KARELIA-AMMATTIKORKEAKOULU
Tietotekniikan koulutusohjelma

Joonas Litmanen, Ville Kyllönen

KENTTÄTYÖNOHJAUSJÄRJESTELMÄ, LÄMPÖKARELIA OY

Opinnäytetyö
Huhtikuu 2016

Karelia AMMATTIKORKEAKOULU	**OPINNÄYTETYÖ** **Huhtikuu 2016** **Tietotekniikka** Tikkarinne 9 80220 JOENSUU Keskuksen puhelinnumero

Tekijä(t)
Ville Kyllönen, Joonas Litmanen

Nimeke
Kenttätyönohjausjärjestelmä, Lämpökarelia Oy

Toimeksiantaja
Lämpökarelia Oy

Tiivistelmä

Tämän opinnäytetyön tarkoituksena oli jatkokehittää aiemmin opintojen aikana aloitettua työnhallintajärjestelmää Lämpökarelia Oy:lle. Sovellus toteutettiin web-pohjaisena.

Opinnäytetyö toteutettiin toimeksiantona paikalliselle LVI-yritykselle, Lämpökarelia Oy:lle. Sovellukselle oli tarve, koska entinen järjestelmä oli A3-kokoinen paperi seinällä, johon oli piirretty taulukko.

Opinnäytetyö oli pääasiassa toiminnaillinen. Työn teoria osuus koostuu pääsääntöisesti tekniikoiden valitsemisesta ja niihin perehtymisestä. Työ aloitettiin vuonna 2014 ja itse toiminnallinen osuus valmistui samana keväänä.

Kieli	Sivuja 25
suomi	Liitteet 1

Asiasanat

web-sovellus, javascript, php, kalenteri, mysql, ajax

THESIS
April 2016
Information technology

Tikkarinne 9
80220 JOENSUU
FINLAND
Telephone number of the centre

Author (s)
Ville Kyllönen, Joonas Litmanen

Title
Work management application, Lämpökarelia Oy

Commissioned by
Lämpökarelia Oy

Abstract

The purpose of this thesis was to further the development of the work management system for Lämpökarelia Oy. It was implemented as a web-based application.

The research and assignment was conducted under contract to a local company, Lämpökarelia Oy. The application was needed, because the former system consisted of an A3-sized paper on the wall with a chart on it.

The thesis was mainly functional. The theoretical part is mainly a summary of selected techniques and orientation of said techniques. The work was started in 2014 and the functional part was completed same spring.

Languages	Pages 25
Finnish	Appendices 1

Keywords

Web application, javascript, php, calendar, mysql, ajax

Sisältö

Liitteet

1 Johdanto

Tämän opinnäytetyön tarkoituksena on läpikäydä Lämpökarelialle luodun järjestelmän toiminallisuus ja antaa lukijalle käsitys käytetyistä tekniikoista ja menetelmistä. Lähtökohtana on, että lukija on perehtynyt ohjelmistokehitykseen ja ymmärtää webkehityksen periaatteet.

Teknisen kuvauksen lisäksi käymme läpi järjestelmän implementoinnin, josta ilmenevät käytetyt tekniikat, niiden toimintaperiaate sekä työssä käytetyt lisäkirjastot. Tämän jälkeen esittelemme sovelluksen käyttöliittymän sekä testausprosessin.

Opinnäytyön raportti on kirjoitettu työn valmistumisen jälkeen, minkä ansiosta pystyimme pohdinnassa keskittymään laajalti siihen, kuinka olisimme tehneet työn nykyisillä tiedoilla ja minkälaisia tekniikoita olisimme hyödyntäneet.

Opinnäytetyö tehtiin parityönä. Jokaisen version suunnitteluvaiheessa tehtiin yhteistyötä parhaan tuloksen takaamiseksi. Toteutusvaiheessa työ jaettiin puoliksi siten, että frontend-kehityksestä vastasi Joonas Litmanen ja backend-ohjelmoinnista Ville Kyllönen. Työnjako suunniteltiin siten, että frontendistä tulleet kutsut suunniteltiin mahdollisimman yksinkertaisiksi, jolloin niiden yhdistäminen backend-ohjelmistoon oli helppoa. Sivuston tyyli suunniteltiin mahdollisimman selkeäksi molempien kehittäjien yhteistyönä. Versiopäivitykset ja niiden testaus, yrityksen palvelimella, toteutettiin yhdessä.

2 Toimeksianto

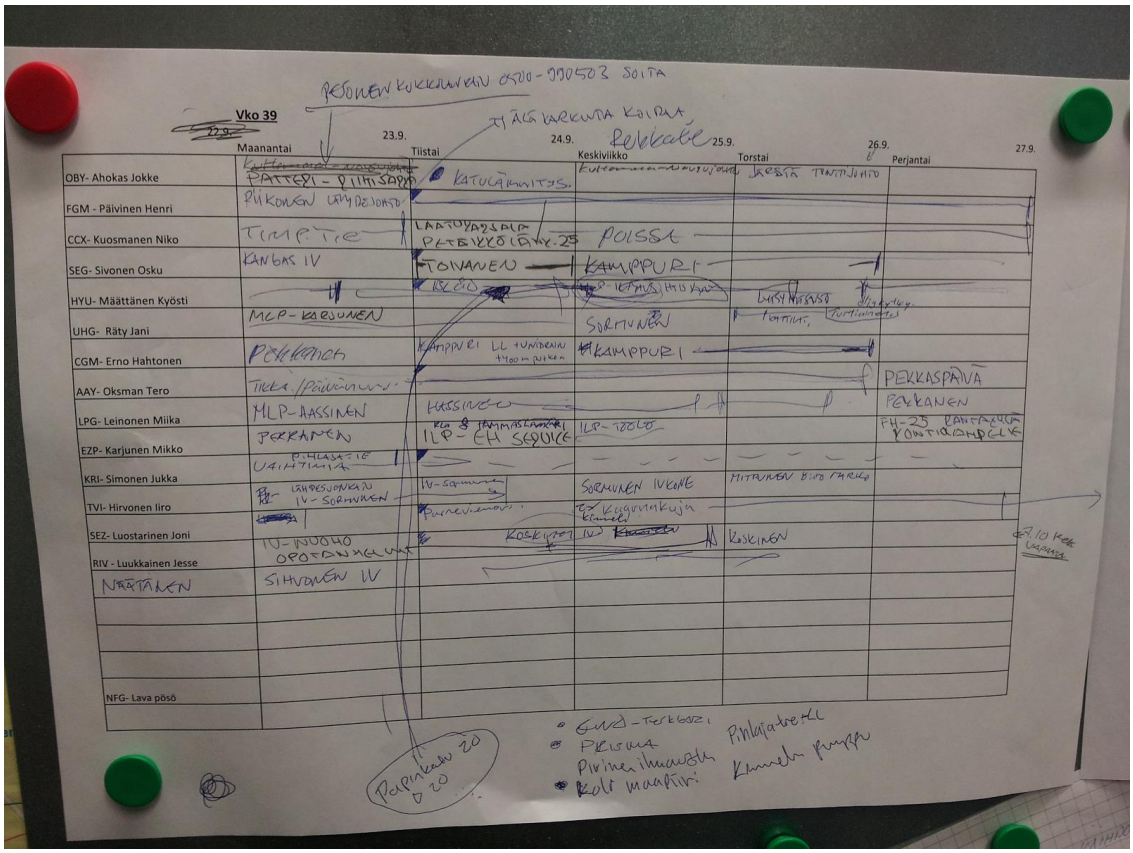
Opinnäytetyö saatiin toimeksiantona Lämpökarelia Oy:ltä. LVI-alan yritys on perustettu vuonna 1992 ja sillä on toimipisteet Joensuussa, Kuopiossa, Kiteellä ja Vantaalla.

Sovelluksen suunnittelu ja ohjelmointi aloitettiin jo Monialainen projektityö-opintojaksolla, johon Lämpökarelia Oy oli yhteistyöyrityksenä antanut tämän toimeksiannon.

Kurssilla aloitetun työn tarkoituksena oli korvata yrityksellä jo olemassa oleva järjestelmä, joka oli A3-paperille piirretty taulukko. Tähän taulukkoon työnjohtajat merkitisivät käsin viikon tapahtumat. Tämä ratkaisu täyttyessään muuttui sekavaksi ja vaikeutti näin yrityksen työnjohtoa. Uudella järjestelmällä pyritään helpottamaan ja selkeyttämään kenttätyön ohjausta yrityksessä.

Sovelluksen tarkoituksena ei ole ainoastaan korvata jo olemassa oleva järjestelmä, vaan myös luoda dynaaminen pohja tulevaisuuden ohjelmistoratkaisuille ja tuoda yrityksen työnohjaus tälle vuosituhannelle.

