Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Sebastian Söderholm

WordPress-verkkosivustojen tuotannon ja ylläpidon tehostaminen

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Ohjelmistotekniikka

Insinöörityö

1.3.2019

Tekijä Otsikko Sivumäärä Aika	Sebastian Söderholm WordPress-verkkosivustojen tuotannon ja muokkaamisen tehostaminen 30 sivua 26.4.2019
Tutkinto	insinööri (AMK)
Tutkinto-ohjelma	Tietotekniikan insinöörityötkinto
Ammatillinen pääaine	Ohjelmistotekniikka
Ohjaajat	Lehtori Ilpo Kuivanen

Insinöörityössä käsiteltiin tapoja toteuttaa WordPress-sivuja siten, että niiden ylläpito ja kehitys ovat mahdollisimman tehokkaita. Työn tavoitteena oli löytää ratkaisuja, joilla valmiin verkkosivuston ylläpito voitaisiin antaa käyttäjälle ilman, että tältä vaadittaisiin ohjelmointitaitoa.

Työ toteutettiin tarkastelemalla sekä WordPressin hallintapaneelin kautta tehtyjä ratkaisuja että ohjelmoimalla lisätoiminnallisuuksia. Lisäksi työssä käsiteltiin joitakin hyväksi todettuja tapoja tuottaa lisätoiminnallisuuttaa WordPressiin ilman, että ylläpito vaikeutuisi esimerkiksi päivitettäessä järjestelmää.

Työssä tutkittiin WordPressin sivuston rakentamista ja ylläpitoa sivunrakennustyökalun avulla. Lisäksi sivunrakennustyökaluun luotiin oma moduuli lisäosamuodossa ja tarkasteltiin sen käytettävyyttä käyttäjän näkökulmasta. Tämä todettiin hyväksi menetelmäksi luoda käyttäjäystävällinen tapa ylläpitää ja rakentaa sivua, sillä menetelmällä oli helppo luoda graafinen käyttöliittymä, joka ei vaadi ohjelmointitaitoa.

Työssä luotiin myös esimerkkinä WordPress-lisäosa ja sille yksinkertainen valikko WordPressin hallintapaneeliin. Tämän lisäksi tarkasteltiin WordPress REST APIn rakennetta ja käyttöä. Työssä käytiin myös läpi valmiiden lisäosien käyttöä sivuston rakentamisen, ylläpidon ja tietoturvallisuuden kannalta.

Työn tuloksena saatiin käsiteltyä tapoja, joita WordPress-verkkosivujen suunnittelussa voidaan hyödyntää luomaan helpommin ylläpidettävä järjestelmä. Jotkut tavoista todettiin mahdollistavan lisätoiminnallisuuksien luonnin käyttäjälle, joka ei tarvitse ohjelmointitaitoa. Toiset menetelmät todettiin hyödyllisiksi ohjelmoijan ylläpidon tehostamisen näkökulmasta.

Avainsanat	WordPress, verkkosivustojen suunnittelu

Author Title Number of Pages Date	Sebastian Söderholm Enhancing the creation and maintenance of WordPress-websites 30 pages 26 April 2019
Degree	Bachelor of Engineering
Degree Programme	Information technology
Professional Major	Software engineering
Instructors	Ilpo Kuivanen, Lecturer

This thesis covered methods on producing websites with WordPress in a way that enhances its production and maintenance. The purpose of the work was to find methods to create a product for the user, who could maintain the webpage without knowledge of programming.

The work was done by examining methods of extending WordPress' functionality both in its administration dashboard and programmatically. The work also covered some best practices of how to extend WordPress' functionality without creating conflicts when updating the system.

The work examined the building of WordPress websites and maintenance with page builder tools. The page builder was further extended with a custom module done as a WordPress plugin and the usability of the extension was examined from the perspective of the user. This method was regarded as a user friendly due to the possibility of easily creating a graphical user interface the use of which requires no knowledge of programming.

