OULUN AMMATTIKORKEAKOULU

Jani Leinonen

PÖYTÄVARAUSJÄRJESTELMÄ-LISÄOSA WORDPRESSIIN

PÖYTÄVARAUSJÄRJESTELMÄ-LISÄOSA WORDPRESSIIN

Jani Leinonen
Opinnäytetyö
Syksy 2018
Tietojenkäsittely koulutusohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn koulutusohjelma, Web-sovelluskehityksen sv

Tekijä(t): Jani Leinonen
Opinnäytetyön nimi: Pöytävarausjärjestelmä-lisäosa WordPressiin
Työn ohjaaja: Pekka Ojala
Työn valmistumislukukausi ja -vuosi: Syksy 2018 Sivumäärä: 36

Opinnäytetyön tarkoituksena oli kehittää pöytävarausjärjestelmä ravintoloiden verkkosivuille. Järjestelmä toteutettiin lisäosana WordPress -sisällönhallintajärjestelmään. Toimeksiantajana toimi mainostoimisto Asema10 Oy. Toimeksiantajan toiveena oli toimiva pöytävarausjärjestelmä, mikä olisi helposti integroitavissa eri WordPress-sivustoille. Tämän vuoksi lisäosa oli järkevä ratkaisu.

Raportin teoriaosuudessa käsitellään WordPressiä ja sen kehittäjille suunnattuja ominaisuuksia. Käytännön osuudessa käydään läpi toteutetun lisäosan toiminta sekä kerrotaan lyhyesti kehitystyössä käytetyistä tekniikoista ja lisäosan rakenteesta. Työn lopputuloksena oli toimiva pöytävarausjärjestelmä, mikä vaatii vielä kehittämistä, jotta se olisi valmis tuote. Lisäosan koodia voidaan helposti käyttää uudelleen sen luokkarakenteen vuoksi.

Asiasanat: WordPress, sisällönhallintajärjestelmä, lisäosa, PHP, jQuery, AJAX

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems, Web application development

Author(s): Jani Leinonen
Title of thesis: Table reservation service as a WordPress plugin
Supervisor(s): Pekka Ojala
Term and year when the thesis was submitted: Autumn 2018 Number of pages:36

The purpose of this thesis was to develop a WordPress plugin that implements an online table reservation service for WordPress sites. The client requested for a table reservation service app, that could be easily integrated in WordPress sites. Because of this developing a WordPress plugin was a smart way to do it.

The theory section of this thesis covers WordPress and its features for developers. The practical section goes through the plugin and its features step by step. This section also covers some of the techniques used in plugins development and explains the structure of the plugin. The result of this thesis was a functional table reservation service plugin that still requires much development for it to be considered as a finished product. The code used in this plugin can easily be reused when developing other plugins because of its class based structure.

Keywords: WordPress, content-management system, plugin, PHP, jQuery, AJAX

SISÄLLYS

1 JOHDANTO

Tämän opinnäytetyön aiheena on ravintoloille suunnatun pöytävarausjärjestelmän toteuttaminen WordPress sisällönhallintajärjestelmään. Työn toimeksiantajana toimii mainostoimisto Asema10 Oy. Opinnäytetyön tavoitteena on syventää osaamistani WordPressin ja sen lisäosien kehityksessä, sekä kehittää ohjelmointitaitojani. Kehittämistehtävän päämääränä oli saada aikaiseksi toimiva pöytävarausjärjestelmä, jota toimeksiantaja voi suositella ravintoloille WordPress-verkkosivujen toteuttamisen yhteydessä.

Raportin tietoperustassa käydään läpi hieman perustietoa WordPressistä, jonka jälkeen käsitellään sen tarjoamia ominaisuuksia joita lisäosan kehityksessä voidaan hyödyntää. Näihin kuuluvat mm. WordPressin kehittäjille tarkoitetut ohjelmointirajapinnat. Toiminnallisessa osuudessa käydään läpi lisäosan toteutus sekä selostetaan hieman käytettyjä tekniikoita sekä lisäosan rakennetta. Raportin lopussa on pohdintaosio, jossa käydään läpi hieman työn haasteita sekä tuloksia joita saatiin aikaan.

2 WORDPRESS

Blogien julkaisualustana alkunsa vuonna 2003 saanut WordPress, on sittemmin kehittynyt maailman suosituimmaksi sisällönhallintajärjestelmäksi, jota käytetään yli 31%:ssa kaikista nettisivuista. WordPress on kehitetty PHP -ohjelmointikielellä MySQL -tietokantaa hyödyntäen ja se on julkaistu avoimen lähdekoodin ohjelmistojen julkaisuun tarkoitetulla GPLv2 -lisenssillä (General Public License). Avoimen lähdekoodin julkaisualustana WordPressiä kehittävät jatkuvasti osaavat web -kehittäjät, suunnittelijat ja bloggaajat ympäri maailmaa. (WordPress 2018a, viitattu 15.08.2018.)

2.1 Käyttäjälle

Perusominaisuuksien lisäksi WordPressin tuhannet lisäosat tarjoavat käyttäjälle lähes rajattomasti eri käyttömahdollisuuksia laajentaa WordPressin toimintaa. WordPressillä voi sen joustavuuden vuoksi luoda millaisen nettisivun tahansa ja käyttäjä voi valita tuhansista teemoista ulkoasultaan ja ominaisuuksiltaan mieleisensä. (WordPress 2018b, viitattu 15.08.2018.)

WordPress mahdollistaa helppokäyttöisen sisällönhallinnan ja julkaisemisen käyttäjälle. Käyttäjä voi luoda sivuja ja postauksia, muokata tekstiä editorissa ja lisätä mediatiedostoja. WordPressin julkaisutyökalut mahdollistavat luonnosten tekemisen, julkaisujen ajoittamisen ja aiempien luonnosversioiden palauttamisen. Lisäksi käyttäjä voi merkitä sisällön julkiseksi tai yksityiseksi ja suojata sivuja tai yksittäisiä postauksia salasanalla. (WordPress 2018b, viitattu 15.08.2018.)

WordPressiin voi lisätä useita eri käyttäjiä, joilla on eri käyttöoikeuksia. Näitä eri käyttäjärooleja on WordPressissä oletuksena viisi:

- Ylläpitäjä (kaikki oikeudet)
- Päätoimittaja (oikeudet julkaista ja muokata sisältöä)
- Kirjoittaja (oikeudet julkaista ja muokata itse luomiaan postauksia)
- Avustaja (oikeudet kirjoittaa, muokata ja poistaa itse luomiaan julkaisemattomia postauksia, joko ylläpitäjä tai päätoimittaja julkaisee)
- Tilaaja (oikeudet lukea ja kirjoittaa kommentteja, sekä muokata omaa profiilia) (Strojny 2014, viitattu 15.08.2018.)

2.2 Kehittäjälle

WordPress tarjoaa web -kehittäjille paljon ominaisuuksia joita hyödyntämällä he voivat laajentaa ja kehittää WordPressiä haluamallaan tavalla. Sen mukana tulevat ominaisuudet helpottavat mm. erilaisten sovellusten tekemistä, kuten esimerkiksi kielikäännösten tekeminen, tietokannat, HTTP-kyselyt ja URL-reititykset. WordPressin sovellusrajapinta mahdollistaa kehittäjien luoda omia lisäosia ja teemoja, joko omaan käyttöön tai julkiseen levitykseen. WordPressin mukana tulevat myös seuraavat koodikirjastot: jQuery, Plupload, Underscore.js ja Backbone.js. (WordPress 2018b, viitattu 16.08.2018.)

3 WORDPRESS -LISÄOSAN KEHITYS

Lisäosilla voidaan muokata, kehittää ja parantaa WordPress-sivuston toimintaa. WordPressin lisäosien tekoon suunniteltu sovellusrajapinta mahdollistaa lisäominaisuuksien kehittämisen, ilman että kehittäjien tarvitsee muokata WordPressin ydinkoodia. (WordPress 2018j, viitattu 16.08.2018.)

3.1 Hookit eli koukut

WordPress tarjoaa lisäosan kehittäjälle ”hookkeja” eli koukkuja, joita hyödyntämällä voidaan laukaista lisäosan toimintoja haluttuun aikaan ja ylipäätään saadaan lisäosa toimimaan. Näitä koukkuja on kahdenlaisia: toiminta- ja suodatinkoukut (action hook, filter hook). Usein saman lopputuloksen voi saada käyttämällä kumpaa tahansa tapaa. (WordPress 2018f, viitattu 17.08.2018.)

Toimintakoukkujen avulla voidaan WordPressiin lisätä uusia toimintoja tai muuttaa sen toimintaa. Ne mahdollistavat funktioiden ajamisen tietyissä vaiheissa WordPressin suorituksen aikana. Toimintojen lisääminen onnistuu luomalla ensin funktio, joka halutaan suorittaa, jonka jälkeen ”hookataan” kyseinen funktio haluttuun toimintakoukkuun WordPressin add_action() -funktiolla (kuvio 1). (WordPress 2018k, viitattu 17.08.2018.)

```
10
11
12  public function init() {
13
14      add_action( 'rest_api_init', array($this, 'register_rest_routes' ));
15
16  }
17
18  public function register_rest_routes() {
19
20      // Code to execute
21
22  }
23
```

KUVIO 1. Toimintafunktion ”hookkaaminen” toimintakoukkuun PHP-luokassa

Suodatinkoukut mahdollistavat funktioiden luomisen, joilla voi muokata toisten funktioiden sisältämää dataa, kuten esimerkiksi muuttujia. Suodatinkoukkujen avulla voidaan ainoastaan muokata jo olemassa olevaa dataa. Tämä tapahtuu välittämällä suodatinkoukkuun ”hookatulle” funktiolle da-

taa, jonka jälkeen tämä funktio käsittelee datan ja palauttaa sen (kuvio 2). Toisin kuin toimintakoukkujen kohdalla, suodatinkoukut aina lähettävät ja vastaanottavat dataa. (Hayes 2018, viitattu 23.08.2018.)

```
4
5   function atr_smtp_options($phpmailer){
6
7       $phpmailer->SMTPOptions = array(
8           'ssl' => array(
9               'verify_peer' => false,
10              'verify_peer_name' => false,
11              'allow_self_signed' => true
12          )
13      );
14      return $phpmailer;
15  }
16
17  add_filter( 'wp_mail_smtp_custom_options', 'atr_smtp_options' );
18
```

KUVIO 2. Suodatinfunktion ”hookkaaminen” suodatinkoukkuun

3.2 Options API

Options API mahdollistaa yksinkertaisen tavan tallentaa tietoa tietokantaan. Sen tarjoamien funktioiden avulla voi helposti lisätä, muokata, hakea ja poistaa tietoa tietokannasta. Kaikki tiedot tallennetaan tietokantaan wp_options -nimiseen tauluun nimi/arvo pareina. Options API tarjoaa seuraavat funktiot tietokannan hallintaan:

- add_option (lisää uuden tiedon)
- delete_option (poistaa tiedon)
- update_option (päivittää/lisää tiedon)
- get_option (hakee tiedon) (WordPress 2018e, viitattu 27.08.2018.)

