Jukka Lähetkangas

Pienyrityksen tapahtumanhallinta

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Mediatekniikan koulutusohjelma

Insinöörityö

28.11.2014

Metropolia

Tekijä Otsikko Sivumäärä Aika	Jukka Lähetkangas Pienyrityksen tapahtumanhallinta 28 sivua + 4 liitettä 28.11.2014
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikan koulutusohjelma
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaaja	Lehtori Aarne Klemetti

Insinöörityössä toteutettiin Metropolia Ammattikorkeakoulun opiskelijakunta METKAn Metka opiskelijapalveluille laaja tapahtumanhallintasovellus Kaljaasi-risteilyn järjestämistä varten. Toteutettu sovellus mahdollistaa tietojenkäsittelyn Kaljaasi-risteilyn järjestäjien työmäärän vähentämiseksi ja Kaljaasi-risteilylle ilmoittautumisen käyttäjäystävällisellä tavalla.

Insinöörityössä tutkittiin käyttäjätapauksia, vaihtoehtoisia tekniikoita, tietokantamalleja ja jälkituotantoa. Työssä painotettiin tuotesuunnittelua ja ratkaisuja, joilla kierrettiin Metka opiskelijapalveluiden käytössä olleen pilvipalvelun teknisiä rajoitteita. Sovellus kehitettiin siten, että sen suorittaminen Metka opiskelijapalveluiden pilvipalvelussa oli mahdollista. Työ toteutettiin web-sovelluksena PHP-ohjelmointikielellä käyttäen Yii Framework -sovelluskehystä. Tietokantana käytettiin MySQL-tietokantaa.

Työn tarkoitus oli mahdollistaa Metka opiskelijapalveluiden Kaljaasi-risteilyn järjestäminen vuonna 2014. Risteilyn järjestämisessä onnistuttiin ja sovelluksesta päätettiin tehdä tavoiteltua laajempi. Laajennetun sovelluksen jatkokehitys jatkuu syyskuuhun 2015, minkä jälkeen sovellus jää Metka opiskelijapalveluiden käyttöön.

Avainsanat	Yii, PHP, MySQL, tapahtumanhallintajärjestelmä

Author	Jukka Lähetkangas
Title	Event management software as a web application
Number of Pages	28 pages + 4 appendices
Date	28 November, 2014
Degree	Bachelor of Engineering
Degree Programme	Media Engineering
Specialisation option	Digital media
Instructor	Aarne Klemetti, Senior Lecturer

The web application presented in this thesis was commissioned by Metka opiskelijapalvelut to be used as an event management tool to enable extensive data manipulation required in organizing large scale events, specifically the Kaljaasi-cruise, organized in November, 2014.

This paper focuses on product design and necessary technical solutions required to enable the developed application to be deployed in the cloud service used by Metka opiskelijapalvelut. Additionally, this paper places emphasis on use case scenarios, possible alternative technologies, database models and post production. The application was developed with Yii framework in PHP using a MySQL database.

The purpose of the application was to make the organizing of the Kaljaasi-cruise possible. The Kaljaasi-cruise of 2014 was organized successfully and ultimately the application covered an larger set of features than was necessary for the cruise of 2014. Development of the extended features will be completed in September 2015 and the application will remain in use of Metka opiskelijapalvelut.

Keywords	Yii, PHP, MySQL, event management

Sisällys

Termit ja lyhenteet

AJAX	Asynchronous Javascript and XML. Nimitys asynkroniselle tiedonsiirrolle.
CSS	Cascading Stylesheets. Tyylimääritelmä HTML-dokumentille.
CSV	Comma-separated values. Taulukkotiedostomuoto, jossa arvot erotellaan toisistaan pilkulla.
EAV	Entity-Attribute-Value. Tietomalli, joka mahdollistaa ennalta määrittelemättömän tiedon tallentamisen relaatiotietokannoissa.
HTML	Hypertext Markup Language. Merkintäkieli, jolla koostetaan web-dokumenttien rakenne.
JSON	JavaScript Object Notation. Merkintätapa selväkielisen tiedon välittämistä varten.
MySQL	Suosittu tietokantasovellus.
Pakka	Tässä työssä toteutetun sovelluskokonaisuuden kutsumanimi. Pakkasovellukseen kuuluu ilmoittautumissovellus ja hallintasovellus.
PHP	Hypertext Preprocessor. Komentosarjakieli, jolla suoritetaan palvelinlogiikkaa ja kirjoitetaan HTML-dokumentteja.
SQL	Structured Query Language. Kyselykieli, jolla ohjataan tietokannan toimintaa.
Yii	PHP-kielelle saatavilla oleva sovelluskehys.

1 Johdanto

Insinöörityössä selvitetään pienyrityksen järjestämän vuosittaisen tapahtuman organisoinnissa käytetyn ilmoittautumis- ja hallintasovelluksen puutteita, esitetään toimenpiteet puutteiden korjaamiseksi ja toteutetaan uusi sovellus pienyrityksen käyttöön.

Työ tehdään Metropolia Ammattikorkeakoulun opiskelijakunta METKAn omistaman yhtiön Metka opiskelijapalvelut oy:n tilauksesta. Metka opiskelijapalvelut oy järjestää vuosittaista opiskelijaristeilyä, Kaljaasia, jolle osallistuu arviolta 1 500 opiskelijaa. Tapahtuman kaikki tietojenkäsittely tehdään yhden www-sovelluksen kautta. Sovelluksen tarkoitus on mahdollistaa tapahtumaan ilmoittautuminen, tietojenkäsittely käyttäjäystävällisillä työkaluilla sekä erilaisten tilastojen ja tulosteiden käsittely tapahtuman järjestäjiä ja varustamoa varten. Työssä toteutettava sovellus korvaa aikaisemmin käytössä olleen sovelluksen. Tilatussa sovelluksessa on kaksi osa-aluetta: tapahtumanhallintasovellus ja ilmoittautumissovellus. Kahden osa-alueen kokonaisuudelle annettiin projektinimeksi Pakka-sovellus.

Yksi suurimmista haasteista tilatun sovelluksen teknisessä toteuttamisessa on tyytyminen käytössä olevan webhotellipalvelun tarjoamaan tekniikkaan. Sovellus tulee kehittää siten, että rakenteeltaan määrittelemätön ilmoittautumistietokanta on mahdollista toteuttaa rakenteeltaan tarkasti määritellyissä relaatiotietokannoissa, kuten MySQL:ssä. Sovelluksen tilaajalle tärkeintä on sovelluksen toimiminen yleisellä tasolla ja kaiken tarvittavan tiedon tallentaminen ja listaaminen.

Työn painopiste on palvelinlogiikan suorittamassa tietojenkäsittelyssä, minkä vuoksi sovelluksen ulkoasu toteutetaan mahdollisimman vähällä työllä Bootstrap-tyylikehyksen avulla. Sovelluksen toimintaan liittyy suuri määrä ominaisuuksia erityisesti hallintapaneelin puolella, ja siksi sovelluksen kehitys toteutetaan kahdessa vaiheessa. Insinöörityönä tehdään sovelluskehityksen ensimmäinen vaihe, joka valmistuu syksyllä 2014. Sovelluske-

hityksen toinen vaihe toteutetaan vuonna 2015.

2 Vanha sovellus

Metka opiskelijapalvelut oy järjestää kerran vuodessa suuren opiskelijaristeilyn, johon osallistuu noin 1 500 opiskelijaa. Suuren ja kysytyn tapahtuman järjestäminen edellyttää jatkuvaa tiedonkäsittelyä tapahtumaan ilmoittautuneiden henkilöiden osalta [1]. Tapahtuman järjestäjä tarvitsee lomakkeen ilmoittautujia varten, hyvät työkalut lomakkeesta kerätyn tiedon käsittelyä varten ja sopivat tulosteet varustamoa varten.

Ilmoittautujien suuren määrän vuoksi moni ominaisuus on automatisoitava, jotta ylläpitäjän ei tarvitse kuluttaa työtunteja toistuviin rutiineihin. Toistuvia tehtäviä, kuten laskujen ja maksuvahvistusten lähettämisiä ilmoittautujille, tulee niin paljon, että tapahtumaa ei voitaisi järjestää käytössä olevilla resursseilla. Lisäksi tulosteiden tulee olla tietynlaisia varustamoa ja lähtöselvityksen logistiikkaa varten (liite 1).

Ilmaisia valmiita lomakeohjelmia on tarjolla, kuten Google Forms, joka tallentaa lomakkeen tiedot Googlen Spreadsheets-taulukkolaskentaohjelmaan [2]. Vastaavissa lomakeohjelmissa on kaikissa sama puute: automatisoinnin mahdollisuudet ovat hyvin rajalliset ja tallennettu tieto vain yhdessä, vaikeasti käsiteltävässä, muodossa.

Ilmoittautumissovelluksena käytettävän www-sovelluksen tulee mahdollistaa lomakkeiden luominen siten, että luotu lomake saadaan vastaamaan tapahtumakohtaisesti sitä, mitä ilmoittautujilta halutaan kysyä. Lomakkeen hallinnoinnissa tulee olla mahdollisuus asettaa muutoksia ilmoittautumisen hintaan riippuen siitä, miten käyttäjä lomakkeen täyttää. Lisäksi sovelluksen tulee tarjota tapahtumakohtaisia palveluja, kuten sähköpostin lähetystä ja viitenumeroiden laskemista. Ominaisuuksien tulee olla tapahtuman järjestäjälle niin helppokäyttöisiä, että erillistä teknistä asiantuntijaa ei tarvita tapahtuman ylläpitäjäksi valmistuneen tuotteen tuotantoonviennin jälkeen.

Metka opiskelijapalveluissa vuodesta 2011 käytössä ollut ilmoittautumissovellus oli kaksi

PHP-komentosarjasovellusta, jotka käyttivät samaa tietokantaa. Ensimmäinen sovellus oli ilmoittautumislomake ja toinen oli hallinta- ja tilastosovellus [3]. Käyttäjäkokemus oli ilmoittautujalle seuraava:

1. Käyttäjä saapuu etusivulle ja näkee alasvetovalikon, joka listaa hyttiluokat, joista käyttäjä valitsee yhden.
2. Valittuaan hyttiluokan käyttäjä saa lomakkeen, johon pitää täyttää kaikkien hytissä matkustavien henkilöiden tiedot.
3. Tiedot täytettyään käyttäjä painaa hyväksymisnappia, jolloin käyttäjää kiitetään lomakkeen täyttämisestä. Käyttäjä saa sähköpostiviestin, joka sisältää lisäohjeita ja tiedot hytistä.

