LAPIN AMK
Lapland University of Applied Sciences

Ingo-ilmoittautumisjärjestelmän muuttaminen SaaS-tyyppiseksi palveluksi

Tommi Finnilä

Kaupan ja kulttuurin toimialan opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
Tradenomi

TORNIO 2014

TIIVISTELMÄ

LAPIN AMMATTIKORKEAKOULU, Kauppa ja kulttuuri

Koulutusohjelma:	Tietojenkäsittelyn koulutusohjelma
Opinnäytetyön tekijä:	Tommi Finnilä
Opinnäytetyön nimi:	Ingo-ilmoittautumisjärjestelmän muuttaminen SaaS-tyyppiseksi palveluksi
Sivuja (joista liitesivuja):	29 (0)
Päiväys:	18.6.2014
Opinnäytetyön ohjaaja:	Yrjö Koskenniemi

Tämän opinnäytetyön tarkoituksena on muokata olemassa oleva ilmoittautumisjärjestelmä yhdeksi SaaS-tyyppiseksi palveluksi. Tavoitteena on selvittää järjestelmän muutostyön avulla, mitä palvelun muuttaminen vaatii ja ovatko muutokset mahdollista tehdä.

Ilmoittautumisjärjestelmä on alun perin rakennettu yhden asiakkaan käyttöön, minkä jälkeen sitä on kysynnän vuoksi tarjottu myös muille. Tämä aiheutti ongelmallisen tilanteen, jossa ylläpidettäviä järjestelmiä oli useita. Tilanne vaikeutti ja hidasti sekä ylläpito- että kehittämistyötä. Tämän vuoksi oli tarkoituksenmukaista kehittää järjestelmää tukemaan useita käyttäjiä yhden palvelun alaisena.

Järjestelmä on rakennettu PHP-sovelluskehyksen päälle, joka tarjoaa työkalut ja pohjan nopeaan ja helppoon kehittämistyöhön. Käytetyn sovelluskehyksen hyötyjä hyödynnetään tehokkaasti kyseisessä järjestelmässä ja se mahdollistaa myös joustavamman kehittämisprosessin. Palvelun uudelleen rakennettu hallintapaneeli tukeutuu vahvasti JavaScript-toimintoihin ja niiden avulla sen käyttöön onkin saatu joustavuutta ja johdonmukaisuutta.

Työn tuloksena on järjestelmä, joka tukee useita käyttäjiä, heidän tapahtumiaan, sekä mahdollistaa koko ilmoittautumisketjun toteuttamisen palvelun avulla. Palvelun kehittäminen SaaS-tyyppiseksi mahdollistaa paremman jatkokehittämisen ja helpomman ylläpidon, mikä auttaa asiakkaita ja ylläpitäjiä myös tulevaisuudessa.

Asiasanat: PHP, JavaScript, SaaS, Ilmoittautumisjärjestelmä

ABSTRACT

LAPLAND UNIVERSITY OF APPLIED SCIENCES, Business and culture

Degree programme:	Bachelor of Business Administration
Author(s):	Tommi Finnilä
Thesis title:	Converting Ingo-enrollment service to SaaS type of service
Pages (of which appendixes):	29 (0)
Date:	18.6.2014
Thesis instructor(s):	Yrjö Koskenniemi

The objective of this thesis research is to convert an existing enrollment service to a single SaaS type of service. The objective is to find out what this conversion requires and if it is possible to make the conversions.

This enrollment service is originally built for one customer. Because of the increasing demand for this service, it has also been offered to other customers. The fact that there are several customers using the service caused a problematic situation, because there were multiple systems to maintain. It complicated and slowed down the maintenance and development work. Because of this, it was practical to develop the system to support multiple customers under one service.

The system is built on top of PHP-framework, which provides tools and base for a quick and easy development work. Benefits of the used framework are utilized effectively in the system and it makes more flexible development work possible. Service's rebuilt control panel relies heavily on JavaScript functions and therefore its usage is more flexible and coherent.

Outcome of this work is system, which supports multiple customers and their events, as well as makes it possible to offer whole enrollment chain via one service. Developing the system to be SaaS type of service makes further development and maintenance work easier, which helps customers and developers over time.

Keywords: PHP, JavaScript, SaaS, Enrollment service

SISÄLLYS

1 JOHDANTO

Erilaisiin urheilutapahtumiin ilmoittautuminen on muuttunut viime vuosien aikana entistä enemmän verkkopohjaiseksi. Tämän vuoksi erilaisille internetissä toimiville ilmoittautumispalveluille on kysyntää. Alalle onkin tullut viime vuosina uusia yrittäjiä ja vanhoja järjestelmiä on kehitetty edelleen.

Tämän opinnäytetyön aiheena on Ingo-ilmoittautumisjärjestelmän muokkaaminen SaaS-tyyppiseksi pilvipalveluksi. Tässä työssä on tarkoituksena tutkia mitä vaaditaan, jotta palvelua voidaan kehittää kyseiseen suuntaan ja myös toteuttaa itse palvelun kehittäminen tämän tyyppiseksi. Näin saadaan hyvä kuva siitä, mitä työn kaltainen muutos vaatii ja samalla voidaan tehdä myös havaintoja nykyisen järjestelmän puutteista.

Vanhanaikaisen järjestelmän käyttäminen kuluttaa nykymuodossaan turhan paljon resursseja ylläpitäjän näkökulmasta ja aiheuttaa ongelmia järjestelmän kehittämistyössä, jonka vuoksi järjestelmän tehokas edelleenkehittäminen on haastavaa. Tämän vuoksi uudistamiselle oli tarvetta, koska tällä halutaan parantaa tulevaisuuden näkymiä sekä ylläpitäjän että käyttäjän osalta.

Opinnäytetyössä käydään läpi uuden järjestelmän taustoja, vaatimuksia, käytettyjä tekniikoita, suunnitelmia järjestelmän toteuttamiseksi ja toteutuksen eri vaiheita.

Olemassa oleva järjestelmä on luotu käyttäen PHP-, HTML-, CSS-, ja JavaScript ohjelmointi- ja kuvauskieliä. Itse järjestelmä on rakennettu PHP-sovelluskehyksen päälle, jonka avulla voidaan tuottaa selkeämpää koodia, jota on helpompi ylläpitää. Uuteen järjestelmän rakenteeseen tehdään myös pieniä muutoksia ja lisäyksiä, jotka keskittyvät lähinnä tietoturvaan ja käyttöliittymän rakentamiseen.

Tärkeä osa työstä liittyy eri käyttäjien tietojen hakemiseen ja siihen, että käyttäjällä on mahdollisuus päästä käsiksi vain niihin tietoihin joihin hänellä on oikeus. Tämä on tärkeää, koska tiedot sisältävät henkilötietoja ja niihin käsiksi pääseminen voi mahdollistaa tietojen väärinkäytöksiä.

2 TAUSTA

2.1 Toimeksiantaja

Mainostoimisto Heinäkuu on kokkolalainen vuonna 1994 perustettu mainosalan yritys. Yritys kertoo sen tärkeimmiksi osaamisalueisiin kuuluvan muun muassa brändisuunnittelun, mainonnan ja markkinoinnin suunnittelun, graafisen suunnittelun, verkkopalveluiden ja kotisivujen suunnittelun ja toteutuksen sekä ohjelmistokehityksen (Mainostoimisto Heinäkuu 2013, hakupäivä 20.10.2013). Suurin osa töistä on graafista suunnittelua, kuten mainoksia tai uutislehtiä ja internet-palveluiden kehittämistä. Yritys tarjoaa nykyisin myös tärkeitä työkaluja markkinointiin ja mainostamiseen sekä verkossa että sosiaalisessa mediassa. Yrityksellä on tällä hetkellä 7 työntekijää, joista 2 työskentelee Oulun konttorissa.