The work also covered an example of how to create a WordPress plugin and a simple menu for it in the WordPress dashboard. In addition, the architecture and use of the WordPress REST API was examined. The work also covered the use of already made plugins to enhance the production, maintenance and security of the website.

As a result, the work showed methods that can be utilized to create a more easily maintainable WordPress-website. Some of these methods were considered to enable the creation of functionality in a way that doesn't require programming skills to maintain. Other methods were concluded as useful for the programmer to create websites more effectively.

Keywords	WordPress, Website design

Sisällys

Lyhenteet

API	Application programming interface. Ohjelmointirajapinta, jonka avulla ohjelma voi kommunikoida toisen ohjelman kanssa.
CSS	Cascading Style Sheets. Tyyliohje, joka määrittä esimerkiksi HTML-tiedostojen tulosteen tyylitykseen.
GPL	GNU General Public License. Vapaiden ohjelmistojen julkaisemiseen tarkoitettu lisenssi.
GPLv2	GPL lisenssin toinen versio.
HTML	Hypertext Markup Language. Kuvauskieli, jolla verkkosivustojen rakenne ja sisältö määritellään.
JSON	JavaScript Object Notation. Tiedostomuodon standardi, joka on tarkoitettu erityiesti tiedonvälitykseen.
REST	Representational State Transfer. Arkkitehtuurimalli, joka perustuu HTTP-protokollaa.
SASS	Syntactically Awesome Style Sheets. Tyyliohje, joka lisää toiminnallisuuksia perinteiseen CSS-syntaksiin.
URL	Uniform resource locator. Verkossa löytyvän tiedon paikan määrittämä viite.
WP	WordPress. Avoimen lähdekoodin sovellus, jolla voidaan rakentaa verkkosivustoja.
WPML	WordPress Multilingual Plugin. WordPressin lisäosa, joka mahdollistaa monikielisten sivustojen luonnin.

1 Johdanto

WordPress on verkkosivustojen kehitysalusta, joka sai alkunsa 2003 Mike Littlen sekä Matt Mullenwegin johdosta nimellä B2/cafelog. WordPressiä kehitetään jatkuvasti ja sitä käyttää jo yli 33 % verkkosivustoista. (Our Story, verkkoaineisto.) WordPress on noussut suureen suosioon muun muassa sen helppokäyttöisyyden sekä monipuolisuuden kannalta, sillä sen käyttöliittymä mahdollistaa verkkosivujen toteutuksen ilman ohjelmointiosaamista, mutta samalla se on hyvin muokattavissa käyttäjän tarpeisiin sen avoimen lähdekoodin takia. Koska lähdekoodi on avoin, on WordPressille myös kehitetty tuhansia kolmannen osapuolen lisäosia, jotka mahdollistavat entistä laajempien lisätoiminnallisuuksien liittämisen järjestelmään. Vaikka se saikin alkunsa blogialustana, sitä käytetään nykyään suurempienkin verkkosivustojen ja verkkokauppojen, sekä jopa portaalien rakentamiseen. (Features, verkkoaineisto.)

Tässä työssä esittelemme tapoja, jolla WordPress-sivustojen tuotantoa voidaan tehostaa siten, että niistä saadaan mahdollisimman helposti ylläpidettäviä ylläpitokustannusten alentamiseksi. Käymme myös läpi, kuinka sivustojen eri osioita voidaan kehittää sille tasolle, että voidaan tarjota käytäjälle mahdollisimman helposti itse päivitettävä tuote. Tavoite on luoda tuote, joka on mukautettu käyttäjän tarpeiden mukaisesti, mutta samalla mahdollisimman helposti päivitettävä järjestelmä, jonka muokkaaminen ei vaadi käyttäjältä ohjelmointitaitoa. Tällöin voidaan valmiin sivuston mukana tarjota opastusta sekä kirjallinen ohje, jonka avulla saadaan vähennettyä ylläpitokustannuksia.