Toimeksiantajan vaatimukset olivat yksinkertaiset ja työn edistyessä lähdettiin rakentamaan alkuperäisen näkemyksen päälle lisää. Korvattava järjestelmä oli hyvin alkeellinen kynä ja paperi mentaliteettiä käyttävä taulu, johon lukujärjestyksistä tutulla taulukolla merkattiin päivittäiset työtehtävät, kenelle ne kuuluvat ja kuinka kauan ne kestävät (kuva 1).



Kuva 1. Alkuperäinen ratkaisu työnohjaukselle

Pienemmissä organisaatioissa tai täysin muuttumattomissa tilanteissa tällainen järjestelmä toimisi oikein hyvin ja olikin aikaisemmin työvoiman vähyyden johdosta ollut tarkoituksenmukainen.

Ongelmia esiintyy, kun asiat muuttuvat tai kun esimerkiksi työntekijöitä tulee lisää. Tästä syystä tulevan ratkaisun tulisi olla dynaaminen, jolloin ongelmatilanteita ei muuttuvien olosuhteiden johdosta synny.

Alkuperäinen vaatimusluettelo:

-Kalenterinäkymä - Autom. työn siirto
-Prioriteetit
-Drag-n-Drop

-Tehtävä (objekti)
-kohde
-prioriteetti
-ajastettu muistutus
-asiakas (tiedot)

3 Käytetyt tekniikat

Projektin alussa lähdettiin selvittämään, onko työnohjauksen toteuttamiseen jo olemassa olevaa ohjelmallista ratkaisua. Jo alkuvaiheen selvityksessä kävi ilmi, että vaikka ratkaisu itsessään vaikuttaa yleismaalliselta sekä yksinkertaiselta, ei räätälöityä ratkaisua juuri toimeksiannossa olevaan ongelmaan löytynyt.

Pohjatyön tuloksena löydettiin kuitenkin kehityskirjasto, jota hyödyntämällä vaatimuksiin pystyttiin vastaamaan niiden vaatimalla tavalla.

3.1 Tekniikan valitseminen

Oli alunperin selvää, että tekniikka sekä työnohjausjärjestelmä tultaisiin rakentamaan web-sovellukseksi. Tällä tavalla varmistimme järjestelmän toimivuuden mahdollisimman monella laitteella kokoonpanosta, valmistajasta tai käyttöjärjestelmästä riippumatta.

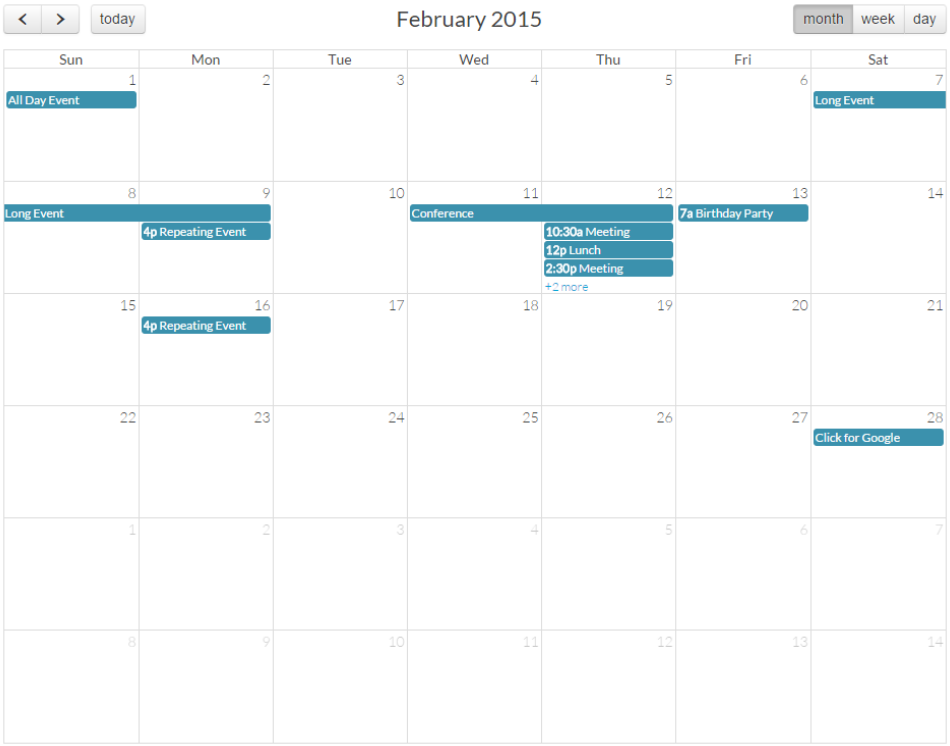
Sovelluksen oli tarkoitus tulla näkyviin vain yrityksen lähiverkkoon. Verkkosovelluksena järjestelmä vaatii toimiakseen palvelimen, joka kykenee toteuttamaan yksinkertaisen Apache, MySQL ja PHP ympäristön. Koska suurin osa työstä haluttiin toteuttaa käyttäjän puolella, Javascript ja sen lisäkirjastot tulisivat olemaan päätekniikka projektissa.

3.2 FullCalendar

FullCalendar on ilmainen, vapaa ja monipuolinen Javascript-kirjasto, joka on tehty verkossa toimivien kalenterisovellusten pohjaksi. FullCalendarin on kehittänyt amerikkalainen Adam Shawn[1]. Kirjaston tärkeimpiä ominaisuuksia ovat monipuolinen muokattavuus sekä niin kutsuttu ”drag 'n' drop” toiminnallisuus, joka oli yksi tärkeimpiä vaatimuksia dynaamisen kalenterin kannalta.

Tällä hetkellä FullCalendarista on julkaistu jo 2.3.1-versio (8.3.2015), mutta työssä käytetty versio pohjautuu aiempaan 1.0 versioon.

Muista tärkeistä ominaisuuksista mainittakoon AJAX-yhteensopivuus, jota tullaan tarvitsemaan kalenterisovelluksen tietokantayhteyden luonnissa. FullCalendar-kirjasto valittiin myös, koska se toimii MIT-lisenssin alaisuudessa, joten sen muokkaaminen, uudelleenhyödyntäminen sekä jopa sen kaupallinen käyttö on sallittua vapaasti ja ilmaiseksi. Alla olevassa kuvassa Fullcalendarin perusnäkymä.



Kuva 2. FullCalendar-kirjaston perusversio

3.2.1 Resource view

Jarno Kurlinin luoma Resource view –näkymä[2] hyödyntää nimenomaista laajennettavuutta lisäten FullCalendariin “resurssinäkymän”, jolloin Kalenterio minaisuuksien lisäksi näkymään voi lisätä resursseja (tässä tapauksessa työntekijöitä) ja antaa näille omat sarakkeet erottelua varten, jolloin kalenteri muuttuu resurssienhallintaan soveltuvaksi aikatauluksi.

3.2.2 MIT-lisenssi

MIT-lisenssi on alunperin amerikkalaisen Massachusetts Institute of Technologyn (tästä lähtien MIT) tuottama vapaa ohjelmistolisenssi. Lisenssi oikeuttaa käyttäjän vapaasti muokkaamaan, kopioimaan tai käyttämään teosta omissa projekteissaan, ehtona ollen, että lisenssi säilyy teoksen mukana. Lisenssi ei velvoita tekijäänsä julkaisemaan lähdekoodia. (liite 1)

3.3 Javascript ja olennaiset kirjastot

JavaScript on Netscape Communications Corporationin tekemä skriptikieli, sen ensimmäinen versio julkaistiin vuonna 1995 ja tämänhetkinen vakaa versio on 1.8.5. Se on prototyyppipohjainen olio-ohjelmointikieli ja sitä käytetään luomaan verkkosivuille dynaamista sisältöä.