2.2 Ingo-ilmoittautumisjärjestelmä

Mainostoimisto Heinäkuu on tarjonnut Ingo-ilmoittautumisjärjestelmää asiakkaidensa käyttöön jo vuosien ajan. Se on pääasiassa urheiluseuroille suunnattu palvelu, jolla ne voivat tarjota helpon ilmoittautumistavan ja hoitaa ilmoittautumisiin liittyvät toimet.

Tällä hetkellä jokaiselle asiakkaalle luodaan oma ympäristönsä, jossa palvelu toimii. Eli käytännössä järjestelmä kopioidaan ja muokataan vastaamaan asiakkaan vaatimuksia. Näin ollen ylläpidettävänä on niin monta järjestelmää kuin on asiakastakin.

Tämän johdosta ohjelmavirheiden korjaaminen on työlästä, koska ne joudutaan korjaamaan käsin jokaiseen järjestelmään erikseen. Näin ollen eri järjestelmissä voi olla eri määrä korjattuja virheitä ja nämä saattavat aiheuttaa uusia ongelmia, koska ylläpitotyötä ei voida hallita organisoidusti. Myös joillekin asiakkaille tehdään laajemmin räätälöityjä järjestelmiä uusilla ominaisuuksilla, jolloin tätä pirstaloitumista tapahtuu vielä enemmän.

Nykyinen ilmoittautumisjärjestelmä tarjoaa asiakkailleen mahdollisuuden luoda tapahtumia ja hoitaa niihin liittyvien ilmoittautumisten hallitsemisen palvelun avulla.

Palveluun luotuun tapahtumaan voidaan syöttää tarvittavat perustiedot ja kuvaus julkista tapahtumasivua varten. Tapahtumalle voidaan lisätä erilaisia rajoituksia, joilla voidaan rajata tapahtuma tietylle yleisölle. Mahdollisuutta ulkoasun muokkaamiseen ei juuri ole, ja kaikki tapahtumasivut ovat samalla pohjalla.

Asiakas pystyy selaamaan tiettyyn tapahtumaan ilmoittautuneita käyttäjiä, näkemään heidän maksutilanteensa ja muuttamaan ilmoittautumisen tietoja tarpeen mukaan. Ilmoittautuneen kaikkia tietoja voidaan muokata jälkikäteen ja maksutietoja voidaan myös käsitellä, jos ilmoittautuja ei ole maksanut ilmoittautumismaksua verkkomaksupalvelun kautta. Myös erilaisten raporttien ja yhteenvetojen luominen palvelun kautta onnistuu, jolloin asiakas saa helposti tulostettavat listat ilmoittautuneista ja maksaneista osallistujista.

3 TYÖ

3.1 Tavoitteet ja rajaukset

Alkuperäisenä työn tavoitteena oli luoda järjestelmä, jolla pystytään palvelemaan useaa asiakasta yhdessä palvelussa. Näin voitaisiin helpottaa järjestelmän ylläpitoa ja hallintaa, kun tarvittavat korjaukset ja kehittämistyö voitaisiin tehdä keskitetysti yhdessä järjestelmässä.

Asiakkaiden tulisi voida helposti luoda palveluun uusia tapahtumia ja määritellä niiden sisältö. Tämä tarkoittaa ilmoittautumislomakkeiden muokkaamista tapahtuman mukaan, itse sivun tekstisisällön muokkaamista ja mahdollisten yhteistyökumppaneiden logojen ja mainoksien lisäämistä sivuille.

Palvelun tulisi olla käytettävissä mahdollisimman monella eri päätelaitteella ja näyttökoolla. Tämän vuoksi käyttöliittymä luodaan responsiivisen sovelluskehyksen päälle, jolloin se voidaan tarpeen mukaan optimoida toimimaan erilaisilla päätelaitteilla, kuten älypuhelimilla ja tablet-tietokoneilla.

Palvelun avulla käyttäjät voivat helposti etsiä erilaisia tapahtumia ja seurata tapahtumakalenterista oman alueensa tapahtumia. Rekisteröityneet käyttäjät voivat helposti ilmoittautua tapahtumiin ja heidän tietonsa lisätään ilmoittautumislomakkeeseen tietokannasta automaattisesti. Muut käyttäjät joutuvat lisäämään omat tietonsa aina ilmoittautumisen yhteydessä.

Lähinnä yrityksille on tarjolla rekisteröitysmismahdollisuus palveluun, joka mahdollistaa ryhmäilmoittautumiset. Näin yritysten ryhmät voisivat tehdä ilmoittautumisen helposti. Ryhmät voivat myös käyttää edellisen vuoden tietoja uudelleenilmoittautumiseen ja tehdä niihin muutoksia tarpeen mukaan.

Palvelu luotaisiin aluksi vain suomenkielisenä, mutta järjestelmä rakennettaisiin tukemaan uusien kielien lisäämistä. Kun järjestelmä rakennetaan alusta alkaen monikieliseksi, on tulevaisuudessa uusien kieliversioiden luominen yksinkertaista.

Työn edetessä tuli eteen tilanne, jossa päätettiin muuttaa työn tavoitteita. Uusien ominaisuuksien lisäämisestä ja laajemmasta muokattavuudesta luovuttiin ja päätettiin, että olemassa olevan järjestelmän ominaisuudet tuodaan sellaisenaan uuteen järjestelmään. Kaikkien tuotavien toimintojen toiminta usean käyttäjän järjestelmässä tulisi kuitenkin varmistaa. Palvelun hallintaosio luodaan kokonaan uudelleen käyttäen valittua käyttöliittymäsovelluskehystä responsiivisen käyttöliittymän tarjoamiseksi. Julkisen puolen käyttöliittymä päivitetään myös kyseisen sovelluskehyksen mukaiseksi, ilman suuria muutoksia ulkoasuun. Eli uusi järjestelmä luodaan tukemaan useita käyttäjiä ja samalla sen eri osat päivitetään uusimpiin versioihin. Näitä osia olivat muun muassa CodeIgniter, jQuery ja muut käytettävät kirjastot.

Ennen työtä asetettiin järjestelmälle myös joitain rajoituksia, joiden sisällä toimittaisiin työtä tehdessä. Järjestelmän rakentamisessa käytetään ohjelmointikielenä PHP:ta ja komentosarjakielenä JavaScriptiä. Ulkoasua varten käytetään kuvauskielenä HTML:ää ja tyyliohjeet kirjoitettaisiin CSS:nä.

Nykyinen järjestelmä on rakennettu vanhan CodeIgniter-sovelluskehysversion päälle ja tämä tuli päivittää uusimpaan versioon, joka työtä tehdessä oli versio 2.1.4. Tämän päälle lisätään myös BootStrap-käyttöliittymäsovelluskehyksen uusin versio, jolloin saadaan käyttöön sen tarjoamat työkalut ja ominaisuudet muun muassa responsiivisen käyttöliittymän luomiseksi.

Tässä työssä ei tutkita, mitä rajoituksia tai vaatimuksia palvelu asettaa palvelimelle tai tietoliikenneyhteyksille.

3.2 Työvälineet

Järjestelmän kehitystyö tapahtui enimmäkseen Linux Mint 15 -käyttöjärjestelmässä. Järjestelmää varten olen asentanut XAMPP-ohjelmistopaketin, joka sisältää Apache-palvelinohjelmiston, tuen MySQL-tietokannoille, sekä Perl ja PHP-kielille. Paketissa on myös phpMyAdmin-työkalu tietokannan hallintaan internetselaimella. Kyseisen paketin avulla pystyn helposti luomaan paikallisen ympäristön järjestelmän kehittämistä varten. Itse järjestelmän ohjelmistokoodi on kirjoitettu käyttäen Sublime Text 3 -editoria.

Pienet graafiset asiat hoidin käyttäen GIMP 2-ohjelmistoa. Tietokantasuunnittelua varten käytössäni oli MySQL Workbench 6.1 CE. Järjestelmän testaamisen toteutin käyttäen Google Chrome, Mozilla Firefox ja Microsoft Internet Explorer -selaimia.