2 WordPress kehitysalustana

WordPress on ohjelmoitu PHP-kielellä, ja se käyttää MySQL- tai MariaDB-tietokantaa. WordPressin lähdekoodi on avoin GPLv2-lisenssin alla (Our Story, verkkoaineisto). Vaikka WordPress toimii PHP 5.2.4+- sekä MySQL 5.0+ versioilla, WordPress ei virallisesti enää kehitä alustaansa näille versioille. Sen sijaan viralliset suositukset WordPressin ajamiselle palvelimella ovat seuraavat:

- PHP-versio 7.3+
- MySQL-versio 5.6+ tai MariaDB versio 10.0+
- HTTPS-tuki. (Requirements, verkkoaineisto.)

2.1 Verkkosivustot

WordPress sisältää blogitaustansa vuoksi artikkeleiden (eng. posts) julkaisujärjestelmän, joka mahdollistaa niiden ajastetun julkaisemisen, luonnosten luonnin, näkyvyysmääreiden määrittelyt (myös salasanalla suojaus) sekä kategorioinnin (Features, verkkoaineisto). Artikkeli toimii WordPressissä objektina, josta voidaan luoda mukautettuja vaihtoehtoja (eng. custom post types) moneen eri käyttöön (Post Types, verkkoaineisto).

WordPress sisältää myös teema-järjestelmän, joka mahdollistaa sivuston visuaalisen ilmeen muokkaamista asennettavien lisäteemojen avulla. Teemat antavat myös mahdollisuuden muokata sivuston ulkonäköä WordPressin muokkausvalikon avulla, jossa voi muokata fontteja, valikoita, ylä- ja alareunaa, artikkeli- ja arkistosivun asettelua ja muita sivuston yleisiä ominaisuuksia.

2.2 Sivut

Sivut on yleisesti tarkoitettu näyttämään hierarkkista tietoa, joka ei ole riippuvaista julkaisuajasta. Tämän vuoksi WordPress-sivuja käytetään yleensä näyttämään tietoa, joka ei vanhene ja ne voidaan järjestää ala- ja yläsivuihin (Pages, verkkoaineisto). Sivuja ei kuitenkaan voi laittaa kategorioihin, kuten artikkeleita (What Pages Are Not, verkkoaineisto). Etusivun voi asettaa näyttämään blogisivun sijaan staattisen sivun WordPressin hallintapaneelista kohdasta Asetukset -> Lukeminen. Jos sivulle asettaa toisen sivun yläsivuksi WordPress, luo automaattisesti sivun URLin tyyliin www.sivustoni.fi/yläsivu/alasivu (To create a subpage, verkkoaineisto).

WordPress-sivut generoidaan dynaamisesti, eli ne voivat muuttua valittujen WordPress-asetusten mukaisesti. WordPress-sivu ei siis ole pelkkä staattinen HTML-sivu, vaan se hakee latauksen yhteydessä tietoa ladatusta sivusta, kuten teeman tyylityksiä, MySQL-

tietokannasta. Monesti kuitenkin WordPress-sivuja kutsutaan ”staattisiksi”, jos ne sisältävät vain staattista tietoa, mutta dynaamisen generoinnin takia niitä voitaisiin yhtä hyvin kutsua ”pseudo-staattisiksi” tai ”dynaamisiksi” (The Dynamic Nature of WordPress Pages, verkkoaineisto).