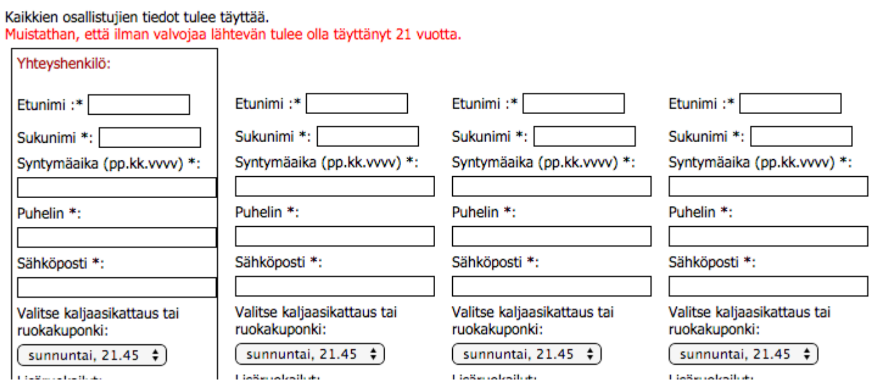
Hallinta- ja tilastosovellus pyysi salasanan sisäänkirjautumista varten, minkä jälkeen sovellus näytti listan tärkeistä tiedoista, kuten ilmoittautuneiden henkilöiden lukumäärä, varattujen hyttien lukumäärä, maksettujen laskujen summa ja kaikkien ilmoittautuneiden velka yhteensä. Tilastosivulla oli myös linkki sivulle, joka listasi varatut hytit ja niiden eräpäivät. Lista mahdollisti hytin tietojen muokkaamisen, kuten maksun vahvistamisen ja henkilötietojen ylikirjoittamisen. Lisäksi sivulla oli työkalu, josta pääsi muokkaamaan hyttien tietoja, kuten sen, kuinka monta mitäkin hyttiä on jäljellä, kuinka monta hyttejä on yhteensä, ja niin edelleen. Tapahtuman jälkeen tietokanta voitiin tyhjentää, minkä jälkeen seuraavan tapahtuman järjestäminen oli vasta mahdollista.

Vanhan ilmoittautumissovelluksen ohjelmakoodista ja tietokantamalleista puuttui abstraktointi. Itse sovellus oli toteutettu ilman olio-ohjelmointia, ja kaikki ohjelmakoodi, HTML-merkinnät ja tyylimäärittelyt olivat yhdessä tiedostossa. Tietokannan taulut eivät olleet normalisoituja, eikä tauluihin tallennetussa tiedossa ollut voimassaoloaikaleimoja. Aikaleimattavilla tietokannoilla saataisi käyttöön tietomalli, jossa voimassaolevaa tietoa on se, jonka alkamishetki on menneisyydessä ja päättymisaikaleima ei ole menneisyydessä tai sitä ei ole asetettu. Aikaleimojen puute aiheutti yleisen tietojenkäsittelyongelman: kaikki muutokset, esimerkiksi hytin kokoonpanoon, helposti ylikirjoitettiin vanhan tiedon päälle. [3.]

Hyttiluokkien tietoja lukuun ottamatta kaikki käyttäjille näkyvä teksti oli upotettu HTML-dokumentin tulostavan PHP-sovelluksen koodiin. Pelkästään lähtevän sähköpostin tekstin vaihtaminen edellytti, että tapahtuman järjestäjän tuli olla yhteydessä tekniseen asiantuntijaan, jotta tämä vaihtaisi tekstin. Alkuperäisessä PHP-ohjelmassa ei käytetty olio-ohjelmointia, sovelluskehyskirjastoja, sisällönhallintajärjestelmiä tai mitään muutakaan tuotekehitystä helpottavaa aineistoa. Myös tietokantaa ohjaavat SQL-kyselyt olivat käsittelemättöminä sellaisinaan PHP-ohjelman seassa. [3.]

Sovellus ei edes kirjoittanut sähköposteja tietokantaan, eli tapahtuman järjestäjällä ei ollut omaa kopiota käyttäjille lähetetyistä kuiteista. Ilmoittautumislomake ei myöskään suorittanut minkäänlaista validointia. Ottaen huomioon suuren risteilyn taloudellisen riskin järjestävälle taholle, edes jatkuvasti läsnä oleva tekninen asiantuntija ei olisi pystynyt paikkaamaan sovelluksen puutteiden aiheuttamaa tietoturvaongelmaa. Sovelluksen puutteellinen rakenne edellytti, että tapahtuman järjestäjällä oli oltava vuosittain tekninen asiantuntija mahdollistamassa sovelluksen käyttöönoton.

Hallintapaneelin ja ilmoittautumissovelluksen käyttöliittymä oli alkeellinen. Hallintapaneelin käyttöliittymän etuna oli kuitenkin se, että tietyt sovelluksen osa-alueet olivat minimalistisuuden vuoksi erittäin selkeitä ja helppokäyttöisiä. Kuvassa 1 esitetty ilmoittautumissovelluksen käyttöliittymä puolestaan kyseenalaisti koko sovelluksen luotettavuuden. Käyttäjä pystyi jättämään ilmoittautumissovelluksen auki selaimen ikkunaan ja palaamaan myöhemmin, esimerkiksi viikon päästä, täyttämään lomakkeen loppuun, vaikka risteily olisi jo loppuunmyyty. Kaikki lomakkeessa tarvittava logiikka tuli kirjoittaa PHP-ohjelmakoodina suoraan ilmoittautumissovelluksen ulkoasun määrittelevään tiedostoon.



Kuva 1: Vanhan ilmoittautumissovelluksen hytinvaraussivu.

3 Uusi sovellus

3.1 Ominaisuudet ja kehitys

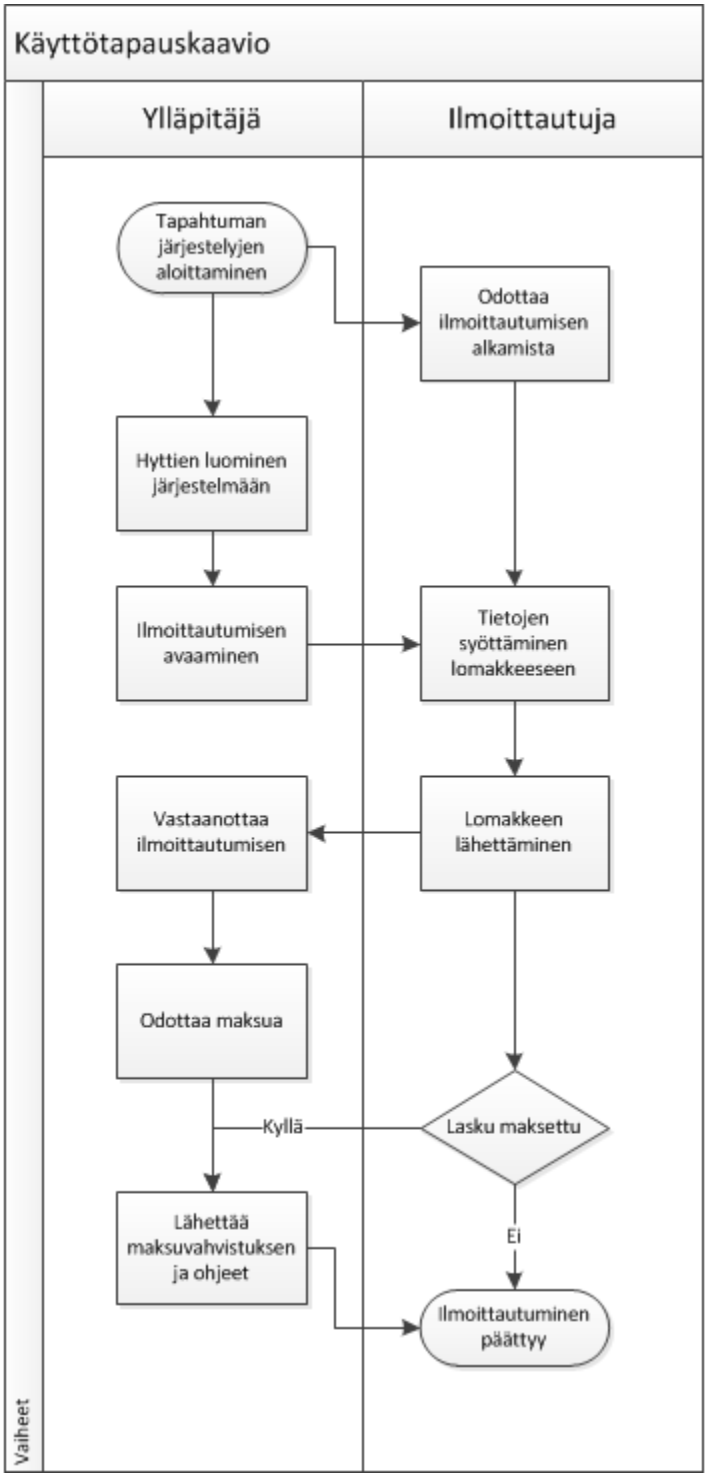
Kevään ja kesän 2014 aikana tehtyjen haastattelujen yhteydessä ilmoittautumissovelluksen vertauskuvana käytettiin korttipakkaa. Ajatuksena oli, että käyttäjän avaama ilmoittautumislomake olisi kuin pelikorttikäsi – jokainen lomakkeen henkilösarake olisi kuin kortti, jotka otetaan korttipakasta. Korttipakkaa olisi jäljellä niin kauan, kuin hyttejä olisi jäljellä. Sovellus sai tästä lempinimen Pakka. Korttipakkamaista visuaalista ilmettä ei aikataulun takia tehty, mutta palvelinlogiikka on sama.

Metka opiskelijapalveluiden uudessa Pakka-sovelluksessa tuli olla vain kahdenlaisia käyttäjiä: ylläpitäjiä ja ilmoittautujia. Ylläpitäjällä on pääkäyttäjätunnus, jolla kirjautua hallintapaneeliin. Mikäli ylläpitäjiä on useita, he käyttävät samaa pääkäyttäjätunnusta. Ilmoittautujilla ei ole minkäänlaisia käyttäjätunnuksia, sillä heidän ei odoteta palaavan ilmoittautumislomakkeeseen onnistuneen ilmoittautumisen jälkeen.

Käyttöliittymäsuunnittelua varten on tärkeää tietää vaiheet, jotka sovelluksen käyttäjän tulisi käydä läpi missäkin käyttötapauksessa. Vaiheiden selvitystä varten toteutettiin käyttäjäryhmien haastattelut, joiden pohjalta laadittiin liitteet 1 ja 2. Ilmoittautuminen on prosessi, jossa ilmoittautuja ja ylläpitäjä reagoivat vuorotellen. Lopullinen prosessi on esitetty kuvassa 2.

Alun perin Pakka-sovellus suunniteltiin toteutettavaksi JavaScriptillä Node.js-palvelinalustalla käyttäen MongoDB-tietokantaa. MongoDB on dokumenttitietokanta, jolla voi tallentaa tietoa, jonka muoto ei ole ennalta määritelty [4]. Tämä olisi sopinut täydellisesti ilmoittautumislomaketta varten, jonka tietomalli ei ole tiedossa ja joka on aina tapahtuman järjestäjän tai ylläpitäjän muokattavuuden varassa. Dokumenttitietokannan heikkous on relaatioiden

puuttuminen kokonaan. Node.js-palvelinalusta nojaa vahvasti MongoDB:n käyttöön, joten sen valinta sovelluskehykseksi olisi ollut luonnollista. Muiden ohjelmointikielten ja sovelluskehysten tuki dokumenttitietokannoille on vielä rajallinen. MongoDB Inc on toteuttanut eri ympäristöille useita ajureita, joiden avulla tietokantaa voi käyttää, mutta sovelluskehyksien oma tuki ajureille on rajallinen. [5.]