3.3 Tutkimus

Tämä opinnäytetyö on soveltavaa tutkimusta konstruktiivisella tutkimusotteella, jossa tarkoituksena on luoda jotain uutta. Alussa ei vielä tiedetty, että kuinka se toteutetaan. Koska tein työn yksin käytin työmenetelmänä iteratiivista tapaa, joka noudatteli prototyyppimenetelmän spiraalimuotoa. Iteratiivisessa tavassa tiettyä kaavaa toistetaan ja toteutus suoritetaan pienissä osissa. Näin työn eri osat valmistuvat kerroksittain, kunnes niistä syntyy kokonaisuus. Jokaisen kierroksen jälkeen on valmiina prototyyppi, joka sisältää tietyn osan järjestelmästä. Näin voidaan edetä iteratiivisesti läpi kaikki palvelun toiminnot, kunnes on saavutettu lopullinen tavoitetila.

Itse työn tarkoituksena on vastata seuraaviin tutkimuskysymyksiin:

- Voidaanko palvelu muuttaa SaaS-tyyppiseksi ja mitä se vaatii?
- Mitkä palvelun toiminnot vaativat parantamista?
- Miten palvelun ylläpito ja kehitys tehdään helpommaksi?

Työtä varten hankin tietoa haastattelemalla Mainostoimisto Heinäkuun työntekijöitä, joista löytyy nykyisen järjestelmän tekijöitä, suunnittelijoita ja ylläpitäjiä. Taustatietoa hankin myös kirjallisuudesta, jota löytyi niin painettuna kuin sähköisenäkin.

4 KÄYTETYT TEKNOLOGIAT

Web-palveluita kehitettäessä, on käytettävissä lukuisia eri teknologioita, joiden avulla voidaan järjestelmä ja sisältö luoda. Seuraavassa käyn läpi tähän työhön liittyvät teknologiat.

4.1 Ohjelmointi- ja kuvauskielet

PHP, eli PHP: Hypertext Preprocessor on suosittu tulkattava ohjelmointikieli, jota käytetään yleisesti www-sivustojen ja -palveluiden luomiseen. Palvelinohjelmisto lukee ja suorittaa tulkattavien ohjelmointikielien lauseet yksi kerrallaan, eikä niitä käännetä etukäteen ohjelmakieleksi. PHP:n tapauksessa tulkkaus tapahtuu, kun palvelimelle annetaan hakupyyntö tietystä tiedostosta, ennen kuin se lähetetään pyytäjälle. Käyttäjän internetselaimelle lähetetään tulkattu tiedosto, joka sisältää vain sen tukemia merkintäkieliä. Käytetyn XAMPP-ohjelmistopaketin Apache-palvelinohjelmisto sisältää tulkin PHP-kielelle, jonka avulla järjestelmän koodi tulkataan näytettävään muotoon.

HTML, eli Hypertext Markup Language on merkkauskieli, jolla kuvataan niin sanottua hypertekstiä (W3Schools 2014b, hakupäivä 19.4.2014). Sen avulla voidaan luoda esimerkiksi www-sivuja. HTML-kielestä on käytettävissä eri versioita, joista viimeisimpänä on käytettävissä HTML5-versio, jota ei kuitenkaan ole vielä vahvistettu uudeksi standardiksi. Uusin HTML5-määrittely sisältää joukon uudistuksia, joiden avulla voidaan luoda monimutkaisia HTML5-sovelluksia.

JavaScript on web-ympäristössä käytettävä tulkattava olio-pohjainen kieli, jolla voidaan luoda toiminnallisuutta ja joka tunnetaan parhaiten internetsivujen komentosarjakielenä. Sen suuren suosion takana on laaja selaintuki, käyttöjärjestelmäriippumattomuus sekä ohjelmointityylien laaja kirjo. Sitä voidaan käyttää myös muissa kuin internetselainpohjaisissa ympäristöissä (Mozilla Developer Network 2014, hakupäivä).

CSS, eli Cascading Style Sheets, on www-sivujen tyyliohjeiden kuvaukseen käytetty kieli. Sen avulla www-sivulle voidaan luoda tyyliohjeita, joiden perusteella eri elementtien ulkoasu muodostuu. Se lisättiin HTML 4.0 standardiin ratkaisemaan ongelmaa

elementtien ulkoasun määrittelyyn ja vähentämään työmäärää. Tyyliohjeet tallennetaan erillisiin CSS-tiedostoihin (W3Schools CSS 2014a, hakupäivä 19.4.2014.)

Uusin CSS3-versio mahdollistaa muun muassa erilaiset animoinnit ja elementtien kääntämisen eri akselien suhteen verrattuna aikaisempaan CSS2-versioon. Näiden avulla voidaan luoda näyttäviä ja toiminnallisia internetsivuja.

Monimutkaisissa järjestelmissä on mahdollista käyttää myös LESS- ja SCSS-kuvauskieliä. Ne eroavat perinteisestä CSS-kielestä siinä, että ne pystyvät hyödyntämään muun muassa muuttujia ja sisäkkäisiä määrityksiä. Näistä luodaan ennen ajoa tai sen aikana selaimen tarvitsema CSS-kielinen tiedosto.

MySQL, eli My Structured Query Language, on tietokantajärjestelmä, joka käyttää hyväkseen relaatioita. Tätä järjestelmää käytetään hyvin laajasti erilaisissa web-palveluissa. Relaatiotietokanta perustuu tietokannan taulujen välisiin yhteyksiin eli relaatioihin. Järjestelmän käyttö perustuu SQL-kielisiin käskyihin, joilla voidaan hallita ja lukea tietokannassa olevaa tietoa. Relaatiotietokanta soveltuu hyvin isoihinkin järjestelmiin ja se pystyy käsittelemään tuhansia rivejä sisältäviä tauluja nopeasti.

4.2 Käytetyt sovelluskehykset ja koodikirjastot

Bootstrap on käyttöliittymän kehittämiseen tarkoitettu sovelluskehys, joka on alkujaan Twitterin työntekijöiden kehittämä. Sovelluskehys koostuu erinäisistä CSS- ja JavaScript-tiedostoista. Näiden avulla tarjotaan toimintoja ja tyyliohjeita, jotka helpottavat käyttöliittymän luomista ja kehittämistä antaen valmiita komponentteja, toimintoja ja ominaisuuksia käyttöön.

Eräs nykypäivän tärkeimmistä ominaisuuksista on mahdollisuus responsiivisen käyttöliittymän luomiseen. Sen mahdollistaa eräänlaisen taulukkopohjaisen rakenteen luominen, joka tarpeen mukaan skaalautuu pitäen ulkoasun halutun näköisenä. Tämän avulla voidaan luoda helposti halutunlainen käyttöliittymä, joka on erilainen tietokoneella, tablet-tietokoneella ja älypuhelimella, sisältäen kuitenkin saman sisällön.

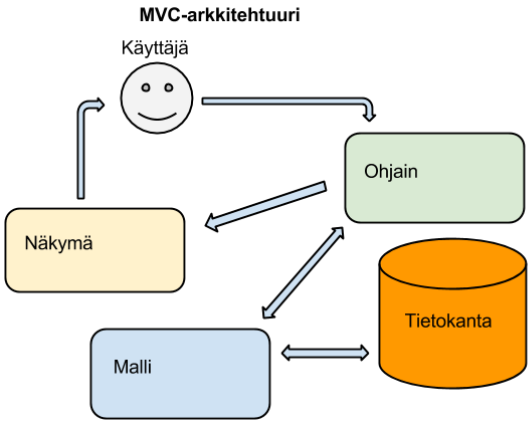
Toinen melko usein käytetty ominaisuus on valmiiksi määritellyt komponentit, joita ovat muun muassa valikkorakenteet, lomakkeet, taulukot ja painonapit.