2.3 Luokittelut

WordPressissä luokittelu (eng. taxonomy) on tapa yhdistää artikkeleita ryhmiin. Luokittelujen nimiä kutsutaan arvoiksi (eng. terms). Jotkut luokittelut voivat olla hierarkkisia, kuten artikkelien kategoriat, eli kategoria voi sisältää toisen kategorian, joka voi sisältää toisen kategorian jne. WordPress sisältää oletuksena neljä luokittelua (Taxonomies, Verkkoaineisto):

- Kategoria (eng. Category): Mahdollistaa artikkeleiden ryhmityksen
- Avainsana (eng. Tag): Mahdollistaa myös artikkeleiden ryhmityksen. Avainsanoja käytetään yleensä vapaamuotoisemmin kuin kategorioita (esim. artikkeleiden hakutoimintoihin).
- Linkkikategoria (eng. Link Category): Mahdollistaa linkkien ryhmityksen (linkkikategorioita ei yleensä näytetä käyttäjälle).
- Artikkelimuoto (eng. Post Format): Osa artikkelin metadataa, joka on tarkoitettu käytettäväksi WordPressin teemoille, jotta artikkelin ulkoasua voidan muokata. (Taxonomies, Verkkoaineisto.)

2.4 Teemat

WordPress-teema muuttaa sivuston ulkonäköä ja antaa sivustolle mahdollisia lisätoimintoja. Teema ei kuitenkaan ole lisäosa ja hyvän ohjelmointitavan mukaisesti on suositeltavaa, että teeman ei tulisi määrittää kriittisiä toiminnallisuuksia sivulla. Tällöin käyttäjä voi vaihtaa teemaa ilman, että sivuston toiminnallisuus vaihtuu. Teeman tulisi siis määrittää ainoastaan sisällön näyttämiseen liittyviä toiminnallisuuksia (What is the difference between a theme and a plugin?, verkkoaineisto). Teeman tulee sisältää vähintään seuraavat tiedostot.

- Index.php – päätiedosto sivupohjan määrittämiseen
- Style.css – päätiedosto tyylityksen määrittämiseen.

(Required files, verkkoaineisto)

3 Verkkosivustojen tuotannon tehostamismenetelmät

3.1 Sivunrakennustyökalun käyttö

WordPressiin on viime vuosina ilmestynyt monia lisäosia, jotka mahdollistavat sivustojen rakentamisen tehokkailla menetelmillä, jotka eivät vaadi ohjelmointitaitoa. Näitä lisäosia kutsutaan usein sivunrakennustyökaluiksi (eng. Page Builder), ja ne sisältävät yleensä graafisen käyttöliittymän, jolla käyttäjä voi muokata sivustoa Drag & Drop -menetelmin. Monet sivunrakennustyökalut sisältävät myös valmiita komponentteja, joita voi suoraan raahata sivustolle ja muokata ilman ohjelmointitaitoa. Nämä työkalut tehostavat verkkosivujen luontia huomattavasti ja antavat mahdollisuuden tarjota käyttäjälle helposti ylläpidettävän järjestelmän.

Beaver Builder on sivunrakennustyökalu, joka tarjoaa graafisen käyttöliittymän suoraan sivulla, jota työkalulla muokataan. Kun halutaan muokata sivua Beaver Builderillä, siirrytään muokattavalta sivulta suoraan Beaver Builderin muokkausnäkymään WordPressin ylävalikkoriviltä. Muokkausnäkymässä voidaan muokata rivejä ja sarakkeita Drag & Drop -tyylillä. Jokainen rivi, jossa on sisältöä, käyttää hyväkseen moduulia, joka voi olla joko Beaver Builderin tai sen lisäosan oma moduuli tai WordPressin vimpain. Joitakin Beaver Builderin vakiomoduuleja ovat:

- Kuva – mahdollistaa WordPressin mediakirjastoon ladatun kuvan lisäämisen ja muokkaamisen sivulla.
- Tekstinmuokkain – mahdollistaa tekstin kirjoittamisen sivulle sekä graafisella tekstieditorilla että suoraan HTML-koodina.
- Kuvagalleria – mahdollistaa kuvagallerian luonnin WorpPressin mediakirjastoon lisätyistä kuvista, sekä joidenkin kuvagallerian toimintojen muokkaamisen.
- Kartta – luo Google Maps -kartan sekä karttamerkin moduulin muokkausnäkymässä kirjoitettuun osoitteeseen. Karttamoduuli käyttää Google Mapsin iframe-määrittelyä, eli se ei tarvitse Google Maps API -tunnusta.
- HTML – mahdollistaa HTML-koodin lisäämisen sivulle. Moduuli sisältää HTML editorin, joka värittää koodin luettavuuden parantamiseksi ja sisältää

joitakin toimintoja koodin kirjoittamisen helpottamiseksi, esimerkiksi automaattisen sulkutagin generoinnin. Editori sisältää myös yksinkertaisen virhetarkastuksen HTML koodille.

