennen kuin näkymä ladataan esimerkiksi jos näkymässä on kuvion 11 kaltainen lähdekoodi, jossa HTML-koodin seassa on PHP-muuttujia.

```
<body>
    <h1>Tervehdys <?=$nimi?>!</h1>
    <p>Tänään on <?=$date?></p>
```

KUVIO 11. PHP-muuttujia HTML-tagien sisällä.

Normaalisti nämä muuttujat aiheuttavat virhekoodin, ellei niille anneta arvoa. CodeIgniterin tapauksessa näille muuttujille voidaan asettaa arvot jo kontrollerissa. Tällöin näkymä ei puutu lainkaan tiedon prosessointiin, vaan kaikki tapahtuu kontrollerista käsin. Lisäksi näkymän lähdekoodi ei monimutkaistu.

```
1  <?php
2  class Hello extends CI_Controller {
3
4      function index()
5      {
6          $data = array(
7                          'nimi'=>'Matti',
8                          'date'=>date('d.m.Y',time())
9                          );
10         $this->load->view('etusivu',$data);
11     }
12 }
```

KUVIO 12. Tiedon välitys kontrollerista muuttujaan.

Kuviossa 12 on asetettu $data-muuttujaan taulukko, jossa ensimmäisen avaimen nimeksi asetettu nimi ja tämän arvoksi ”Matti”. Toisen avaimen nimi on date, ja arvoksi lasketaan juuri prosessointihetkellä oleva päivämäärä. Näkymää ladattaessa muuttuja lisätään näkymän nimen perään pilkulla ne erottaen. Näin

kaikki ne muuttujat näkymässä, joille löytyy vastaavan niminen avain-arvo $data-taulukosta saavat itselleen taulukon arvon.

3.4 Malli

Kuten jo aikaisemmin mainittiin, ovat mallit vapaaehtoisia CodeIgniterissä. Mallit ovat tarpeen vain, jos halutaan pitää MVC-mallin mukainen rakenne projektissa. Jos malleja ei haluta käytettävän, kaikki mallien toiminnallisuudet voidaan sijoittaa kontrolleriin.

Rakenteeltaan mallit ovat PHP-luokkia, joiden pääasiallinen tarkoitus on käsitellä dataa järjestelmän ja tietokannan välillä. Kuviosta 13 voi havainnoida mallin perusrakenteen. Esimerkkinä malliin on lisätty get_users()-funktio, joka CodeIgniterin tietokantaluokkaa käyttäen palauttaa users-tietokantataulun sisällön kutsujalle.

```
class Example_model extends CI_Model {

    function __construct()
    {
        parent::__construct();
    }

    function get_users()
    {
        $query = $this->db->get('users');
        return $query->result();
    }
}
```

KUVIO 13. Mallin perusrakenne ja esimerkkifunktio.

Mallin lataus kontrollerii tapahtuu kuvion 14 mukaisesti. Kun malli on ladattu, ovat sen kaikki funktiot heti käytettävissä.

```
$this->load->model('Example_model');

$users = $this->example_model->get_users();
```

KUVIO 14. Mallin lataus ja käyttö.

Tässä esimerkissä muuttujaan $users säilötään tietokantakyselyn tulos. Jos mallin toiminnallisuudet ovat käytössä useammassa kontrollerissa, voi mallin myös asettaa automaattisesti latautuvaksi CodeIgniterin asetuksista. Tällöin kuvion 14 ylempi rivi on turha, koska malli latautuu jo kontrolleria luotaessa.

3.5 Avustajaluokat

Avustajaluokat toimivat nimensä mukaan avustajina. Toisin kuin muut CodeIgniterin osat, avustajaluokat eivät noudata oliomallia vaan ovat yksinkertaisesti kokoelma oman ryhmänsä funktioita. CodeIgniter tarjoaa lukuisia avustajaluokkia valmiina kehittäjälle. Ainoa vaadittu toimenpide luokkien käyttöönottoon on kuvan 15 mukainen lataus, sillä oletuksena CodeIgniter ei lataa avustajaluokkia käyttöön. (EllisLab, 2013)

```
$this->load->helper('email');
```

KUVIO 15. Avustajaluokan lataus kontrollerissa.

Tässä esimerkissä ladataan sähköpostiavustajaluokka. Sähköpostiosoitteen varmistaminen standardien mukaiseksi on hyvin työläs tehdä itse, mutta sähköpostiavustajaluokan valid_email()-funktio tarjoaa valmiin ratkaisun ongelmaan. Funktiolle syötetään testattava sähköpostiosoite, ja se palauttaa boolean-arvon validioinnin tuloksena.

Jos laajasta avustajaluokkien kirjosta ei löydä hakemaansa, voi käyttäjä laajentaa jo olemassa olevia luokkia tai luoda kokonaan omia. Luokkaa laajennettaessa luodaan uusi tiedosto, jolle annetaan täysin sama nimi muuten, mutta sen eteen lisätään MY_ -etuliite. Näin CodeIgniter tietää, mitä avustajaluokkaa laajennetaan. Jos laajennustiedostossa käytetään funktion nimissä samoja nimiä kuin alkuperäisessä avustajaluokassa, CodeIgniter ylikirjoittaa alkuperäisen toiminnallisuuden korvaten sen laajennustiedoston versiolla.

3.6 Sparks

Vaikka CodeIgniter tarjoaa lukuisia kirjastoja verkkosovellusten kehitystä helpottamaan, on tarjonta aika geneeristä. Erikoisia ja pienemmälle käytölle jääviä kirjastoja ei kannata pakettiin lisätä, jos framework halutaan pitää kevyenä ja nopeana. Näitä puuttuvia luokkia varten on luotu Sparks-palvelu.

Sparks on käyttäjien luoma ja ylläpitämä palvelu, joka keskittyy tarjoamaan erilaisia kirjastoja CodeIgniter-ympäristöön. Palvelu tekee kehittäjille mahdolliseksi jakaa omia kirjastoja muille ja saada palautetta työstään. Keskitetty rakenne mahdollistaa pakettihallintaluokan asennuksen omaan CodeIgniter-projektiin. Pakettihallintaa hyödyntäen kehittäjä voi vaivattomasti pitää omaan projektiin asentamansa kirjastot ajan tasalla.

Valitettavasti nykyään CodeIgniterin suosion hiipuessa myös Sparks-palvelun kirjastot päivittyvät yhä harvemmin ja harvemmin.

4 JÄRJESTELMÄN KUVAUS

Tässä kappaleessa käydään läpi järjestelmälle asetettuja teknisiä vaatimuksia. Järjestelmäkuvaukset on tarkoitettu suuntaa-antaviksi, eli tilaa muutoksille on myös annettu.

Järjestelmän käyttöliittymän tulee olla selainpohjainen. Jo heti alkuun päätettiin, ettei järjestelmän tarvitse tukea vanhempia versiota Microsoft Internet Explorer -Internetselaimesta. Näin ollen vanhin tuettu selain on Internet Explorer 9.

Järjestelmään kirjauduttuaan käyttäjä tulee voida luoda uusi postitus, muokata ja luoda osoitelistoja viestin vastaanottajista, hallinnoida tiedostoja tiedostohallinnassa ja tarkastella edellisten lähetysten raportteja. Sivustolta tulee löytyä myös palautelomake, jolla käyttäjä voi pyytää teknistä tukea ja antaa palautetta.

4.1 Uuden postituksen luominen

Uusi postitus tulee voida luoda joko tallennetun viestipohjan pohjalta, tallennetun ja muokatun viestin pohjalta tai kokonaan puhtaalta pöydältä. Viestin vastaanottajat valitaan tallennetuista osoitelistoista tai ne syötetään käsin. Myös molemmat vaihtoehdot on voitava valita samaan aikaan. Lisäksi käsinsyötetyn listan on pystyttävä tallentamaan nimellä tulevia lähetyksiä varten, jos käyttäjä niin haluaa. Kun vastaanottajat on valittu, on lähettäjän määriteltävä lähetyksen ajankohta. Ajankohta voi olla heti tai valittuna päivänä tulevaisuudessa minuutin tarkkuudella. Lähetysajan jälkeen käyttäjälle avautuu sivu, josta hän voi nähdä yhteenvedon viestin lähetykseen liittyvistä tiedoista ja esikatsella viestiä vielä selaimessa. Jotta viestin lähetys voi alkaa, on ruksittava ruutu, jolla käyttäjä vakuuttaa viestin sisällön olevan lainsäädännön mukaista. Käyttäjällä on myös oltava viestikapasiteettia lähettää viestit kaikille osoitelistan osotteille. Jos kapasiteetti ei riitä, järjestelmä ei anna aloittaa postitusta. Samalla järjestelmän tulee lähettää tästä tiedon DevNet Oy:n myyntihenkilöille, jotka voivat tarjota suurempaa kapasiteettia käyttäjälle. Viestin lähetystä tulee seurata edistymispalkilla voida lähetyksen ajan. Jos toisella käyttäjällä on lähetys kesken, tulee odottavalle käyttäjälle näkyä aika-arvio lähetyksen ajankohdasta.