Bootstrapin tueksi on saatavilla erilaisia lisäosia ja teemoja, joiden avulla voidaan tuoda uusia ominaisuuksia tai erilainen ulkoasu sovelluskehyksen tarjoamiin komponentteihin. Tässä järjestelmässä Bootstrapin tueksi otettiin käyttöön valmis ulkoasumalli, joka sisältää uudelleen suunniteltuja komponentteja ja toimintoja, joita voidaan käyttää samalla tavalla kuin Bootstrapin omiakin komponentteja. Tämä kyseinen Ace - Responsive Admin Template on suunniteltu käytettäväksi palvelun hallintapuolella, johon sitä aiotaankin käyttää. Se valittiin miellyttävän ulkoasun, tarvittavien ominaisuuksien, responsiivisuuden ja muokattavuuden vuoksi. Käyttäen tämän mallin komponentteja, yrityksen käyttöliittymän suunnittelija teki luonnoksen palvelun hallintapaneelin ulkoasusta, joka rakennetaan käyttäen kyseistä mallia.

PHP-sovelluskehyksenä on laajassa käytössä oleva CodeIgniter. PHP-sovelluskehyksien avulla voidaan rakentaa monipuolisia järjestelmiä helposti, koska ne tarjoavat tehokkaat työkalut ja toiminnot niiden luomiseksi. Tämä myös omalta osaltaan nopeuttaa kehittämistyötä.

Sovelluskehykset, kuten käytettäväkin sovelluskehys, sisältävät useita helpottavia kirjastoja, toimintoja ja hallintajärjestelmiä, joiden avulla voidaan helposti toteuttaa esimerkiksi tietokanta- ja kirjautumistoimintoja. Näiden sovelluskehykseen kuuluvien osien avulla vältetään koodin uudelleenkirjoittamista ja nopeutetaan kehittämistyötä.

CodeIgniter perustuu MVC-arkkitehtuuriin, joka tulee sanoista Model, View ja Controller. Tässä mallissa järjestelmä perustuu kolmeen eri osioon: Malliin (Model), jossa suoritetaan tietokantakyselyt, Näkymään (View), jossa määritellään sivun ulkoasu, ja Ohjaimeen (Controller), jossa tehdään itse tietojen käsittely ja näkymän muokkaus näiden perusteella. Arkkitehtuurin rakennekuviosta (Kuvio 1) voidaan nähdä kuinka ne kommunikoivat keskenään tässä järjestelmässä.



Kuvio 1. MVC-arkkitehtuurin rakenne

Kyseessä olevassa järjestelmässä käyttäjän toimet ohjataan ohjaimeen, jossa kutsutaan haluttua funktiota. Ohjain kutsuu tarpeen mukaan malli-luokan funktiota, joka hakee tietokannasta tietoa ja palauttaa sen ohjaimelle. Tämän jälkeen ohjain voi luoda olemassa olevien tietojen perusteella uuden näkymän, joka näytetään käyttäjälle.

MVC-mallin etuna voidaan pitää sitä, että eri osia voidaan kehittää toisistaan riippumatta. Näin kehittämistyötä voidaan jakaa erikseen ulkoasun kehittäjälle, joka hallitsee CSS:n ja HTML:n, ja tietokanta- ja ohjaintoiminnot taas niistä vastaavalle kehittäjälle, joka hallitsee PHP:n.

jQuery on JavaScript-kielellä toteutettu koodikirjasto. Se on suunniteltu tuomaan toimintoja ja ominaisuuksia esimerkiksi www-sivuille yksinkertaisessa muodossa ja onkin sen vuoksi maailman suosituin JavaScript-kirjasto (W3Techs.com 2014, hakupäivä 24.10.2013.)

jQueryn käyttämä syntaksi eroaa hieman perinteisestä JavaScript-kielen syntaksista. Se on räätälöity valitsemaan HTML-elementti ja suorittamaan jokin toiminto siinä (jQuery Syntax 2010, hakupäivä 1.5.2014). Taulukossa 1 on esimerkki siitä kuinka jQueryn avulla dokumentin latauksen valmistuttua yhdistetään p-elementteihin toiminto, joka piilottaa sen sitä klikattaessa.

```
$(document).ready(function(){
 $("p").click(function(){
   $(this).hide();
 });
});
```

Taulukko 1. Esimerkki jQueryn avulla toteutetusta elementin piilottamisesta

5 VALMISTELUT JA SUUNNITTELU

Aloitin työn tutustumalla toimeksiantajan kehityspalvelimella olemassa olevaan järjestelmään ja sen toimintoihin. Näin sain hyvän kuvan siitä, miten kyseinen järjestelmä on rakennettu ja minkälaisia funktioita siinä käytetään. Testasin kaikki perustoiminnot, kävin läpi niiden funktiot ja tutkin kuinka ne käyttävät tietokantaa hyväkseen.

Tutustuin myös käytettäviin sovelluskehyksiin, joista Bootstrap oli itselleni melko outo. Toimintaperiaate oli selvä, mutta jotkin sen toiminnoista olivat uusia. Sille löytyy laaja virallinen dokumentaatio ja useilta www-sivuilta löytyy runsaasti lisäohjeita sen käyttämiseksi, jotka olivat hyödyksi sen käyttöä opetellessa.

Samalla tutustuin myös CodeIgniterin uusimpaan versioon ja sen tuomiin muutoksiin, koska olemassa olevassa järjestelmässä käytetään vanhempaa sovelluskehystä. Päivittäminen uudempaan versioon oli kuitenkin dokumenttien mukaan melko yksinkertaista, koska siihen ei ollut tullut merkittäviä muutoksia. Tämän jälkeen oli sopiva tilanne kopioida kyseinen järjestelmä omaan kehitysympäristöön ja aloittaa järjestelmän päivittäminen ja muokkaaminen.

Suunnittelun aikana otin huomioon etukäteen määritellyt linjaukset liittyen palveluun, sen käytettävyyteen ja sisältöön. Koska palvelussa säilytettiin olemassa olevat toiminnot ja haluttiin varmistaa niiden toimivuus usean käyttäjän palvelussa, piti kyseinen asia pitää mielessä jo suunnitteluvaiheessa.

Omien taitojen kehittämisen otin huomioon jo suunnitteluvaiheessa siten, että järjestelmän rakentamisessa käyttäisin mahdollisimman vähän valmiita ratkaisuja niissä tilanteissa, joissa voisin myös itse tehdä tarvittavat toiminnot. Samoin ongelmatilanteet oli tarkoitus ratkaista mahdollisimman pitkälle omin voimin, jotta saisin hyvää kokemusta myös niistä.

5.1 Käyttöliittymä ja visuaalinen ulkoasu

Koska palvelussa luodaan hallintapaneeli kokonaan uusiksi, tuli siihen toimeksiantajan puolesta luonnos sen rakenteesta ja ulkoasusta. Luonnos sisälsi mallin hallintapaneelin sivusta, jonka tyyliä seurattaisiin muissakin osissa.

Käyttöliittymän julkinen puoli rakennetaan uudelleen käyttäen Bootstrapin luokkavalitsimia säilyttäen kuitenkin nykyinen ulkoasu. Käyttöliittymä rakennetaan kuitenkin siten, että ulkoasun muuttaminen jälkeenpäin olisi mahdollisimman helppoa.

Käyttöliittymän rakentamisessa pyritään siihen, että se skaalautuisi hyvin myös mobiililaitteille eli sen rakentamisessa tullaan hyödyntämään Bootstrapin tarjoamia ominaisuuksia responsiivisen käyttöliittymän luomiseksi.

Käyttäjien käytössä oleva hallintapaneeli rakennetaan aiemmin mainittua Bootstrapin Ace - Responsive Admin Template ulkoasumallia hyväksi käyttäen. Tämä sen vuoksi, koska se on hyvin suunniteltu, tarjoaa nykyaikaisen ulkoasun ja tarvittavat toiminnot hallintapaneeliin. Tällöin pystyin keskittymään järjestelmän toimintojen kehitystyöhön, jotta ne olisivat mahdollisimman toimivia ja virheettömiä.