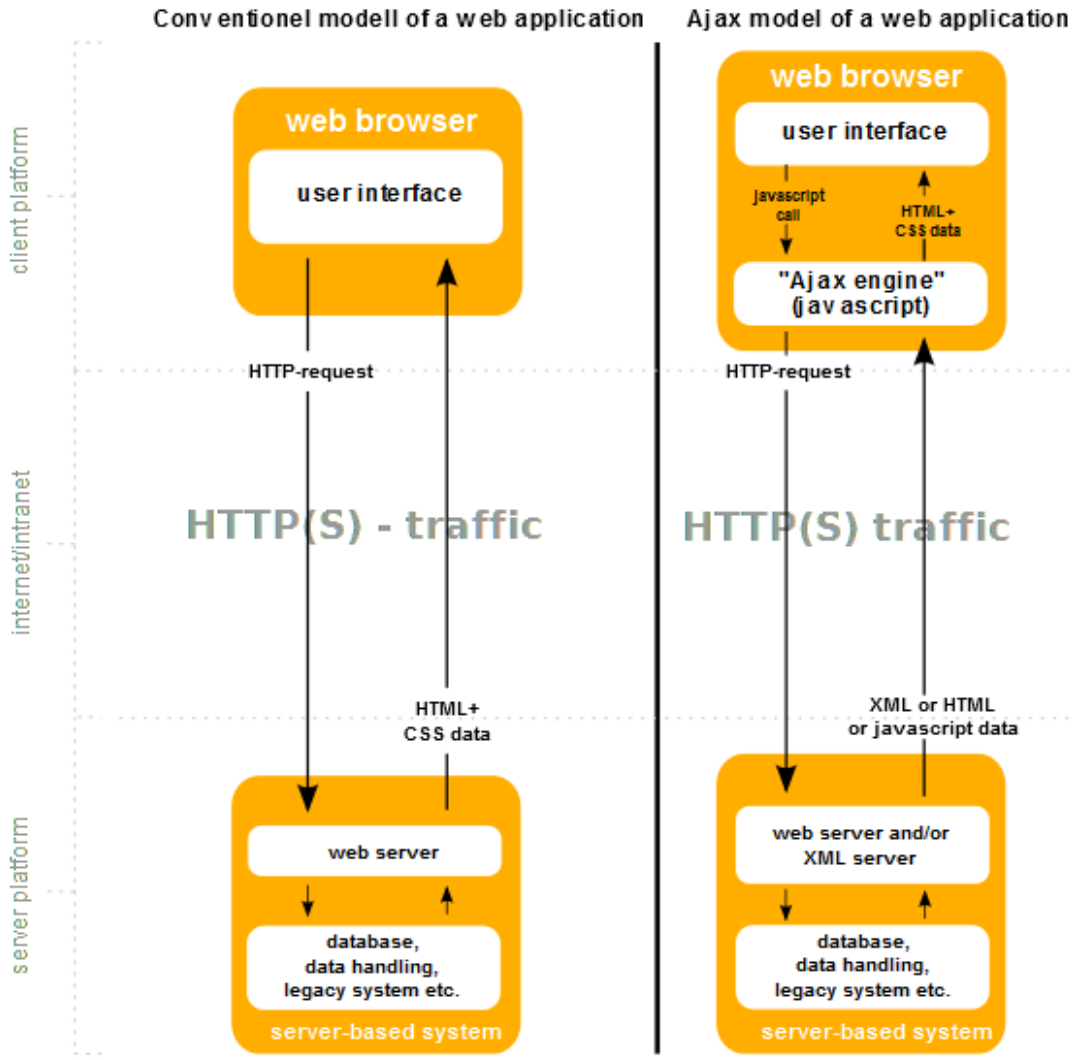
Javascript toimii osana verkkoselainta ja skriptikielenä se antaa työkalut asiakasohjelman, eli selaimen puoleiseen interaktiivisuuteen, sisällön dynaamisen muokkaukseen sekä asynkronoituun kommunikaatioon, johon projektissakin käytetty AJAX antaa lisää toiminnallisuutta.

Javascriptillä voidaan toteuttaa erilaisia ominaisuuksia verkkosivuilla yksinkertaisista tyyliin vaikuttavista animaatioista aina projektissakin tarvittuun tietokantakutsujen ohjaukseen ja kokonaisen kalenterilisäkirjaston ajamiseen.

3.3.1 AJAX

AJAX on kokoelma tekniikoita, jotka toimivat kokonaisuutena verkkosivustolla. AJAX-kokonaisuudella kyetään asiakasohjelman sisällä suorittamaan ajon aikana tehtäviä, jotka yleensä vaatisivat sivuston uudelleen latauksen tai sieltä poistumisen. Tällaisia tehtäviä voivat olla esimerkiksi tietokannasta haku tai sinne tallennus, sähköpostin lähetys tai sen väliaikainen tallennus.

AJAX yhdistää yleensä erillään olevat tekniikat yhteen. Sitä käytetään HTML ja CSS tiedon näyttämiseen sekä tyylittämiseen, Document Object Model (DOM) tiedon manipulointiin, XML:llää tiedon siirtämiseen sekä Javascriptiä tämän kaiken yhdistämiseen.



Kuva 3. Websovelluksen perinteinen sekä AJAXia hyödyntävä ratkaisu

AJAXin yhdistämät tekniikat toimivat ikäänkuin puskurina tiedon näyttämisen, käsittelyn sekä sen varastoinnin välillä, luoden käyttäjälle dynaamisen ja reaaliaikaisen sivustokokonaisuuden.

3.3.2 JQuery

JQuery on maailman suosituin, ilmainen Javascript-kirjasto. Se on kokoelma funktioita, joiden tarkoitus on koota yhteen yleisesti käytettyjä Javascriptin ominaisuuksia yksinkertaisessa ja tehokkaassa muodossa. Nämä funktiot toimivat Javascript-kehittämisen tukena.

JQueryn käytetyimpiin ominaisuuksiin kuuluvat nimenomaan verkkosivujen toimintojen käsittely, elementtien animaatiot, valitseminen ja manipulointi. JQuery tukee myös AJAX-kirjastoa ja helpottaa sen käyttöä tuomalla käyttöön tehokkaat funktiot tätä varten.

3.4 PHP

PHP eli PHP: Hypertext Preprocessor on serverillä käännettävä komentosarjakieli, jota käytetään dynaamisen sisällön luomiseen web-sivustoilla. Lisäksi se on alustariippumaton, eli sitä voidaan käyttää missä tahansa käyttöjärjestelmässä.

PHP:n kehitys alkoi vuonna 1994, kun tanskalais-grönlantilainen Rasmus Lerdorf julkaisi kokoelman C-kielellä kirjoitettuja skriptejä [3]. Sen ensimmäinen versio julkaistiin jo vuonna 1995 nimellä PHP/FI (Personal Home Page / Forms Interpreter). Viimeisin vakaa julkaistu versio on 5.6.7 ja se julkaistiin maaliskuussa 2015.

Muuttujat ovat PHP:ssa heikosti tyypitettyjä. Tämä tarkoittaa sitä, että muuttujat voivat saada minkä tahansa arvon, näitä arvoja ovat

- boolean (totuusarvo)
- integer (kokonaisluku)
- float/double (liukuluku)
- string (merkkijono)
- array (taulukko)
- Object (olio)
- Resource (ulkoinen resurssi)
- null (tyhjä)

Jo PHP4 sisälsi tarvittavat toiminnot olioiden käsittelyyn, mutta PHP5 uudisti kaiken ja siinä olioiden ominaisuuksia ovat muun muassa public, private ja protected. Julkiset (public) muuttujat olioissa voidaan lukea mistä tahansa ohjelman osasta, yksityisiä (private) voi lukea vain kyseisestä luokasta ja suojatut ovat julkisia kyseessä olevan luokan alaluokille, mutta muille ohjelman osille ne ovat yksityisiä.

3.5 MySQL

MySQL on tietokantaohjelmisto. Sen ensimmäinen versio on luotu vuonna 1995 ja julkaistu vuonna 1996 . Sen ovat kehittäneet suomalainen Michael Widenius

ja ruotsalainen David Axmark. [4] Sitä kehittää ruotsalainen MySQL AB, joka nykyisin on Oracle Corporation:n omistuksessa[5].

MySQL on hyvin suosittu web-palveluissa tietokantana vakautensa ja nopeutensa ansiosta. Siitä on olemassa GNU-GPL lisensioitu sekä kaupallinen versio.

4 Sovellus

Sovellus on verkkopohjainen, LAMP-serverille asennettu, edellä selvitetyillä tekniikoilla toteutettu työnohjausjärjestelmä. Sen kehitys aloitettiin marraskuussa 2013 ja ensimmäinen versio asennettiin Lämpökarelia Oy:n käyttöön joulukuussa 2013. Samalla asennettiin käyttöön yrityksen tiloihin sovellukselle dedikoitu palvelin. Tämän jälkeen kehitysversioita päivitettiin yritykseen kuukauden välein sovitun ajan puitteissa.

Lähtökohtana oli pitää sovellus mahdollisimman yksinkertaisena ja noudattaa Lämpökarelialta saatuja vaatimuksia. Sovellusta lähdettiin toteuttamaan niin, että siirtäisimme olemassa olevan ratkaisun ohjelmalliseen muotoon noudattaen mahdollisuuksien mukaan vanhan järjestelmän yksinkertaista ja tuttua toimintaa. Sovelluksen toiminta kyettiin säilyttämään alkuperäistä mukailevana FullCalendarin ja resourceView lisäosan avulla.

Sovelluksen käyttöön Linux-palvelimelle asennettiin LAMP-kokonaisuus, joka toimi sovelluksen pohjana, ja jota hyödynnettiin sovelluksen tietokannan ja taustaprosessien hallinnassa.

4.1 FullCalendar-kirjaston implementaatio

Kirjaston ominaisuuksista keskeisin, kalenteri, on pohja sovelluksemme ulkonäölle sekä toiminnallisuudelle. Kirjasto on kokonaisuudessaan Javascriptillä toteutettu, jolloin kykenemme hyödyntämään kielen ominaisuuksia kehityksessä.