Kuva 2: Prosessikaavio ilmoittautumissovelluksen ylläpitäjän ja käyttäjän toiminnoista.

Node.js-sovellukset ovat itsenäisiä prosesseja, jotka toimivat HTML- ja JSON-dokumenttien kirjoittajina. Ne eivät toimi kuten perinteiset tiedostonjakopalvelimet, jotka tulkitsevat erillisissä tiedostoissa sijaitsevia komentosarjaohjeita [6]. PHP on yleisesti käytetty komentosarjakieli, jota kehitetään edelleen, ja vanhasta iästään huolimatta se on edelleen suosittu, koska sillä saa toteutettua hyvän tuotteen hyvin vähällä vaivalla vuosien saatossa valmistettujen kattavien työkalujen ansiosta. Muita vaihtoehtoja käytetyiksi kieliksi ja ympäristöiksi harkittiin pikaisesti, kuten Ruby on Rails- ja Django-sovelluskehyksiä, mutta ne eivät tarjonneet mitään, mikä ei olisi ollut mahdollista PHP:llä tai Node.js:llä. [7.]

Asiakkaalla oli tarve tallentaa palvelimelle muutakin materiaalia ja muita www-sivuja, kuten Wordpress-sivustoja, joiden käyttäminen edellyttää tiedostonjakopalvelinta, joka kykenee tulkitsemaan PHP-komentosarjoja [8]. Ilmoittautumisohjelman tilaajalla oli jo käytössään pilvipalvelin, joka kykeni suorittamaan PHP-sovelluksia ja MySQL-tietokantaa ja lähettämään sähköpostia. Päätettiin, että uuden sovelluksen tulisi toimia tässä ympäristössä, jottei sovelluksen toiminta jäisi riippuvaiseksi alkuperäisestä kehittäjästä.

Sovelluksen ohjelmointikieleksi valittiin PHP ja tietokannaksi MySQL, sillä niille on laajaa tukea ja Metka opiskelijapalveluilla oli jo käytössään pilvipalvelin, jossa PHP- ja MySQL-sovelluksia voi suorittaa. PHP on usealle kehittäjälle niin kutsuttu aloituskieli, eli PHP:n taitajia löytyy jatkokehitystä varten helposti tulevaisuudessakin. Haasteeksi jäi ilmoittautumislomakkeen tietojen tallentaminen vaivattomasti, minkä MongoDB olisi ratkaissut. Pakka-sovellus päätettiin toteuttaa projektiluontoisena kahdessa vaiheessa. Kehitystyön tuli alkaa keväällä 2014 ja päättyä syksyllä 2015. Projektin jälkeen sovellusta jatkokehitettäisiin ja ylläpidettäisiin vielä enintään vuosi, mikäli asiakkaan ja sovelluskehittäjän sopimusta ei purettaisi ennenaikaisesti.

Projektin ensimmäisen vaiheen tuli olla valmis lokakuun alussa 2014, jolloin sovelluksen oli tarkoitus vastata toiminnoiltaan vanhaa, käytöstä poistettavaa sovellusta liitteen 3 mukaisesti. Projektin toisen vaiheen tulisi olla valmis lokakuun alussa 2015, jolloin sovellus olisi sellaisessa tilassa, että sovelluksen käyttäminen ja hallinnointi ei tarvitsisi sovelluskehittäjän väliintuloa.

Insinöörityöraporttia kirjoitettaessa projektin ensimmäinen vaihe on päättynyt. Kehitystyö on jäädytetty siihen saakka, että syksyn 2014 risteily on ohi, jolloin sovelluksen tuotantovalmiudella ei ole enää merkitystä. Ensimmäisen ja toisen vaiheen välissä kerätään palautetta ja kehitetään suunnitelmaa jatkokehitystoimenpiteistä, tulevista korjaustöistä ja puuttuvien ominaisuuksien kehittämisestä.

Pakka-sovellus tuli toteuttaa web-sovelluksena, jonka toteutuksessa tuli painotta sulavaa käyttökokemusta. Sivut ovat HTML-dokumentteja, joille luodaan ulkoasu CSS-tyylimäärittelyillä. HTML on merkintäkieli, jolla koostetaan www-sivun rakenne. HTML-dokumentin elementit koostavat kaiken sisällön HTML-dokumentissa. CSS-tyylimäärittelyt toteuttavat kaiken visuaalisen asettelun.

Sovelluksen painopiste oli tietojenkäsittelyssä, joten ulkoasun toteuttamisessa edettiin siten, että ilmoittautumissivun tuli näyttää intuitiiviselta, mutta yleiseen asetteluun tuli käyttää vain rajallinen määrä työtunteja. Lopputuloksesta tuli tyydyttävä, mutta viimeistään projektin toisessa vaiheessa saadaan toteutettua riittävä viimeistely. Sovelluksessa käytettiin yleisilmeeseen suosittua Bootstrap-tyylikehystä. Se tarjoaa valmiin CSS- ja JavaScript-pohjan, jonka päälle sovelluskehittäjän on helppo jatkaa [9]. Ilmoittautumissovelluksessa kaikki käyttäjäinteraktio tapahtuu www-sivulla ajettavan JavaScript-sovelluksen kautta. Perinteisiä HTML-lomakkeita ei käytetä, vaikka HTML-lomakkeiden *input*-elementtiä käytetään lomakkeen toteuttamiseen.

Tietoa käyttäjän ja palvelimen välillä liikutetaan JavaScript-ohjelman mahdollistamilla AJAX-kutsuilla. Termillä AJAX (Asynchronous Javascript and XML) tarkoitetaan sovellusta, joka lähettää HTTP-pyyntöjä palvelimelle ilman, että www-selain lataa sivua uudestaan. AJAX-kutsut palvelimelle voivat olla täysin huomaamattomia. Ilmoittautumisohjelman kaikissa vaiheissa, joissa käyttäjä painaa hyväksymisnappia siirtyäkseen seuraavalle sivulle, käytetään huomaamatonta AJAX-kutsua. Käyttäjä siirretään seuraavalle sivulle vasta, kun AJAX-kutsu on palannut palvelimelta palvelimen hyväksynnän kanssa. [10, s. 339.]

Kokemus vanhasta ilmoittautumissovelluksesta osoitti, että ilmoittautumisohjelman ruuhkautuessa käyttäjät usein painavat www-selaimen *päivitä*-nappia sen toivossa, että so-

vellus päästäisi käyttäjän nopeammin eteenpäin. Tästä seurasi, että ruuhkautuneet palvelinpyynnöt saattoivat mennä palvelimelta läpi useamman kerran. Ylläpitäjä joutui usein etsimään tietokannasta varauksia, joissa oli samat tiedot kuin jossain toisessa varauksessa.

AJAX-lähestymistapa lomakkeisiin oli tehokkain keino vähentää toistettuja varauksia, joita perinteinen HTML-lomake helposti aiheuttaa. Mikäli käyttäjä painaa *päivitä*-nappia, www-selain ei yritä lähettää lomaketta uudestaan, koska keskustelun palvelimen kanssa suorittaa JavaScript-sovellus. Koodiesimerkki 1 näyttää ilmoittautumissovelluksen etusivulla sijaitsevan hyttiluokkavalinnan HTML-merkinnän. HTML-elementille *button* on annettu *data-deckid*-arvo, joka kertoo JavaScript-sovellukselle, mikä on valitun hyttiluokan tunnistenumero. Sivulla ei ole varsinaista HTML-lomaketta, ja jokaisen *button*-elementin tarkoitus on palvella JavaScript-sovellusta.

```
<tr>
  <td> A-luokan hytti 2 hengelle </td>
  <td>
      <button type="button"
          class="btn btn-primary selectDeck"
          data-toggle="modal"
          data-deckid="9"
          data-target="#selectDeckModal">
        Valitse
      </button>
  </td>
</tr>
```

Koodiesimerkki 1: Hytin valitsemisen merkintä.

Pakka-sovellus toteutettiin kahdessa erillisessä osassa käyttäen hyväksi Yiin liitännäisrakennetta. Yii-sovellukset koostuvat yhdestä isäntäsovelluksesta ja useasta liitännäissovelluksesta, jotka ovat rakenteeltaan samankaltaisia isäntäsovelluksen kanssa [10, s. 479]. Pakka-sovellus toteutettiin kahtena liitännäissovelluksena, ilmoittautumisohjelmana ja hallintapaneelina, käytöstä poistettavan ilmoittautumissovelluksen tapaan. Isäntäsovellusta käytettiin vain hallinnoimaan tietokantamalleja.

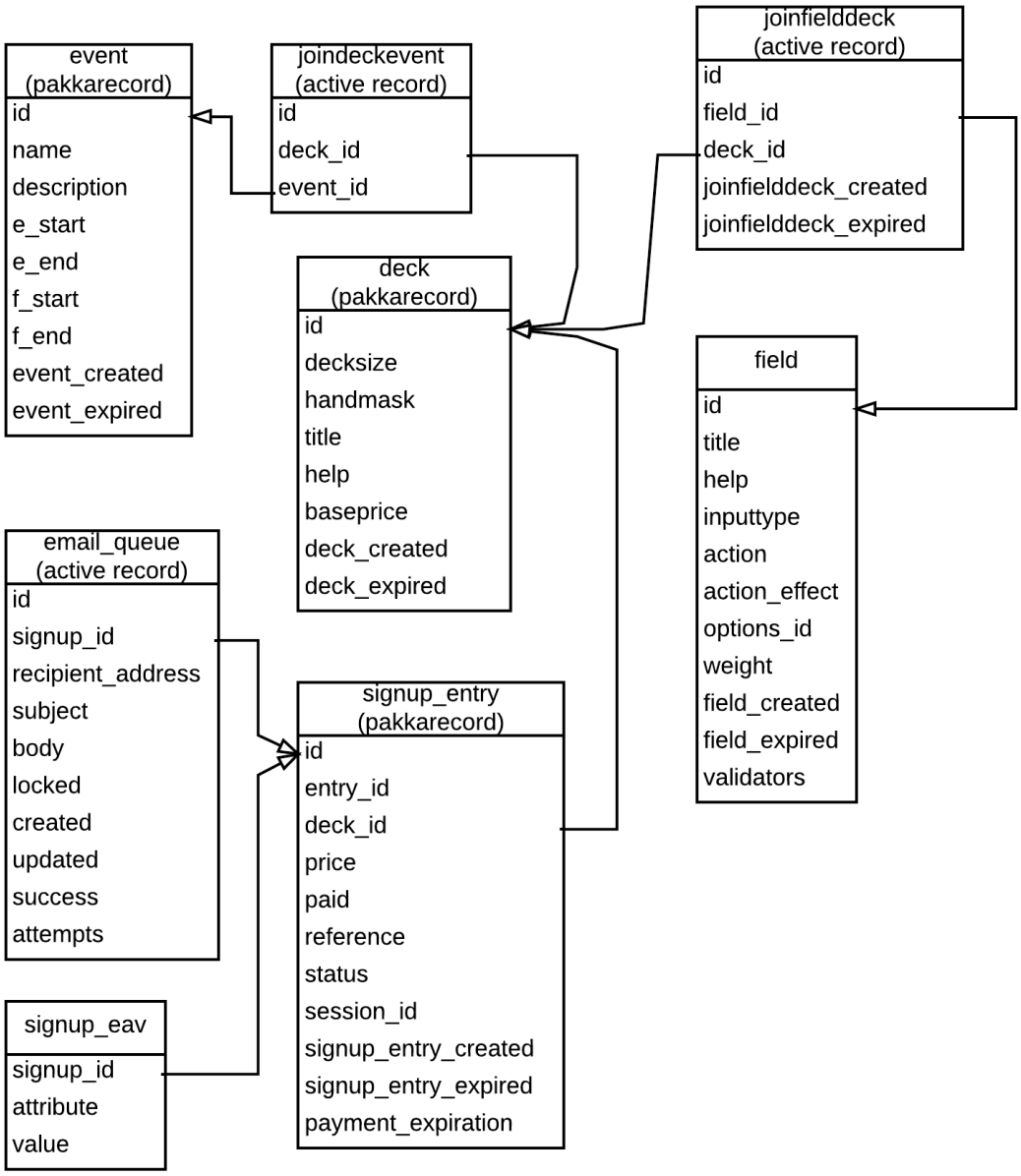
Sovelluksen tärkeimpiä ominaisuuskokonaisuuksia olivat ilmoittautumislomakkeen validointi ja ylläpitäjän tiedonkäsittely. Ilmoittautumislomakkeen validointi oli sovelluksen kehityksen haastavimpia vaiheita, sillä ilmoittautumislomakkeen muoto ei ole tiedossa, en-