5.2 Tietokantarakenne

Järjestelmän käyttämät tiedot tallennetaan MySQL-tietokantaan. Tietokanta sisältää useita tauluja, joihin tallennetaan palvelun eri osissa käytettäviä tietoja. Tietokantaan tallennettuja tapahtumakohtaisia tietoja ovat muun muassa tapahtumat tiedot ja rajoitukset, tarjolla olevat matkat, hintatiedot, ilmoittautumislomakkeen kysymykset ja lisävalinnat sekä niiden vastaukset, ilmoittautumiset, ryhmäilmoittautuneiden tarkemmat tiedot, laskut ja niiden rivit. Myös palveluun rekisteröityjen käyttäjien tiedot on tallennettu omaan tauluunsa tietokannassa.

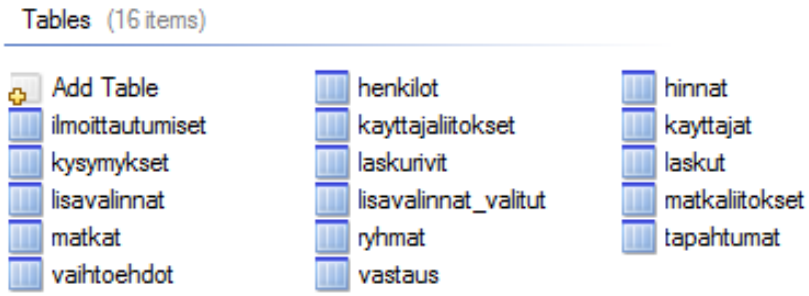
Aloitin tietokantamuutosten suunnittelun ja toteuttamisen ottamalla tietokannasta vedoksen, jonka toin MySQL Workbench ohjelmaan. Kyseinen ohjelma on suunniteltu tietokantarakenteiden suunnitteluun ja sen avulla pystyin hahmottamaan tietokannan

rakennetta paremmin. Kirjasin ylös huomaamiani muutostarpeita ja niihin muutosehdotuksia.

Aluksi muokkasin vain käyttäjiin liittyviä tietokantatauluja, koska alussa oli hyvä saada palvelun perusajatus useiden käyttäjien palvelusta toimimaan. Palvelun tulee voida erotella asiakkaat, käyttäjät ja ryhmät toisistaan.

Käyttäjätauluihin tein vain pienimuotoisia korjauksia, koska niiden alkuperäinenkin rakenne mahdollisti eri käyttäjäluokkien käytön. Käytännössä erottelin eri käyttäjäluokat tarkemmin toisistaan, lisäsin muutaman tarpeellisen sarakkeen ja poistin käyttämättömiä.

Seuraavassa kuvassa (Kuva 1) näkyy kaikki järjestelmässä olevat taulut. Järjestelmän käyttämä tieto on hajautettu useaan eri tauluun, jotta voidaan pitää niiden rakenteet selkeinä ja tehdä selkeitä yhteyksiä taulujen kesken.



Kuva 1. Tietokannan sisältämät taulut

Seuraavassa kuvassa (Kuva 2) nähdään käyttäjät-taulun rakenne ja eri kenttien tyypit. Kyseiseen tauluun sijoitetaan kaikkien rekisteröityneiden käyttäjien tiedot.

Column Name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default
kayttaja_id	INT(11)	☑	☑	☐	☐	☐	☐	☑	
kayttaja_tyyppi	INT(1)	☐	☐	☐	☐	☐	☐	☐	NULL
kayttaja_tunnus	VARCHAR(64)	☐	☑	☑	☐	☐	☐	☐	
kayttaja_salasana	VARCHAR(40)	☐	☑	☐	☐	☐	☐	☐	
kayttaja_yritys_nimi	VARCHAR(100)	☐	☐	☐	☐	☐	☐	☐	NULL
kayttaja_ytunnus	VARCHAR(12)	☐	☐	☐	☐	☐	☐	☐	NULL
kayttaja_enimi	VARCHAR(35)	☐	☐	☐	☐	☐	☐	☐	NULL
kayttaja_snimi	VARCHAR(35)	☐	☐	☐	☐	☐	☐	☐	NULL
kayttaja_email	VARCHAR(100)	☐	☐	☐	☐	☐	☐	☐	NULL
kayttaja_gsm	VARCHAR(15)	☐	☐	☐	☐	☐	☐	☐	NULL
kayttaja_osoite	VARCHAR(250)	☐	☐	☐	☐	☐	☐	☐	NULL
kayttaja_pnumero	VARCHAR(5)	☐	☐	☐	☐	☐	☐	☐	NULL
kayttaja_paikkakunta	VARCHAR(250)	☐	☐	☐	☐	☐	☐	☐	NULL
kayttaja_luotu	TIMESTAMP	☐	☐	☐	☐	☐	☐	☐	'0000-00-00 00:00:00'
kayttaja_muokattu	TIMESTAMP	☐	☑	☐	☐	☐	☐	☐	CURRENT_TIMESTAMP
kayttaja_kirjautuminen	TIMESTAMP	☐	☐	☐	☐	☐	☐	☐	'0000-00-00 00:00:00'
kayttaja_markkinointilupa	BOOLEAN	☐	☐	☐	☐	☐	☐	☐	'1'
kayttaja_merkattu_testiksi	BOOLEAN	☐	☑	☐	☐	☐	☐	☐	'0'
kayttaja_status	INT(1)	☐	☐	☐	☐	☐	☐	☐	'1'

Kuva 2. Käyttäjät-taulu, johon tallennetaan rekisteröityjen käyttäjien tiedot

5.3 Luokkasuunnittelu

Aiempi järjestelmä on rakennettu käyttäen MVC-arkkitehtuuria, joten siinä oli valmiina suurin osa luokista, joita tarvitaan myös tulevassa järjestelmässä. Tästä huolimatta halusin tehdä myös luokkasuunnittelua ja kävin läpi kaikki olemassa olevat luokat ja niiden yhteydet toisiinsa. Huomasin joitain funktioita, joiden käyttölogiikkaa tulee muuttaa ja toimintoja, jotka tarvitsevat täydellistä uudelleen luomista, jotta ne toimisivat johdonmukaisesti uudessa järjestelmässä.

5.4 Tietoturva

Koska palvelu tulee olemaan julkisessa käytössä ja tietoturvakysymykset ovat aina ajankohtaisia, tutkin myös tietoturvan asettamia vaatimuksia ja muutostarpeita järjestelmään. Yksi tärkeä osa tietoturvaa on salasanojen tallennus kryptograafisena tarkistussummana, jota ei voida suoraan kääntää selkokieliseksi. Tähän voidaan lisätä myös niin sanottu salasanan suolaus, joka tarkoittaa sattumanvaraisen merkkijonon lisäämistä salasanamerkkijonoon ennen kuin siitä lasketaan tarkistesumma. Suolauksen avulla päästään jo melko turvalliseen tapaan tallentaa salasanat. Huomion arvoinen asia on tarkistussumman laskemiseen käytetty tekniikka, joista on viime vuosina löytynyt tietoturva-aukkoja. Esimerkiksi yleiset MD5- ja SHA-1-algoritmit ovat nykypäivänä liian turvattomia, joten paremman kryptograafisen teknologian valitseminen on ensisijaisen tärkeää. Otin tutkimuksen alle SHA-2- ja bcrypt-tekniikat, koska ne ovat suosittuja ja laajasti käytettyjä tekniikoita.

SHA-2 sisältää neljä eri funktiota, joita voidaan käyttää tarkistussumman laskemiseksi halutusta tiedosta. SHA-2:sta on korjattu monia ongelmia, jotka vaivasivat aiempaa SHA-1-versiota. Mutta koska SHA-2 perustuu samaan algoritmiin kuin SHA-1, epäillään, että sekin tullaan murtamaan lähitulevaisuudessa (Jašek, Roman & Sarga, Libor & Benda, Radek 2013, 21).

Bcrypt on Blowfish-algoritmiin pohjautuva skaalautuva salausalgoritmi, jonka turvallisuus perustuu siihen, että tarkistussumma muodostetaan monimutkaisella algoritmilla (Shelley 2002, hakupäivä 23.10.2013). Tarkistussumman murtamiseen menee sitä

enemmän aikaa mitä monimutkaisempaa menetelmää sen tekemiseen on käytetty (Viestintävirasto 2011, hakupäivä 23.10.2013).