4.2 Osoitelistat

Osoitelistan muokkaus -sivulla käyttäjän tulee voida luoda itselleen uusi osoitelista kolmella eri tapaa. Ensimmäinen vaihtoehto on luoda tyhjä nimetty lista, johon käyttäjän tulee voida lisätä sähköpostiosotteita koska tahansa lisää. Toinen vaihtoehto on käsin syötetyt osoitteet. Toiminnallisuuden tulee olla samanlainen kuin uutiskirjeen lähetyksen yhteydessä tallennettavassa käsin syötetyssä listassa. Osoitteet kirjoitetaan tekstikenttään pilkulla erottaen. Koska yleensä tämänlaiset osoitelistat ovat lyhyitä, ei niille tulla antamaan mahdollisuutta lisätä sähköpostiosoitteeseen sidottua mahdollista nimeä luonnin yhteydessä. Jos käyttäjä haluaa lisätä nimet osotteille, tulee ne lisätä osoitelistojen muokkaustyökalulla. Kolmas tapa luoda uusi osoitelista on lähettää palvelimelle CSV-tiedosto.

CSV (lyhenne sanoista comma-separated values) on tiedostomuoto, jolla tallennetaan yksinkertaista taulukkomuotoista tietoa tekstitiedostoon. CSV on toteutukseltaan tekstitiedosto, jonka taulukkorakenteen eri kentät on eroteltu toisistaan pilkuilla ja tietueet rivinvaihdoilla. (RFC-4180 2005.)

CSV-tiedoston lähetyksen yhteydessä käyttäjältä kysytään nimi-sarakkeen järjestysnumero, jos nimi-sarake on olemassa, sähköpostisarakkeen järjestysnumero sekä CSV-tiedoston erotinmerkki.

Erotinmerkki on tarpeellista kysyä, koska esimerkiksi Microsoft Excel hyväksyy myös muita merkkejä pilkun lisäksi kenttien erottimeksi. Suosituin epästandardi merkki on puolipiste. Tämä johtuu siitä, että pilkkua käytetään etenkin Pohjois-Amerikan ulkopuolella myös desimaalieroittimena luvuissa. (Wikipedia 2013.)

Järjestelmä ei tule hyväksymään virheellisiä sähköpostiosotteita. Ennen kuin sähköposti tullaan merkkaamaan epäkelvoksi, tarkastetaan ensin, onko siinä mahdollisesti käytetty aksentteja tai umlautteja ja onko osoitteen perään tullut sinne kuulumaton merkki kuten piste. Lisäksi järjestelmän tulee poistaa CSV-tiedostosta mahdolliset duplikaatit.

Osoitelistasivulla tulee myös olla edellä mainittu muokkaustyökalu, jolla käyttäjä voi listata osoitelistan sisällön ruudulle, ruksia poistettavat osoitteet, tehdä muutoksia nimiin ja sähköpostiosoitteisiin sekä lisätä uusia osoitteita listoihin.

Käyttäjän tulee voida myös halutessaan avata osoitelista rajatulla haulla. Tämän listan tulee myös näyttää osoitelistalta poistuneet käyttäjät ja mahdolliset toimimattomat sähköpostiosoitteet. Toimimattomilla sähköpostiosoitteilla tarkoitetaan tässä osoitteita, jotka ovat standardien mukaisia, mutta on esimerkiksi poistettu käytöstä.

4.3 Tiedostojen hallinta

Tiedostohallintasivulta käyttäjän tulee voida helposti hallinnoida mahdollisia liitetiedostoja sekä sähköpostien kuvatiedostoja jos haluaa niiden jakelun toimivan palvelun kautta. Yksittäisen liitetiedoston koko tullaan rajaamaan kahteen megatavuun vastaanottajan internetkaistan säästämiseksi.

4.4 Raportit

Raporttisivulta käyttäjän voi nähdä kaikkien aikaisempien uutiskirjeiden lähetystiedot, itse viestin, viestin avanneet käyttäjät, virheellisen palauteviestin sähköpostipalvelimelta palauttaneet sähköpostiosoitteet sekä sähköpostilistalta poistuneet sähköpostiosoitteet. Sähköpostin avanneiden listassa tulee olla kaupankäyntitarkoituksia varten toiminto, jolla esimerkiksi myyjä voi merkata ne sähköpostin avanneet, joihin hän on ollut yhteydessä. Tämä tulee palvelemaan tilanteissa, jossa myyjä myyjä soittaa viestin avanneille tarjoten lisäinfoa tuotteista, ja merkkauksen ansiosta myyjä voi pitää kirjaa soitoista.

4.5 Uutiskirjeiden lähetys

Uutiskirjeiden lähetykseen tulee luoda erillinen taustaohjelma, joka tasaisin väliajoin tarkastaa, onko tietokannassa uusia lähetettäviä postituksia. Jos lähetettävä työ löytyy, daemonin tulee luoda tietokantaan action-tietokantatauluun uusi rivin merkiksi siitä, että lähetys alkaa. Jokaisen lähetetyn sähköpostin jälkeen tietokantariviin tulee päivittyä lähetettyjen viestien määrä. Kun kaikki viestit on

lähetetty ja lähetettyjen viestien määrä vastaa osoitelistan osotteiden määrää, daemonin tulee asettaa lähetys päättyneeksi ja alkaa taas odotella uutta työtä.

Ohjelmasta pyritään tekemään vikasietoinen. Kesken jäänyt lähetys havaitaan siitä, että action-tietokantataulussa rivi ei ole asetettu ”Lähetys valmis”-tilaan. Kun kesken jäänyttä lähetystä aletaan käsitellä, voidaan helposti huomata lähetettyjen viestien määrästä, mikä sähköpostiosoite osoitelistalla on seuraavana lähetysvuorossa ja näin lähetys voi jatkua normaalisti.

Daemonin tulee käydä läpi myös paikallinen sähköpostilaatikko aika-ajoin virheviestien varalta. Mahdolliset virheviestit tulee kirjata tietokantaan virheellisinä, jottei näihin sähköpostiosoitteisiin enää lähetettäisi viestejä. Kun sähköposti on käsitelty, siirretään se arkistokansioon.

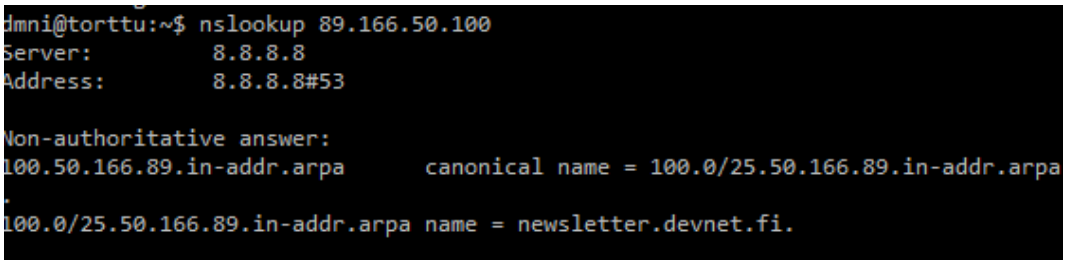
4.6 Odotettavissa olevat ongelmat

Kaikkia muuttujia ei millään voida ottaa kaiken kattavasti huomioon. Tässä kappaleessa listataan aiheita, joiden oletetaan aiheuttavan vastoinkäymisiä järjestelmän kehityksen aikana.

4.6.1 Roskaposti

Suurin haaste tulee olemaan uutiskirjeen saaminen sellaiseksi, etteivät sähköpostipalvelimet merkitse viestejä roskapostiksi. Kriteereitä tähän on useita.

Tärkeimmät toimet, mitä tulee tehdä, on asettaa käänteisnimipalvelu, SPF (Sender policy framework) sekä DKIM (DomainKeys Identified Mail). Käänteisnimipalvelulla (Reverse DNS) tarkoitetaan toimintoa, jolla pystytään hakemaan IP-osoitetta vastaava nimipalvelunimi. Jotkin palvelimet voivat hylätä viestit, joille käänteisnimipalvelu ei palauta mitään arvoa tai jos saatu arvo eroaa siitä, mitä sähköposti antaa ymmärtää. SPF on validointiympäristö roskapostien estämiseen varmentamalla lähettäjän IP-osoitteen. DKIM puolestaan on toiminto, jolla lähettäjän domain nimi liitetään sähköpostiin, ja näin ollen voidaan määrittää, kuka postista on vastuussa.



KUVIO 16. Käänteisnimipalvelukysely.

Kuviosta 16 voidaan nähdä, että annettu IP-osoite palauttaa käänteisnimipalvelun mukaisen kirjoitusmuotoisen osoitteen ”newsletter.devnet.fi” nslookup-komennolla.