3.3 Settings API

WordPress 2.7 versiossa lisätty Settings API mahdollistaa ylläpitäjän asetuksia sisältävien sivujen lisäämisen ja hallitsemisen puoliautomaattisesti. Sen avulla voidaan määrittää asetussivuja, jotka voidaan jakaa osioihin (section). Näille osioille voidaan märittää kenttiä (field), jotka sisältävät yk-

sittäisen asetuksen. Nämä asetukset tallentuvat samaan tietokannassa olevaan ”wp_options” tauluun kuin Options API:n kautta suoraan tallennettavat tiedot. (WordPress 2018g, viitattu 27.08.2018.)

Yksinkertaisimmillaan asetussivujen luominen Settings API:a hyödyntäen vaatii seuraavat vaiheet: asetusten rekisteröinti, asetussivujen lisääminen WordPressin hallintapaneelin navigaatioon, sekä itse asetussivujen sisällön luonti. Asetusten rekisteröinti tapahtuu hyödyntämällä Settings API:n tarjoamia funktioita: add_settings_section(), add_settings_field() ja register_setting(). Kyseinen kehittäjän luoma funktio, joka sisältää asetusten rekisteröinnin, ”hookataan” admin_init -nimiseen toimintakoukkuun. (Hayes 2015, viitattu 27.08.2018.)

Navigaation lisääminen onnistuu käyttämällä yhtä WordPressin monista funktioista, joilla saadaan lisättyä WordPressin hallintapaneelin sivupalkin valikkoon uusia sivuja. Funktiolla add_menu_page() kehittäjä voi lisätä uuden päätason valikkoelementin. Mikäli kehittäjä haluaa lisätä alasivun jollekin jo valikossa sijaitsevalle elementille, voi hän käyttää funktiota add_sub_menu_page(). Kehittäjä voi myös käyttää kyseiselle valikkoelementille suunnattua funktiota, mikäli sellainen on, kuten esimerkiksi add_posts_page(), jolla lisätään päätason valikkoelementille ”Postit” uusi alasivu. Itse sivun sisällön luonti tapahtuu funktiossa, joka välitetään edellä mainittujen funktioiden parametrinä niin sanottuna callback-funktiona. (Hayes 2017, viitattu 28.08.2018.)

Asetussivun sisällön rakentavassa funktiossa voidaan tulostaa ensimmäisessä vaiheessa rekisteröidyt asetukset lomakkeeksi seuraavilla Settings API:n funktioilla: settings_fields() ja do_settings_sections(). Settings API:n funktiot renderöivät ainoastaan lomakkeen sisältämät asetuskentät, joten kehittäjän tulee itse luoda lomakkeen aloitus – ja lopetustagit sekä submit-painikkeen. (Hayes 2015, viitattu 29.08.2018.)

3.4 Shortcode API

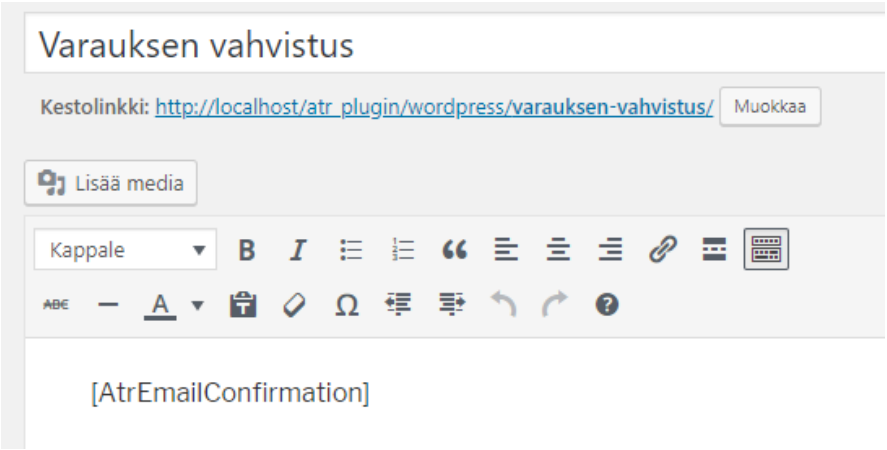
Shortcode API mahdollistaa funktioiden kehittämisen, joilla lisäosan käyttäjä voi lisätä sivuille tai postauksiin uutta sisältöä, kuten esimerkiksi lomakkeen. Tämä tapahtuu hyödyntämällä Shortcode API:n add_shortcode()-funktiota. Kyseinen funktio ottaa vastaan kaksi parametriä: shortcoden nimi

ja callback-funktio, joka sisältää uuden sisällön rakentavan koodin (kuvio 3A). (WordPress 2018h, viitattu 29.08.2018.)

```
10
11  public function init() {
12
13      add_shortcode('AtrEmailConfirmation', array($this, 'shortcode_email_confirmation'));
14
15  }
16
17  public function shortcode_email_confirmation(){
18
19      do_action( 'atr_before_email_confirmation_form', 10 );
20
21      echo  __('Varaus vahvistettu onnistuneesti');
22
23      do_action( 'atr_after_email_confirmation_form', 10 );
24
25  }
26
```

KUVIO 3A. Lyhytkoodin luonti PHP-luokassa.

Lisäosan käyttäjä voi lisätä haluamalleen sivulle tai postaukseen kyseisen sisällön lisäämällä ”shortcoden” eli lyhytkoodin kyseisen sivun/postauksen tekstikenttään (kuvio 3B). Halutessa voidaan lyhytkoodia lisätessä lähettää callback-funktiolle parametrejä seuraavasti: [lyhytkoodin_nimi parametri1=” ” parametri2=” ”], mikäli niitä on kyseisessä funktiossa määritelty. (WordPress 2018h, viitattu 29.08.2018.)



KUVIO 3B. Lyhytkoodin lisääminen sivustolle.

3.5 WP-Cron

Cron on Unixin tukema ominaisuus, joka mahdollistaa ajastettujen tehtävien (Cron job) suorittamisen tiettyihin kellonaikoihin tai tietyin väliajoin. WordPressin WP-Cron, toisin kuin serverillä suoritettava Cron-tehtävä, ajetaan sivunlatauksen yhteydessä. Tästä syystä WP-Cronilla ajastettujen

tehtävien suorittaminen ajallaan on riippuvainen sivuston liikenteestä. Jonossa olevat tehtävät, joiden ajastettu aikaväli on mennyt jo umpeen, ajetaan seuraavan sivunlatauksen yhteydessä. (WordPress, 2018l, viitattu 30.08.2018.)

WordPressissä on oletuksena kolme aikaväliä: tunnin välein, kahdesti päivässä ja kerran päivässä. Mukautettujen aikavälien lisääminen onnistuu helposti hookkaamalla ”cron_schedules” -suodatinkoukkuun suodatinfunktion, jolla lisätään uusi aikaväli (kuvio 4A). Uuden tehtävän lisääminen onnistuu funktiolla wp_schedule_event(), mutta ennen uuden tehtävän lisäämistä tulee tarkastaa ettei kyseinen tehtävä ole jo jonossa käyttämällä funktiota wp_next_scheduled() (kuvio 4B). (WordPress 2018m, viitattu 31.08.2018.)

```
24
25  add_filter( 'cron_schedules', 'atr_schedules' );
26
27  function atr_schedules( $schedules ) {
28
29      $schedules['5min'] = array(
30          'interval' => 5*60,
31          'display' => __( '5 minuutin välein' )
32      );
33      return $schedules;
34
35  }
36
```

KUVIO 4A. Mukautetun aikavälin lisääminen cron_schedules -suodatinkoukkua käyttäen

```
35
36  if (! wp_next_scheduled ( 'atr_remove_unconfirmed_reservations' )) {
37
38      wp_schedule_event(time(), '5min', 'atr_remove_unconfirmed_reservations');
39
40  }
41
```

KUVIO 4B. Uuden WP-Cron tehtävän lisääminen

WordPressin tapaan lisätyt Cron-tehtävät on mahdollista ajaa myös täsmällisesti, ilman että ne ovat riippuvaisia sivun latauksista eli sivuston kävijämäärästä. Tällöin kehittäjän tulee ottaa pois käytöstä WP-Cron lisäämällä seuraava rivi koodia wp-config.php -tiedostoon: define(’DISABLE_WP_CRON’, true). Tämän jälkeen kehittäjän tulee lisätä serverille uusi ”oikea” Cron-tehtävä, joka ajetaan sen aikavälin mukaan, mikä on pienin WordPressissä määritetyistä aikaväleistä. Tämä serverillä määritetty Cron-tehtävä tekee pyynnön edellä määritetyin aikavälein wp-cron.php

-tiedostoon, joka tarkastaa WordPressin aikataulutetut tehtävät. (McFarlin 2013, viitattu 31.08.2018.)