Tekniikan kehittyminen ja laskentatehojen nouseminen tietokoneissa, aiheuttavat haasteita tietoturvan ylläpitämiseksi järjestelmissä. Monimutkaisilla tekniikoilla tarkistussumman laskemiseksi voidaan suojata tehokkaasti käyttäjien salasanat, mutta ne voivat aiheuttaa myös suorituskykyongelmia. Salaustekniikan valinta onkin aina kompromissi halutun salauksen tehokkuuden ja nopeuden väliltä (Verens 2010, 37).

Toinen huomioitava asia on erilaisten SQL-injektioiden estäminen. Ne käyttävät hyväkseen tietoturva-aukkoja tietokantaa hyödyntävissä järjestelmissä. Niissä hyökkääjä pystyy ujuttamaan SQL-kyselyyn SQL-koodia, jonka avulla hän pyrkii ohittamaan esimerkiksi käyttäjän tunnistamistoimintoja. Tällöin vahvakaan salasanan suojaus ei auta, jos järjestelmä antaa käyttöoikeuden tällaisen haavoittuvuuden kautta.

SQL-injektioita vastaan voidaan suojautua käyttämällä erilaisia suodattimia tai kirjastoja, jotka estävät ongelmia aiheuttavien merkkien hyväksikäytön SQL-kyselyissä. Esimerkiksi käytettävä sovelluskehys, CodeIgniter, sisältää useita tietoturvaan liittyviä toimintoja ja kirjastoja. Näillä voidaan estää muun muassa XSS- ja CSRF-hyökkäykset, SQL-injektiot ja muut erilaiset web-palveluille ongelmia aiheuttavat tietoturvaongelmat.

6 PALVELUN RAKENTAMINEN

Tässä luvussa käyn läpi työn toteuttamisen eri osa-alueet ja niihin liittyvät asioita, jotka ovat huomioimisen arvoisia kehittäjän näkökulmasta.

6.1 Sovelluskehykset ja kirjastot

Aluksi päivitin vanhan järjestelmän käyttämät sovelluskehykset uusimpiin versioihin, jolloin pääsin eroon vanhoja versioita vaivanneista ohjelmistovirheistä ja tietoturvaongelmista. Samalla poistin myös joitain komponentteja, joista haluttiin päästä eroon, koska ne aiheuttivat yhteensopivuusongelmia tai sisälsivät paljon päällekkäisiä ominaisuuksia muiden komponenttien kanssa. Yksi poistettu komponentti oli jQuery UI-koodikirjasto, joka sisältää osittain samoja ominaisuuksia kuin Bootstrap.

CodeIgniter sovelluskehyksen päivittäminen versioon 2.1.4 toi mukanaan joukon muutoksia jotka jouduin tekemään järjestelmään, jotta se toimi edelleen virheettömästi. Näitä muutoksia olivat muun muassa luokkien konstruktoreiden eli muodostimien määrittelyn muuttuminen, tiettyjen funktioiden toimintalogiikoiden muutokset, funktioiden nimien muutokset ja tiettyjen funktioiden korvaaminen kokonaan uusilla. Muodostimet päivittyivät vastaamaan PHP5:n tapaa luokan muodostamisesta. Kyseinen sovelluskehys vaatiikin vähintään PHP 5.1.6 version. Muut isot muutokset liittyivät tietoturvaan ja ydintoimintojen parantamiseen. Esimerkiksi tietoturvaan liittyvät toiminnot löytyvät nyt omasta Security-kirjastosta, josta löytyy muun muassa toiminnot XSS-suodatukseen ja CSRF-suojaukseen. Myös erinäisiä tietoturva-aukkoja on korjattu, jotka mahdollistivat hyökkäyksiä käyttäen SQL-injektioita tai HTML5-haavoittuvuuksia.

Päivitin myös käytössä olevan jQuery-koodikirjaston uusimpaan versioon, joka toi mukanaan paljon muutoksia. Vanha versio oli vuodelta 2010 ja uusimpaan versioon verrattuna muutoksia oli melko runsaasti. Muutoksia oli tullut muun muassa funktioiden toimintoihin ja monia funktioita oli poistettu tai muutettu. Tämän vuoksi järjestelmässä olevien JavaScript-kirjastojen tarkka läpikäynti ja jQuery-versioiden julkaisutietojen tarkka lukeminen oli tärkeä osa päivittämisprosessia, jotta kaikki muutokset tulisivat tehtyä. Näin kehittämistyön edetessä ei tullut eteen isompia ongelmia jQueryn suhteen ja uusien funktioiden kirjoittaminen oli nopeaa.

6.2 Kehittämistyö

Kehittämistyössä seurasin tehtyä listaa järjestelmän ominaisuuksista ja rakensin niistä toimivia kokonaisuuksia. Koska noudattelin kehittämistyössä prototyyppimallin spiraalimuotoa, määrittelin jokaisen kierroksen alussa osion tai toiminnon sisällön. Kun kyseessä oli monimutkaisempi toiminto, suunnittelin sille rakenteen, jota koodi noudattelisi ja lopuksi toteutin sen suunnitelman mukaisesti. Toteutuksen jälkeen testasin toiminnon tarkasti käyttäen eri argumenttiyhdistelmiä, jotta pystyin näkemään kuinka se toimisi eri tilanteissa. Kehittämistyön edetessä jouduin palaamaan joihinkin toimintoihin uudelleen, koska järjestelmän muiden osioiden muutokset aiheuttivat yhteensopivuusongelmia. Tämä aiheutti jonkin verran samojen funktioiden muokkaamista useaan otteeseen, mutta näin sain mielestäni hallittua paremmin kehitystyön etenemistä.

Kun aluksi olin päivittänyt järjestelmän käyttämän CodeIgniter-sovelluskehyksen uusimpaan versioon, oli järjestelmä tilassa, jossa palvelua pystyi jo käyttämään. Tämän jälkeen kävin läpi järjestelmän luokat ja niiden sisältämät funktiot ja tein tarvittavia muistiinpanoja ja muutosehdotuksia. Näitä tuli esimerkiksi tiettyjen toimintojen kulusta ja koodin ulkoasusta.

Läpikäynnin jälkeen pystyin aloittamaan järjestelmän näkymien uudelleenkirjoittamisen. Koska järjestelmä rakentuu nyt Bootstrap-käyttöliittymäsovelluskehyksen päälle, käytännössä kaikki näkymät tuli kirjoittaa uusiksi, jotta pystyisin hyödyntämään sen tarjoamia ominaisuuksia. Aluksi Bootstrapin käyttämien luokkavalitsimien käyttö tuntui hieman vieraalta, koska olin tutustunut responsiiviseen toteuttamiseen vain pintapuolisesti ja käyttäen muita sovelluskehyksiä. Kun opin tuon kuvaustavan, oli sen kirjoittaminen melko nopeaa, eikä se tuntunut enää jälkeenpäin oudolta.

Käytännössä Bootstrapin luokkavalitsimia käytetään siten, että aluksi määritellään rivi div-elementtinä, jonka jälkeen se jaetaan sarakkeisiin halutulla tavalla div-elementtien avulla. Bootstrap käyttää 12 sarakkeen rakennetta, josta voidaan siis tarpeen mukaan lohkaista osa halutulle elementille. Pystyin siis helposti määrittelemään rivin, jossa oli kaksi saraketta, ensimmäinen 8 osaa leveä ja toinen 4 osaa leveä. Bootstrap mahdollistaa sarakkeiden määrittelyn erikseen neljälle eri näytön koolle. Näin voidaan luoda mobiililaitteilla toimiva käyttöliittymä samalla, kun kirjoitetaan käyttöliittymää esimerkiksi työpöytäkäyttöön sopivaksi.