Final-Recipient: rfc822; spfinternational@terra.com.br
Original-Recipient: rfc822;spfinternational@terra.com.br
Action: failed
Status: 4.7.1
Remote-MTA: dns; vip-us-br-mx.terra.com
Diagnostic-Code: smtp; 450 4.7.1 Client host rejected: cannot find your hostname, [89.166.50.48]

--45481C1BE6.1390993593/smtp-1.wmhost.com
Content-Description: Undelivered Message
Content-Type: message/rfc822
Content-Transfer-Encoding: 7bit

KUVIO 17. Hylätty sähköpostiviesti.

Virhe lähetyspalvelimen käänteisnimipalvelussa johti useisiin kuvion 17 kaltaisiin virheviesteihin.

4.6.2 Uutiskirjeiden avausten seuranta

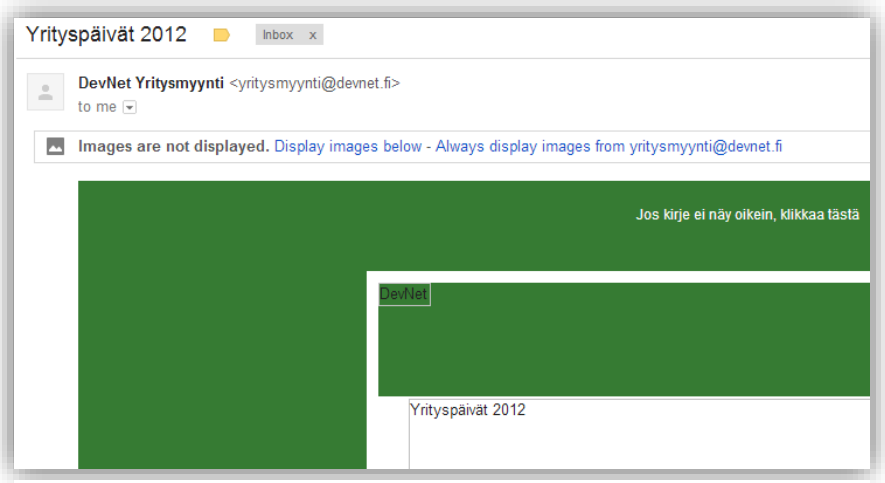
Kauniista uutiskirjeestä ei ole mitään hyötyä, jos sitä ei kukaan lue. Siksi on tärkeätä, että kerätään statistiikkaa avattujen viestien lukumäärästä.

Ongelmana kuitenkin on se, että ei ole olemassa täysin varmaa tapaa seurata viestien avauksia. Ja jos sellainen löytyisi, se estettäisiin hyvin nopeasti. Monet

ihmiset eivät halua antaa tietoja lähettäjälle, ja suurin osa sähköpostiohjelmistoista pitävät huolen, ettei käyttäjä automaattisesti niitä lähettäjälle lähetä.

Suosituin tapa on lisätä näkymätön yhden pikselin kokonen yksilöity kuva viestiin, joka haetaan palvelimelta, ja kun kuvan parametrit luetaan, kirjoitetaan ne tietokantaan. Toinen tapa on lisätä samat parametrit myös kaikkiin linkkeihin niin, että kun linkkiä klikataa, kulkee se välityspalvelimen kautta, joka kirjaa klikkauksen tietokantaan. Näillä menetelmillä on kuitenkin varjopuolensa. Pahimmassa tapauksessa viestin vastaanottanut sähköpostipalvelin merkkaa viestin suoraan roskapostiksi. Toinen ongelma on myös se, että sähköpostiohjelmistot eivät oletuksena näytä näitä viestin kuvia, eli käyttäjän on erikseen haettava kuvat palvelimelta. Jos käyttäjä valitsee olla lataamatta viestin kuvat, kuten kuviossa 18, jää viestin avaus-statistiikka täysin pimentoon. Toinen ääripää tälle taas on Googlen tarjoama GMail-sähköpostipalvelu, jossa Google itse hakee kaikki kuvat valmiiksi käyttäjälle ennen viestin avausta, mutta käyttäjän avatessa viestin, näytetään kuvat Googlen omilta palvelimilta. Näin ollen Google avaa kaikki viestit jo ennen kuin käyttäjä itse on viestiä nähnytkään. Lisäksi käyttäjät, joilla on käytössä tekstipohjainen sähköpostiohjelmisto, jäävät täysin huomiotta.

Suurimmat sähköpostittajayritykset, kuten MailChimp, tiedostavat myös nämä ongelmat, ja heiltä paras ratkaisu on pyytää uutiskirjeen tilaajia lisäämään lähettäjän sähköpostiosoite luotettuihin lähettäjiin osoitekirjaan, jotta kuvat latautuisivat automaattisesti aina. Hieno idea, mutta todennäköisesti suurin osa jättävää tämän mielummin tekemättä.



KUVIO 18 Tämän viesti on avattu mutta tieto siitä ei välity ilman kuvia

4.6.3 Uutiskirjeiden epäonnistuneet lähetykset

Vaikka osoitelistan kaikki osoitteet noudattaisivat kaikkia standardien asettamia muotoilusääntöjä, tulee silti aina esiintymään epäonnistuneita lähetyksiä. Suurin syy tähän on ihminen; yrityksen osoitelistat eivät välttämättä ole ajan tasalla, tai sähköpostiosotteissa on kirjoitusvirheitä.

Toinen mahdollinen tekijä ovat laitteistot. Vastaanottajan sähköpostilaatikko voi olla täynnä ja hylkää siksi uudet viestit. Sähköpostiosoite voi olla kokonaan poistettu palvelimelta esimerkiksi työsuhteen loputtua. Myös virheellisesti säädetyt roskapostin suodattimet voivat tulkita lähetykset roskapostiksi ja jättää ne ottamatta vastaan. Lähettäjällä ei ole paljoa tehtävissä tälläisille virheille.

Koska jokainen viestinvälittäjäagentti on erilainen, on myös jokaisen virheviesti erilainen. Ei ole olemassa yhtä työkalua, joka osaisi käsitellä kaikki mahdolliset virheviestit, joten virheestä kertovat paluuviestit (bounce, suom. ponnahtaa) joudutaan käymään jokainen läpi rivi riviltä ja etsiä niistä hakusanoja, joilla voidaan luokitella virheet.

Virheiden luokittelu on tärkeää, jotta käyttäjä näkee, miksi asiakas ei mahdollisesti ole viestiä saanut. Jos sähköpostiosoite ei ota viestiä vastaan,

merkkaa järjestelmä osoitteen epäkelvoksi eikä lähetä sinne enää viestejä. Näin säästyy viestikapasiteettia käyttäjältä. Mutta jos vastaanottajan sähköpostilaatikko on vain täynnä, voi käyttäjä silti lähettää seuraavankin viestin kyseiseen osotteeseen halutessaan.

Jos virheviesti ei vastaa mitään hakusanaa, asetetaan virhe tuntemattomaksi hetkellisesti ja lähetetään siitä ilmoitus kehittäjille, jotka käyvät kyseisen sähköpostin läpi ja luovat siitä uuden hakusanasäännön. Ajan myötä järjestelmä oppii yhä paremmin tunnistamaan virheet.

Epäonnistuneiden lähetysten käsittelyssä on myös sellainen tekijä, että jos sähköpostipalvelin huomaa viestin lähetettävän monesti samaan olemattomaan osoitteeseen, voi se merkata lähettäjän mustalle listalle eikä jälkeen enää ota mitään sähköpostiliikennettä vastaan kyseiseltä lähettäjältä tai jopa kyseiseltä IP-osoitteelta.

5 TOTEUTUS

Järjestelmä päätettiin rakentaa täysin omaan ympäristöön virtuaalipalvelimena. Näin mikään muu prosessi ei hidasta toimintoja tai vie resursseja muuhun toimintaan. DevNet Oy:llä on web-hotellitoimintaa, joten palvelimen asennus hoitui lähes automaattisesti. Palvelimen käyttöjärjestelmäksi valittiin Debian 6.0 ”squeeze” joka myöhemmin päivitettiin versioon 7.0 ”wheezy”. Postitusta varten ei tarvinnut erikseen asentaa sähköpostipalvelimia, koska DevNet Oy:n omistama WMHost tarjoaa sähköpostipalveluita ja kaiken sähköpostiliikenteen pystyi lähettäämään näillä palvelimilla.

5.1 Tietokanta

Tietokannan suunnittelu lähti liikkeelle autentintikointikirjastosta. Autentikointiin käytettiin CodeIgniter Frameworkille vartavasten luotua Ion_Auth autentikointikirjastoa.