3.6 Lisäosan tietoturva

WordPressin lisäosan kehityksessä on erittäin tärkeää huolehtia siitä, että lisäosan tietoturva on kunnossa. Lisäosan tietoturva-aukko voi vaarantaa kaikkien lisäosaa käyttävien sivustojen koko sivuston turvallisuuden. Tämän vuoksi on tärkeää tehdä lisäosa tietoturvalliseksi ja päivittää sen tietoturvaa tarpeen vaatiessa.

Tärkein vaihe WordPress -lisäosan tietoturvan rakentamisessa on käyttöoikeuksien tarkistaminen. Tällä voidaan varmistaa, että vain tietyt käyttäjät pääsevät käsiksi haluttuihin ominaisuuksiin. Ylläpitäjän roolilla esimerkiksi on WordPressissä ”manage_options” -käyttöoikeudet. Mikäli kehittäjä haluaa rajoittaa toimintoja WordPressin hallintapaneelin tai sivuston puolelta tietyille käyttöoikeustasoille, voi hän hyödyntää WordPressin current_user_can() -funktiota (kuvio 5). Kyseinen funktio ottaa parametrinä vastaan halutun käyttöoikeuden ja palauttaa joko tosi tai epätosi, riippuen onko käyttäjällä kyseiset käyttöoikeudet. (WordPress 2018n, viitattu 06.09.2018.)

```
22
23
24  if ( current_user_can( 'manage_options' ) ) {
25
26      /* Shows only for administrators
27
28  }
29
30
```

KUVIO 5. Käyttäjän käyttöoikeuksien tarkastaminen

3.6.1 Datan validointi ja siistiminen

Datan validointi tarkoittaa sen tarkistamista siten, että se täyttää halutut kriteerit. Validointi tulisi tehdä aina ennen kuin kyseistä dataa käytetään. Mikäli käyttäjän lomakkeessa syöttämät tiedot eivät ole valideja, voidaan käyttäjä joko ohjata korjaamaan virheellisesti syötetyt tiedot lomakkeessa, tai ne voidaan siistiä/korjata automaattisesti. (Harris 2012, viitattu 06.09.2018.)

Ennen kuin data tallennetaan tietokantaan tai tulostetaan sivustolle, tulisi se siistiä. Tämä tarkoittaa sitä, että siitä poistetaan (filtteröidään) kaikki ei halutut osat ja näin ollen tehdään datasta turvallista. Datan siistimisellä pyritään esimerkiksi estämään se, ettei tulostettava teksti sisällä HTML-koodia tai ettei tietokantaan tallennettava data sisällä SQL-lausekkeita. Useat WordPressin funktiot huolehtivat itsessään datan siistimisestä. Tällöin datan siistiminen erikseen ei ole tarpeellista. (Harris 2012, viitattu 06.09.2018.)

3.6.2 WordPressin noncet

WordPressin noncet auttavat suojaamaan sivustoa monenlaisilta hyökkäyksiltä, kuten esimerkiksi CSRF -hyökkäyksiltä (Cross Site Request Forgery). Nonce on WordPressin luoma kirjaimista ja numeroista koostuva vaikeasti arvattava merkkijono, jota voidaan hyödyntää esimerkiksi lomakkeissa ja linkeissä, jotta voidaan varmistaa, että suoritettava toiminto on käyttäjän aikomuksen mukainen. (WordPress 2018i, viitattu 06.09.2018.)

WordPress tarjoaa valmiita funktioita esimerkiksi noncen sisältävän linkin luomiseksi tai noncen sisältävän kentän lisäämisen lomakkeeseen. Myös esimerkiksi AJAX-pyyntöjä on mahdollista ja hyvä suojata noncella (kuviot 6A ja 6B). (WordPress 2018i, viitattu 06.09.2018.)

```
5
6   public function init() {
7
8       wp_enqueue_script(
9           'adminscripts',
10          plugins_url( '../admin/js/adminscripts.js', __FILE__ ),
11          array('jquery', 'jquery-ui-core', 'jquery-ui-datepicker', 'jquery-ui-sortable'),
12          time(),
13          true
14      );
15
16      wp_localize_script( 'adminscripts', 'my_ajax_obj', array(
17          'ajax_url' => admin_url( 'admin-ajax.php' ),
18          'nonce' => wp_create_nonce( 'my_nonce' ),
19      ));
20
21      add_action( 'wp_ajax_Atr_get_reservations', array($this, 'get_reservations' ));
22
23  }
24
25  public function get_reservations() {
26      check_ajax_referer('my_nonce', '_ajax_nonce');
27
28      /* Return reservations
29  }
30
```

KUVIO 6A. AJAX-pyynnöissä käytettävän noncen luonti ja tarkistus PHP-luokassa

```
28
29
30  $.post(my_ajax_obj.ajax_url, {
31      _ajax_nonce: my_ajax_obj.nonce,
32       action: "Atr_get_reservations",
33       date : date,
34  }, function(data) {
35       /* Do Something with data */
36  });
37
38
```

KUVIO 6B. Noncen lähettäminen AJAX-pyynnön parametrinä

3.7 Tietokannan hallinta

WordPressissä on määriteltynä luokka nimeltään wpdb, jonka päätarkoitus on toimia rajapintana WordPressin tietokannalle. WordPress tarjoaa globaalin $wpdb-nimisen instanssimuuttujan, joka on wpdb-luokan instanssi. Oletuksena wpdb-luokan instanssi ottaa yhteyttä WordPressin tietokantaan, mutta wpdb-luokasta voidaan luoda myös instanssi ottamaan yhteyttä johonkin muuhun tietokantaan. (WordPress 2018c, viitattu 10.09.2018.)

Tähän globaaliin $wpdb-instanssiin pääsee käsiksi märittelemällä $wpdb globaalina muuttujana, käyttämällä avainsanaa ”global”. Tätä kyseistä instanssia ei voida pelkästään käyttää WordPressin mukana oletuksena tuleviin tauluihin, vaan myös esimerkiksi lisäosan luomiin tauluihin (kuvio 7). (WordPress 2018c, viitattu 10.09.2018.)

```
 9
10  public function remove_unconfirmed_reservations() {
11
12      global $wpdb;
13
14      $wpdb->query( "DELETE FROM " . $this->table3 .
15                    " WHERE confirmation = 0 " .
16                    " AND time_created < DATE_SUB(NOW(), INTERVAL 15 MINUTE)"
17      );
18
19  }
20
21
```

KUVIO 7. Tietokantakysely lisäosassa määriteltyyn tauluun $wpdb-instanssimuuttujaa käyttäen

Monet wpdb-luokan metodeista hoitavat automaattisesti SQL-lausekkeissa käytettävän datan siistimisen tietoturvalliseksi, ja silloinkin kun tilanne ei ole tämä, voi kehittäjä käyttää wpdb-luokan prepare-metodia. Tämä kyseinen metodi huolehtii datan siistimisestä poistaen siitä mahdolliset SQL-injektiot (kuvio 8). (WordPress 2018c, viitattu 10.09.2018.)

```
$result = $wpdb->get_row(
    $wpdb->prepare(
        "SELECT * FROM " . $this->table1 . " WHERE name = %s ",
        $room_name
    )
);
```

KUVIO 8. Esimerkki wpdb-luokan prepare-metodin käytöstä

3.7.1 Tietokantataulujen luonti ja päivitys

Lisäosat voivat luoda omia tietokantatauluja lisäosan aktivoinnin yhteydessä hyödyntämällä WordPressin register_activation_hook() -funktiota, jolla voidaan laukaista lisäosan funktioita sen aktivoinnin yhteydessä. Tauluja luovia SQL -lausekkeita ei syötetä suoraan tietokantaan vaan ne sidotaan muuttujaan ja lähetetään parametrinä dbDelta-nimiselle funktiolle. Tämä kyseinen funktio sijaitsee upgrade.php -tiedostossa, mutta koska WordPress ei lataa oletuksena tätä tiedostoa, tulee se ladata erikseen ennen kuin dbDelta-funktiota voidaan käyttää (kuvio 9). (WordPress 2018d, viitattu 10.09.2018.)

```
public static function plugin_activate() {

    global $wpdb;
    $atr_db_version = 1.0;

    $table_name1 = $wpdb->prefix . "atr_room";
    $sql = "CREATE TABLE " . $table_name1 . "(
        id int NOT NULL AUTO_INCREMENT PRIMARY KEY,
        name varchar(25) UNIQUE NOT NULL,
        xwidth int NOT NULL,
        ywidth int NOT NULL,
        unactiveitems blob
    );";

    require_once(ABSPATH . "wp-admin/includes/upgrade.php");
    dbDelta($sql);

    add_option("atr-plugin-db-version", $atr_db_version);

}
```

KUVIO 9. WordPress-lisäosan tietokantataulujen luonti

Edellä mainittu dbDelta-funktio tutkii senhetkistä tietokannan taulurakennetta ja vertaa sitä haluttuun taulurakenteeseen, jonka perusteella se joko luo taulun tai muokkaa jo olemassa olevaa taulua. Kun taulujen luontilausekkeet on syötetty, tallennetaan tietokantaan wp_options -tauluun tietokannan versionumero add_option -funktiolla (kuvio 9). (WordPress 2018d, viitattu 10.09.2018.)