nen kuin ylläpitäjä tekee lomakkeen. Validointi tuli siis toteuttaa ilman tietokantamallia, jota vasten lomakkeeseen syötettyä tietoa voisi peilata.

3.2 Active record ja pakkarecord

Tilatun ilmoittautumissovelluksen kannalta oli tärkeää, että mitään tietoa ei katoa ja että tietoa voidaan etsiä aikaleiman avulla. Tällöin esimerkiksi varustamolle voidaan antaa tietoa siitä, kuka varaaja korvaa kenet, jos hytteihin tulee muutoksia. Aikaleimattu tieto voisi auttaa myös tarvittaessa palauttamaan sovelluksen tiettyyn tilaan. Aikaleimattavilla tietokannoilla voidaan ottaa käyttöön tietomalli, jossa voimassaolevaa tietoa on se, jonka alkamishetki on menneisyydessä ja päättymisaikaleima ei ole menneisyydessä tai sitä ei ole asetettu. Kuvassa 3 on ER-kaavio Pakka-sovelluksen koko tietokantarakenteesta. Mainittavimpana erikoisuutena ovat pakkarecord-mallia käyttävät tietokantataulut, joiden määrittelyssä ovat aikaleiman merkinnät muodossa *taulun_nimi_created* ja *taulun_nimi_expired*.

Yii käyttää oletusarvoisesti yksinkertaista active record -arkkitehtuuria tietokantamalleissaan. Active record -arkkitehtuuri on tietokantarajapinta, joka mahdollistaa tietokannan käsittelyn abstraktoinnin ohjelmointikielelle. Yiin active record -malli sisältää ominaisuudet yksinkertaiseen lukemiseen ja kirjoittamiseen [10, s. 166]. Pakka-sovellusta varten kehitettiin *pakkarecord*-nimellä kulkeva malli, jonka ominaisuudet periytyvät Yiin active record -mallista. Pakkarecord automatisoi Pakka-sovelluksen kannalta tehtävät, jotka tulisi tehdä joka kerta, kun tietokantaan kirjoitetaan tai tietokannasta poistetaan rivejä. Tärkein pakkarecord-mallin toiminto on aikaleimaamisen automatisointi jokaisen tapahtuman yhteyteen siten, että sovelluskehityksen aikana siihen ei tarvitse puuttua.



Kuva 3: Pakka-sovelluksessa käytetyn tietokannan kaavio.

Koodiesimerkin 2 esimerkki expire-jäsenfunktiosta korvaa tavallisessa active record -mallissa käytetyn delete-jäsenfunktion. Delete-funktio tuhoaisi tietuerivin tietokannasta, kun expire-funktio kirjoittaa rivin expired-sarakkeeseen kirjoittamishetken aikaleiman, minkä jälkeen Pakka-sovellus tulkitsee rivin vanhentuneeksi. Funktio pyytää ensimmäiseksi parametriksi aikaleiman ja toiseksi parametriksi tiedon siitä, asetetaanko vanhentumisaikaleima sellaisenaan ensimmäisen parametrin mukaan vai asetetaanko vanhentumisaikaleima muutoksena nykyhetkestä ensimmäisen parametrin arvon verran.

```
1   public function expire($seconds = 0, $absoluteMoment =
       false)
2   {
3     if ($absoluteMoment === false) {
4       $time = time()+$seconds;
5     } else if ($absoluteMoment === true) {
6       $time = $seconds;
7     }
8
9     $expiredField = $this->tableName().'_expired';
10    $this->{$expiredField} = $time;
11    return $this->save();
12  }
```

Koodiesimerkki 2: Pakkarecord Expire.

Yii mahdollistaa active record -mallien vakioulottuvuuden (default scope) uudelleenmäärittelyn [10, s. 173]. Pakkarecord-malli syrjäyttää vakioulottuvuuden siten, että kaikki palautusarvot, joiden expired-aikaleima on menneisyydessä, hylätään vanhentuneina (koodiesimerkki 3). Täten sovelluksen kehityksen aikana ei tarvitse kirjoittaa tarkistuksia siitä, ovatko kyselyistä palautuneiden tietueiden aikaleimat valideja, vaan pakkarecord-malli käyttäytyy, ikään kuin vanhentuneet tietueet olisi poistettu tai tietueita ei olisi olemassakaan

```
1   public function defaultScope()
2   {
3     return array(
4         'condition' => $this->tableName().'_expired>'.time()
5     );
6   }
```

Koodiesimerkki 3: Pakkarecord defaultScope.

3.3 Tietomallit

Koska ilmoittautumislomakkeen tallentaman tiedon muoto ei määräydy suoraan mistään olemassa olevasta tietomallista, lomakkeen käsittelyn eri vaiheissa tuli käyttää vapaamuotoisempaa lähestymistapaa. Jokaista ilmoittautumistapausta käsitellään SignupEntry-nimisellä mallilla. SignupEntry on pakkarecord-malli, joka sitoo yhteen käyttäjän syöttämän ilmoittautumistiedon ja yksityiskohdat siitä, minkä "korttipakan" käyttäjä on valinnut, sekä korttipakkaan liittyviä muuttujia, kuten lopullisesta ilmoittautumisesta laskettava hinta.

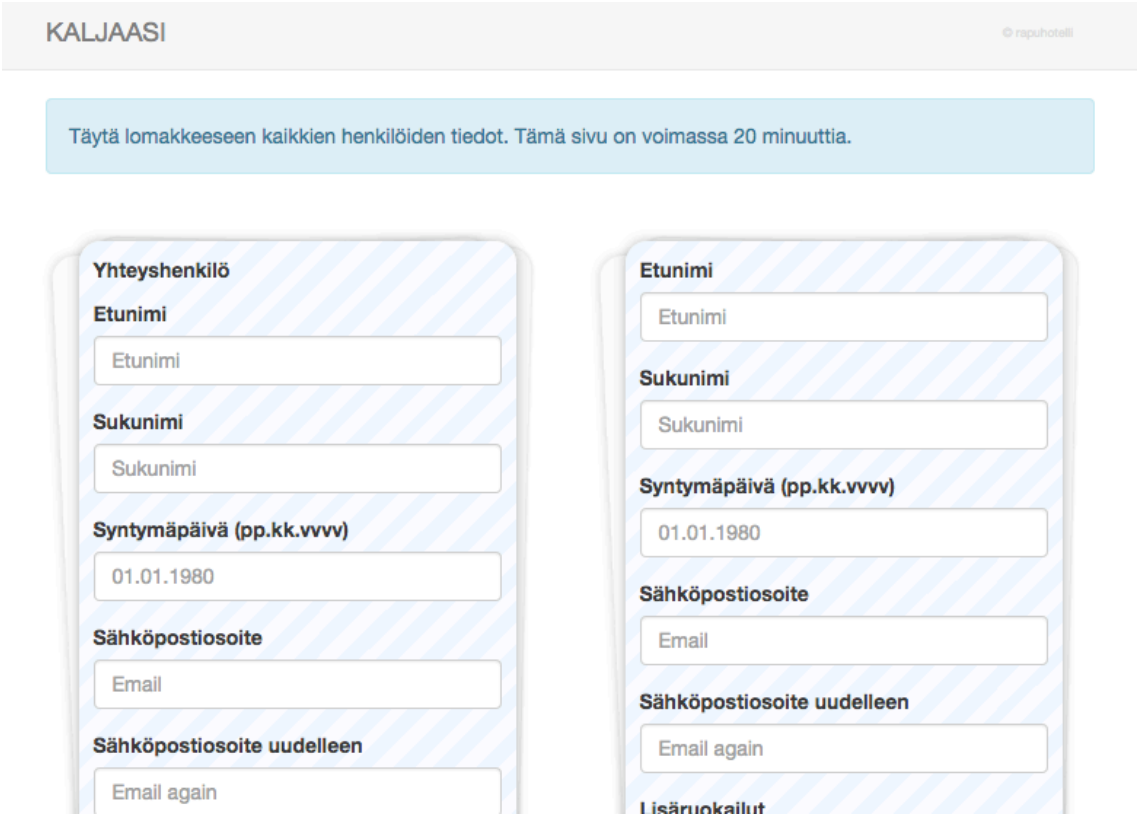
Ilmoittautumissovelluksen käyttäjän valittua hyttiluokkansa, ilmoittautumissivulla sijaitseva JavaScript-sovellus lähettää POST-pyynnön palvelimelle. Pyynnön saavuttua palvelimelle palvelin tarkistaa vielä, voiko valintaa toteuttaa. Mikäli valinta onnistuu, sovellus kirjoittaa signup_entry-tietokantatauluun merkinnän taulukossa 1 esitetyn esimerkin mukaisesti.