5.2 Ion_Auth

Ion_Auth on kevyt, helppo ottaa käyttöön, ja se tarjoaa pienellä vaivalla rekisteröintilomakkeen ja muut toiminnot, mitä käyttäjätunnushallinnoinnilta voi odottaa. Ion_Auth-kirjaston asennus onnistui CodeIgniterin Sparks-pakettihallintasovelluksella. Kirjaston käyttöönotto vaatii vain tiedostopaketin latauksen. Paketti puretaan CodeIgniter projektikansioon, ja tämän jälkeen tietokantaan tuodaan tarvittavat taulut paketin mukana tulleesta SQL-kyselytiedostosta.

Kun tämä toimenpide on suoritettu, tarvitsee kirjasto enää vain ladata ohjaimeen kuvion 19 mukaisesti ja käyttö voi alkaa.

```
$this->load->library('ion_auth');
```

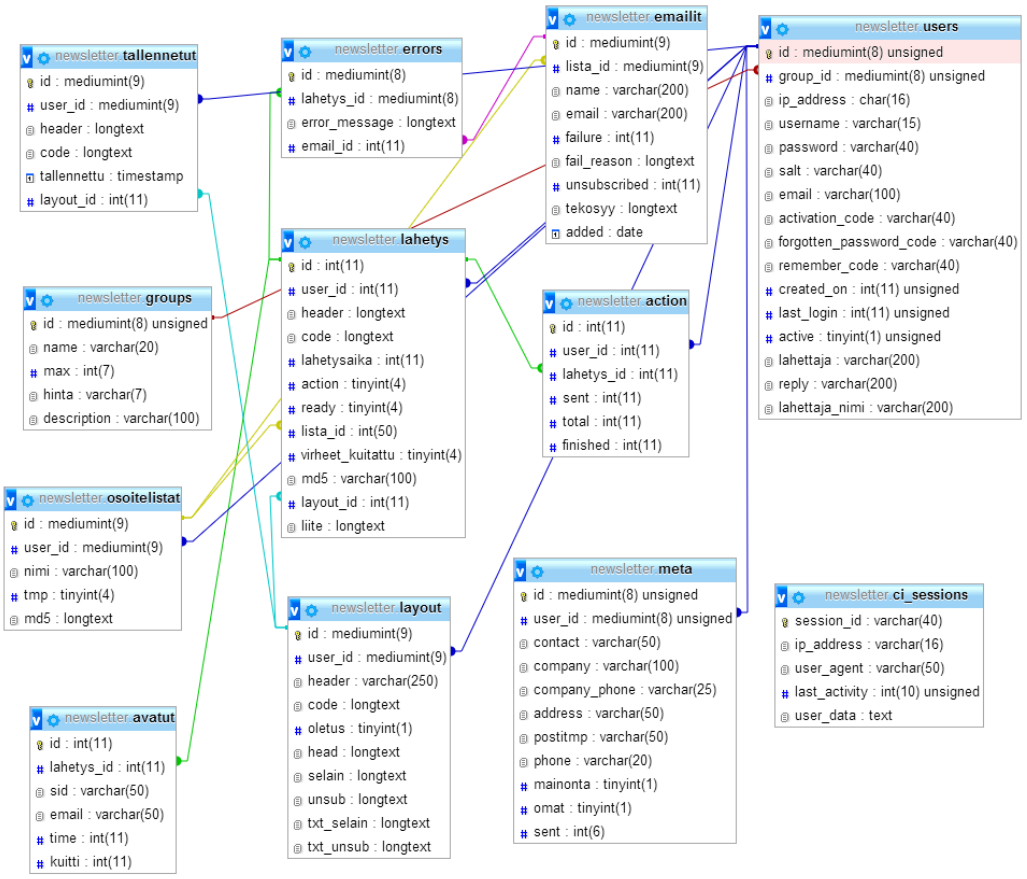
KUVIO 19. Ion_authin lataus kontrolleriin (Ben Edmunds 2012.)

Kirjaston malliesimerkit havainnollistavat kuviossa 18 selkeästi, kuinka perusoperaatiot toimivat ja hyvä dokumentointi tekee helpoksi koodin muokkauksen.

```
$identity = 'ben.edmunds@gmail.com';
$password = '12345678';
$remember = TRUE; // remember the user
$this->ion_auth->login($identity, $password, $remember);
```

KUVIO 20. Ion_authin kirjautumistoiminto yksinkertaisimmillaan

Koska Ion_auth luo itselleen tietokantataulut, jotka se vaatii toimiakseen, oli muiden tietokantataulujen suunnittelu helppo lähteä laajentamaan tältä pohjalta. Lopullinen tietokantakuvaus esitellään kuviossa 21.



KUVIO 21. Tietokannan graafinen kuvaus relaatioineen

Talut ”users”, ”groups” ja ”meta” ovat Ion_auth-kirjaston luomia, mutta näihin on lisätty kenttiä, jotka ovat sopineet kyseiseen tauluun. Esimerkkinä groups (ryhmät) on saanut lisäkenttinä ryhmälle määrätyn kuukausihinnan ja kuukausittaisen viestikiintiön. Meta-taulu puolestaan sisältää asiakkaan metatietoja, kuten lähetettyjen viestien määrä ja valinta DevNet Oy:n suoramainonnan vastaanottamisesta. Users (käyttäjät)-taulun lisäyksinä ovat lähettäjän sähköpostiosoite, vastausosoite ja lähettäjän nimi. Useimmiten nämä arvot ovat samat kuin käyttäjän omat tiedot, mutta joissakin tapauksissa halutaan, että esimerkiksi viestit tulevat osoitteesta uutiskirje@domain.com, ja jos viestiin vastaa, niin viesti lähteekin osoitteeseen myynti@domain.com.

Asiakkaalle tallennetut viestipohjat tallennetaan layout (ulkoasu)-tauluun. Ulkoasulle annetaan nimi ja lisätään optimoitu koodi. Jos asiakkaalla on useita viestipohjia, valitaan haluttu pohja oletuspohjaksi. Viestipohjan ylä- ja alaosan

lähdekoodi annetaan eri kenttiin, koska daemon lisää näihin personoidun seuranta- sekä listaltapoistumiskoodin lähetyksen yhteydessä. Myös HTML-kieltä tukemattomia selaimia varten on tarpeen antaa viestin ylä- ja alaosa, mutta niin, että linkkitekstinä on varsinainen linkki itse. Muussa tapauksessa linkkiä ei voi klikata, koska sähköpostiohjelma tulostaa vain linkin tekstin.

Kun asiakas tekee muutoksia viestipohjaan ja tallentaa ne, tallennetaan muutokset ”tallennetut”-tauluun. Tauluun tallentuvat tallennuksen nimi sekä muokattu lähdekoodi. Tallennuksen yhteydessä tallennus saa myös pohjan tunnisteen layout-taulusta. Kun viestejä läheteään, katsoo daemon tämän layout_id tiedon perusteella, minkälaisen viestin ylä- ja alaostan sähköpostiin lisätään.

Lähetysvalmiista viestistä luodaan tietokannan lahetys-tauluun instanssi. Lähetyksen tietojen lisäksi tallennetaan lähetysaika, edellä mainittu layout_id ja sähköpostilistan tunnus. Daemon monitoroi yksinomaan tätä tietokantataulua uusilta lähetyksiltä. Jos rivejä, joissa lähetysaika on käsillä, ilmenee tarkastaa daemon, onko rivissä action asetettu nollaksi vaiko ykköseksi. Tämä action-tieto ilmoittaa, onko lähetys jo aloitettu. Jos actionin arvo on yksi mutta ready-tiedon arvo on vielä nolla, voidaan todeta, että lähetys on jäänyt kesken ja lähetystä jatketaan. Jos molemmat arvot ovat nollia, lähetys alkaa alusta, ja jos taas arvot ovat ykkösiä, on lähetys suoritettu onnistuneesti loppuun.

Daemon pitää kirjaa lähetyksen tilasta action-taulussa. Tauluun tallennetaan lähetyksen alussa lähettäjä, lähetyksen tunnus, lähetettävien viestien määrä sekä lähetettyjen viestin määrä. Jos lähetys on jäänyt kesken, voidaan lähetystä jatkaa helposti, kun otetaan lähetyksen osoitelista käsittelyyn ja jatketaan lähetystä järjestysnumeroltaan seuraavana vuorossa olevaan osoitteeseen kunnes lähetettävien ja lähetettyjen viestien määrä on sama. Lopuksi lähetys merkataan valmiiksi asettamalla finnished-tieto ykköseksi. Tästä daemon tietää myös asettaa lähetys-tauluun ready-tiedon ykköseksi.

Kaikista syötetyistä osoitteista tehdään osoitelista. Vaikka lähetys onnistuu myös syöttämällä listan käsin, lähetystä varten käsin syötetystä listasta generoidaan väliaikainen lista joka tuhoutuu lähetyksen päätteeksi. Käsin syötettyjä listoja käytetään pääaisassa vain testauksessa omien viestien lähettelyssä.