Kun olin kirjoittanut uusiksi tärkeimpien toimintojen näkymät käyttäen Bootstrapin luokkavalitsimia, siirryin itse koodin pariin. Koska palvelu muutetaan usean käyttäjän palveluksi, luonnollinen aloituskohta oli eri toiminnot liittyen kirjautumiseen ja käyttäjätilien hallintaan. Samalla oli hyvä tilaisuus ottaa tarkasteluun myös tietoturvaan liittyvät asiat. Tein muutamia käytännön testejä liittyen salasanojen tarkistesummien luomiseksi, mutta en kuitenkaan lähtenyt tässä työssä muuttamaan käyttäjätilien salasanoihin liittyvää logiikkaa, vaan säilytin sen ennallaan.

Tietokanta itsessään sisälsi jo alun perin tarvittavat kentät käyttäjien erottamiseksi toisistaan, jolloin riitti, että määrittelyn mukaisesti järjestelmä ohjaa käyttäjät oikeisiin näkymiin käytetyn käyttäjätyypin mukaisesti. Järjestelmässä on käytössä kirjasto, jonka avulla voidaan tehdä käyttäjien oikeuksien tarkistukset helposti. Käytännössä tämä tapahtuu aina mahdollisimman aikaisessa vaiheessa, eli jo luokan muodostimessa. Muodostin suoritetaan luokassa aina ensimmäisenä, ennen funktioita. Jos käyttäjä jostain syystä pyrkii käyttämään jonkin luokan toimintoa, johon hänellä ei ole oikeutta, niin hänet kirjataan ulos palvelusta ja ohjataan etusivulle. Näin pyritään estämään se, että käyttäjä ei pääse koskaan näkemään tietoja, joihin hänellä ei ole oikeutta.

Käyttäjien hallintaa varten loin myös ylläpitäjälle perustoiminnot, joilla voidaan hallita käyttäjätilejä. Ylläpitäjä voi listata kaikki palvelun käyttäjät, muokata heidän tietojaan tai vaihtaa salasanoja. Myös uusien käyttäjien lisääminen palveluun onnistuu tätä kautta.

Kun oli mahdollista lisätä uusia käyttäjiä, aloitin rakentamaan uutta hallintapaneelia tapahtumien hallitsemiseksi. Näkymän perusrunko oli jo luotu, jonka jälkeen loin siihen lisäsisältöä tarpeen mukaan. Tässä kohtaa rakensin toiminnot sen mukaisesti kuin niitä käytetäänkin eli ensimmäisenä luodaan tapahtuma, jonka jälkeen voidaan muokata sen tietoja. Uuden tapahtuman luominen on käytännössä vain nimen perusteella uuden rivin lisääminen tietokantaan. Muokkaustilassa lisätään kaikki muut tapahtumaan liittyvät tiedot. Tapahtuman muokkausnäkymä sisältää suuren määrän eri toimintoja, joiden avulla voidaan lisätä ja muokata:

- Perustietoja
- Tapahtuman matkoja
- Tapahtuman rajoituksia ja tarkentavia tietoja

- Tapahtuman kuvausta
- Hintoja
- Ilmoituslomakkeen kysymyksiä
- Ilmoituslomakkeen lisävalintoja ja alennuksia

Myös ilmoittautuneiden selaaminen ja heidän tietojensa muokkaus onnistuu muokkausnäkymän kautta. Koska tämä tapahtuman muokkaukseen tarkoitettu osio pitää sisällään näin suuren määrän eri toimintoja, jaoin näkymän välilehtiin. Loin jokaiselle välilehdelle oman näkymänsä, jolloin niiden muokkaaminen on helppoa. Näin eri toiminnot ovat helposti käytettävissä, mutta ruudulla ei tarvitse näyttää suurta määrää toimintoja kerrallaan. Välilehtitoiminto on toteutettu käyttäen Bootstrap teeman välilehti-toimintoa.

Koska halusin tehdä tietojen muokkaamisen mahdollisimman helpoksi, ilman turhia sivunlatauksia, käytin niiden hallinnassa paljon AJAX-kutsuja. Niiden avulla haetaan tai viedään tietoa sen mukaan onko tarkoitus tallentaa vai hakea sisältöä käytettäväksi.

JavaScript-funktioille joilla tietoa viedään tallennettavaksi, tuodaan tietoa lomakkeilta, kalenteri-ikkunoista tai Jeditable-kentistä, jotka mahdollistavat tekstin muokkaamisen suoraan tekstirivillä. Kun nämä toiminnot ja AJAX-kutsut yhdistetään jQueryn ja Bootstrapin tarjoamiin efekteihin, voidaan helposti luoda miellyttävän näköisiä toimintoja, joissa poistettava tieto häivytetään näkyvistä ja uusi kenttä liukuu näkyville. Näin käyttäjän on miellyttävämpi seurata toimintoja, kun tieto ei häviä tai ilmaannu yllättäen.

Kysymyksien ja lisävalintojen muokkauksessa käytän niin sanottua haitariominaisuutta, jolla voidaan näyttää aluksi vain otsikkotieto ja jota klikkaamalla aukeaa näkymä, jossa voidaan muokata kaikkia tietoja. Tietoja poistettaessa käytin Bootbox nimistä JavaScript-kirjastoa, joka käyttää Bootstrapin modaali-elementtejä luodakseen dialogi-ikkunoita esimerkiksi poiston varmistamiseksi. Tämä on yksi tapa jolla voidaan vähentää turhia sivulatauksia.

Edellä mainittuja AJAX-kutsuja käytetään monessa eri paikassa järjestelmän sisällä. Niissä tiedon välittäminen kontrolleriin tapahtuu käyttäen POST-metodia tiedon ollessa koodattuna JSON-muotoon. Ohjaimessa tietoa käsitellään tarpeen mukaan, se voidaan ajaa validointifunktion läpi tai tietoja voidaan käsitellä, ennen niiden viemistä malliin, joka tekee tarpeelliset muutokset tietokantaan. Tämän jälkeen ohjain voi vielä luoda

mahdollisen uuden osan näkymään, joka JavaScript-funktioon palautumisen jälkeen lisätään näkymään dynaamisesti.

Tapahtuman muokkausnäkymän rakentaminen oli suurin yksittäinen tehtävä joka liittyi hallintapaneeliin. Koska tiettyjä järjestelmän komponentteja oli päivitetty reilusti uudempiin versioihin ja kyseisen osion toiminnot tukeutuvat vahvasti JavaScript-funktioihin, päätin tehdä ne alusta alkaen uudelleen. Näin pystyin karsimaan vanhentuneita toimintoja ja luomaan vain tarvittavat, jolloin kyseisien tiedostojen määrä väheni ja hallinta helpottui.

Koska järjestelmä on alun perin rakennettu melko kauan sitten ja sitä on laajennettu useaan otteeseen eri kehittäjien toimesta, oli sen ohjelmakoodi osittain hankalasti seurattavaa. Esimerkiksi ilmoittautumisprosessi, joka on käytössä palvelun julkisella puolella, tarvitsee hyvin laajan funktion, koska se sisältää useita hakuja liittyen tapahtuman tietoihin. Samoin samassa funktiossa hoidetaan samalla muun muassa ilmoituslomakkeen tietojen validointi, syötettyjen tietojen muokkaus sen mukaan mitä toimintoja tapahtumalle on asetettu, erilaisten tarkistusten läpikäynti, mahdollisten laskutietojen luominen, tietokantafunktioiden kutsuminen ja vahvistussähköpostin lähettäminen. Tämä laaja funktio oli hyvä saada purettua muutamaan erilliseen funktioon, jotta sen kulkua olisi helpompi seurata. Loin omat funktiot ilmoittautumislomakkeen kysymysten, tapahtuman matkojen hakemiseksi sekä lomaketietojen validoimiseksi. Tämä lyhensi aikaisempaa funktiota n. 300 riviä ja selkeytti sen rakennetta.