Jotta tietokantaa voidaan päivittää lisäosan päivitysten yhteydessä, tulee lisäosan aktivoinnin yhteydessä suoritettavaan funktioon tehdä joitain muutoksia (Vrt. kuvio 9 ja kuvio 10). Uusimmasta tietokannan versionumerosta tulee tehdä globaali muuttuja, jotta voidaan verrata sitä tietokannassa olevaan versionumeroon ja mikäli ne eroavat toisistaan, ajetaan tällöin uudet taulujen luontilausekkeet. Tietokannassa oleva tietokannan versionumero voidaan hakea get_option() -funktiolla. Uuden tietokannan versionumeron tallennus tulee tehdä add_option() -funktion sijaan update_option() -funktiolla, joka joko lisää tiedon wp_options -tauluun tai päivittää jo olemassa olevaa tietoa. (WordPress 2018d, viitattu 10.09.2018.)

```
4
5   global $atr_db_version;
6   $atr_db_version = 1.0;
7
8   public static function plugin_activate() {
9
10      global $wpdb;
11      $installed_db_version = get_option("atr-plugin-db-version");
12
13      if ( $installed_db_version != $atr_db_version ){
14
15          $table_name1 = $wpdb->prefix . "atr_room";
16          $sql = "CREATE TABLE " . $table_name1 . "(
17              id int NOT NULL AUTO_INCREMENT PRIMARY KEY,
18              name varchar(25) UNIQUE NOT NULL,
19              xwidth int NOT NULL,
20              ywidth int NOT NULL,
21              unactiveitems blob
22          );";
23
24          require_once(ABSPATH . "wp-admin/includes/upgrade.php");
25          dbDelta($sql);
26
27          update_option("atr-plugin-db-version", $atr_db_version);
28      }
29
30  }
31
```

KUVIO 10. WordPress-lisäosan tietokantataulujen luonti päivitysmahdollisuudella

Koska uudet tietokantataulut luova funktio kutsutaan vain lisäosan aktivoinnin yhteydessä, tulee plugins_loaded -nimiseen toimintakoukkuun lisätä toimintafunktio jolla tarkistetaan, että tietokanta

on ajan tasalla. Mikäli globaalin tietokantaversiomuuttujan arvo eroaa tietokannassa olevasta, ajetaan tällöin päivityksen tekevä funktio (kuvio 11). (WordPress 2018d, viitattu 10.09.2018.)

```
6
7   public function init() {
8       add_action( 'plugins_loaded', array($this, 'atr_update_db_check') );
9   }
10
11  public function atr_update_db_check() {
12      global $atr_db_version;
13      if ( get_site_option( 'atr_plugin_db_version' ) != $atr_db_version ) {
14          $this->plugin_activate();
15      }
16  }
17
18  public static function plugin_activate() {
19      .
20      .
21      .
22  }
23
```

KUVIO 11. Tietokannan version tarkastus

3.8 Asennus ja poisto

WordPress tarjoaa omat ”hookit” lisäosan aktivoinnin ja deaktivoinnin yhteydessä suoritettaviin toimintoihin. Edellisessäkin kappaleessa mainittu register_activation_hook() -funktio on tarkoitettu lisäosan aktivoinnin yhteydessä suoritettavien toimenpiteiden ajamiseen, kuten esimerkiksi tietokantataulujen luontiin tai oletusarvojen tallentamiseen tietokantaan. Lisäosan deaktivoinnin yhteydessä suoritettavien toimenpiteiden ajaminen onnistuu funktiolla register_deactivation_hook(). Näitä toimenpiteitä ovat esimerkiksi väliaikaisen datan, tiedostojen tai kansioiden poistaminen. Molemmat edellä mainituista funktioista ottavat vastaan parametrinä viittauksen lisäosan päätiedostoon ja ajettavan callback-funktion nimen. (WordPress 2018o, viitattu 10.09.2018.)

Lisäosan poiston yhteydessä on hyvä poistaa kaikki sen lisäämät asetukset wp_options -taulusta sekä mahdolliset taulut, joita lisäosa on lisännyt. Tähän löytyy kaksi eri lähestymistapaa, joko käyttäen register_uninstall_hook() -funktiota tai luomalla lisäosan kansiorakenteen juureen uninstall.php -nimisen tiedoston. Ensimmäinen vaihtoehdoista toimii samalla periaatteella kuin edellisessä kappaleessa mainitut, nimeltään samankaltaiset funktiot. Tällöin lisäosan poiston yhteydessä halutut toimenpiteet suoritetaan callback-funktiossa. Toista vaihtoehtoa käytettäessä taas halutut toimenpiteet suoritetaan suoraan uninstall.php -tiedostossa, jonka WordPress ajaa auto-

maattisesti, kun lisäosa poistetaan. Uninstall.php -tiedostossa tulisi ensimmäisenä ennen kuin mitään suoritetaan tarkastaa, että WP_UNINSTALL_PLUGIN -niminen vakio on määriteltynä WordPressin toimesta, jotta saadaan estettyä suora pääsy tiedoston ajamiseen (kuvio 12). (WordPress 2018p, viitattu 10.09.2018.)

```
 9   if (!defined('WP_UNINSTALL_PLUGIN')) {
10       die;
11   }
12
13   delete_option("atr-plugin-db-version");
14
15   global $wpdb;
16
17   $table_name = $wpdb->prefix . "atr_settings";
18   $sql = "DROP TABLE IF EXISTS $table_name;";
19   $wpdb->query($sql);
```

KUVIO 12. Esimerkki uninstall.php -tiedostosta

4 TOIMEKSIANTAJA JA TYÖN TAVOITE

Tämän opinnäytetyön toimeksiantajana toimii kempeleläinen mainostoimisto Asema10 Oy. Kyseessä on vuonna 2015 perustettu yritys, jonka palveluihin kuuluvat mm. WordPress-suunnittelu, graafinen suunnittelu, valokuvaus, videotuotanto sekä hakukoneoptimointi.

Itse työn tavoitteena oli toimeksiantajan toiveesta toteuttaa ravintoloille suunnattu pöytävarausjärjestelmä lisäosana WordPressiin. Toimeksiantajan toiveet lisäosalle olivat seuraavat:

- Ylläpitäjä voi luoda saleja ja pöytiä WordPressin hallintapaneelin puolella ja näiden luontiin tulee olla graafinen hallintanäkymä.
- Pöytävarauksille tulee olla päiväkohtainen näkymä, josta käy ilmi seuraavat tiedot: sali, pöydän numero, kellonaika, varauksen nimi ja varauksen koko.
- Sivuston puolella tulee olla lomake jolla asiakkaat voivat tehdä pöytävarauksia. Lomakkeesta käy ilmi ajankohdat jolloin pöytiä on varattavissa ja vain vapaita pöytiä on mahdollista varata.
- Mahdollisuus priorisoida pöytien varaamista niiden sijainnin mukaan.
- Lisäosan tulee osata tehdä varaukset pöydän koon ja priorisoinnin mukaan.

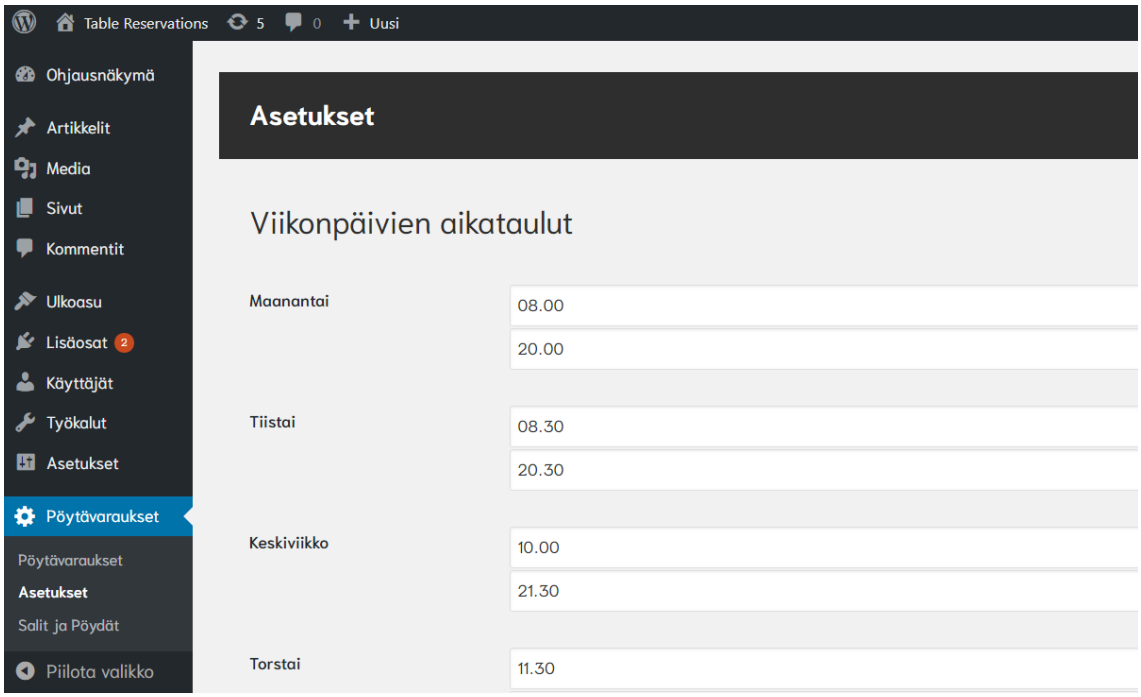
Toimeksiantajalla olisi ollut työlle muitakin vaatimuksia, mutta työn laajuuden ja työhön käytetyn työmäärän vuoksi joitain ominaisuuksia täytyi jättää myöhemmin kehitettäväksi. Nämä ominaisuudet on huomioitu lisäosan kehityksessä, jotta niiden integrointi myöhemmin olisi mahdollisimman vaivatonta.