Beaver Builder sisältää myös CSS- ja JavaScript -muokkaimet. Nämä voidaan joko lisätä Beaver Builderin muokkausnäkymän valikosta kahdesta kohdasta: ”Sommitelman CSS / Javascript” sekä ”Yleiset asetukset”. Ensimmäinen vaihtoehto kääntää kirjoitetun CSS- ja JavaScript-koodin vain sivulle, jolla käyttäjä on. Toinen vaihtoehto on kääntää koodit joka sivulla. Tämä mahdollistaa yksinkertaisen optimoinnin sivustolle, jos CSS/JavaScript-koodia on paljon, sillä koodin, joka koskee vain yhtä sivua, voi sijoittaa ladattavaksi vain tälle yhdelle halutulle sivulle. Yleinen koodi koskee kaikkia sivuja, joilla Beaver Builder -lisäosa on asetettu toimimaan (Add custom Javascript, verkkoaineisto). Jos CSS halutaan toimimaan muilla sivuilla, kuten esimerkiksi artikkelisivulla, on joko Beaver Builder enabloitava myös artikkelisivulla tai CSS on kirjoitettava WordPressin teemavalikossa sijaitsevaan CSS-editoriin. Jos CSS-koodeissa on päällekkäin määritellyt muotoilut, ylikirjoittaa WordPressin teemavalikkoon kirjoitettu CSS Beaver Builderin ”Yleisen” CSS-koodin, joka puolestaan ylikirjoittaa Beave Builderin sivukohtaisen CSS-koodin.

Beaver Builderiin voi myös luoda omia moduuleja. Oman moduulin luomiseksi on suositeltavaa luoda oma WordPressin lisäosa, joka sisältää luodut moduulit, jotta WordPressiä päivitettäessä päivitysvalikon kautta nämä eivät poistu. Lisäosan luontia käsitellään myöhemmin. Luodaan lisäosalle kansio nimeltä mt-bb-modules ja sen sisään tiedosto mt-bb-modules.php. Lisäosan PHP-tiedostoon lisätään koodi: (Create a plugin, verkkoaineisto)

```
<?php

/**
 * Plugin Name: Custom Beaver Builder Modules
 * Plugin URI: https://www.musetech.fi/en
 * Description: Custom modules for the Beaver Builder Plugin.
 * Version: 1.0
 * Author: MuseTech
 * Author URI: https://www.musetech.fi/en
 */

define( 'MT_DATE_DIR', plugin_dir_path( __FILE__ ) );
define( 'MT_DATE_URL', plugins_url( '/', __FILE__ ) );

function mt_load_modules() {
    if ( class_exists( 'FLBuilder' ) ) {
        require_once 'mt-date/mt-date.php';
    }
```

```
}

add_action( 'init', 'mt_load_modules' );

?>
```

Esimerkkikoodi 1. Määritellään lisäosan perustiedot kommenttien avulla sekä lisäosan polkuihin viittaavat merkkijonot MT_DATE_DIR sekä MT_DATE_URL.