Taulukko 1: Ote signup_entry-tietokantataulusta.

entryId	deckId	price	paid	reference	created	expired	status
600f56444d	5	0	0	52552	1411545872	1411547079	0

Taulukossa 1 esitetty tietokantataulu näyttää varauksen tunnisteen, korttipakan eli hyttiluokan tunnisteen, hinnan, jo maksetun hinnan, viitenumeron, tilanteen ja aikaleimat. Tilanne arvolla "0" kertoo sovellukselle, että käyttäjä on avannut lomakkeen, mutta käyttäjä ei ole tallentanut lomaketta. Käyttäjä tunnistetaan evästellä, joka on sama kuin varauksen *entryId*. Mikäli sovellus tunnistaa palaavan käyttäjän lomakkeen tilanteen arvolla 0 ja sen erääntymishetki on tulevaisuudessa, voidaan lomake palauttaa käyttäjän täytettäväksi ja erääntymishetkeä siirtää. Lopuksi palvelin palauttaa ilmoittautumissivun JavaScript-sovellukselle hyväksyvän ilmoituksen, jolloin JavaScript-sovellus siirtää käyttäjän itse lomakesivulle. Ilmoittautumislomake on pystysuuntaisesti aseteltu HTML-lomake, joka esitetään rinnakkain niin monta kertaa, kuin valitun hyttiluokan henkilömäärä edellyttää. Rinnakkaiset lomakkeet esitetään siten, että eri henkilöiden tiedot erotellaan juoksevalla numerolla. Esimerkiksi sukunimet kahden hengen hytissä ilmoitettaisiin palvelimelle kenttien

tunnisteilla *sukunimi[0]* ensimmäiselle henkilölle ja *sukunimi[1]* toiselle henkilölle. Sovellus tunnistaa tällä tavalla eritellyn lomakkeen siten, että sana *sukunimi* tunnistetaan avainsanaksi, jolloin sukunimien listaus voidaan luoda automaattisesti. Kuvassa 4 on ilmoittautumislomake kahden hengen hyttiin.



Kuva 4: Kahden hengen hytin varaustilanne.

Lomakkeen kentät luo ylläpitäjä. Ylläpitäjä voi hallintatyökalussa luoda kenttiä ja antaa kentille toiminnallisuutta. Jokaiselle kentälle voi antaa myös parametrin, miten kentän tietoa validoidaan. Lisäksi kenttien nimeäminen tietyllä tavalla mahdollistaa automatisoituja prosesseja. Nimeämällä yhden kentän esimerkiksi "posti" ja toisen kentän "posti-again", syntyy automaattinen validointi, jossa molempien kenttien käyttäjältä tulevan syötteen tulee olla identtinen, esimerkiksi sähköpostiosoitteen oikeinkirjoituksen varmentamista varten.

Käyttäjän lähettäessä lomakkeen ilmoittautumissivulla sijaitseva JavaScript-ohjelma lähettää jälleen POST-pyynnön palvelimelle syötteen validointia varten. Mikäli validointi epäonnistuu, sovelluksen validointirajapinta palauttaa listan epäonnistuneista tilanteista eroteltuna lomakkeen kenttien tunnisteilla, jolloin JavaScript-sovellus värjää lomakkeessa punaiseksi ne kentät, joiden varmentaminen epäonnistui. Taulukossa 2 on esimerkki sii-

tä, miten saatettaisiin tallentaa sovellukselle ohje, miten lomakkeen jäsenyyskenttä voisi käyttäytyä.

Taulukko 2: Esimerkkikooste field-tietokantataulusta.

title	inputtype	action	action_effect	weight	validator
jasenyys	checkbox	price	-600	8	checkbox

Esimerkistä ilmenee, että kenttä on rasti ruutuun -kenttä, sen toiminto on hintamuutos ja hintamuutoksen arvo on -6 euroa, eli rastin rastittaneet ilmoittautujat saavat hinnasta kuuden euron alennuksen. Weight-parametri kertoo lomakesivun piirtävälle sovellukselle, missä järjestyksessä kentät luodaan. Toinen kenttä weight-sarakkeen arvolla "7" loisi kentän jäsenyyskentän yläpuolelle.

Yksi Pakka-sovelluksen suurimmista teknisistä haasteista oli ilmoittautujan täyttämän lomakkeen tallentaminen ja varmentaminen. Lomakkeelle ei ole tietomallia, vaan se tallennetaan ylläpitäjän määrittelemän rakenteen mukaisesti lomakkeen kentän tunnisteen ja henkilön järjestysnumeron mukaan. Lomakkeelle ei siis ole myöskään tietomallia, jota vasten käyttäjän syöttämiä tietoja voitaisi varmentaa. Luonnollinen ratkaisu on tallentaa tiedot dokumenttitietokantaan, mutta sovellus tuli asentaa ympäristöön, jossa oli mahdollista toteuttaa vain relaatiotietokantoja. [4.]

Ratkaisu oli tallentaa lomakkeen tiedot väljästi määriteltyyn niin sanottuun vertikaaliseen tietokantatauluun. Entity-attribute-value (EAV) -periaatteen mukainen tietokantataulu koostuu vain kolmesta sarakkeesta: tietueen tunnisteesta, ominaisuudesta ja ominaisuuden arvosta. EAV-lähestymistapa yhdistetään usein aloittelijavirheisiin, sillä tapa saa tietokantataulun helposti muistuttamaan Excel-taulukkoa. Tapa ei myöskään palvele relaatiotietokannan tarkoitusta, ja rivejä tulee nopeasti tuhansia, mutta hyvällä indeksoinnilla lukuhitautta voi kuroa kiinni [11]. Relaatiotietokannoissa ei juuri ole muita vaihtoehtoja tallentaa tietoa tavalla, joka ei olisi riippuvainen tietokannan rakenteesta.

Yksi syy EAV-mallin käyttämiselle oli Yii-sovelluskehykselle saatavilla oleva lisäosa, joka tuo EAV-ominaisuudet active record -mallin käytettäväksi [12]. Lisäosa vähensi huomattavasti sovelluskehitykseen käytettäviä työtunteja. Koodiesimerkissä 4 on kooste Pak-

kaController-luokasta, joka hallinnoi ilmoittautumissivua. Koodiesimerkin 4 PHP-koodi hakee käyttäjän selaimen evästeen perusteella kaiken käyttäjään liittyvän ilmoittautumistiedon. Rivillä 3 suoritetaan kysely tietokantaan selaimen evästeen mukaisesti ja tallennetaan muuttujaan tietokantakyselyn tulos. Rivillä 5 suoritetaan kysely EAV-tietokantatauluun ja tallennetaan tämän käyttäjän tiedot muuttujaan. Rivillä 3 esitetty SignupEntry on taulukossa 1 esitetyn tietokantataulun pakkarecord-olio. Haku evästeen perusteella onnistuu vain, mikäli vastaava SignupEntry on merkitty "status"-sarakkeessa arvolle "0", eli varausta ei ole palvelimella hyväksytetty.

```
1 $userCookie = Yii::app()->request->cookies['signup_entry'];
2 $entryId = $userCookie->value;
3 $reservation = SignupEntry::model()->
4              findByAttributes(['entryId' => $entryId]);
5 $reservationData = $reservation->getEavAttributes();
```

Koodiesimerkki 4: Pakkarecord EAV.

Taulukossa 3 on esitetty ote EAV-taulusta, jossa näkyy varaus avaintunnisteella 23. Varaukseen 23 liittyy taulukon mukaan kahden hengen tiedot nimestä, sähköpostista ja jäsenyydestä. Tunniste 23 viittaa signup_entry-taulun avaintunnisteeseen, joka toimii relaationa taulujen välillä.

Taulukko 3: EAV-tietokantataulu.

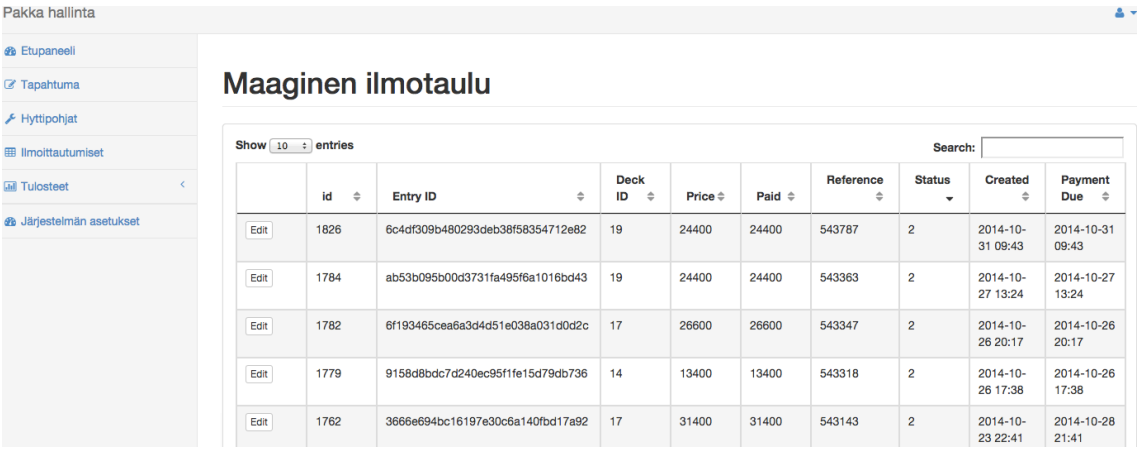
signup_id	attribute	value
23	etunimi[0]	Maija
23	etunimi[1]	Matti
23	sukunimi[0]	Meikäläinen
23	sukunimi[1]	Muukalainen
23	email[0]	maija.meikalainen@osoite.com
23	email[1]	matti.muukalainen@osoite.com
23	jasenyys[0]	x
23	jasenyys[1]	

4 Hallintasovellus

4.1 Ilmotaulu

Hallintasovelluksessa *ilmotaulu*ksi kutsutaan hallintasovelluksen keskeisintä osaa, jolla käsitellään varaustietoja. Ilmotaulu hakee hallintasovelluksen JSON-rajapinnasta suuren, kaikki varaustiedot sisältävän JSON-objektin ja luo interaktiivisen ilmoittautumislistauksen sen perusteella. Ilmotaulu ei ole varsinaisesti AJAX-sovellus, sillä kaikki tiedot haetaan kerralla ja vain kerran, heti ilmotaulun sivun avauduttua. JavaScript-lähestymistapa ilmotaulun tapauksessa oli edellytys sille, että ylläpitäjä voi selata satoja ilmoittautumisia lähettämättä uusia pyyntöjä palvelimelle. Ylimääräisten pyyntöjen lähettämisen välttäminen katsottiin tärkeäksi etenkin tilanteissa, joissa ilmoittautumissovellus on käytössä, jolloin ilmoittautumissovelluksen ja hallintasovelluksen jaettuja palvelinresursseja tulisi säästää.

Ilmotaulun tehokkuus perustuu automaattiseen suodatukseen. Ilmotaulun yläreunassa (kuva 5) on hakukenttä. Kun hakukenttään kirjoitetaan esimerkiksi "Jaakko", suodattuvat jo käyttäjän kirjoittaessa listaan pelkästään ne varaukset, joiden tiedoissa on mainittu ylläpitäjän kirjoittama teksti. Jokaiselle pystysarakkeelle on myös omat hakukenttänsä sarakkeen alareunassa (ei kuvassa), ja niistä voi yhdistellä sopivan hakukriteerin.



Kuva 5: Ilmotaulun käyttöliittymä.