5 PÖYTÄVARAUSJÄRJESTELMÄN TOTEUTUS

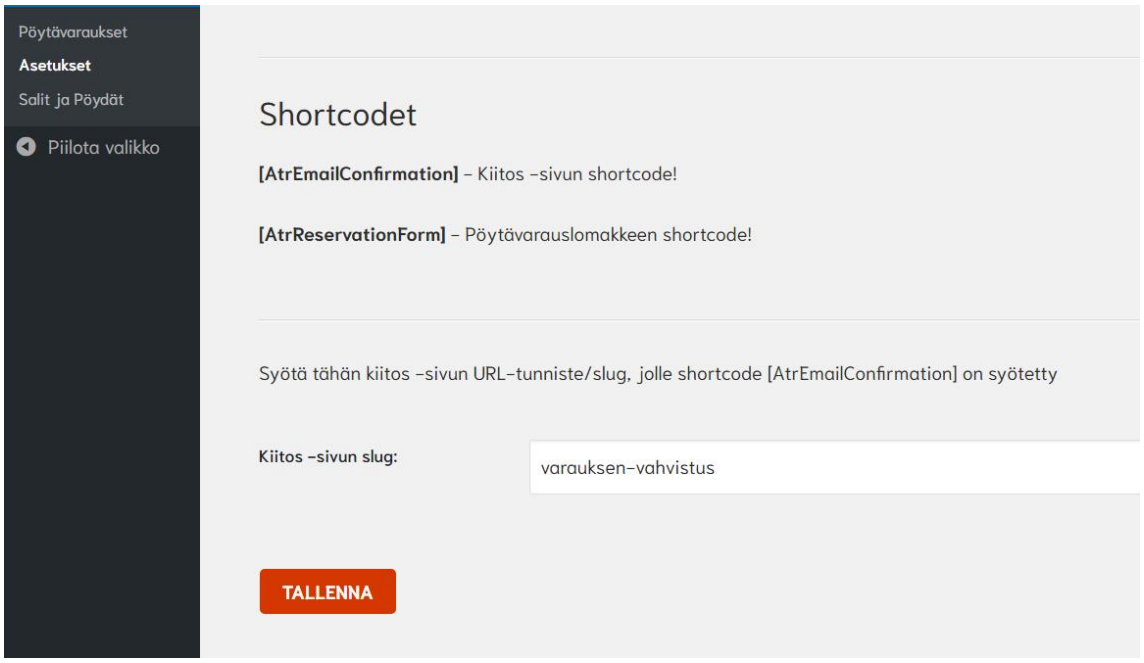
Lisäosaa lähdettiin kehittämään ajatuksella, että se koostuisi kolmesta eri sivusta hallintapaneelin puolella: salien ja pöytien luontiin tarkoitettu sivu, varausten tarkasteluun päiväkohtaisesti tarkoitettu sivu sekä asetusten sivu. Itse sivuston puolelle tuleva pöytävarauslomake toteutettiin hyödyntämällä WordPressin Shortcode API:a eli siitä tehtiin lyhytkoodi. Näin lisäosan käyttäjä voisi itse lisätä lomakkeen mille tahansa haluamalleen sivulle.

5.1 Asetukset-sivu

Asetukset -sivulta löytyvät kaikki lisäosan toiminnan kannalta tarpeelliset käyttäjän asetukset, sekä lyhytkoodit. Tämän opinnäytetyön osalta asetukset-sivulle ei tehty muita käyttäjän asetuksia, kuin varausten aikavälit viikonpäivittäin (kuvio 13A), sekä kiitossivun sivutunniste, jolle asiakas ohjataan pöytävarauksen yhteydessä tehtävän sähköpostivahvistuksen jälkeen. Kuviossa 13B näkyvät lisäosassa käytettävät lyhytkoodit sekä kiitossivun sivutunnisteen tekstikenttä.



KUVIO 13A. Asetukset-sivun lomake aikavälien asettamiseen viikonpäivittäin



KUVIO 13B. Asetukset-sivun lyhytkoodit ja kiitossivun slug.

Viikonpäivien aikataulujen tallentamiseen on hyödynnetty WordPressin Options API:sta löytyvää add_option() -funktiota. Kyseisillä aikatauluilla tarkoitetaan aikaväliä, jolle varauksia voidaan tehdä ja ne määritellään päiväkohtaisesti jokaiselle viikonpäivälle.

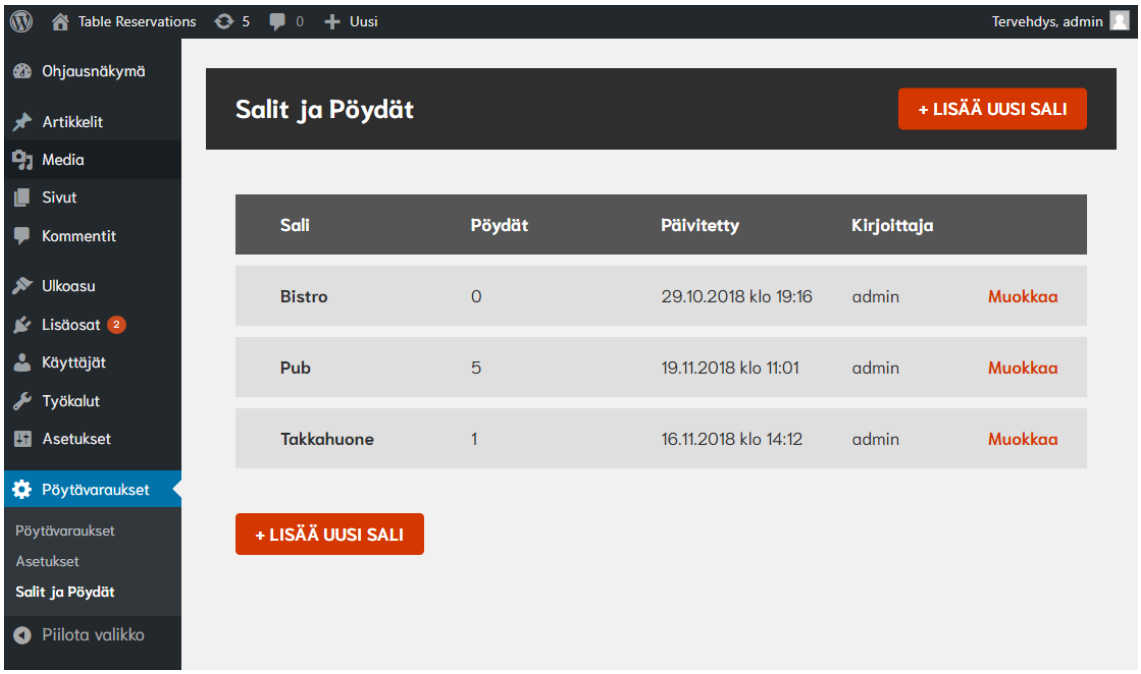
Asetukset-sivulle on tarkoitus myöhemmin lisätä kalenteri, jolla lisäosan käyttäjä voi tarpeen tullen lisätä poikkeusaikatauluja tai merkitä ravintolan suljetuksi tiettynä päivänä. Sivuston puolella käytössä oleva pöytävarauslomakkeen Datepicker-kalenteri tukee jo tätä ominaisuutta, eli siihen voidaan määrittää päiviä jotka eivät ole valittavissa.

Pöytävarausten yhteydessä lähetettävän sähköpostin muotoilut ja sisältö, vahvistuslinkkiä lukuun ottamatta, ovat tarkoitus myös tulla myöhemmin asetukset sivulle muokattavaksi. Tällä hetkellä vahvistusviestin sisältö on yksinkertainen teksti ja linkki, eikä sitä ole muotoiltu millään tavoin.

5.2 Salit ja pöydät -sivu

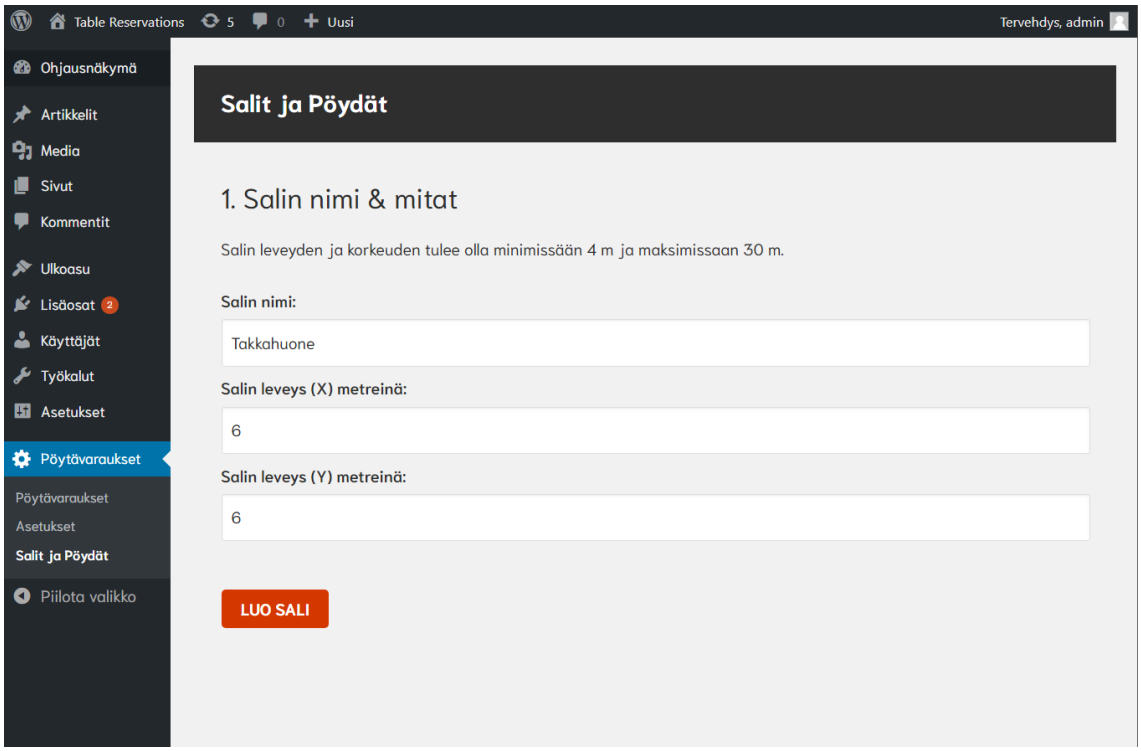
Salit ja pöydät -sivu on tarkoitettu ravintolassa pöytävarauksiin käytettävien tilojen ja pöytien luomiseen. Sen päänäkymässä on listaus tallennetuista saleista (kuvio 14). Listauksessa näkyvät sa-

lin nimi, pöytien määrä, milloin saliin on viimeksi tehty muutoksia ja kuka nämä muutokset on tehnyt. Lisäksi listassa on jokaiselle salille oma painike, jolla pääsee muokkaamaan salin sisältöä eli pöytiä. Päänäkymässä on myös kaksi painiketta, joista molemmista pääsee luomaan uutta salia.