Jokainen sähköpostiosoite lisätään emailit-tauluun. Jos käyttäjä poistuu sähköpostilistalta, asetetaan unsubscribed-tiedoksi poistumisajankohta ja tekosyy-kenttään lisätään mahdollinen listaltapoistumis-syy. Jos unsubscribed-kentässä on syöte, daemon ei lähetä viestiä tähän osoitteeseen enää.

Epäonnistuneen lähetyksen sattuessa daemon käy palautuneen viestin läpi etsien siitä hakusanoilla virhekoodeja. Jos virhe havaitaan, lisätään tieto siitä errors-tauluun. Sähköpostiosoitteet, jotka ovat lisätty errors-tauluun, sivuutetaan lähetyksissä kyseisistä osoitelistoista.

Avatuista viesteistä pidetään kirjaa avatut-taulussa. Tauluun tallentuu lähetyksen ja sähköpostin tunnus, sekä avausaika. DevNet Newsletter huomioi myyntipainotteiset sähköpostit, ja avattujen viestien seurantaan on luotu toiminto palvelun käyttäjälle kuitata viestin avanneet. Tämä toiminto on hyödyllinen esimerkiksi, jos viestin avanneita soitellaan läpi. Näin pysyy kirjanpito soitetuista asiakkasita helposti ajantasalla muutamalla klikkauksella.

5.3 Swift Mailer

Sähköpostien lähetykseen sopi parhaiten Swift Mailer PHP-kirjasto. Myös Swift Mailer asennettiin käyttäen Sparks-pakettihallintaa. Swift Mailer tarjoaa kaiken, mitä sähköpostin lähettämiseen tarvitaan yhdessä paketissa, ja ennen kaikkea, dokumentaatio on erittäin selkeä ja sisältää paljon esimerkkejä. Koska viestien lähetykseen oli jo palvelimet olemassa WMHost:lta, Swift Mailerin käyttöönotto vaati ainoastaan SMTP-palvelimen käyttäjätunnusten sekä palvelimen osoitteen syötön kuvion 22 esimerkin mukaisesti.

```php
require_once 'lib/swift_required.php';

// Create the Transport
$transport = Swift_SmtpTransport::newInstance('smtp.example.org', 25)
  ->setUsername('your username')
  ->setPassword('your password')
  ;

/*
You could alternatively use a different transport such as Sendmail or Mail:

// Sendmail
$transport = Swift_SendmailTransport::newInstance('/usr/sbin/sendmail -bs');

// Mail
$transport = Swift_MailTransport::newInstance();
*/

// Create the Mailer using your created Transport
$mailer = Swift_Mailer::newInstance($transport);

// Create a message
$message = Swift_Message::newInstance('Wonderful Subject')
  ->setFrom(array('john@doe.com' => 'John Doe'))
  ->setTo(array('receiver@domain.org', 'other@domain.org' => 'A name'))
  ->setBody('Here is the message itself')
  ;

// Send the message
$result = $mailer->send($message);
```

KUVIO 22. Swift Mailerin eri sähköpostin lähetysmenetelmiä. (Swift Mailer 2013.)

Swift Mailer mahdollistaa myös viestien massalähetyksen. Viestin vastaanottajat tulee vain asettaa taulukkoon kuvion 23 mukaisesti, ja kirjasto hoitaa iteroinnin itse.

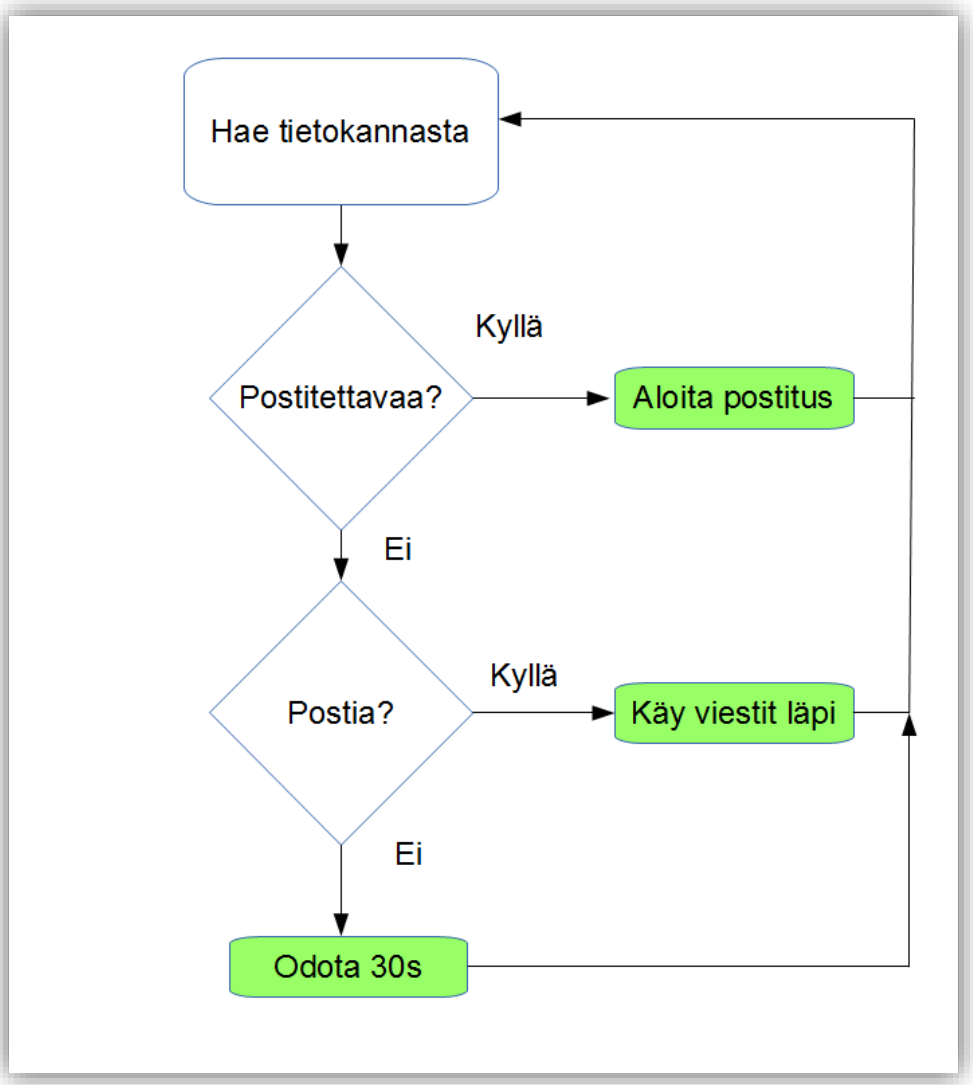
```php
// Send the message
$failedRecipients = array();
$numSent = 0;
$to = array('receiver@domain.org', 'other@domain.org' => 'A name');

foreach ($to as $address => $name)
{
  if (is_int($address)) {
    $message->setTo($name);
  } else {
    $message->setTo(array($address => $name));
  }

  $numSent += $mailer->send($message, $failedRecipients);
}
```

KUVIO 23. Swift Mailer ja massapostitus

Viestien lähetys toteutettiin niin, että palvelimella ajettava daemon-ohjelma tarkastaa aika-ajoin, onko tietokannassa lähetettävää, ja jos lähetettäviä viestejä löytyy, alkaa ohjelmisto lähetyksen välittömästi, ellei lähetysaikaa ole asetettu tulevaisuuteen. Daemon päätettiin toteuttaa PHP:lla.

5.4 Daemon

Daemonin toteutuksessa käytettiin PEAR-pohjaista System_Daemon kirjastoa. Kuviosta 22 voi havainnoida yksinkertaistetun toimintakaavion daemonin toiminnoista.



KUVIO 24. Daemonin toimintaperiaate

System_Daemon paketin tarkoitus on tehdä PHP-CLI scripteistä daemonin kaltaisia ”ainapäällä” olevia ohjelmia. Ohjelmiston tarjoamia etuja verrattuna ”oikeisiin” daemoneihin, jotka on toteutettu C/C++ kielillä, on muun muassa lähdekoodin käytettävyys. Sama lähdekoodi, jota verkkosivusto käyttää, toimii myös tässä tapauksessa. Lisäksi PHP-kielen tarjoamat lukuisat funktiot nopeuttavat kehitystä ja mikä parasta: web-kehittäjän ei tarvitse opiskella konekieliä tai yrityksen palkata erikseen osaajaa. (KVZ 2009.)