Kuvitteellisessa tilanteessa, jossa ylläpitäjän tulee tietää sukunimeltään tuntemattoman Jaakon ilmoittautumismaksun eräpäivä, voisi ylläpitäjä toimia esimerkiksi seuraavasti:

1. Ylläpitäjä klikkaa Payment Due -sarakkeen otsikkoa asettaakseen järjestyksen pienemmästä suurempaan Payment Due -sarakkeen arvon mukaisesti.
2. Ylläpitäjä asettaa Status-sarakkeen hakuehdoksi arvon "1", eli lomake on lähetetty, mutta varaus on vielä maksamaton.
3. Ylläpitäjä kirjoittaa yläreunan yleisen haun kriteeriksi "Jaakko".

Esimerkin lopputuloksena Ilmotaulun listauksessa näkyvät kaikki varaamattomat hytit, joiden varaustiedoissa esiintyy sana "Jaakko". Lisäksi listaus on järjestynyt siten, että kaikista vanhin eräpäivä on ylimpänä. Edit-nappia painamalla avautuu valikko, josta ylläpitäjä voi lähettää sähköpostia ilmoittautujalle sekä muuttaa ilmoittautumislomakkeesta vastaanotettuja tietoja.

4.2 Tulosteet

Risteilyn järjestämisessä tarvitaan kahdenlaista listausta: varustamolle tiedot hyteistä ja risteilyn järjestäjille tiedot henkilöistä lähtöselvityksen logistiikkaa varten. Varustamo ei ole ilmoittanut, minkälaisen asettelun mukaisesti hyttitiedot tarvitaan. Edellinen hallintasovellus mahdollisti yksinkertaisten Excel-taulukoiden tulostamisen.

Pakka-sovelluksessa päätettiin käyttää CSV-tiedostomuotoa taulukoiden tulostamisessa. CSV, *comma-separated values*, on yksinkertainen tekstitiedosto, mutta toimii taulukkotiedostona pelkän merkintätapansa vuoksi. Muoto on yleinen ja helposti luettavissa ohjelmallisesti ja ihmisen silmin. CSV-muoto on vanha, eikä sille ei ole tarkkaa määritelmää, vaikka käytännön ohjeita sen määrittelyyn on olemassa [13]. Taulukkolaskentaohjelmat osaavat avata CSV-tiedostot taulukkona samaan tapaan, kuin varsinaiset taulukkotiedostomuodot. Ratkaisulla vältettiin se, että varustamo edellyttäisi tiettyä tiedostomuotoa. Lisäksi CSV-muoto on niin yksinkertainen, että sen muokkaamiseen ei tarvita edes taulukkolaskentaohjelmaa, vaan tekstinkäsittelyohjelma riittää.

CSV-muodossa ensimmäinen rivi kuvaa pystysarakkeen nimen ja kaikki seuraavat rivit ovat arvoja sarakkeille. Koodiesimerkissä 5 on tapaus, jossa pyydetty listaus sisältää arvot ilmoittautumislomakkeen kentistä "etunimi", "sukunimi", "birthday", "school" ja "jasenyys".

```
1 "etunimi";"sukunimi";"birthday";"school";"jasenyys"
2 "Matti";"Muukalainen";"16.02.1988";"Metropolia";"x"
3 "Maija";"Meijäläinen";"23.08.1989";"Metropolia";"x"
4 "Teemu";"Testaaja";"17.12.1988";"Metropolia";"x"
```

Koodiesimerkki 5: Listausmuoto.

Sarakkeiden nimet tulevat ylläpitäjän määrittelemän ilmoittautumislomakkeen kenttien nimistä, ja sarakkeiden arvot ovat suoraan siitä, mitä käyttäjä on syöttänyt ilmoittautumislomakkeeseen. Lomakkeen kysymys jäsenyydestä on rasti ruutuun -kenttä, jonka hallintasovellus tulkitsee kirjaimeksi "x".

4.3 Sähköpostiominaisuudet ja laskutus

Kaljaasi-risteilyn järjestämisessä on aikaisempina vuosina vahvasti luotettu sähköpostiin perustuviin ilmoituksiin, joten uuden ilmoittautumissovelluksen tuli hallita vastaavat sähköpostitoiminnot. Käytöstä poistettu ilmoittautumissovellus muodosti ilmoittautumisen päätteeksi pitkän merkkijonon, josta tuli lähtevän sähköpostin sisältö. Tekstin muodostettuaan sovellus lähetti sähköpostin välittömästi. Sähköpostia ei kirjoitettu tietokantaan tai lähetetty ylläpitäjälle kopiona. Tapahtuman järjestäjällä ei siis ollut tositetta ilmoittautujalle lähetetystä laskusta eikä myöskään mitään tilastotietoa siitä, onko sähköpostien lähetys edes onnistunut. Maksuohje-sähköpostit ovat voineet mennä käyttäjän roskapostikansioon, eikä kummallekaan osapuolelle ole jäänyt maksuohjeen olemassaolosta merkintää.

Pakka-sovelluksen sähköpostinkäsittelyn tuli korjata vanhan sovelluksen puutteet. Sähköpostinkäsittelyä ei varsinaisesti kirjattu tuotetta määriteltäessä, mutta sähköpostit kirjaamalla saadaan samalla kerättyä tositteet laskuista. Kun käyttäjä vahvistaa varauksensa ilmoittautumisohjelman puolella, kirjoitetaan email_queue-tietokantatauluun taulukossa 4 esitetyn esimerkin mukainen rivi. *Signupid*-sarake mahdollistaa hallintasovellukselle sähköpostien hakemisen varauskohtaisesti. *Ok*-sarake kertoo, onko sovellus saanut lähetettyä sähköpostin onnistuneesti, ja *att*-sarake kertoo, kuinka monta kertaa sovellus on

yrittänyt sähköpostia lähettää. Mikäli *att*-sarakkeen arvo on 6 tai enemmän ja ok-sarake on yhä arvolla 0, ohjelma lähettää hälytyksen ylläpitäjälle. Sähköpostitietokantataulun tutkiminen ja sähköpostien uudelleenlähettämisyritykset tehdään palvelimella suoritettavalla cron-ajastusohjelmalla, jolle syötetään Pakka-sovelluksen komentosarjoja ohjeeksi.

Taulukko 4: Ote email_queue-tietokantataulusta.

id	signup id	recipient address	subject	body	created	ok	att
61	56	jukka@ test.net	Tervetuloa Kaljaasille!	Hei! Tervetuloa Kaljaasille! Alla näet maksuohj...	1410801815	1	1

Ilmoittautujille lähtevä ja tietokannan *body*-sarakkeeseen kirjoitettava sähköposti on HTML-muodossa pelkän merkkijonon sijaan. HTML-sähköposti mahdollistaa CSS-tyylimääritellyn viestin ja muuta viestin lukemista helpottavia HTML-merkintöjä, kuten taulukkoja. Esimerkiksi maksuohje on muodostettu ilmoittautujan vahvistettua varauksensa, joten sama HTML-taulukko voidaan näyttää ilmoittautumissivun viimeisellä sivulla sen lisäksi, että se liitetään lähtevään sähköpostiin. Hallintatyökalussa ilmoittautumisten hallinnan sivulla on hyttivarausten kohdalla "Re-Send Invoice" -nappi, jolla ylläpitäjä voi lähettää laskun uudelleen siltä varalta, että ilmoittautuja antaa palautetta, ettei ole saanut maksuohjeita.

5 Pakka-sovelluksen jälkituotanto

5.1 Tuotantoonviennin vaiheet

Pakka-sovelluksen tuotantoonviennillä oli peruuttamaton määräaika, koska Kaljaasi-risteilyä ei voitu siirtää. Tapahtumaan ilmoittautumisen alkamispäivämäärä oli julkisesti tiedossa, ja tapahtuman järjestäjän puolelta risteilyn järjestelyihin liittyvä logistiikka oli aloitettu. Ilmoittautumisen risteilylle tuli alkaa 18.9.2014, jolloin ohjelman tuli olla testattu ja käyttövalmis siten, että vähimmäistuote olisi toiminnoiltaan vähintään yhtenevä syrjäytetyn sovelluksen toimintojen kanssa. Huoli tuotantovalmiudesta oli suuri, sillä koko tapahtuman onnistuminen nojaa ilmoittautumisohjelman onnistumiseen. Mikäli sovelluksessa oli-

si vakavia ilmoittautumiseen liittyviä virheitä, se näkyisi negatiivisesti myyntituloksessa ja tapahtuman järjestäjän maineessa.

Syyskuun aikana luotiin Pakka-sovellukseen kaksi testiristeilyä testausta varten. Ensimmäinen testiristeily oli koeajo, jossa painopiste oli löytää käytettävyyteen liittyviä ongelmia ja odottamatonta käyttäytymistä. Kaikista tärkeintä oli saada tapahtumaan ilmoittautujien tiedot tallennettua oikein. Selvät poikkeamat ilmoittautumissivun käyttäytymisessä olisivat vihjanneet ongelmista palvelinlogiikassa. Lisäksi tietokantaan syötettävää tietoa monitoroitiin testauksen aikana.

Toisen testiristeilyn tavoite oli selvittää erikoistilanteita ja varmistaa käyttäjäkokemuksen toimivuus alusta loppuun. Kriittisiä toimintoja olivat muun muassa sähköpostin lähettämisen toimivuus ja ilmoittautujalle lähetettävän maksuohjeen paikkansapitävyys, etenkin ilmoittautumisen hinnan suhteen. Toisen testiristeilyn aikana löydettiin vielä vaarallisia virheitä, jotka olisivat vaikuttaneet ilmoittautumisohjelman toimintakykyyn. Kaikki vakavat ongelmat saatiin ratkaistua ennen todellisen tapahtuman ilmoittautumisen julkaisua. Sovelluksen toimintakyvyn kannalta merkityksettömät ominaisuudet siirrettiin jatkokehityssuunnitelmaan (liite 4).

Hallintapaneelin toiminta ei ollut kriittinen, koska ilmoittautumisen jälkeen oli vielä lähes kuusi viikkoa aikaa risteilyn alkamiseen, jolloin hallintapaneelia pystyttiin kehittämään ja korjaamaan matalan riskin aikoina altistumatta tilanteisiin, joissa sovelluksen tuotantovalmius olisi uhattuna. Lisäksi, mikäli ilmoittautumisohjelma toimii tarkoitetulla tavalla, kaikki ylläpitotoiminnot pystytään tekemään viimeistään käsin tietokantaa muokkaamalla.

Kaljaasi-risteilyn profiiliin on jo lähes perinteeksi muodostunut ilmoittautumissovelluksen ruuhkautuminen. Ruuhkautuminen on erikoinen haaste siksi, että Kaljaasi-risteilyn kotisivuja ja ilmoittautumisohjelmaa katsotaan lähes ainoastaan risteilylle ilmoittautumisen yhteydessä, kerran vuodessa. Ruuhkautumista voi estää palvelin, joka on suunniteltu kestämään ruuhkautumispiikkejä, mutta erikoispalvelimet ovat kalliita vain kerran vuodessa käytettäviksi.

Sovelluksen kehittämisen ensimmäisessä vaiheessa ei tehty toimenpiteitä ruuhkautumisen estämiseksi eikä sovellusta kehitetty ruuhkaan reagoimaan, vaan tärkeintä oli ohjelmiston yleinen toimivuus ja tiedon tallentaminen onnistuneesti. Vasta projektin toisessa vaiheessa, keväällä 2015, kehitetään toimenpiteitä ruuhkan estämiseksi. Tavoitteena on ilmoittautumista porrastamalla estää ruuhkan syntyminen kokonaan.