Koska FullCalendar on itsessään pelkästään kalenteripohja, jolle annetaan arvoina asetuksia, projektissa hyödynnetään Javascriptin ja sen kirjastojen laajennettavuutta. Näihin laajennuksiin kuuluvat muun muassa AJAX:in sisällyttäminen kalenterin toimintaan, jolloin kalenterin tietueet voidaan saattaa tietokantaan. Kirjastossa itsessään oli valmiiksi ominaisuus kuunnella kalenterissa tapahtuvia muutoksia, esimerkiksi klikkaus- , siirto- tai luonti-tapahtumia. Näillä tapahtumankäsittelyfunktioilla tässä tapauksessa ohjattiin tilanteen vaatimaa AJAX-kutsua. FullCalendarin, AJAX:in ja tietokannan yhdistämiseen löydettiin valmis implementaatio, jota hyödynnettiin tässä projektissa.

ResourceView-lisäkirjasto muuttaa kalenterin toiminnallisuutta siten, että kalenteri rivitetään sen mukaan, kuinka monta resurssia kulloinkin on asetettu. Tässä järjestelmässä resurssit ovat työntekijöitä, jolloin tapahtumat voidaan osoittaa tietylle työntekijälle, eli resurssille.

Koska FullCalendar on Javascript-kirjasto, se ajetaan ennen kuin verkkosivu näytetään. Alussa FullCalendar katsoo saamansa asetukset läpi ja valmistelee kalenterin pohjautuen näihin arvoihin.

Tässä tapauksessa kalenterin tietueet tulevat tietokannasta, joten tässä kohtaa FullCalendar tekee AJAX-kutsun tietokantaan yhdistävälle PHP-funktiolle, joka hakee kaikki kalenterimerkinnät ja tuo sen kalenterille. Sen jälkeen FullCalendar luo saamastaan datasta tietueet kalenteriin. Vähimmäisvaatimus tapahtumille on resurssin ID, alku- ja loppupäivämäärä, allDay ja otsikko. allDay on kalenterin sisäinen muuttuja, joka kertoo kestääkö tapahtuma koko päivän vai N määrän tunteja. Koska kalenterin lisäksi käytössä on resourceView-lisäkirjasto, tarvitaan kalenterin perustietueiden lisäksi numeerinen arvo, joka kertoo mille resurssille tapahtuma osoitetaan. Kalenterin sekä resourceView kirjastojen vaatimien kenttien lisäksi tietokantaan tallennetaan tapahtuman prioriteetti ja lisätiedot. Näiden tietojen pohjalta kalenteri generoi kalenterinäkymän sivulle.

Tämän jälkeen kalenteriin lisättiin bootstrapista dialogit. Dialogien tarkoituksena on toimia kalenterin ja tietokannan välikätenä sekä lisätä käyttäjäystävällisyyttä. Dialogit tukevat tällöin myös jatkokehitystä, jolloin tietueiden lisäys, poisto tai muokkaus olisi suoraviivaisempaa.

4.1.1 Käyttöönotto

FullCalendarin initialisointi tapahtuu init.js-tiedostossa. Aluksi kalenterille asetetaan sen tarvitsemat päivämäärämuuttujat sekä luodaan itse kalenterimuuttuja.

```
var date = new Date();
var d = date.getDate();
var m = date.getMonth();
var y = date.getFullYear();
var calendar = $('#calendar').fullCalendar({ kalenterin asetukset ja funktiot }):
```

Tämän jälkeen kaikki toiminnallisuus ja funktiot asetetaan kalenterille. Ensiksi sovellus vaatii perustietoja, kuinka kalenteri toimii, kuinka sitä voidaan käsitellä ja mistä kalenterin tarvitsemat tiedot haetaan.

```
table: true,
```

Asetetaan kalenteri taulukkomuotoon

```
selectable: true,
```

Asetaan kalenteri hyväksymään drag ’n’ drop toiminnot

editable: true,

Asetetaan kalenteri hyväksymään muokkaustoimenpiteet

selectHelper: true,

Ulkoasumuuttuja, näyttää raahausta helpottavan “varjon”

defaultView: 'resourceWeek'

Kalenterin oletusnäkymä, Tässä tapauksessa resurssinäkymä ulkoista kirjastoa hyödyntäen.

allDayDefault: true,

Asettaa allDayDefault muuttujan arvoksi “true”.

events: "functions/events.php",

Asetetaan kalenterin tapahtumat, tässä tapauksessa ne haetaan tietokannasta events.php tietokantakutsulla.

resources: "functions/resources.php",

Asetetaan kalenterin resurssit eli työntekijät, tässä tapauksessa ne haetaan tietokannasta resources.php tietokantakutsulla.

Seuraavaksi muotoillaan kalenterin otsikkorivi.

```
header: {
                left: 'prev,next today',
                center: 'title',
                right: 'resourceWeek,resourceNextWeeks',
},
```

Vasemmalle osiolle laitetaan kalenterin navigaationäppäimet, joilla pääsee viikoissa eteen ja taaksepäin, sekä takaisin nykyhetkeen. Keskelle pääotsikko, joka sisältää kuukauden nimen sekä aikavälin, joka näytetään. Oikealla on lisää navigaationäppäimiä, joista voi valita viikkonäkymän sekä seuraavan kuukauden tapahtumat.

Kun tapahtuma luodaan, lähetetään tapahtuman kalenteri ja resurssitiedot kalenterista dialogiin. Dialogissa lisätään puuttuviin tietueisiin halutut arvot, jonka jälkeen syötettyjen arvojen pohjalta laaditaan AJAX-kutsu PHP-funktiolle. Kyseinen funktio lähettää arvot tietokantaan.

```
select: function(start, end, allDay, jsEvent, view, resource) {
        var start = $.fullCalendar.formatDate(start, "yyyy-MM-dd 00:00:00");
        var end = $.fullCalendar.formatDate(end, "yyyy-MM-dd 01:00:00");
        var reso = resource.id;

        $(function() {
                $( "#dialog2" ).dialog();
                document.getElementById('start').value = start;
                document.getElementById('end').value = end;
                document.getElementById('resource_id').value = res_id;
```

```
		});

		calendar.fullCalendar('renderEvent',
		{
			title: title,
			start: start,
			end: end,
			allDay: allDay,
			resource: resource.id
		},

		true // make the event "stick"
);

calendar.fullCalendar('unselect');
```

Kun tapahtumaa klikataan, kalenterin klikkauksen tapahtumakäsittelyfunktio avaa dialogin, hakee kyseessä olevan tapahtuman tietueet ja asettaa ne dialogissa niille osoitettuihin kenttiin. Dialogista poistuttaessa kalenteri tekee AJAX-kutsun PHP-funktiolle, joka tallentaa muutetut tietueet tietokantaan.

```
eventClick: function ( event, jsEvent, view )  {
		$(function() {
			$( "#dialog" ).dialog();
			var otsikko = event.title;
			var startti = event.start;
			var id = event.id;
			var info = event.info;
			var prior = event.priori;

			document.getElementById('result2').value = otsikko;
			document.getElementById('starter').value = startti;
			document.getElementById('del').value = id;
			document.getElementById('ident').value = id;
			document.getElementById('infor').value = info;
			document.getElementById('priority').value = prior;

		});
},
```

Kun tapahtuman ajankohtaa muutetaan sen kokoa muuttamalla tai sitä raahaamalla, koonmuokkauksen tai vaihtoehtoisesti pudotuksen tapahtumakäsittelyfunktio hakee kyseisen tapahtuman uuden aloitus- ja lopetuspäivämäärän ja lähettää ne AJAX-kutsuna PHP-funktiolle, joka tallentaa muutokset tietokantaan.

```
eventResize: function( event, dayDelta, minuteDelta, allDay, revertFunc, jsEvent, ui, view ) {
		var start = $.fullCalendar.formatDate(event.start, "yyyy-MM-dd HH:mm:ss");
		var end = $.fullCalendar.formatDate(event.end, "yyyy-MM-dd HH:mm:ss");

		$.ajax({
			url: 'functions/update_events.php',
			data: '&start='+ start +'&end='+ end +'&id='+ event.id,
			type: "POST",
		});
},
```

4.2 MySQL ja PHP implementaatio

Koska sivuston sisältö rakentuu lähes kokonaan Javascriptin avulla tavanomaista HTML-pohjaa lukuunottamatta, PHP:n päärooli sivustolla oli taustaprosessien ohjauksessa, näistä keskeisimpänä tietokantakutsujen ohjauksessa. Tästä johtuen MySQL-implentaatio on toteutettu PHP:n avulla, pohjautuen tarvittaviin AJAX-kutsuihin. Sivuston tapauksessa tietokantakutsut ovat jaoteltu kukin omaan tiedostoonsa, joista ne kutsutaan tapahtumakäsittelyfunktiolla.