Esimerkkikoodissa 1 on tärkeää määritellä MT_DATE_DIR sekä MT_DATE_URL suffikseilla _DIR sekä _URL ja asettaa ne vastaamaan moduulin nimiavaruutta, joka riippuu luodun moduulin kansion ja sen tiedoston nimistä. Funktion mt_load_modules_() sisältö kaataa sivuston tässä vaiheessa, koska kansiota mt-date ja tiedostoa mt-date.php ei ole vielä luotu. Funktion sisällön voi kommentoida siksi aikaa, kunnes moduulin kansio ja tiedosto ovat luotuja ja valmiiksi määriteltyjä. Kommenttirivit määrittävät lisäosan metadatan, josta osa näkyy WordPressin lisäosat-valikossa. Lisäosa on aktivoitava WordPressin lisäosat-valikosta, jotta seuraavat muutokset tulevat voimaan.

Luodaan esimerkkinä yksinkertainen lisämoduuli Beaver Builderiin, joka näyttää automaattisesti päivittyvän päivämäärän ja mahdollistaa päivämäärän muodon muokkaamisen moduulin omasta valikosta. WordPressin lisäosan luonnin jälkeen tehdään moduulille oma kansio lisäosan kansion sisällä nimeltä ”mt-date” ja sen sisään samanniminen PHP-tiedosto ”mt-date.php”. Tämän tiedoston sisään asetetaan Beaver Builderin FLBuilderModule-luokan perivä luokka määrittelemään moduulin ominaisuudet Beaver Builderin valikossa: (Add a module to your plugin, verkkoaineisto.)

```
<?php

class MtDate extends FLBuilderModule {

    public function __construct()
    {
        parent::__construct(array(
            'name'            => __( 'MT Date', 'fl-builder' ),
            'description'     => __( 'Automatically updating date', 'fl-build-
er' ),
            'category'        => __( 'Custom', 'fl-builder' ),
            'group'           => __( 'MuseTech Modules', 'fl-builder' ),
            'dir'             => MT_DATE_DIR . 'mt-date/',
            'url'             => MT_DATE_URL . 'mt-date/',
            'icon'            => 'clock.svg',
            'editor_export'   => true, // Defaults to true and can be omitted.
            'enabled'         => true, // Defaults to true and can be omitted.
            'partial_refresh' => false, // Defaults to false and can be omit-
ted.
```

```
        ));
    }
}

?>
```

Esimerkkikoodi 2. Määritellään oma luokka, joka perii Beaver Builderin FLBuilderModule-luokan. Name- ja group-ominaisuudet näkyvät Beaver Builderin moduulivalikossa, mutta category-ominaisuutta ei vielä käytetä Beaver Builderissä (ehkä tulevaisuudessa). Icon-ominaisuus määrittää Beaver Builderin moduulivalikossa moduulin vieressä näkyvän ikonin. Tässä esimerkissä käytämme Beaver Builder -lisäosan wp-content/bb-plugin/img/svg kansiossa olevaa valmista ikonia.

Lopuksi rekisteröimme moduulin Beaver Builder -lisäosalle siten, että se näkyy moduulivalikossa. Esimerkkikoodissa 3 lisätään mt-date.php tiedostoon: (Define module settings, verkkoaineisto.)

```
FLBuilder::register_module( 'MtDate', array(
    'my-tab-1'      => array(
        'title'         => __( 'General', 'fl-builder' ),
        'sections'      => array(
            'section-format'  => array(
                'title'         => __( 'Format', 'fl-builder' ),
                'fields'        => array(
                    'select_date_format' => array(
                        'type'          => 'select',
                        'label'         => __( 'Select what will be shown',
'fl-builder' ),
                        'default'       => 'day',
                        'options'       => array(
                            'day'      => __( 'Show day', 'fl-builder' ),
                            'month'    => __( 'Show month', 'fl-builder' ),
                            'year'     => __( 'Show year', 'fl-builder' )
                        ),
                        'multi-select'  => true
                    ),
                )
                ),
            'section-styling'  => array(
                'title'         => __( 'Styling', 'fl-builder' ),
                'fields'        => array(
                    'text_align' => array(
                        'type'    => 'align',
                        'label'   => 'Text Align',
                        'default' => 'center',
                    ),
                    'font_style' => array(
                        'type'          => 'font',
                        'label'         => __( 'Font', 'fl-builder' ),
                        'default'       => array(
                            'family'        => 'Helvetica',
                            'weight'        => 300
                        )
                    ),
                    'text_color' => array(
                        'type'          => 'color',
                        'label'         => __( 'Text Color', 'fl-builder' ),
```

```
                        'default'       => '333333',
                        'show_reset'    => true,
                        'show_alpha'    => true
                    ),
                    'background_color' => array(
                        'type'          => 'color',
                        'label'         => __( 'Background Color', 'fl-build-
er' ),
                        'default'       => 'ffffff',
                        'show_reset'    => true,
                        'show_alpha'    => true
                    ),

                )
            )
        )
    )
) );
```