Risteilylle ilmoittautumisen alkamisajankohta oli 18.9.2014 kello 12.00. Tilaamalla uutiskirje oli mahdollista varata hytti risteilyltä kello 10.00, ennen varsinaisen ilmoittautumisohjelman julkistamista. Varsinaista ennakkovaraustoiminnallisuutta ei sovelluksen ensimmäisessä vaiheessa ole, vaan ennakkovaraus ratkaistiin siten, että osoite ilmoittautumisohjelmaan pidettiin salassa, kunnes se lähetettiin sähköpostitse uutiskirjeen tilanneille. Kello 12.00 lisättiin tapahtuman promootiosivulle *kaljaasiristeily.fi*-linkki ilmoittautumisohjelmaan, jolloin ilmoittautumisohjelma katsottiin julkaistuksi.

Jo muutamassa minuutissa pilvipalvelin osoitti heikkouden merkkejä, ja ensimmäinen tukipyyntö palveluntarjoajalle kapasiteetin lisäämisestä lähetettiin kello 12.10. Palveluntarjoajan asiakaspalvelun kanssa käytiin kirjeenvaihtoa noin tunti, kunnes palvelimella sijainneet sovellukset päätettiin siirtää tehokkaammalle palvelimelle. Kello 13.22 *kaljaasiristeily.fi*-osoitteeseen asetettiin HTTP 403 -käyttökielto, ja siirtäminen toiselle palvelimelle alkoi [14]. Siirto katsottiin valmiiksi kello 13.27, jolloin palveluntarjoaja ilmoitti uuden palvelimen tiedot. Käyttökieltoa ei voitu poistaa ennen uuden palvelimen tietojen vastaanottamista, koska ilmoittautumissovellus lähettää SMTP-protokollalla sähköpostia, jonka käyttö edellyttää uuden palvelimen tietojen asettamista Pakka-sovelluksen asetuksiin.

Sovelluksen asetuksien päivittämisen jälkeen HTTP 403 -kielto voitiin poistaa. Siirto-operaatiosta huolimatta sovellus käyttäytyi erittäin hitaasti. Yksi syy palvelun hitauteen oli resurssien jakaminen promootiosivun kanssa, joka on Wordpress-alustalla tehty blogi. Lisäksi vasta pilvipalvelun asiakaspalvelun kanssa käydyn keskustelun jälkeen selvisi, että palvelimella on mahdollista käyttää välimuistia www-sovelluksissa. Wordpress-sovelluksesta ja ilmoittautumissovelluksesta puuttui välimuistin tuki. Wordpress-sivuille välimuistituki on asennettavissa lisäosan avulla, mutta ilmoittautumissovellukseen välimuistituen rakentaminen vaatii suuria muutoksia sovelluksen arkkitehtuuriin.

Rasitetta palvelimelle antoi se, että jokainen käyttäjä ohjattiin ilmoittautumissovellukseen promootiosivun kautta. Jokainen käyttäjä avasi siis ensin Wordpress-blogin, jossa käyttäjälle näytettiin ilmoittautumiseen liittyviä yksityiskohtaisia tietoja. Tällä tapahtuman järjestäjät yrittivät varmistaa sitä, että ilmoittautujat tietävät, mihin ovat ilmoittautumassa. Esimerkiksi hyttien hinnat eivät olleet esillä itse ilmoittautumissovelluksessa (kuva 6). Vasta blogisivun avaamisen jälkeen käyttäjät pystyivät linkkiä klikkaamalla siirtymään ilmoittautumissovellukseen. Seurauksena oli suuri määrä turhia sivulatauksia, jotka kuormittivat kevyttä pilvipalvelinta.

Kuva 6: Uuden ilmoittautumissovelluksen etusivu.

Palvelimen kantokyvyn heikkouden pelättiin haittaavan sovelluksen myyntikykyä, mutta huoli osoittautui turhaksi. Risteily oli loppuunmyyty kello 18.00:aan mennessä ja hallintapaneeli raportoi noin 1 600 ilmoittautujasta. Palvelimen siirtäminen kesken ilmoittautumisen aiheutti muutamia erikoistapauksia, joissa hyttivaraukset rekisteröityivät tietokantaan vain osittain, mutta ilmoittautujiin saatiin yhteys ja tiedot saatiin korjattua.

5.2 Sovelluksen jatkokehitys

Pakka-sovellus toteutetaan kahdessa vaiheessa, joista ensimmäinen päättyi 5.11.2014 ja jälkimmäisen on määrä valmistua lokakuuhun 2015 mennessä. Ensimmäisen vaiheen aikana pyrittiin koko Pakka-sovelluskokonaisuudesta saamaan valmiiksi mahdollisimman suuri osa, erityisesti abstraktion ja arkkitehtuurin kannalta. Pyrkimyksenä oli minimoida ylimääräiseen suunnitteluun ja arkkitehtuurin muuttamiseen kuluvaa aikaa, jota toisen vai-

heen alussa todennäköisesti joudutaan tekemään. Tarkempi listaus ensimmäisen vaiheen jälkeen puuttuvista ominaisuuksista on liitteessä 4.

Ensimmäisenä toteutetaan uusi tietomalli henkilötietojen tallentamisesta. Yksi sovelluksen keskeisiä teemoja on henkilötietojen jatkuva käsittely. Henkilötietoja tulisi voida käsitellä objekteina muusta varausinformaatiosta erotettuna. Ylläpitäjällä tulee olla mahdollisuus muokata henkilötietoja kokonaisuuksina ja luoda uusia tietueita henkilöistä ilman ilmoittautumislomaketta. Lisäksi hyttivarauksiin liitettyjen henkilöiden vaihtamisen tulisi onnistua vain henkilötietuetta vaihtamalla.

Koska ensimmäisen vaiheen tavoite oli saada vain vuoden 2014 tapahtuma onnistumaan, ei sovelluksessa ole työkaluja tekstien muokkaamiseen. Sovellukseen tulee kehittää tekstimuokkain, jolla ylläpitäjä voi muuttaa sähköpostiviestejä ja ilmoittautumissivun ilmoituksia. Lisäksi ilmoittautumissovelluksen kaikki tekstit tulee voida kirjoittaa suomeksi ja englanniksi. Yii tukee yksinkertaista sanastotietokantaa, mutta käyttöliittymän kehittäminen sille saattaa osoittautua työlääksi, sillä sanasto-ominaisuutta ei ole tarkoitettu jatkuvasti muokattavaksi. Osana ylläpitäjän työkaluja tulee myös olla huoltotoiminnallisuudet, jotka mahdollistavat uusien tapahtumien järjestämisen ja ehkäisevät tietokannan kasvamista liian suureksi.

Viimeisin suuri kokonaisuus on käyttäjäkokemuksen viimeistely. Ensimmäisen vaiheen jälkeen ilmoittautumissovellus ei anna käyttäjälle riittävästi palautetta toiminnastaan. Ilmoittautumislomakkeen lähettämien POST-kutsujen aikakatkaisu esimerkiksi palvelimen ruuhkan vuoksi ei ilmene käyttäjälle mitenkään. Sovellus tarvitsee kunnolliset latausindikaattorit ja virheiden raportoinnin käyttäjälle. Sen sijaan hallintasovelluksen käyttäjäkokemuksella ei ole suurta merkitystä, mikäli toimintojen on testattu tekevän se, mitä pitää.

6 Johtopäätökset ja yhteenveto

Insinöörityönä tehtiin vuoden 2014 Kaljaasi-risteilyn tarpeita vastaava sovellus. Työn pääpaino oli tuotesuunnittelussa. Tuotesuunnittelun rinnalle syntyi ylimääräisiä haasteita aikatauluttamisesta ja Yii-sovelluskehykselle erikoisten teknisten ratkaisujen loppuunviennistä. Projektin aloittaminen myöhästyi hieman, minkä vuoksi jouduttiin tekemään kompromisseja teknisissä ratkaisuissa. Dokumenttitietokannasta luopuminen palvelintekniikan vuoksi oli suurin yksittäinen kompromissi, joka vaikeutti sovelluksen kehitystyötä jo alkuvaiheessa. Dokumenttitietokantaa vastaava toiminnallisuus saatiin kuitenkin toteutettua onnistuneesti tyydyttävällä tavalla.

Projektin ensimmäisen vaiheen aikana toteutettu hallintasovellus tuli lopulta paljon ennakoitua kattavammaksi. Suuri osa projektin toiselle vaiheelle suunnitelluista ominaisuuksista saatiin valmiiksi ja käyttöön jo ensimmäisen vaiheen aikana, mutta ominaisuuksilla ei ole kunnollista käyttöliittymää. Esimerkiksi lähteviä HTML-sähköposteja ylläpitäjä ei voi itse muuttaa, ennen kuin projektin toisessa vaiheessa valmistuvat muokkaimet sen mahdollistavat.

Toisen vaiheen ominaisuuksien toteuttaminen jo ensimmäisessä vaiheessa oli valintana riskialtis. Valinta perustui strategiaan, jolla pyrittiin vähentämään sovelluksen purkamista projektin toisen vaiheen aikana. Mikäli ominaisuuksissa olisi kitsasteltu ensimmäisen vaiheen aikana, olisi niiden asettaminen myöhemmin vaikeampaa. Yksi hallintasovelluksen kehitystyön tavoitteita oli modulaarisuus jatkokehitystä ajatellen, joten sovelluksen rakenteen yhtenäisyyden säilyttäminen ensimmäisen ja toisen vaiheen välissä vähentäisi huomattavasti työmäärää projektin toisessa vaiheessa. Täten riskinä kattavamman sovelluksen kehittämisen aikana olivat mahdollisesti liian myöhäisessä vaiheessa ilmenevät komplikaatiot, joilla olisi vaarannettu koko sovelluksen tuotantovalmius ilmoittautumissovelluksen julkaisupäivänä. Komplikaatioita ei ilmennyt.

Pakka-sovelluksen tuli ensisijaisesti mahdollistaa Kaljaasi-risteilyn järjestäminen vuonna 2014, missä onnistuttiin kiitettävästi, vaikka sovelluksen julkaisupäivään liittyvät yksityiskohdat lisäsivät huolia sovelluksen ja pilvipalvelun toimintakyvystä. Verrattuna käytöstä syrjäytettyyn sovellukseen Pakka-sovellus tarjoaa samat toiminnot, eli projektin ensimmäisen vaiheen voidaan katsoa onnistuneen. Projektin koko mittakaavan mukaan sovelluksen tulee mahdollistaa ilmoittautumislomakkeen hallinnointi täysin ilman sovelluskehittäjän tai muun ulkopuolisen konsultaation väliintuloa. Ensimmäisen vaiheen aikana saatiin toteutettua sovelluksesta niin suuri osa, että sovelluksen valmistuminen ajallaan ja suunnitellussa mitassa on mahdollista.