Funktiokutsut rakentuivat kaikki samalla kaavalla lukuunottamatta itse tietokantakutsua. Ensimmäiseksi kutsu sai pääsovellukselta lähetetyt arvot muuttujiin, jossa muuttuja saa arvon $_POST tapahtumalta, kun pääohjelma lähettää POST kutsun PHP-funktiolle.

```
$muuttuja = $_POST['arvo'];
```

Seuraavaksi tietokantakutsu yrittää avata yhteyden tietokantaan ja palauttaa virheen, jos tämä ei onnistu.

```
try {
$bdd = new PDO('mysql:host=localhost;dbname='tietokannan_nimi', 'käyttäjä', 'salasana');
} catch(Exception $e) {
exit('Unable to connect to database.');
}
```

Tähän kohtaan asetetaan MySQL tietokantaserverin sijainti verkossa, tietokannan nimi sekä käyttäjätiedot. Try ja catch ehtolauseke pitää huolen virheistä, jos esimerkiksi tietokantaan ei saada yhteyttä tai käyttäjätiedot ovat väärät. Tässä tapauksessa palautetaan sovellukselle, eli sivustolle, palaute, ettei tietokantaan yhdistäminen onnistunut.

Jos kaikki onnistuu tämän jälkeen tietokantakutsu valmistelee tietokantakutsussa määritetyn tehtävän ja lähettää sen tietokantaan.

```
$sqlLause = "UPDATE taulu SET kentta0=?, kentta1=?, kentta2=? WHERE tunniste=?";
$q = $bdd->prepare($sqlLause);
$q->execute(array(muuttuja0,$muuttuja1,$muuttuja2,$tunniste))
```

Tässä kohtaa muuttujaan $sqlLause tallennetaan tietokantakutsuun tarvittava tehtävä. Esimerkissä päivitämme “taulu” nimisestä taulusta kentät 0,1 ja kaksi muuttujien 0,1 ja 2 sisältämillä arvoilla jossa taulun kenttien tunniste on X. Tämä tarkoittaa, että suorituksen jälkeen taulun tiedot näiltä osin ovat muuttuneet tunnisteen X kohdalta.

Tämän jälkeen $q muuttujaan yhdistetään avattu tietokantayhteys ja sinne lähetettävä sql-lause. Tässä kohtaa myös lause “valmistellaan” prepare() -

funktiolla, jolloin käyttäjältä tullut syöte tarkastetaan virheiden ja vaarallisen koodin varalta.

Tämän jälkeen lause suoritetaan execute() -funktiolla, jolle annetaan sql-lauseen tarvitsemat arvot sen vaatimassa järjestyksessä.

Tämän jälkeen ohjelma palaa takaisin pääohjelmaan, eli etusivulle.

```
header('Location: ../');
exit();
```

Seuraavassa kohdassa käymme läpi eri funktioiden tietokantakutsut ja miten niitä käytettiin.

Events.php hakee kaikki tapahtumat kalenteriin sivustoa ladattaessa. tietokantakutsussa ladataan tietokannasta “evenements” taulun kaikki tiedot FullCalendarin käytettäväksi tietokantakutsulla.

```
"SELECT * FROM evenement ORDER BY id"
```

Tietokantakutsu valitsee kaikki taulusta “evenement” ja järjestää ne tunnistekentän mukaan.

Resources.php hakee kaikki resurssit, eli työntekijät kalenteriin sivustoa ladattaessa.

```
"SELECT * FROM resources ORDER BY ID DESC"
```

Tietokantakutsu valitsee kaikki taulusta “resources” ja järjestää ne tunnistekentän mukaan laskevaan järjestykseen. Tällöin näytettävät resurssit ovat aikajärjestyksessä, jolloin vanhin työntekijä on ensimmäisenä ja uusin viimeisenä.

Drop_events.php vie tietokantaan tapahtuman uuden aloitus- ja lopetuspäivämäärän sekä työntekijän tiedot, kun tapahtuma raahataan eri kohtaan.

```
"UPDATE evenement SET start=?, end=?, resource=? WHERE id=?"
```

Tietokantakutsu päivittää “evenement” taulun tietueen, eli työtehtävän, aloitus- sekä lopetusajankohtaa tietueen tunnistekentän perusteella. Tämä kutsutaan, kun työtehtävä raahataan ja pudotetaan kalenterissa uuteen kohtaan.

Update_events.php vie tietokantaan tapahtuman uuden aloitus- ja lopetuspäivämäärän,

```
"UPDATE evenement SET start=?, end=? WHERE id=?"
```

Edellinen lause päivittää uuden arvon tauluun “evenement” tunnistetietueen mukaan. Tämä kutsutaan, kun tapahtuman kestoa muutetaan sitä pidentämällä tai lyhentämällä kalenterissa.

del_event.php tapahtuman poisto tietokannasta.

```
"DELETE FROM evenement WHERE id=?"
```

Kutsu poistaa taulusta “evenement” tunnistetiedon näyttämän kentän.

Mod_event.php vie tietokantaan tapahtuman tiedot, kun tapahtumaa muokataan muokkausdialogin kautta.

```
"UPDATE evenement SET title=?, info=?, priori=? WHERE id=?"
```

Tietokantakutsu päivittää “evenement” taulun arvoja title, info ja priori. Tämä kutsutaan, kun jo luodun tapahtuman tarkempia tietoja muutetaan.

Add_events.php luo tietokantaan uuden tapahtuman ja antaa arvoksi sille annetut tietueet.

```
"INSERT INTO evenement (title, start, end, resource, info, priori) VALUES (:title, :start, :end, :resource, :info, :priori)"
```

Tässä kohtaa tietokantakutsu on rakennettu eri tavalla. Tämä kutsu lisää tauluun “evenement” luodun tapahtuman tiedot title, start, end, resource, info ja priori. Nämä ovat FullCalendarin vaatimia tietueita tapahtuman esitystä ja tallennusta varten.

Add_personel.php-kutsu lisää tauluun “resources” uuden työntekijän.

```
"INSERT INTO resources (name) VALUES (:name)"
```

Kalenteritietueiden lisäksi tietokantaan voidaan lisätä ja sieltä voidaan poistaa työntekijöitä sivuston kautta. Tässä tapauksessa tietokantaan tehdään kaksi kutsua. Ensin työntekijälle osoitetut tapahtumat poistetaan kalenterin tietokannasta jonka jälkeen itse työntekijä poistetaan työntekijöille osoitetusta tietokannasta. Työntekijää lisätessä luodaan työntekijöille osoitettuun tietokantaan uusi työntekijä.

4.2.1 Tietokanta

Tietokanta koostuu useammasta taulusta, osa tauluista on alkuperäisen FullCalendar-kirjaston vaatimia perustietueita ja osa myöhemmin sisällytettyjen ominaisuuksien tietueita. Sovelluksen tietokanta rakentuu seuraavista tauluista:

- Evenement
- fc_hourtable
- login_attempts
- members
- resources
- salary_opt

Evenement-taulu sisältää FullCalendarin tarvitseman datan, jotka pitävät sisällään kalenterin tapahtumien tunnistetiedot, tässä tapauksessa työtehtävät. fullCalendar vaatii perustiedoiksi tapahtuman tietokantatunnisteen, otsikon, tapahtuman alku sekä loppu ajankohdan ja kestääkö tapahtuma koko päivän. Tämän lisäksi sovelluksen vaatimia kenttiä ovat resurssitunniste, työntekijän henkilökohtainen tunnistearvo, tehtävän prioriteetti ja lisätietokenttä.

Resurssitunniste kertoo sovellukselle kenelle työntekijöistä tapahtuma on osoitettu. Työntekijän henkilökohtainen tunnistearvo tallensi yrityksen käyttämän työntekijänumeron, joka ei vaikuttanut sovelluksen toimintaan. Tehtävän prioriteetti sisältää numeerisen arvon, joka kertoo tehtävän tärkeystason. Lisätietokenttä sisältää tekstijonon, jossa tehtävään liittyvät lisätiedot ovat.

```
CREATE TABLE IF NOT EXISTS `evenement` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `title` varchar(255) CHARACTER SET utf8 COLLATE utf8_unicode_ci NOT NULL,
 `w_id` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
 `start` datetime NOT NULL,
 `end` datetime DEFAULT NULL,
 `allDay` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin DEFAULT NULL,
 `resource` varchar(60) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '',
 `info` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL DEFAULT '',
 `priori` int(11) NOT NULL,
 PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=557 ;
```

Fc_hourTable sisältää sovelluksen tuntiraportointi ja seuranta toiminnallisuudelle vaaditut kentät. Taulu sisältää kentät tietokantatunniste, resurssitunnisteen, työntekijänumeron, binääritunniste, palkkalaji, tapahtuman otsikko, tapahtumien eli tuntien määrä, alkupäivämäärä, tuntien kirjaustunniste sekä tuntien hyväksyntätunniste.