Näin olin päässyt tilanteeseen, jossa palvelun kautta voidaan ilmoittautua tapahtumiin. Koska ylläpitäjillä ja asiakkailla on mahdollisuus muokata ilmoittautuneen tietoja, tuli niiden toiminta nyt varmistaa myös tehtyjen muutoksien jälkeen. Samalla loin tarpeelliset varmistukset, että käyttäjällä on oikeus kyseiseen tapahtumaan. Tämä tehdään vertaamalla tietokannassa olevaa käyttäjän yksilöivää tunnistetta session-tietojen perusteella haettuihin tietoihin kirjautuneesta käyttäjästä. Jos havaitaan, että käyttäjä pyrkii lataamaan toisen käyttäjän omistaman tapahtuman tietoja, kyseisen käyttäjän session tiedot tuhotaan ja hänet ohjataan palvelun etusivulle, josta hän joutuu kirjautumaan uudelleen hallintapaneeliin. Näin pystytään estämään tahaton ja tahallinen yritys päästä toisen käyttäjän omistaman tapahtuman tietoihin käsiksi.

Koska eri käyttäjätyypeillä on samankaltaisia toimintoja, mutta eri oikeuksia, ovat järjestelmän kontrollerit erillään toisistaan. Käytännössä ne on siis sijoitettu eri kansioihin. Osittain tämä aiheuttaa koodin ja funktioiden toistumista eri käyttäjätyyppien luokissa, mutta niiden hallitseminen on kuitenkin selkeämpää.

Nyt, kun palvelun kautta voitiin hallita tapahtumia ja niihin ilmoittautuneiden tietoja, oli vuorossa aloittaa järjestelmän systemaattinen testaaminen. Erillistä testaussuunnitelmaa en luonut tätä varten, mutta yritin noudattaa järjestelmällistä testauslogiikkaa.

Käytännössä testaaminen tapahtui siten, että kävin järjestelmän eri toiminnot yksitellen läpi ja yritin löytää niistä toimintavirheitä. Tämä tapahtui syöttämällä vajaita tietoja lomakkeisiin, yrittämällä toimintoja jotka eivät ole sallittuja ja eri toimintojen yhdistelmiä. Näin pystyi näkemään toimiko lomakkeiden tietojen tarkistaminen oikein, palautuivatko lomakkeiden tiedot oikein lomakkeeseen palattaessa tai muuttuiko haluttu tieto halutulla tavalla. Muutoksia seurasin jatkuvasti myös tietokannan tauluista, joista on helppo tarkistaa muutoksien toimivuus.

7 POHDINTA

Työn lähtökohdat olivat melko vaativan oloiset, koska palvelu oli laaja ja moniulotteinen, sekä haluttu lopputulos, eli SaaS-tyyppinen ratkaisu oli myös sisällöltään minulle hieman outo. Kuitenkin, koska useat palvelut toimivat nykyään täysin internetin varassa ja sen käyttö ja vaikutus tulevat lisääntymään, työ vaikutti alusta alkaen erittäin mielenkiintoiselta.

Kyseinen järjestelmän muutos toi kuitenkin eteen monta eri asiaa, jotka jouduin ottamaan huomioon työn edetessä. Ensimmäisenä oli se, että mitä järjestelmältä ja tietokannalta vaaditaan, kun sillä on suuri määrä käyttäjiä ja käyttäjiin liittyviä tietoja. Tuli ottaa huomioon käyttäjien oikeuksien määrittely ja rajauksien toteutus, erilaisten tarkistusten tekeminen monessa eri paikassa ja ohjaimien suuri määrä eri käyttäjätyypeille. Toisena tärkeänä asiana oli tietoturva, joka on aina aiheellinen asia internet-pohjaisia palveluita toteutettaessa. Vaikka järjestelmän ainoat selkeimmät muutokset tietoturvaan olivat käytettyjen sovelluskehysten päivittämiset, asian selvittäminen esimerkiksi salasanojen tallentamisen suhteen toi paljon uutta tietoa sen nykytilasta ja mahdollisista tulevista käyttötavoista.

Työ vaati jonkin verran myös taustatiedon hankkimista. Erilaisten dokumenttien ja ohjeiden lukeminen vei ison osan alkuvaiheen työnteosta, jotta sen pohjalta olisi hyvä jatkaa järjestelmän työstämiseen. Jälkeenpäin ajatellen dokumentaation lukeminen oli hyvä asia ja sen hyödyt tulivat tässä tilanteessa selkeästi esille.

Alussa pelkkä nykyisen järjestelmän läpikäyminen ja testaus antoi hyvän pohjan työn aloittamiselle, koska näin pystyin muodostamaan hyvän kuvan sen rakenteesta ja toimintaperiaatteista. Tästä oli paljon hyötyä työn myöhemmissä vaiheissa, kun tiesin logiikan millä eri toimintojen tulisi toimia. Vaikka kaikkia järjestelmän funktioita ja luokkia ei tarvinnut kirjoittaa uudelleen, sisälsi työ paljon PHP:n, HTML:n, CSS:n ja JavaScriptin kirjoittamista. Näistä PHP:n ja JavaScriptin kohdalla sain hyvää lisäkokemusta koodin tuottamisesta ja samalla oppi myös uusia toiminnallisuuksia.

Raportointi sujui työn ohella melko hyvin. Työn edetessä tuli toki tilanteita vastaan jolloin raporttiin ei ollut tullut lisää sisältöä ja näin raportin sisällön jatkaminen myöhemmässä tilanteessa vaati aikaisemman kertaamista. Hankalaksi raportin kirjoittamisessa

koin joidenkin alaan liittyvien termien käytön, joille ei ole vakiintunutta suomenkielistä vastinetta.

Työtä voitaisiin vielä jatkaa kehittämällä järjestelmän tietoturvaa paremmaksi, luomalla uusi käyttöliittymä myös julkiselle puolelle palvelua tai lisäämällä uusia ominaisuuksia asiakkaiden käytettäväksi. Tämän kaltaisiin palveluihin, joissa on runsaasti käyttäjiä, kehitetään yleensä jatkuvasti parannuksia. Nämä voivat olla joko järjestelmän tai käyttäjäkokemuksen parantamista.

Kaiken kaikkiaan työ ja siihen käytetty aika oli mielenkiintoista, opettavaa ja haastavaa ja jälkeenpäin voinkin todeta oppineeni paljon uutta ja kehittäneeni jo olemassa olevia taitojani. Tämä tulee varmasti olemaan hyödyksi tulevaisuudessa tällä alalla työskennellessä.

LÄHTEET

Jašek, Roman & Sarga, Libor & Benda, Radek 2013. Security Review of the SHA-1 and MD5 Cryptographic Hash Algorithms. WSEAS Press.

jQuery Syntax. Hakupäivä 1.5.2014
<http://www.w3schools.com/jquery/jquery_syntax.asp>

Mainostoimisto Heinäkuu. Hakupäivä 23.10.2013
<http://www.heinakuu.com>

Mozilla Developer Network, 2014. JavaScript. Hakupäivä 15.3.2014
<https://developer.mozilla.org/en-US/docs/Web/JavaScript>

Shelley, Johnny 2002. Bcrypt - Blowfish File Encryption. Hakupäivä 23.10.2013
<http://bcrypt.sourceforge.net>

Verens, Kae 2010. CMS Design Using PHP and JQuery : Build and Improve Your In-house CMS by Enhancing PHP with JQuery. Olton Birmingham, GBR, Packt Publishing Ltd

Viestintävirasto 2011. Ohje, 1/2011 Verkkopalvelun ohjelmistoalustan valinta ja palvelun turvallinen ylläpito. Hakupäivä 23.10.2013
<http://www.cert.fi/ohjeet/2011_23/ohje-1-2011.html>

W3Schools CSS 2014a. CSS Introduction. Hakupäivä 19.4.2014.
<http://www.w3schools.com/css/css_intro.asp>

W3Schools HTML 2014b. HTML Introduction. Hakupäivä 19.4.2014
<http://www.w3schools.com/html/html_intro.asp>

W3Techs JavaScript Libraries. Usage of JavaScript libraries for websites. Hakupäivä 24.10.2013.
<http://w3techs.com/technologies/overview/javascript_library/all>