Lähteet

1 Kaljaasi-risteily. 2014. Verkkodokumentti. Metka opiskelijapalvelut oy. <http://www.kaljaasiristeily.fi/>. Luettu 1.10.2014.

2 Google Forms. 2014. Verkkodokumentti. Google, Inc. <http://www.google.com/forms/about/>. Luettu 19.10.2014.

3 Kaljaasi. 2013. Verkkodokumentti. Metka opiskelijapalvleut oy. <http://metka.metropolia.fi/kaljaasi/>. Luettu 21.10.2014.

4 MongoDB: Data models. 2014. Verkkodokumentti. MongoDB, Inc. <http://docs.mongodb.org/manual/data-modeling/>. Luettu 25.10.2014.

5 MongoDB drivers. 2014. Verkkodokumentti. MongoDB, Inc. <http://docs.mongodb.org/ecosystem/drivers/>. Luettu 25.10.2014.

6 About node.js. 2014. Verkkodokumentti. Joyent, Inc. <http://nodejs.org/about/>. Luettu 20.10.2014.

7 Bernardo, Pires. 2014. Rails vs Django: An in-depth technical comparison. Verkkodokumentti. <https://bernardopires.com/2014/03/rails-vs-django-an-in-depth-technical-comparison/>. March 25, 2014. Luettu 20.10.2014.

8 Wordpress.org. 2014. Verkkodokumentti. Automattic, Inc. <https://wordpress.org/about/requirements/>. Luettu 20.10.2014.

9 Bootstrap. 2014. Verkkodokumentti. Twitter, Inc. <https://github.com/twbs/bootstrap>. Luettu 25.10.2014.

10 Ullman, Larry. 2014. The Yii Book: Developing Web Applications Using the Yii PHP Framework. Revision 0.9. Self-Published.

11 Bertrand, Aaron. 2009. What is so bad about EAV, anyway?. Verkkodokumentti. SQLBlog.com <http://sqlblog.com/blogs/aaron_bertrand/archive /2009/11/19/what-is-so-bad-about-eav-anyway.aspx>. November 19, 2009. Luettu 20.10.2014.

12 Kochetov, Alexander. 2010. EAV behavior. Verkkodokumentti. GitHub. <https://github.com/yiiext/eav-behavior>. Luettu 20.10.2014.

13 Shafranovich, Y. 2005. Request for comments 4180: Common Format and MIME Type for Comma-Separated Values (CSV) Files. Verkkodokumentti. Network Working Group. <http://www.ietf.org/rfc/rfc4180.txt>. Luettu 1.11.2014.

14 Fielding, R. Berners-Lee, T. jne. 1999. Request for comments 2616: HTTP/1.1. Verkkodokumentti. Network Working Group. <http://tools.ietf.org/html/rfc2161> Luettu 18.11.2014.

Uusi Kaljaasi-sovellus, suunnittelutapaaminen, käyttötapausmuistiinpanot

Olen ylläpitäjä ja laivayhtiö haluaa minulta seuraavanlaisen listauksen

1. 17.10. listaus sen hetkisestä tilanteesta.
    a) Hytti kokoonpano
    b) hyttiluokka
    c) etunimi
    d) sukunimi
    e) synt.aika
    f) buffat/brekut
2. 24.10 listaus muutoksista: henkilömuutokset 17.10 jälkeen. (sekä uuden että vanhan matkustajan nimi)
    a) Uudet hyttivaraukset
    b) Peruuntuneet hyttivaraukset
3. 30.10 sama listaus 24.10 jälkeen tulleista muutoksista

Olen ylläpitäjä ja haluan tulosteen josta ilmenee

1. Listattuna hyttiluokittain:
2. Hyttikooonpano
3. Hyttiluokka
4. Etunimi
5. sukunimi
6. synt. aika
7. buffa
8. Breku
9. Jäsenyys

Olen ylläpitäjä ja kuoritusta varten haluan seuraavanlaisen tulosteen

1. Henkilöt aakkosjärjestyksessä
2. Buffetit kappalemäärä
3. Aamupalat kapplaemäärä
4. Opiskelijakunnan jäsenyys kyllä/ei

Muut

- Mikäli varustamo pakottaa ottamaan buffetin ja valitsemaan kattauksen: Olen ylläpitäjä ja haluan, että varausjärjestelmä näyttää myös loppuun myydyn buffan/buffa-ajan
- Olen ylläpitäjä ja haluan, että varausjärjestelmä ilmoittaa myös loppuun varatut hyttiluokat
- Olen ylläpitäjä ja haluan listauksen sähköpostiosoitteista
- Olen ylläpitäjä ja haluan että varausjärjestelmässä varaukset pysyvät listattuna oikeassa numeraalisessa järjestyksessä
- Olen ylläpitäjä ja haluan että järjestelmä ilmoittaa nimikaimoista ja järjestelmässä on työkalu, jolla etsiä kaimat tietokannasta
- Olen ylläpitäjä ja haluan, että tietokantaan tehtävistä matkustajamuutoksista henkilöt pysyvät hytti varauksissa paikoillaan
- Olen ylläpitäjä ja haluan, että varaus hetki kestä xx aikkaa, jolloin sessio umpeutuu
- Olen ylläpitäjä ja haluan, että järjestelmä vaatii sähköpostiosoitteen ilmoittamisen kahteen kertaan
- Olen ylläpitäjä ja haluan, että varausjärjestelmä pakottaa lukemaan säännöt ja infon
- Olen ylläpitäjä ja haluan että tietokanta eräpäivien osalta ja merkinnöiltä pysyy ennallaan

Vapaamuotoisia toivomuksia toiminnoista -tapaaminen, kesä

Muistiinpanot

- Luokka ja hytin henkilömäärä heti ensiksi käyttäjälle näkyviin
  - Myös näkyvillä loppuunmyydyt
- Pakka-ajatus, jotain muistiinpanoja:
  - kentän tyyppi, label, täsmenne
  - Visuaalisesti erittäin selvä
  - Muokattavat kentät
  - Validaattoreita:
    - required/not required
    - 2 kenttää pitää olla samaa
    - 2 kenttä ei saa olla samaa (eri korteissa)
    - Mikä tahansa regex
    - nimi+syntymäaika+sähköposti konflikti
      - tarkista myös lähetettävästä lomakkeesta
  - email delivery fault -kuuntelu
  - Yhteishinta-kenttä
  - Hinta-per-kortti -kenttä
  - Yks valinta per käsi
  - Aikaa muuttaa linkin takaa hytin detailsseja
- Tulosteet
  - per henkilö/kortti
  - per hytti/käsi
  - per pakka?/hyttiluokittain
  - Tulostetyökalu, järjestä x parametreilla y prioriteetilla
    - Tapahtumat aikavälillä
- Ylläpitoon:
  - Etsi yhteisiä duplikaattiarvoja lomakkeen tietokannasta

        * Kustomi valikko kannan sarakkeiden mukaan

- Temporaalinen tietokanta
- Pääkäyttäjälle kantadumpit kantaa tyhjennettäessä ja cronjobeilla
    - "haluatko varmasti pelata 52:ta"

  Tilastojen kerääminen

Tilattavan sovelluksen määrittely

Elevator pitch

Palvelun tarkoitus on tarjota mahdollisuus järjestää suurien ihmismäärien organisointia edellyttäviä tapahtumia. Tapahtuma voi olla esimerkiksi risteily, ulkoilmatapahtuma tai urheilutapahtuma. Palvelu eroaa yleisistä lomakesovelluksista tarjoamalla mahdollisuuden räätälöityyn ulostuloon ja tulosteisiin. Palvelu toteutetaan kahdessa vaiheessa. Ensimmäinen vaihe on käyttövalmis viimeistään 1.10.2014. Toinen vaihe on valmis viimeistään 1.10.2015.

Ensimmäinen vaihe

Palvelun ensimmäisen vaiheen toteuduttua palvelu mahdollistaa

- Palvelun käyttäjille
  - Tapahtumaan ilmoittautuminen ryhmässä
  - Automaattisen ilmoittautumisvahvistuksen
- Palvelun ylläpitäjille
  - Yhden tapahtuman ylläpidon
  - Tapahtumaan liittyvien tekstien muokkaaminen
  - Tulosteet

Toinen vaihe

Ensimmäisen vaiheen tuloksen lisäksi palvelu mahdollistaa

- Palvelun ylläpitäjille
  - Uusien tapahtumien luomisen
  - Tapahtumien tyylittelyn muokkaamisen
  - Ilmoittautumislomakkeen kenttien muokkaaminen
- Toisen vaiheen toteuduttua palvelun käyttäminen ei tulisi vaatia palvelun kehittäjää.

Jatkokehitysmuistiinpanot

Mutinaa

- Mitä METKA haluaisi, jos haluaisi?
- Tapahtuman formaattimuutos? ilmoittautuminen speksattava jotenkin porrastettavaksi, 0-budjetilla ilmopamahdusta ei voi estää
- Wordpress-sivuille on ylläpitäjien asennettava cachet

Mietintään

- Miten määritellään "henkilö"deckieditorissa eli mikä tekee esim "etunimi"ja "sukunimi"sarakkeparista merkityksellisemmän kuin "etunimi"ja "buffetit"sarakkeista
- Hae duplikaatit valituista sarakkeista
- Tapahtuman luonnissa määriteltäväksi "pohja"lomake, johon kaikki tulevat "deckit"perustuvat
- Matkustajamuutoksesta lähtevä varausvahvistus uudelle henkilölle?

Hallinta ToDo unsorted

- Varaajan nimi näkyviin ilmotauluun. (ID-numero puolestaan piiloon?)
- Pakkojen editoinnin user experience
- Sähköpostieditori
- Uuden tapahtuman luominen
- Ehkä: salli useampi tapahtuma samaan aikaan (poistamatta edellisiä)
- Ilmoittautumisdatan tyhjennys
- Mallipohja uudelle tapahtumalle

- Tulosteet: valitse avainsanat ihmislistaan
- Määritä standarditapa muutoksien laskutukseen
- Sähköpostin lähetyksen porrastus / cronjobit
- Tietokantadumppaus
- Ilmotaulu: nappi "poista"
- Ilmotaulu: ruksi "[x] näytä poistetut"
- Ehkä: Ilmotaulu: ilmojen editoinnin user experience
- Lisää mahtavia tilastoja etupaneeliin
- Järjestelmän asetukset: yläreunan ilmoitusten editori
- Järjestelmän asetukset: käännöseditori
- Selvistys käännöseditorin tekniikasta
- Järjestelmän asetukset: tuhoa 0-varaukset
- SMTP-logit sähköposteista ja feedbackaus käyttäjälle
- Deckille näkyvyys-kytkin ja mahdollisuus suoralinkittää

Ilmo ToDo unsorted

- Ennakkovaraustoiminnallisuus
- Vahvistussivu järkeväksi mobiililaitteilla
- Tarjoa lomakkeen palautusta palaavalle käyttäjälle
- Tuhoa palanneen käyttäjän 0-varaus
- Feedbackaus käyttäjälle: latausindikaattorit nappia painettua
- Feedbackaus käyttäjälle: järkeistä yläreunan ilmoituksia
- Hytti/henkilö hinnat näkyviin vahvistussivulle