Binääritunniste lisäsi tuntikirjauksen tulosteeseen seuraavan ohjelman vaatiman tunnistetiedon 00,01. Palkkalaji kenttään tallennettiin numeerinen arvo, jolla kerrottiin palkkatyyppi. Tuntienmäärään tallennettiin työtehtävän pituus tunteina. Alkupäivämäärässä tallennetaan työtehtävän aloituspäivämäärä. Tuntien kirjaustunniste sisältää arvon, jolla ohjelma tietää oliko työntekijä hyväksynyt

tunnit tarkastettavaksi ja hyväksymistunniste vastaavasti esimiehen hyväksymän tai hylkäämän.

```
CREATE TABLE IF NOT EXISTS `fc_hourTable` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `resource_id` int(11) NOT NULL,
 `personel_num` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
 `binary_opt` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
 `salary_opt` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
 `event_title` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
 `event_qty` float NOT NULL,
 `startdate` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
 `isSaved` tinyint(1) NOT NULL DEFAULT '0',
 `isAccepted` tinyint(1) NOT NULL DEFAULT '0',
 PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=125 ;
```

Login_attempts-taulu, joka tallentaa tietokantaan kirjautumisyritykset. Tunnistekenttään tallennettiin yrittäneen käyttäjän tunniste ja toiseen kenttään kirjautumisen ajankohta seurantaa varten.

```
CREATE TABLE IF NOT EXISTS `login_attempts` (
 `user_id` int(11) NOT NULL,
 `time` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Members-tauluun tallennetaan sovelluksen käyttäjien tiedot kirjautumista varten. Taulu sisältää tunnistekentän, käyttäjänimen, salasanan, sähköpostin ja suolan. Suola kenttä oli salasanan suolaamista varten, jolloin salasana ei ole tietokantaa tarkastelevan tahon tarkasteltavissa.

```
CREATE TABLE IF NOT EXISTS `members` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `username` varchar(30) NOT NULL,
 `email` varchar(50) NOT NULL,
 `password` char(128) NOT NULL,
 `salt` char(128) NOT NULL,
 PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=4 ;
```

Resources-tauluun tallennettiin sovelluksen resurssit, tässä tapauksessa työntekijät. Resurssit jakoivat kalenterin osiin ja ne kertoivat sovellukselle työntekijöiden nimet ja henkilönumeron. Taulu sisälsi tunnistekentän, yrityksen henkilönumerot työntekijöille sekä työntekijän nimet.

```
CREATE TABLE IF NOT EXISTS `resources` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `h_nro` int(11) NOT NULL,
 `name` varchar(20) COLLATE utf8_bin NOT NULL,
 PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_bin AUTO_INCREMENT=54 ;
```

Salary_opt-tauluun tallennettiin sen hetkiset palkkalajit, jota tuntikirjauksen tulostus käytti. Taulu sisältää tunnistekentän, palkkalajin tunnistenumeron sekä palkkalajin nimen. Tunnistenumero oli sovelluksen tuntitulostusta varten, joka kertoi tuntityypin seuraavalle sovellukselle.