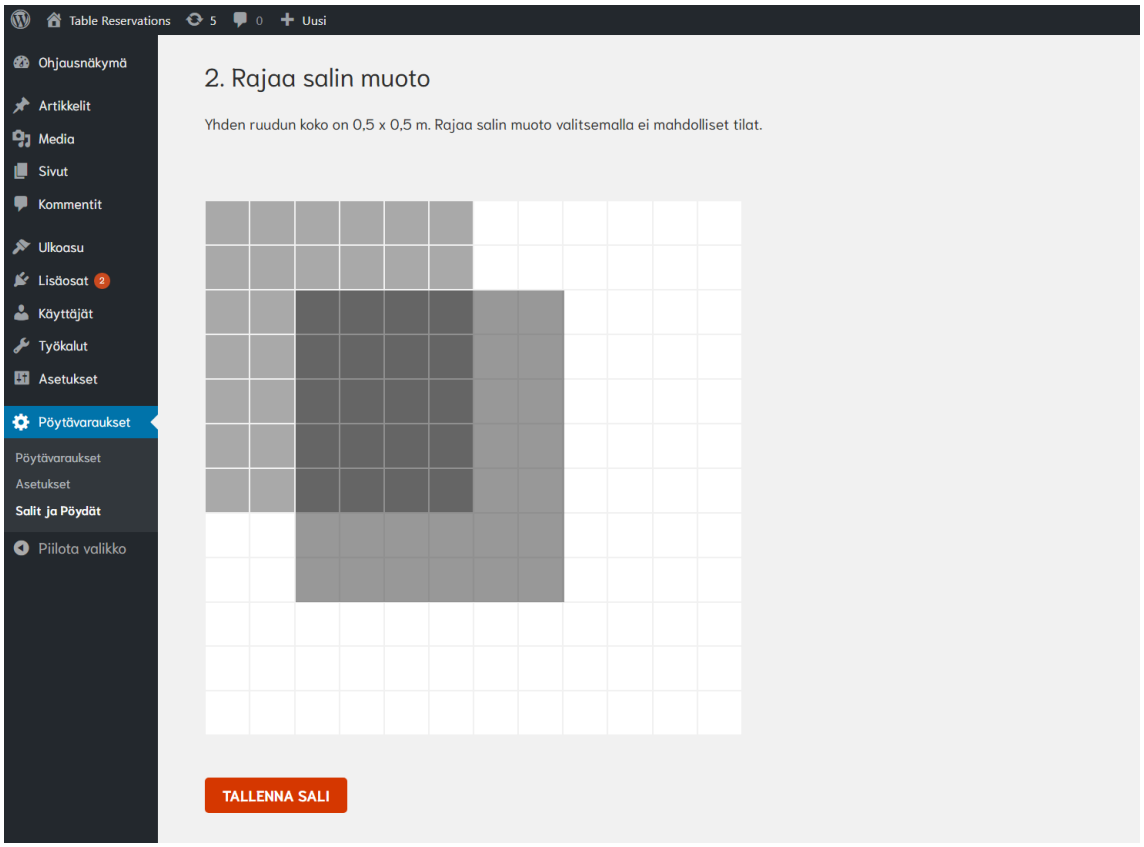


KUVIO 14. Salit ja pöydät -sivun päänäkymä

Uuden salin luontiin tarkoitettu näkymä on kaksivaiheinen. Ensimmäisessä vaiheessa näkyy lomake johon syötetään uuden salin nimi sekä salin mitat (x,y) metreinä (kuvio 15A). Tämän jälkeen näkyviin tulostuu ruudukko käyttäjän antamien mittojen perusteella, jossa yhden ruudun koko on 0.5m (kuvio 15B). Tässä vaiheessa käyttäjä voi muotoilla salin pohjapiirustuksen muodon haluamakseen. Tämä onnistuu maalaamalla harmaaksi alueet, jotka eivät ole käytettävissä, kuten esimerkiksi seinät ja baaritiskit.



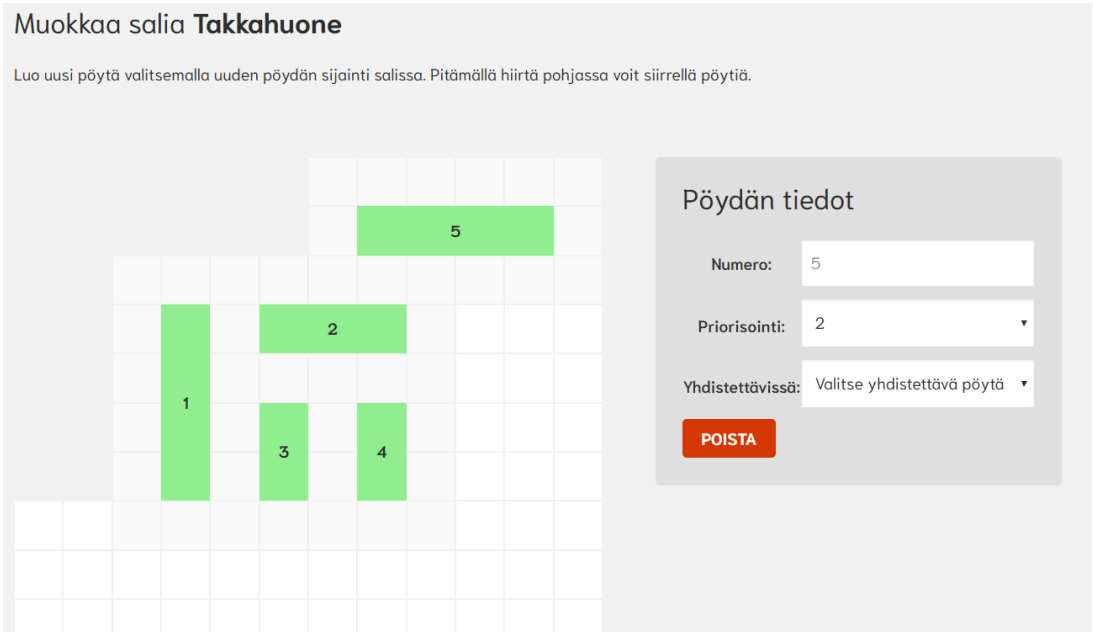
KUVIO 15A. Salin luonti. Vaihe 1



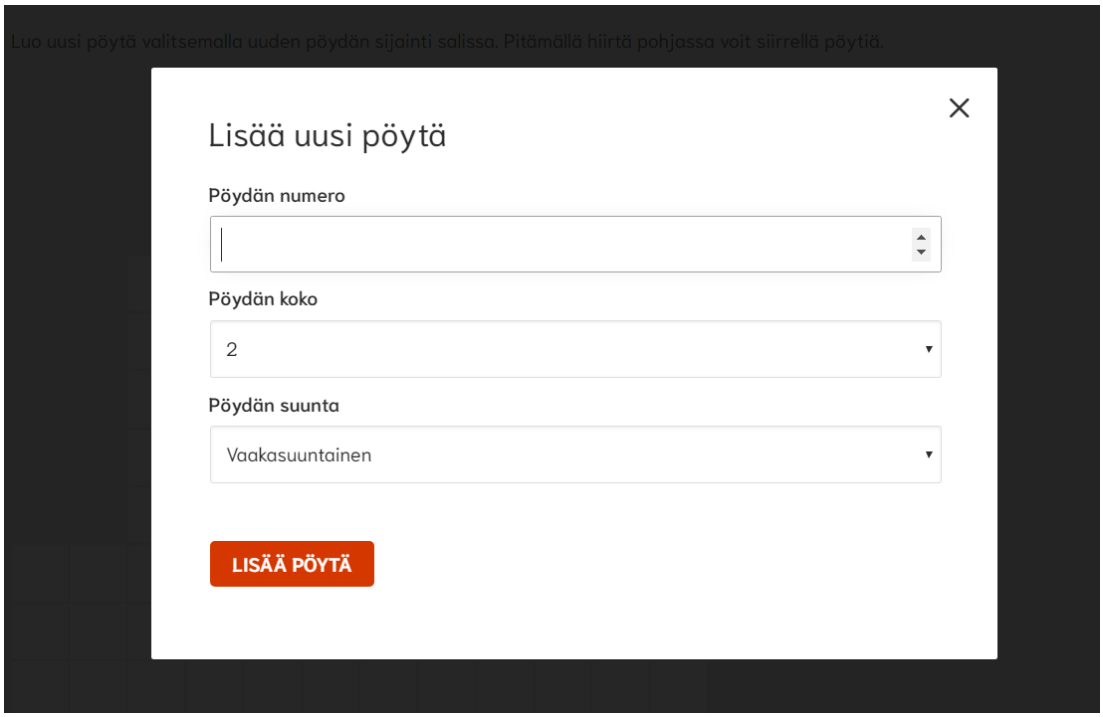
KUVIO 15B. Salin luonti. Vaihe 2

Ruudukon maalaaminen on toteutettu siten, että sillä voi poistaa ja lisätä harmaita alueita yksinkertaisesti pitämällä hiiren painiketta pohjassa ja valitsemalla hiirellä alue joka halutaan maalata harmaaksi tai palauttaa valkoiseksi. Mikäli valitun alueen sisällä on harmaita ruutuja, tällöin kaikki valitun alueen sisällä olevat ruudut palautetaan valkoiseksi, kun hiiren painike vapautetaan. Muutoin valitun alueen ruudut maalataan harmaiksi. Ruudukon maalaaminen onnistuu myös, vaikka hiiren kursori vietäisiin ruudukon ulkopuolelle. Alueen maalaaminen toimii kaikista sulavimmin Chrome -selaimella, koska se on kehitetty käyttäen tätä selainta, joten se tulee myöhemmin vielä optimoida muillekin selaimille.

Kun sali on tallennettu, avautuu automaattisesti kyseisen salin muokkausnäkymä (kuvio 16). Tässä kyseisessä näkymässä käyttäjä voi lisätä pöytiä valitsemalla hiirellä ruudun, johon haluaa uuden pöydän luoda. Tämä avaa popup -ikkunan, joka sisältää lomakkeen pöydän lisäämiseen (kuvio 17). Lomakkeeseen tulee syöttää pöydän numero, pöydän koko ja pöydän suunta. Jokainen pöytä varaa ympäriltään yhden ruudun verran tilaa, joten niitä ei voi asettaa toisiinsa kiinni.