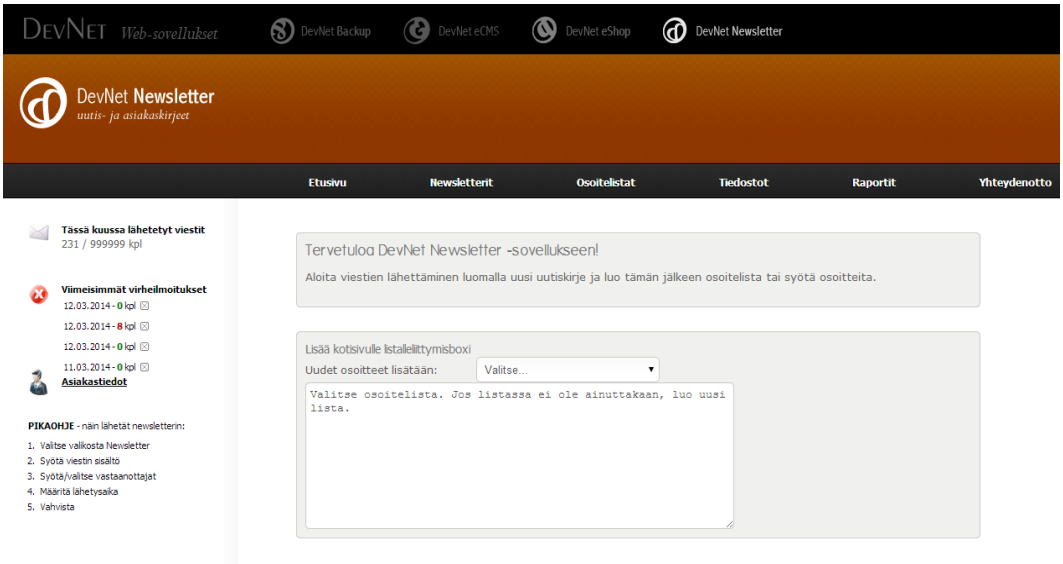
```
31 require_once "System/Daemon.php";
32
33
34 // Bare minimum setup
35 System_Daemon::setOption("appName", "newsletter");
36 System_Daemon::setOption("authorEmail", "niko.salonen@devnet.fi");
37 System_Daemon::setOption("appDescription", "DevNet Newsletter daemon");
38 System_Daemon::setOption("authorName", "Niko Salonen");
39
40
41 if (!System_Daemon::isInBackground() && !System_Daemon::isRunning()) {
42     $init_path = System_Daemon::writeAutoRun(false);
43
44 // Spawn Deamon!
45     System_Daemon::start();
46     System_Daemon::log(System_Daemon::LOG_INFO, "Daemon: '" . System_Daemon::getOption("appName")
47
48
49     $runningOkay = true;
50     while (!System_Daemon::isDying() && $runningOkay) {
```

KUVIO 25. Daemon PHP scriptin minimitiedot.

Kuvio 25 näyttää daemonin luontiin vaadittavat minimitiedot. Riviltä 31 ohjelmaan lisätään ”taustaäly”. Tämän jälkeen sovellukselle annetaan nimi ja kehittäjä. Sitten tarkastetaan, onko sovellus jo ajossa, ja jos ei ole, niin se käynnistetään System_Daemon::start(); komennolla rivillä 45. Nyt sovellus on ajossa, ja se pysyy niin kauan ajossa, kunnes muuttuja $runningOkay saa arvokseen false tai System_Daemon::isDying() funktio ajetaan.

6 ESIMERKKILÄHETYS

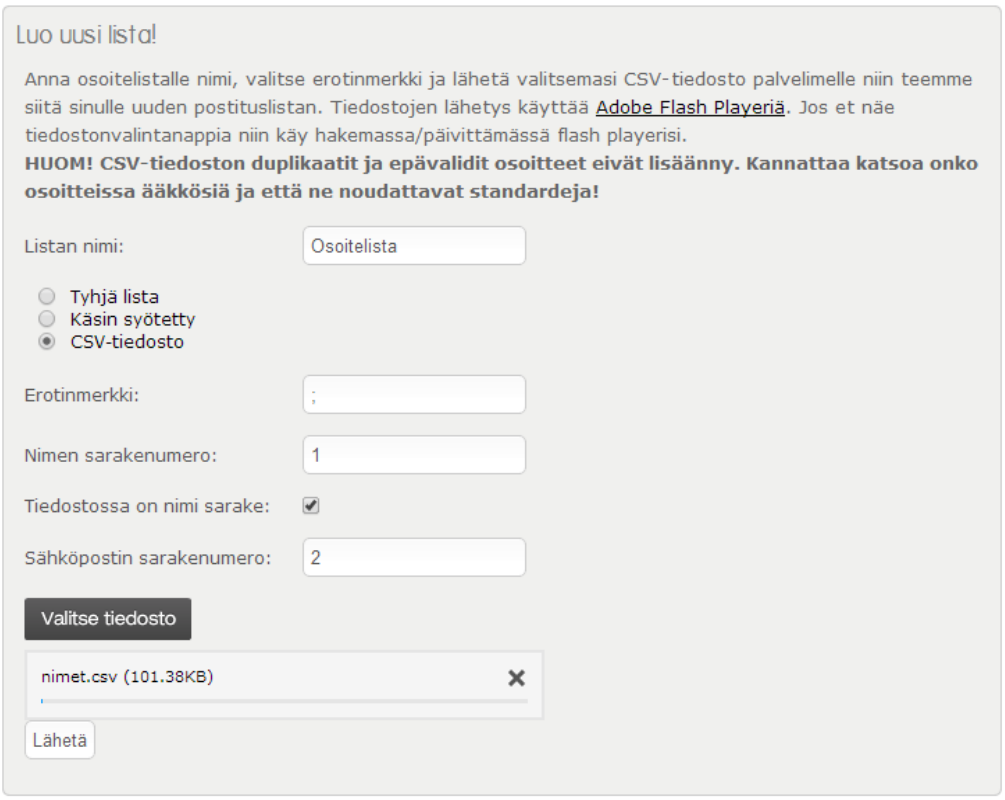
Tässä kappaleessa käydään läpi, kuinka uutiskirjeen lähetys tapahtuu DevNet Newsletter-palvelua käyttäen. Kuvio 26 näyttää palvelun etusivun, josta kaikki navigaatio lähtee.



KUVIO 26. DevNet Newsletter etusivu

6.1 Osoitelistan lisäys

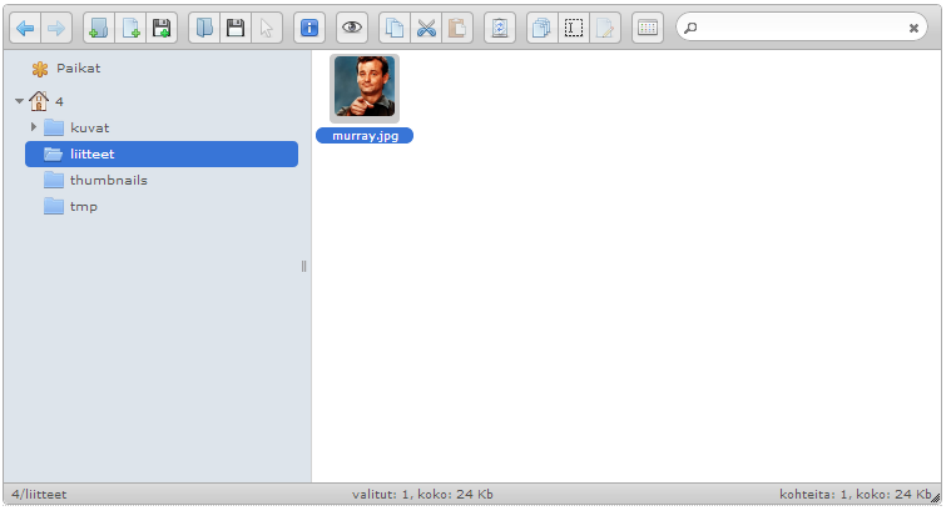
Osoitelistan lisäys tietokantaan on suoraviivainen prosessi. Aluksi valitaan ylävalikosta ”Osoitelistat”-linkki. Aukeavan sivun alaosasta löytyy kuvio 27 mukainen lomake. Listalle annetaan nimi, ja jos lista luodaan CSV-tiedostosta, tarvitsee syöttää myös tiedoston erotinmerkki sekä nimen ja sähköpostiosoitteen sarakenumero.



KUVIO 27. Uuden osoitelistan lisääminen.