```
CREATE TABLE IF NOT EXISTS `salary_opt` (
 `id` int(11) NOT NULL AUTO_INCREMENT,
 `nro` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
 `name` varchar(255) CHARACTER SET utf8 COLLATE utf8_bin NOT NULL,
 PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=14 ;
```

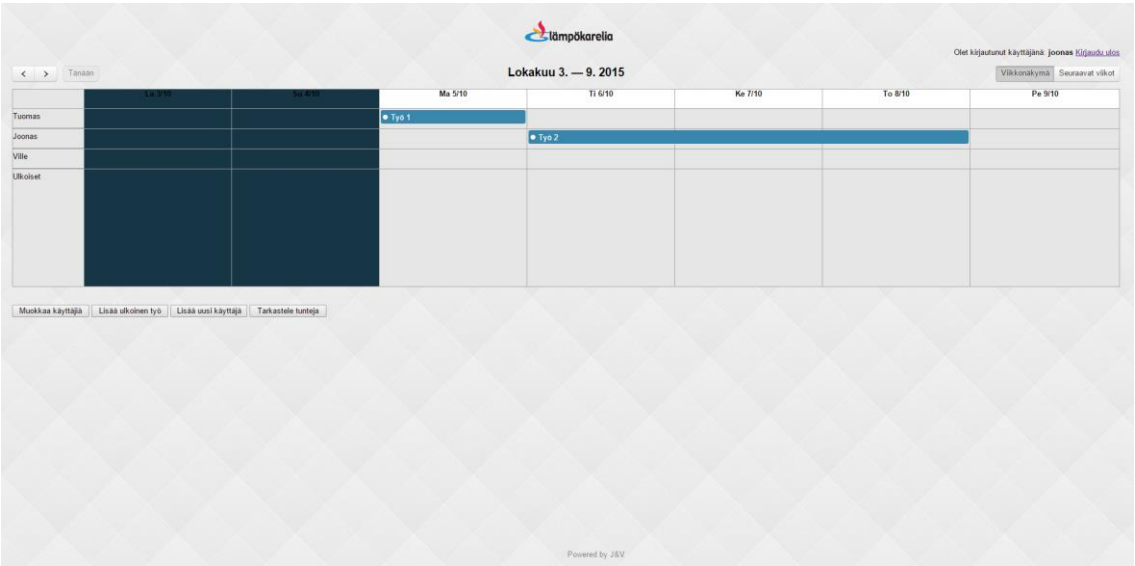
4.3 Käyttöliittymä

Kalenteriin lisättiin asiakkaan toiveiden mukaan ominaisuuksia tietokantayhteyden ja kalenteritietueittein lisäkenttien lisäksi. Hyvän resurssinäkymän lisäksi tietueisiin toteutettiin prioriteetti-arvo, jolloin se asettamalla tietueen väri kentällä muuttui sen tärkeyden mukaan punaisesta vihreään.

Tämän lisäksi tehtiin alpha-vaiheeseen jäänyt työntekijöiden lukulaitteen rajapinta. Tässä perusideana oli, että työntekijöiden lukulaitteelta saatujen arvojen perusteella luotiin automaattisesti tietue kalenteriin, josta se lisättiin suoraan tietokantaan.

Yrityksen käyttämän eri sovelluksen kautta saatu data syötettiin sovellukseen, josta purettiin tarvittavat kentät tietuetta varten.

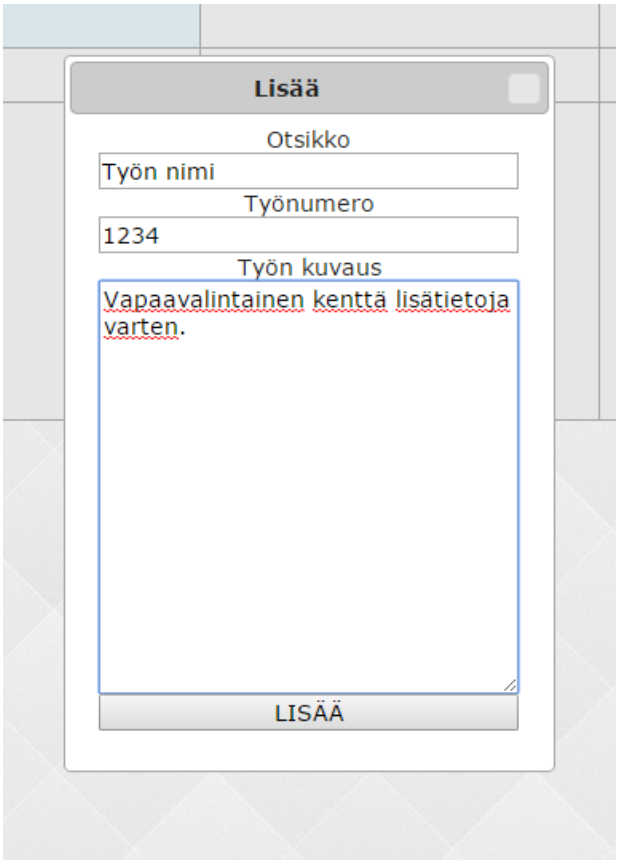
Perusnäkymä on hyvin yksinkertainen näkymä, jossa näkyy yksi viikko kaikkien työntekijöiden kohdalla. Ylhäällä on päivämäärät ja päivät, ja vasemmalla sivussa on työntekijöiden nimet, joiden kohdalla heidän työnsä näkyvät.



Kuva 4. Perusnäkymä

Kun tapahtuma luodaan, dialogi kysyy tarvittavat arvot jatkokäsittelyä varten. Tapahtuma luodaan raahaamalla se halutun työntekijän kohdalle ja halutulle

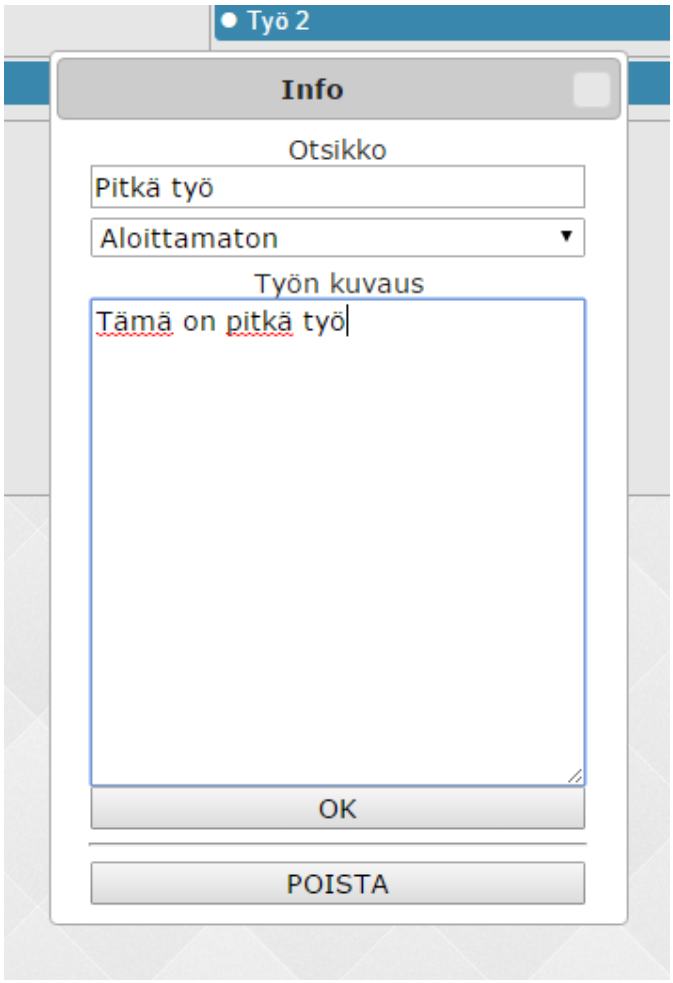
aikavälille. Luontidialogissa määritellään kalenterissa näkyvä otsikko, työpaikan työnumerotunniste sekä vapaavalintainen lisätietokenttä.



Kuva 5. Luontidialogi

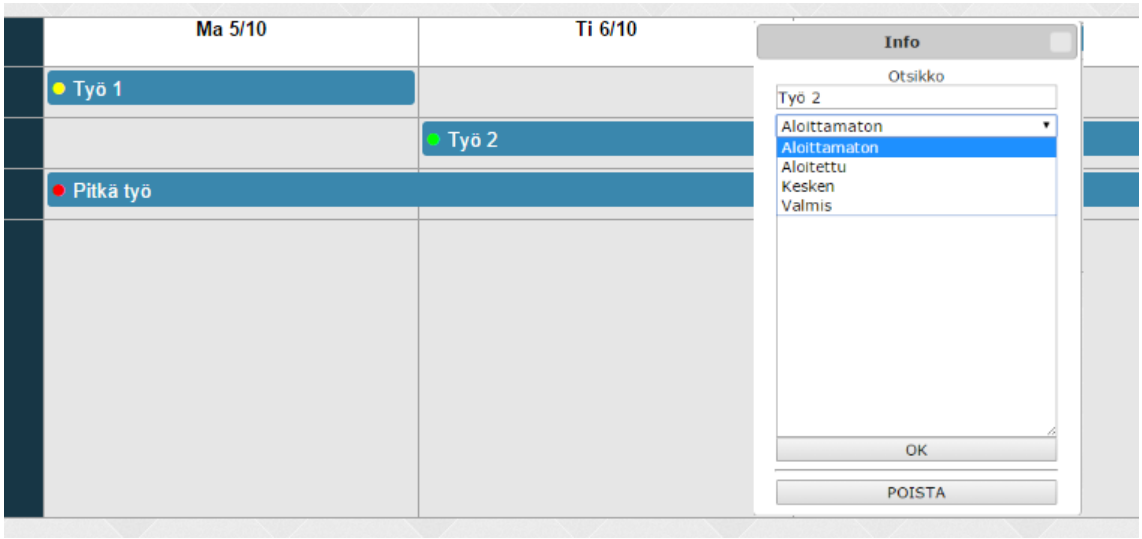
Tämän jälkeen kalenterin tapahtumakäsittelyfunktio vie raahaamisesta selvinneet arvot dialogille, tässä tapauksessa numeerisen arvon resurssista sekä aloitus- ja lopetuspäivämäärän. Nämä arvot eivät näy käyttäjälle, vaan ovat tietokantaan vietäviä tietoja.

Kun jo luotua tapahtumaa painetaan, avautuu ohjelmassa muokkausdialogi. Dialogissa voidaan muokata jo olemassa olevia arvoja, sekä näiden lisäksi tapahtuman prioriteetti. Kun arvot on syötetty, dialogi hyväksymällä tehdään kutsu tietokantaan muokkaukseen tarkoitetulla funktiolla. Muokkausdialogilla voidaan myös tarvittaessa poistaa tapahtuma.

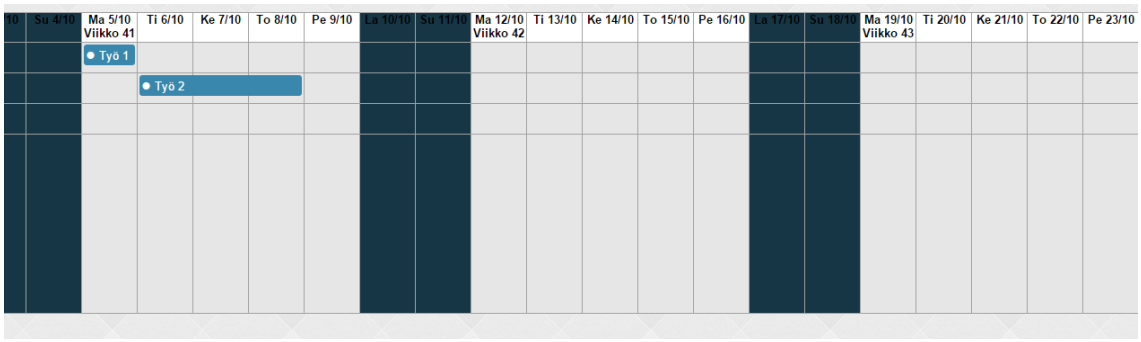


Kuva 6. Muokkausdialogi

Priorisoinnin väreinä käytimme liikennevaloista tuttuja vihreätä, punaista ja keltaista. Vihreä symboloiden valmista, punainen aloittamatonta ja keltainen aloitettua, mutta ei valmista työtehtävää. Nämä toimme suoraan käyttöliittymään kalenterin objektien otsikoihin, joista työtehtäviä oli helppo seurata.



Koska viikkonäkymä ja resurssisolut pilkkoivat kalenterin pienempiin palasiin, ei näkymästä tullut tämän vuoksi epäselvää. Tämän lisäksi kalenteria kyetään käyttämään kuukausinäkymässä samoilla periaatteilla.



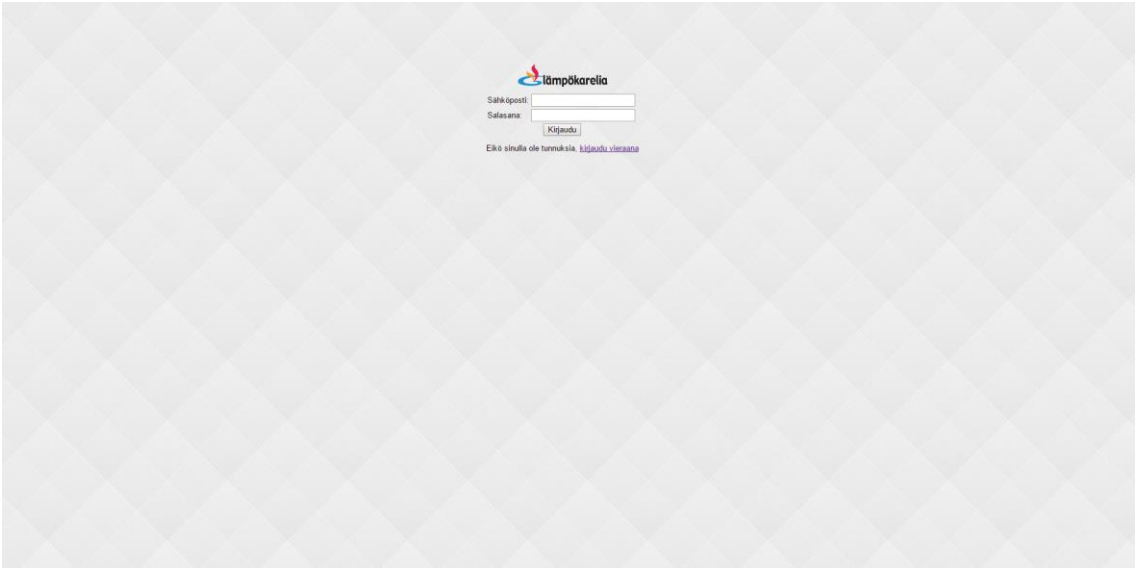
Kuva 7. Kuukausinäkymä

Käyttöliittymäsuunnittelussa lähdettiin siitä, että ohjelman tulisi olla selkeä ja helppolukuinen jokapäiväisessä käytössä. Sen suurempia muutoksia FullCalendarin perusnäkymään ei tehty, koska se täytti nämä vaatimukset lähes suoraan.

Käyttöliittymän pääkohtana toimi tietenkin kalenteri, jonka lisäksi emme yksinkertaista taustakuvaa ja yrityksen logoa kummempaa muutosta tehneet, eikä se olisi tässä projektissa tarkoituksenmukaista.

4.3.1 Kirjautuminen ja käyttäjän luonti

Kirjautumisessa käytettiin perinteistä tietokantaan lisättyjen käyttäjien tarkistamista. Kirjautumisnäkymä noudattaa samanlaista tyyliä pääsivun kanssa.



Kuva 8. Kirjautumisnäkymä

Käyttäjien salasanoihin lisättiin niin sanottu “suola”, joka lisää tietoturvaa. Kirjautuminen toteutettiin PHP:llä ja tietokantakutsut AJAX:lla. Suola on eräänlainen lisä salasanassa, joka muuttaa salasanan niin, ettei se esiinny tietokannassa alkuperäisessä muodossaan. Tähän käytetään erimittaisia merkkijonoja.
Ohjelmaa ja sinne syötettyjä töitä pääsee tarkastelemaan kuitenkin eräänlaisessa vierastilassa, jolloin sinne ei ole mahdollista luoda uusia, muokata eikä sieltä ole mahdollista poistaa vanhoja töitä. Tämän ansiosta jokaiselle työntekijälle ei tarvitse välttämättä luoda omaa käyttäjää, vaan he näkevät omat työnsä ilman kirjautumista. Tarkoituksena olikin, että vain työnjohtajilla on kirjautumiseen sekä kalenterin hallinnointiin käytettävät tunnukset.

5 Testaus

Testaussuunnitelma oli alunpitäen se, että sovellusta testataan sitä käytettäestä. Tästä syystä toimitimme hyvin karkean, alpha julkaisun jo hyvin varhaisessa vaiheessa kehitysprosessia. Tulevien kuukausien aikana n. kuukausien väliajoin toimitimme sovelluksesta uuden version käyttäjän toiveiden mukaisesti.

Asiakas keräsi käyttäessään kokemuksia ja puutteita sovelluksesta, jotka sitten kerrottiin meille versiopäivityksen yhteydessä. Tällä tavalla saatiin aina uutta tehtävää. Jos ominaisuudet olivat valmiita ja myös suoraa palautetta siitä miten ohjelma ja uudet ominaisuudet on otettu vastaan.