KUVIO 16. Salin muokkausnäkymä



KUVIO 17. Uuden pöydän popup -lomake

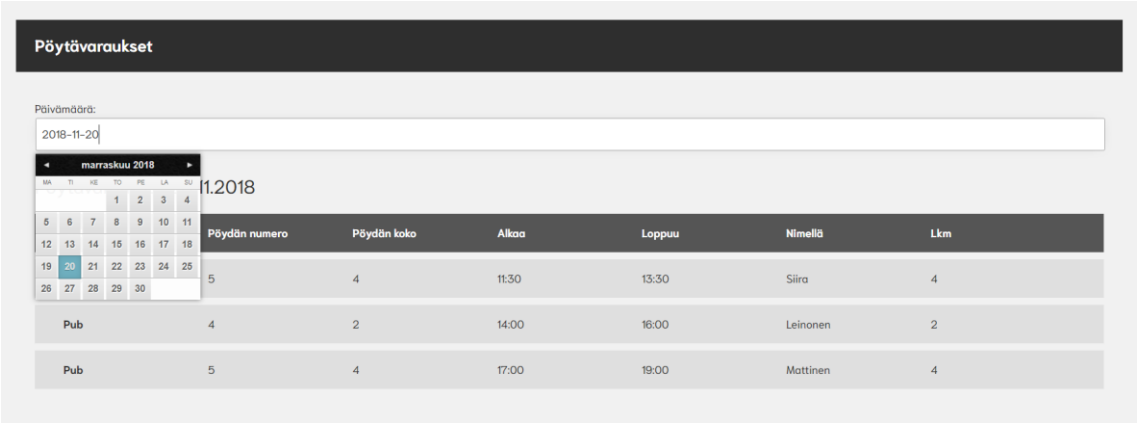
Käyttäjä voi liikutella pöytiä näkymässä pitämällä hiirtä pohjassa pöydän päällä. Pöytiä voi siirtää vain paikkoihin, joissa pöydän ympärille jää yhden ruudun verran tilaa, paitsi mikäli halutaan asettaa pöytä seinään kiinni.

Painamalla pöytää hiiren painikkeella avautuu ruudukon viereen lomake pöydän tiedoista (kuvio 16). Lomake sisältää kolme kenttää: pöydän numero, pöydän priorisointiarvo sekä yhdistettävien pöytien valintakenttä. Lisäksi lomake sisältää painikkeen, jolla voi poistaa kyseisen pöydän. Pöydän priorisointiarvon avulla käyttäjä voi merkitä mitä pöytiä halutaan priorisoida varausta tehtäessä. Priorisointiarvo on luku yhden ja kolmen välillä, yhden ollessa korkea ja kolmen ollessa matala.

Yhdistettävien pöytien valintakentästä käyttäjä voi valita pöydän, jonka kanssa haluaa kyseisen pöydän olevan yhdistettävissä. Tämä tarkoittaa sitä, että mikäli joku yrittää esimerkiksi tehdä 8 hengen varausta ja kaikki 8 hengen pöydät ovatkin varattuja, tarkastaa järjestelmä tällöin onko vapaana kahta, yhdistettävissä olevaa neljän hengen pöytää.

5.3 Pöytävaraukset-sivu

Pöytävaraukset -sivulta voi käyttäjä selailla päiväkohtaisesti kaikkia pöytävarauksia, jotka on vahvistettu sähköpostitse. Tässä näkymässä käyttäjä voi valita Datepicker-kalenterista päivän jota haluaa tarkastella. Kun päivä on valittu, tulostuu näkymään listaus kaikista varauksista aikajärjestyksessä (kuvio 18). Listauksessa käy ilmi salin nimi, pöydän numero, pöydän koko, varauksen alkamis- ja päättymisaika, nimi jolla varaus on tehty sekä henkilöiden lukumäärä.

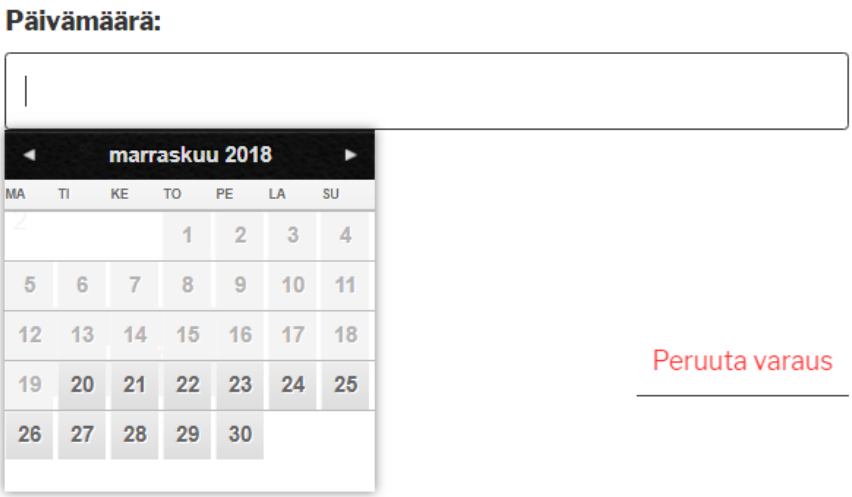


KUVIO 18. Pöytävaraukset-sivun näkymä

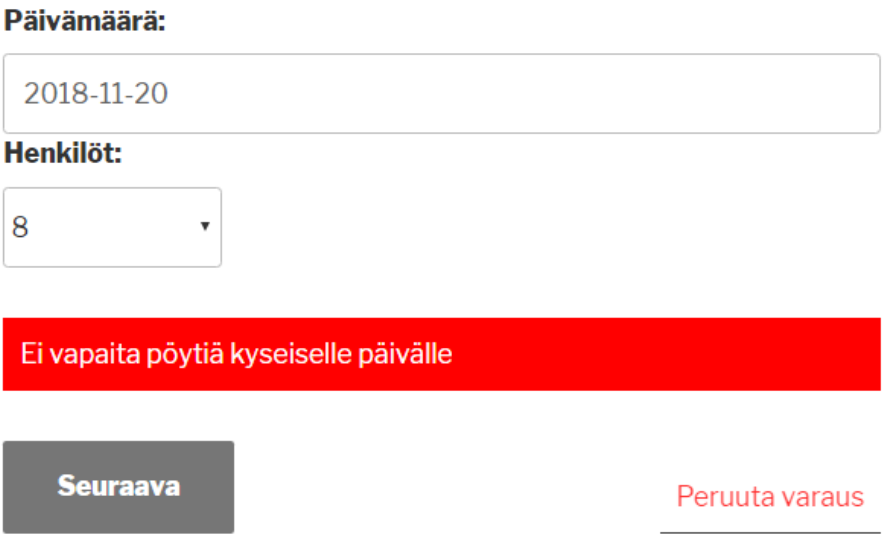
Pöytävaraukset -näkymään on tarkoitus lisätä myöhemmin linkit varausten poistamiseen ja muokkaamiseen käsin, sekä mahdollisuus tarkastella tämänhetkistä tilannetta graafisessa salinäkymässä. Salinäkymässä pöydistä olisi tarkoitus käydä ilmi varauksen nimi ja aika, sekä seuraavan varauksen nimi ja aika.

5.4 Pöytävarauslomake

Sivuston puolelle lyhytkoodina lisättävä pöytävarauslomake on kolmevaiheinen. Ensimmäisessä vaiheessa asiakas valitsee kalenterista päivän, jolle haluaa varauksen tehdä sekä henkilöiden lukumäärän (kuvio 19A). Mikäli annetuilla ehdoilla ei löydy vapaita pöytiä, annetaan käyttäjälle virheilmoitus (Kuvio 19B).



KUVIO 19A. Pöytävarauslomake. Vaihe 1

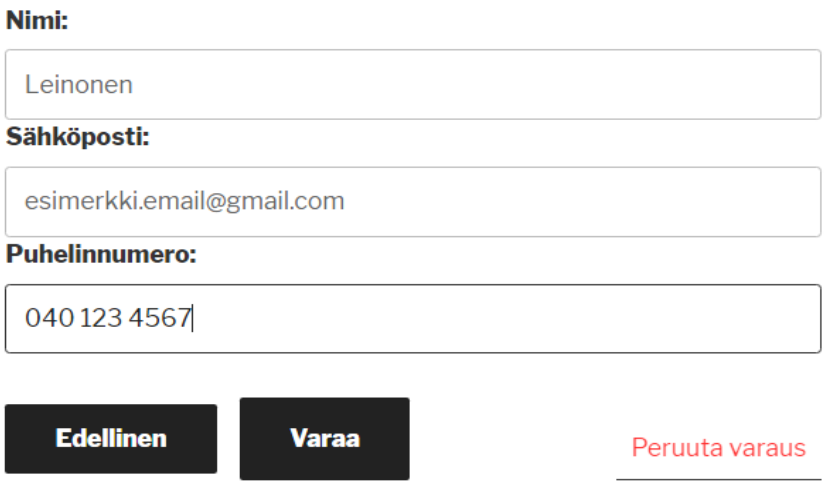


KUVIO 19B. Pöytävarauslomake. Vaihe 1, virheilmoitus käyttäjälle

Vaiheessa kaksi lomakkeeseen tulostuu ruudukko, josta käyttäjä voi valita pöytävarauksen alkamisajan (kuvio 20). Ruudukossa näkyy täyteen varatut ajankohdat harmaina ja niitä ei asiakas pysty valitsemaan. Järjestelmä varaa aikaa varaukselle kaksi tuntia alkamisajasta eteenpäin.