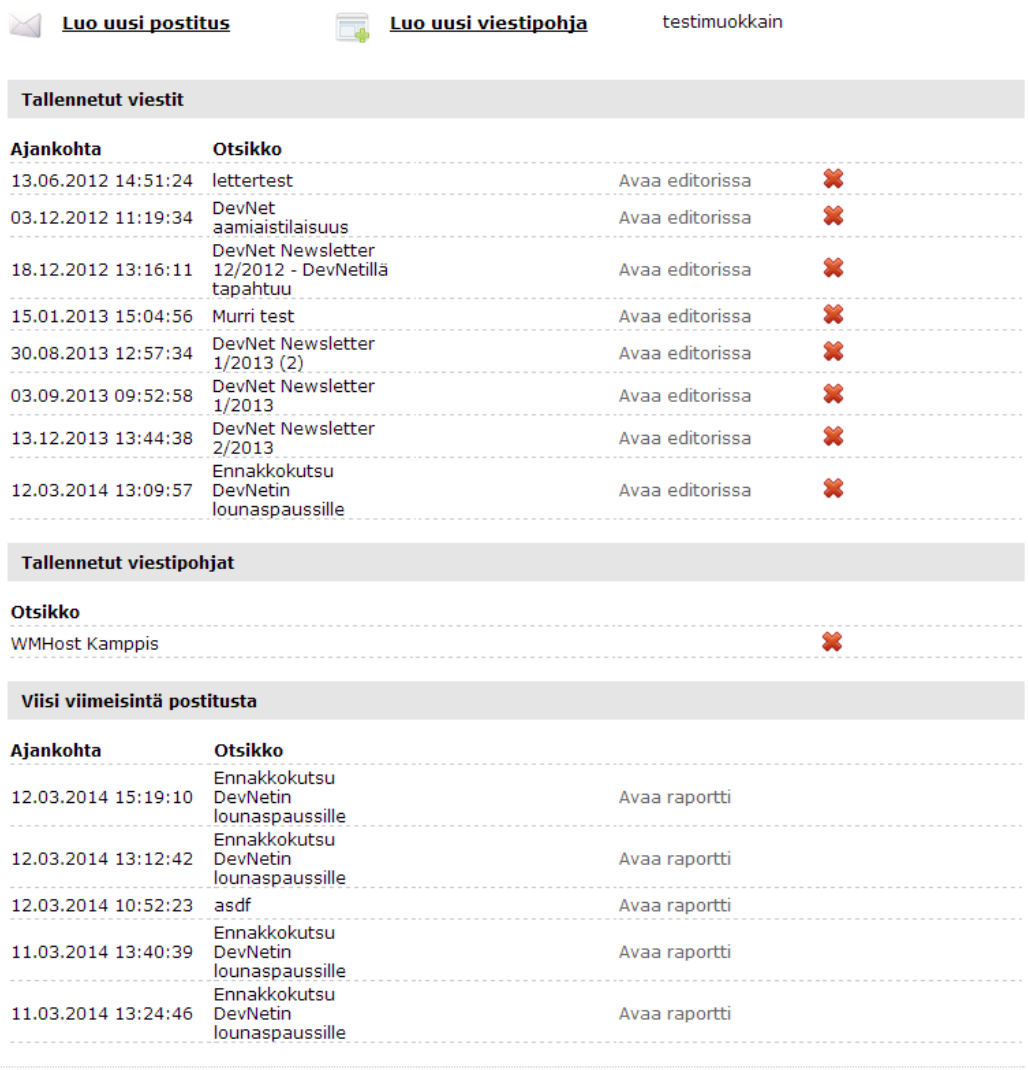
6.2 Uusi lähetys

Uuden postituksen luontiprosessi liitetiedostolla alkaa lisäämällä haluttu liitetiedosto tiedostohallintaan. Ensin valitaan ”liitteet”-kansio, johon tiedoston voi lisätä raahaamalla se hallinnan päälle kuvion 28 tavoin.



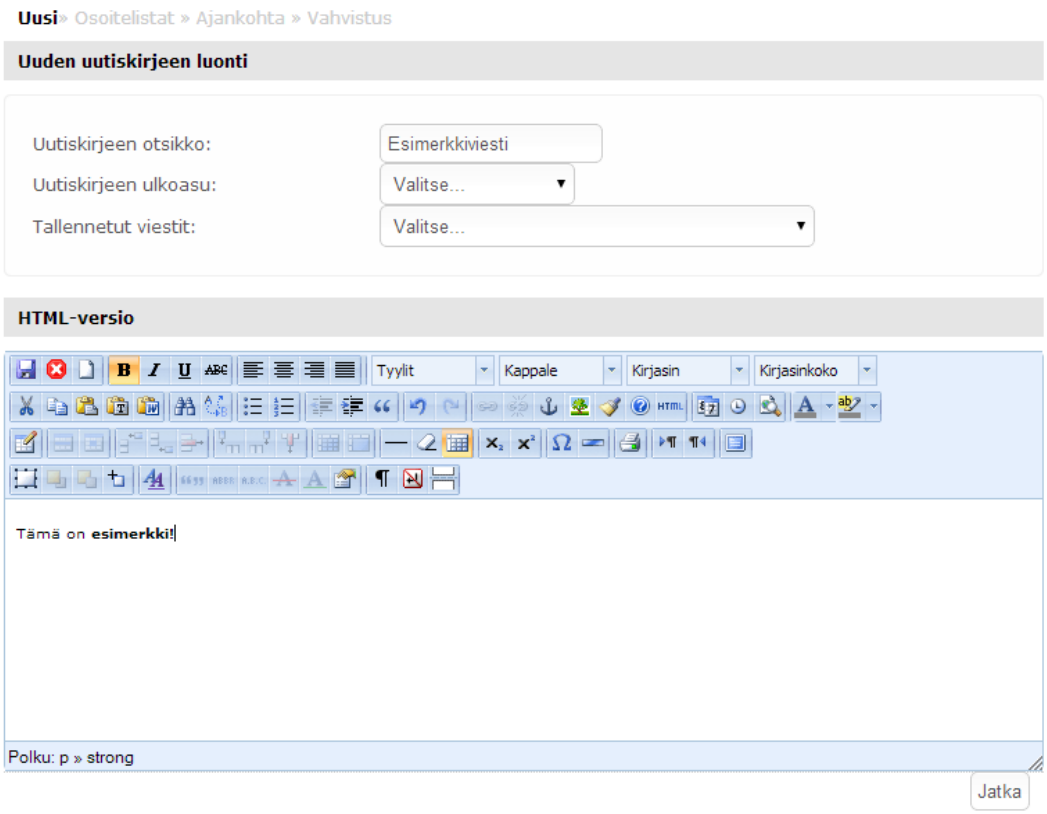
KUVIO 28. Liitetiedostojen hallinta.

Seuraavaksi ylävalikosta valitaan ”Newsletterit”-linkki. Tämän linkin takaa avautuu kuvion 29 mukainen lista tallennetuista viesteistä sekä lista viimeisimmistä postituksista.



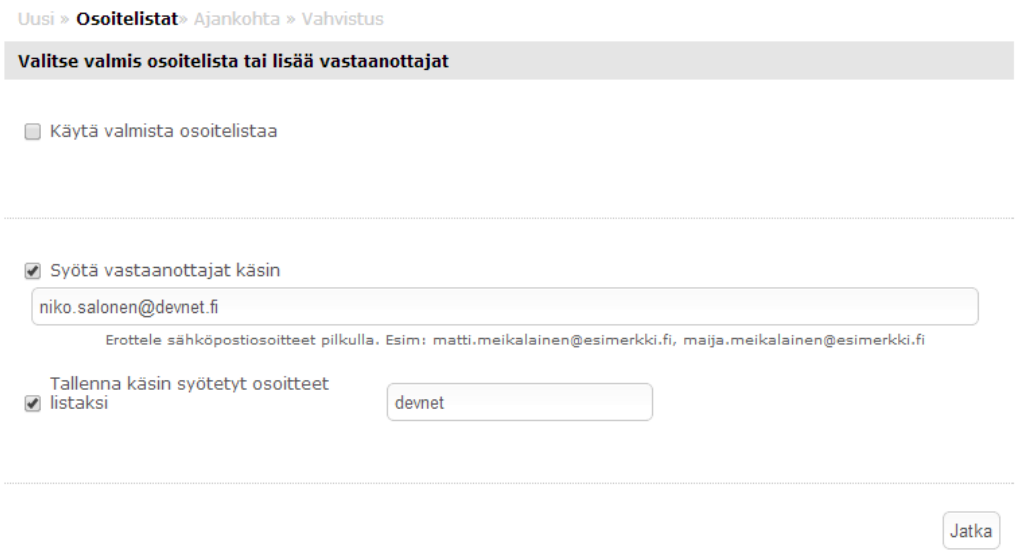
KUVIO 29. Viestinluontisivun päänäkymä

Tältä sivulta klikataan ”Luo uusi postitus”-linkkiä. Vaihtoehtoisesti voidaan myös valita tallennettu viesti ”Avaa editorissa”-linkistä. Seuraavaksi sivulle avautuu viestinrakennusnäkymä kuten kuviossa 30. Näkymässä voi antaa uutiskirjeelle otsikon, valita tallennetun viestin, tallennetun ulkoasun ja muokata viestiä tekstieditorissa. Kun viesti on sellainen, että se on valmis lähetystä varten, painetaan Jatka-nappia. Tässä vaiheessa viesti kannattaa myös tallentaa disketti-ikonista, jotta muutoksia tehtäessa voidaan muokkausta jatkaa tästä kohdasta.