Myöhemmin asiakkaalla oli toimistotyöntekijä, joka käytti tehtävissään kyseistä ohjelmaa. Hän oppi käyttämään sovellusta ilman perehdytystä, joka osoitti, että käyttöliittymä oli tarkoituksenmukainen.

Mielestämme tämä testaustapa toimi tämänkaltaisessa sovelluksessa erittäin hyvin, koska perussovellus oli mahdollista saattaa käyttökuntoon hyvin varhaisessa vaiheessa. Tämä toimi myös, koska työn ajankohta ja aloitus oli niin pikaista, että sen pidempää suunnittelua ei ohjelmassa ollut vaan lähdettiin toteuttamaan ohjelmaa mahdollisimman joutuin.

Jatkuvalla testauksella tarkoitetaan tässä yhteydessä sitä, että ohjelma on ollut koko ajan käytössä yrityksellä ja sitä on päivitetty jatkuvasti pienillä päivityksillä. Tämä mahdollistaa ohjelman aikaisen julkaisun ja käyttäjiltä saatavan palautteen, jonka avulla tuotteesta tehdään entistä toimivampi ja luotettavampi. Kehittäjien mielestä tämä testaussuunnitelma toimi parhaiten tässä tilanteessa, koska ohjelma oli käytössä yrityksellä jo ennen opinnäytetyön aloittamista.

Ohjelman yksinkertaisuus nimenomaan mahdollisti tämän, koska perusversioon saatiin lisättyä ominaisuuksia sitä mukaa kun niitä haluttiin ja tarvittiin.

6 Pohdinta

Mielestämme työ oli kokonaisuutena onnistunut, sillä saimme aikaan toimivan järjestelmän ja se oli kyseisessä yrityksessä jokapäiväisessä käytössä töiden organisoinnissa. Saimme lisättyä järjestelmään halutut lisäominaisuudet ja kaikenkaikkiaan järjestelmä oli helppokäyttöinen.

Työtä jälkeenpäin kokemuksen kasvettua tarkastellessa kävi selkeästi ilmi, että vaikka projekti oli erittäin opettavainen, olivat työskentelymenetelmät ja ammattitaito aloittelijan tasolla. Tämä kävi selkeästi ilmi kirjastojen puolihuolimattomasta hyödyntämisestä, koodin laadusta ja työskentelymenetelmistä. Projekti oli kuitenkin onnistunut ja sovellus toimiva, joten tämä oli toissijaista.

Jos olisimme lähteneet toteuttamaan projektia pidemmällä aikavälillä tai jopa kaupallisesti, olisimme valinneet paremman ohjelmistokehyksen eli alustan työtä varten. Sillä hetkellä emme kuitenkaan tienneet paremmasta ja yksinkertainen LAMP-serveri ja sen päälle rakennettu sovellus oli riittävä. Teknillisesti mielenkiintoinen ja varmasti parempi alusta olisi ollut esimerkiksi python pohjainen Django sovelluskehys, joka on kehitetty vastaavien sovellusten ja sivustojen luomiseen ja hallitsemiseen. Djangon suosioista kertoo jo se, että esimerkiksi suositut sivustot Instagram ja Pinterest ovat rakennettu Djangoa käyttäen.

Sovelluksessa ei myöskään ollut nykystandardien mukaista responsiivisuutta eri kokoisille laitteille ja tämä olisi käytännössä tarkoittanut koko tyyliosion uudelleen suunnittelua ja tekemistä. Myös FullCalendar kirjaston toiminnallisuus pienemmissä laitteissa olisi jouduttu testaamaan erikseen, koska työntekovaiheen versiossa ei mielestämme responsiivisuutta laitteille ollut.

Koska sovellus perustui elementtien raahaukseen ja pudotteluun, myös tämä olisi tarkoittanut sovelluksen uudelleensuunnittelua ja pahimmillaan koko toimintaperiaatteen muutosta kosketusnäytöllisten laitteiden osalta.

Jo projektin aloittaneella kurssilla oli selvää, että työtä tultaisiin jatkamaan opinnäytetyönä ja näin ollen palvelinratkaisu olisi voitu suunnitella paremmin. Yrityksellä oli käytössä kotisivupalvelin, jonne järjestelmä olikin tarkoitus jossain vaiheessa siirtää, mutta tähän asti ei koskaan päästy. Tällä käyttömäärällä ratkaisu kuitenkin oli toimiva, koska sivusto toimi erittäin luotettavasti, ollen pelkästään lähiverkosta saatavilla.

Sovelluksen jatkokehitys tai kaupallistaminen olisi edellyttänyt käytännössä koko sovelluksen uudelleen kirjoittamista, uudella alustalla ja tiukemmalla yhteistyöllä firman kanssa. Vaikka testaus ja versiopäivitykset olivat suhteellisen

nopeaa projektin koosta huolimatta, olisi kehittäminen silti vaatinut enemmän kontaktia yrityksen kanssa ja seikkaperäisempää suunnittelua. Tämä olisi vaatinut erityisesti yhteistyötä yrityksen IT-osaston kanssa, jotta sovellus ja sen vaatima järjestelmä olisi tullut integroitua oikeaoppisesti.

Vaikka sovellusta kyettiin yrityksessä hyvin tuloksin käyttämään ja se helpotti yrityksen toimintaa, se oli selkeästi enemmän tekniikkademo, kuin toimiva ja valmis sovellus.

Kaiken kaikkiaan projekti opetti paljon sovelluskehityksestä ja sitä ympäröivistä muuttuvista tekijöistä. Monet asiat olisi voinut hoitaa paremmin, mutta mikä ammattitaidossa hävittiin, tuli takaisin oppina ja tulee varmasti olemaan hyödyksi tulevaisuuden ammattillisessa kehittyksessä.

Lähteet

1. http://fullcalendar.io/
2. https://github.com/jarnokurlin/fullcalendar
3. http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/
   https://groups.google.com/forum/#!msg/comp.infosystems.www.authoring.cgi/PyJ25gZ6z7A/M9FkTUVDfcwJ
4. http://buytaert.net/the-history-of-mysql-ab
5. http://www.oracle.com/us/sun/index.htm
6. https://soleops.karelia.fi/opsnet/disp/fi/ops_OpetTapTeks/tab/tab/sea?opettap_id=178873446&stack=push

LIITTEET
liite 1