KUVIO 20. Pöytävarauslomake. Vaihe 2

Viimeisessä vaiheessa asiakkaan tulee syöttää nimi, sähköposti sekä puhelinnumero lomakkeeseen (kuvio 21). Painettuaan varaa -painiketta, lähetetään asiakkaalle sähköpostiviesti, joka sisältää varauksen vahvistuslinkin. Mikäli varausta ei vahvisteta 30 minuutin sisällä, poistuu se automaattisesti järjestelmästä ajastetun cron-tehtävän toimesta. Vahvistuslinkki ohjaa asiakkaan asetukset-sivulla määritetylle kiitossivulle.



KUVIO 21. Pöytävarauslomake. Vaihe 3.

6 LISÄOSAN RAKENNE JA TEKNIIKAT

Pöytävarausjärjestelmä -lisäosa on kehitetty PHP:llä, kuten kaikki WordPressin lisäosat. Lisäosan päätiedosto table-reservations.php on hyvin yksinkertainen (kuvio 22). Se sisältää ”file header” -osion, johon on merkitty lisäosan tietoja, kuten lisäosan nimi, kuvaus, tekijä ja versionumero. Nämä tiedot on merkitty kommenttimerkkien sisään nimi/arvo pareina. Lisäksi päätiedostossa haetaan lisäosassa käytettävät luokat, sekä hookataan ”plugins_loaded” -nimiseen toimintakoukkuun funktio, jossa luodaan instanssi lisäosan toiminnallisuuden sisältävästä ”tableReservations” -luokasta.

```php
<?php
/*
Plugin Name: Asema10 Table Reservations
Description: Restaurant table reservation service
Version: 1.0
Author: Jani Leinonen, Asema10
License: GPL2
*/

define ('PLUGIN_NAME', 'atr');
define( 'TR_PLUGIN_DIR', plugin_dir_path( __FILE__ ) );

require_once( TR_PLUGIN_DIR . 'class/class.atr-rest-controller.php' );
require_once( TR_PLUGIN_DIR . 'class/class.table-reservations.php' );

add_action('plugins_loaded', 'table_reservations_init');


function table_reservations_init() {

    $table_reservations = new tableReservations();

}

```

KUVIO 22. Lisäosan päätiedosto table-reservations.php

Lisäosan sivujen sisäinen toiminnallisuus, kuten painikkeiden toiminta, ruudukoiden tulostukset ym. on kaikki toteutettu Javascriptin JQuery-kirjastoa käyttäen. Sivuilla vaihtuvat sisällöt ja näkymät on toteutettu käyttäen JQueryn AJAX-tekniikkaa, jotta sisältö on dynaamista ja latausajat olisivat mahdollisimman lyhyet. Kaikki pyynnöt joita lisäosassa tarvitsee tehdä tietokantaan, tehdään aiemmin mainitusta ”tableReservations” -luokasta. Kyseisessä luokassa on määritelty toimintakoukut kaikille lisäosassa tarvittaville AJAX-pyynnöille, jotta luokan metodeja eli funktioita voidaan kutsua ilman, että tarvitaan sivunlatausta. Jokaisen AJAX-kutsun yhteydessä lähetetään nonce, joka tarkistetaan

kutsuttavan metodin sisällä WordPressin check_ajax_referer() -funktiolla. Tällä varmistetaan pyynnön alkuperä ennen kuin pyynnön prosessointia jatketaan.

Lisäosan tyylitiedostot on kirjoitettu käyttäen SASS -esiprosessori kieltä. Se mahdollistaa monipuolisemman ja helpommin hallittavan tavan kirjoittaa tyylitiedostoja. Se mahdollistaa mm. muuttujien käyttämisen sekä samanlaisen sisennyshierarkian luomisen tyylitiedostossa kuin html:ssä (kuvio 23).

```
/* ----------------------------------------
    ROOM LISTING
---------------------------------------- */

$atr-light: #ffffff;
$atr-dark: #2e2e2e;

#Atr-list-rooms {

   #List-heading {
       padding: 15px 40px; background-color: #555555;
       div.heading-item { display: inline-block; width: 25%; color: $atr-light; font-weight: bold;  }
    }

    div.list-item {
        position: relative; padding: 15px 40px; margin-top: 10px; background-color: #dfdfdf; color: $atr-dark;
        div:not(.wrapper-icon-modify) { display: inline-block; width: 25%; }
    }

    div.wrapper-icon-modify {
        display: inline-block; position: absolute; right: 26px; font-weight: 700;
        &:hover { cursor: pointer; }
    }

}
```

KUVIO 23. Esimerkki SASS -esiprosessorin käytöstä

Tietokantaan lisäosa lisää viisi uutta taulua sen asennuksen yhteydessä. Saleille, pöydille ja varauksille sekä yhdistettävissä oleville pöydille ja salin muokkaustietoja varten on kaikille omat taulunsa. Näiden taulujen luomiseen ja tietokannan kyselyihin käytetään SQL-kyselykieltä, jolla voidaan tehdä muutoksia tai hakuja relaatiotietokantoihin, kuten MySQL-tietokantaan jota WordPress käyttää.

7 POHDINTA

Ennen tätä opinnäytetyötä en ollut koskaan tehnyt lisäosaa WordPressiin, joten opiskelemista oli hyvin paljon itse WordPressin sekä sen lisäosien toiminnassa. WordPress kuitenkin tarjoaa todella hyvät dokumentaatiot kehittäjälle ja valmiit ohjelmointirajapinnat joita käyttää. Näiden avulla on helppo päästä jyvälle siitä, kuinka itse lisäosia toteutetaan.

Työn haastavuus ei tässä opinnäytetyössä kuitenkaan ollut niinkään siinä, kuinka toteuttaa toimiva lisäosa WordPressiin, vaan pikemminkin siinä kuinka toteuttaa toimiva pöytävarausjärjestelmä toimeksiantajan toivomilla ominaisuuksilla. JQueryllä tehdyt toiminnallisuudet kuten pöytien liikuttamiseen toteutettu ”Drag and Drop” -ominaisuus, sekä salien luontiin toteutetut ominaisuudet olivat työn haastavin osuus. Näiden toteuttamiseen kului selvästi eniten työtunteja, jotta ne toimivat sulavasti ilman toimintavirheitä.

Onnistuin pöytävarausjärjestelmän toteuttamisessa mielestäni suhteellisen hyvin ottaen huomioon työn laajuuden ja haastavuuden. Lopputuloksena oli pöytävarausjärjestelmä, joka toimii ja joka sisältää toimeksiantajan toivomat ominaisuudet. Valmis tuote se ei vielä kuitenkaan ole ja siinä on paljon lisäkehitettävää.

Koska lisäosan rakenne perustuu yhteen PHP:llä kirjoitettuun pääluokkaan, on se helposti hyödynnettävissä jatkossa, uusia lisäosia kirjoitettaessa. Luokka sisältää toimivan lisäosan rakenteen, joten sitä voi uudelleen käyttää pohjana muissa projekteissa.

LÄHTEET

Hayes, David 2018. WordPress Hooks, Actions, and Filters: What They Do and How They Work. Viitattu 23.08.2018, https://wpshout.com/wordpress-hooks-actions-filters-work/

Hayes, David 2015. Making an Admin Options Page With the WordPress Settings API. Viitattu 27.08.2018, https://wpshout.com/making-an-admin-options-page-with-the-wordpress-settings-api/

Hayes, David 2017. How to Make a WordPress Admin Options Page (Without Using the Settings API). Viitattu 28.08.2018, https://wpshout.com/wordpress-options-page/

Harris, Stephen 2012. Data Sanitization and Validation With WordPress. Viitattu 06.09.2018, https://code.tutsplus.com/articles/data-sanitization-and-validation-with-wordpress--wp-25536

Strojny, Drew 2014. WordPress user roles explained. Viitattu 15.08.2018, https://themefoundry.com/blog/wordpress-user-roles/

McFarlin, Tom 2013. Properly Setting Up WordPress Cron Jobs. Viitattu 31.08.2018, https://tommcfarlin.com/wordpress-cron-jobs/

WordPress. 2018a. About WordPress. Viitattu 15.08.2018, https://wordpress.org/about/

WordPress. 2018b. About WordPress. Features. Viitattu 15.08.2018, 16.08.2018, https://wordpress.org/about/features/

WordPress. 2018c. Codex. Class Reference/wpdb. Viitattu 10.09.2018, https://codex.wordpress.org/Class_Reference/wpdb

WordPress. 2018d. Codex. Creating Tables With Plugins. Viitattu 10.09.2018, https://codex.wordpress.org/Creating_Tables_with_Plugins

WordPress. 2018e. Codex. Options API. Viitattu 27.08.2018, https://codex.wordpress.org/Options_API

WordPress. 2018f. Codex. Plugin API. Viitattu 17.08.2018, https://codex.wordpress.org/Plugin_API

WordPress. 2018g. Codex. Settings API. Viitattu 27.08.2018, https://codex.wordpress.org/Settings_API

WordPress. 2018h. Codex. Shortcode API. Viitattu 29.08.2018, https://codex.wordpress.org/Shortcode_API

WordPress. 2018i. Codex. WordPress Nonces. Viitattu 06.09.2018, https://codex.wordpress.org/WordPress_Nonces

WordPress. 2018j. Codex. Writing A Plugin. Viitattu 16.08.2018, https://codex.wordpress.org/Writing_a_Plugin

WordPress. 2018k. Developer. Plugin Handbook. Viitattu 17.08.2018, https://developer.wordpress.org/plugins/hooks/actions/

WordPress. 2018l. Developer. Plugin Handbook. Viitattu 30.08.2018, https://developer.wordpress.org/plugins/cron/

WordPress. 2018m. Developer. Plugin Handbook. Viitattu 31.08.2018, https://developer.wordpress.org/plugins/cron/scheduling-wp-cron-events/

WordPress. 2018n. Developer. Plugin Handbook. Viitattu 06.09.2018, https://developer.wordpress.org/plugins/security/checking-user-capabilities/

WordPress. 2018o. Developer. Plugin Handbook. Viitattu 10.09.2018, https://developer.wordpress.org/plugins/the-basics/activation-deactivation-hooks/

WordPress. 2018p. Developer. Plugin Handbook. Viitattu 10.09.2018, https://developer.wordpress.org/plugins/the-basics/uninstall-methods/