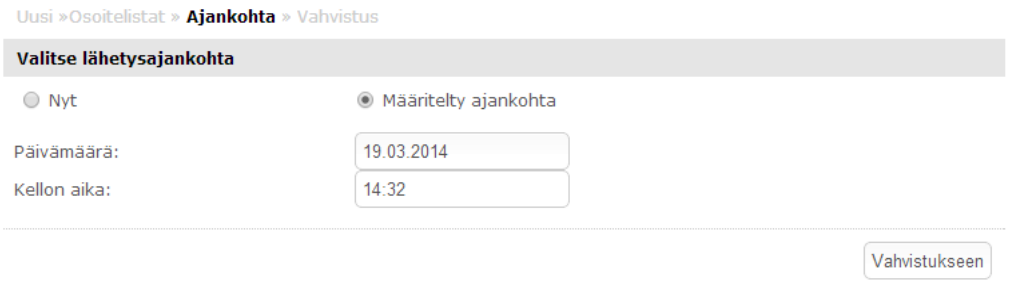
KUVIO 30. Viestin muokkaustyökalu

Seuraavaksi valitaan viestin vastaanottajat. Valintaan voi käyttää aiemmin lisättyä osoitelistaa tai syöttää listan käsin. Kuviossa 31 on valittu vain käsin syötettävä osoite, joka tallennetaan osoitelistaksi ”devnet”.



KUVIO 31. Viestin vastaanottajien valinta

Seuraavassa näkymässä valitaan lähetysajankohta. Ajankohdan voi määrittää minuutin tarkkuudella niin halutessaan kuvion 32 mukaisesti.



KUVIO 32. Lähetysajankohdan asetus

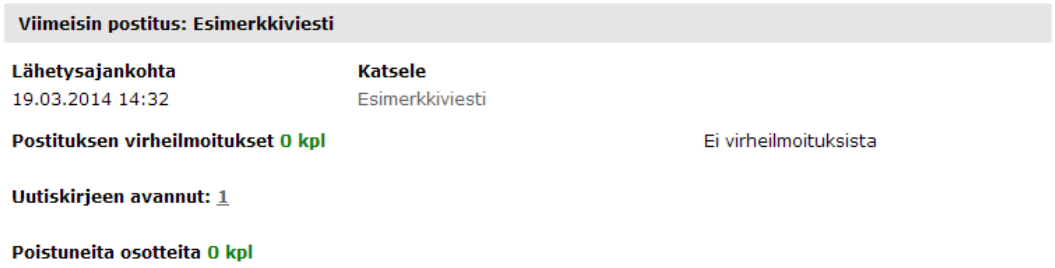
Lopuksi näytetään yhteenveto lähetyksestä. Tässä vaiheessa tulee laittaa rasti ruutuun, jossa vakuutetaan postin sisältö lainsäädännön mukaiseksi kuvion 33 mukaan. Lähettäjä voi vielä halutessaan esikatsella lähetystään selaimessa ja avata vastaanottajien osoitelistan tarkasteluun. Liitetiedosto lisätään kuvan

vetovalikosta. Lähetystiedot kirjataan tietokantaan heti kun ”Lähetä”-nappia on painettu. Jos lähetysaika on heti, myös lähetys aloitetaan välittömästi.

KUVIO 33. Loppuyhteenveto

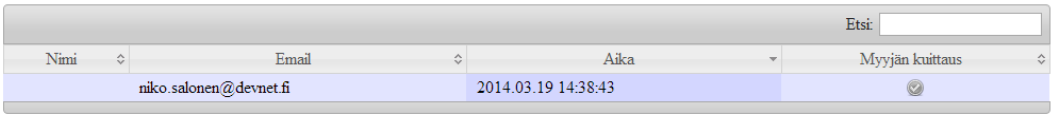
6.3 Raportit

Kun viestit on lähetetty, voidaan avata raportit-sivusto tarkasteluun. Raportista ilmenevät kuvion 34 mukaan lähetysajankohta, viestin otsikko, virheilmoitusten määrä, uutiskirjeen avanneet sekä osoitelistalta poistuneet.



KUVIO 34. Esimerkkiviestin esimerkkiraportti

Kun klikataan uutiskirjeiden avanneiden lukumäärää, avautuu kuvion 35 lista sähköpostiosotteista jotka viestin ovat avanneet. Listasta näkee myös ajankohdan ja avauksen voi kuitata painamalla harmaata nappia. Jos vastaanottaja ei ole ladannut kuvia, ei avaus rekisteröidy tähän listaan.



KUVIO 35. Lista viestin avanneista osoitteista

7 YHTEENVETO JA POHDINTA

Opinnäytetyön aihepiiri oli erittäin mielenkiintoinen. Työssä milenkiintoista oli erityisesti se, että se piti sisällään sekä webbisivujen, että palvelinpuolen työskentelyä.

Tavoitteena oli siis luoda uusi palvelu DevNet Oy:lle. Palvelun avulla DevNet Oy:n asiakkaat voivat lähestyä omia asiakkaitaan sähköpostitse ja samalla saada tärkeää statistiikkaa viesteistä.

Palvelu saatiin valmiiksi ja annettiin käyttöön asiakkaille. Lopputulos vastaa asetettuja tavoitteita, ja suurimmat odotetut ongelmat saatiin ratkaistua. Täysin ilman ongelmia ei kuitenkaan vältytty.

Ensimmäiset käyttökokemukset asiakkailta kertoivat, että ratkaisu, jolla viestin luonti ja muokkaus toteutettiin, osoittautui liian tekniseksi tavalliselle käyttäjälle. HTML-pohjaisille sähköpostiviesteille erittäin tärkeän rakenteen saa hajoitettua kokemattomissa käsissä liian helposti yhdellä ylimääräisellä rivivaihdolla tai askelpalauttimen painalluksella. Muilta osin käyttökokemukset ovat olleet positiivisia.

Jos nyt pitäisi miettiä, mitä olisi tehnyt toisin, olisi viestinmuokkaustyökalu tärkeimpänä listalla. Editorista olisi pitänyt tehdä rajoitetumpi, jotta viestin rakenne ei rikkoudu. Koko viestin muokkauksen sijaan asiakas voisi valita rungon muodon valmiiksi ja sitten muokata aktiivisia tekstikenttiä. Tämä menettely oli ehkä alkuun valittu myös järjestelmäkuvaukseen, jos kilpailevien palveluiden tutkintaan olisi panostettu enemmän.

DevNet Newsletteriksi ristityn palvelun tulevaisuus on tällä hetkellä hieman epäselvä. Aktiivisia asiakkaita on kourallinen, ja heiltä on tullut hyvää palautetta sekä hyviksi todettuja kehitysehdotuksia, mutta toistaiseksi kehitys on ollut jäissä. Aika näyttää, löytyykö jatkokehitykselle aikaa ja panosta.

LÄHTEET

Ben Edmunds – Ion Auth Docs. 2012. [viitattu: 06.04.2014]. Saatavissa: http://benedmunds.com/ion_auth/

CampaignMonitor. 2006. Guide to CSS support in email. [viitattu 01.06.2010]. Saatavissa: http://www.campaignmonitor.com/css/

DevNet Oy - Yritystiedot. 2013. [viitattu 12.09.2010]. Saatavissa: http://www.devnet.fi/yritys

EllisLab. 2013. [viitattu 15.01.2014]. Saatavissa: http://ellislab.com/codeigniter/user-guide/

http://kvz.io/blog/2009/01/09/create-daemons-in-php/

Järvinen, Petteri. 2000. Sinulle on sähköpostia. Teknolit. [viitattu 01.10.2011]. Saatavissa: http://petteri.com/kirjat/email/index.html

KVZ. Create daemons in PHP. 2009. [viitattu 01.06.2011]. Saatavissa:

MAAWG. 2011. Email Metrics Report First, Second and Third Quater 2011. [viitattu 08.12.2012]. Saatavissa: http://www.maawg.org/sites/maawg/files/news/MAAWG_2011_Q1Q2Q3_Metrics_Report_15.pdf

RFC-4180. 2005. [viitattu 24.04.2013].

Saatavissa: http://www.ietf.org/rfc/rfc4180.txt

UTalkMarketing. 2011. Consumers using technology to filter out marketing. 2011. [viitattu 12.12.2012]. Saatavissa: http://www.utalkmarketing.com/pages/Article.aspx?ArticleID=21935&Title=Consumers_using_technology_to_filter_out_marketing

Wikipedia - CSV. 2013. [viitattu 25.04.2013]. Saatavissa: http://fi.wikipedia.org/wiki/CSV

Wikipedia - ESMTP. 2013. [viitattu 11.10.2013]. Saatavissa: http://fi.wikipedia.org/wiki/ESMTP

Wikipedia - SMTP. 2013. [viitattu 28.09.2013]. Saatavissa: http://fi.wikipedia.org/wiki/SMTP

LIITTEET