Esimerkkikoodi 3. Funktio register_module() vaatii kaksi parametria: rekisteröitävän moduulin luokan nimen sekä assosiatiivisen taulukkorakenteen, joka määrittelee moduulin valikon rakenteen ja asetukset.

Seuraavaksi luodaan moduulin ulkoasun määrittävä tiedosto, joka latautuu erikseen jokaiselle moduulin ilmentymälle. Luodaan moduulin kansioon uusi kansio nimeltä includes ja sen sisään tiedosto frontend.php. PHP-tiedoston sisään kirjoitetaan koodi: (Module HTML, verkkoaineisto.)

```
<?php
    $thisDay = date(d);
    $thisMonth = date(m);
    $thisYear = date(Y);

    if(in_array("day" , $settings->select_date_format)) {
        echo $thisDay . " ";
    }

    if(in_array("month" , $settings->select_date_format)) {
        echo $thisMonth . " ";
    }

    if(in_array("year" , $settings->select_date_format)) {
        echo $thisYear;
    }
?>
```

Esimerkkikoodi 4. Yllä oleva koodi tulostaa moduulin rivin sisään päivämäärän niillä tiedoilla, jotka ovat valittu moduulin monivalintavalikossa. Jos esimerkiksi valikossa on valittu ”day” ja ”year”, tulostuu moduulin riviin päivämäärä muodossa pp.vvvv, jos on valittu vain ”year”, niin tulostuu vvvv jne. Päivämäärän tiedot haetaan PHP:n date()-funktiolla, eli ne päivittyvät automaattisesti, kun sivu ladataan uudestaan. Monivalintavalikon tiedot haetaan $settings-muuttujasta, johon Beaver Builder tallentaa register_module() funktioon annetun valikon valintojen tiedot ja päivittää muuttujan sisältöä, kun se muuttuu valikossa. Siksi muutokset tulevat näkyviin sivulle heti, kun moduulin valikon valintoja muutetaan.

Moduuliin voidaan myös lisätä oma tyylitiedosto. Tämä ei ole pakollista moduulin toiminnallisuuden kannalta, mutta Beaver Builder antaa kolme vaihtoehtoa moduulin tyylin muokkaamiseen CSS-koodilla: (Module CSS, verkkoaineisto.)

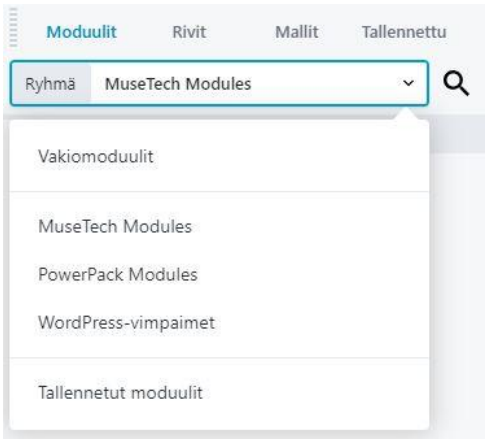
- Globaali CSS – Tämä koodi sijoitetaan tiedostoon mt-bb-modules/mt-date/css/frontend.css. Tiedoston CSS-koodi määrittää yleisen tyylityksen jokaiselle moduulin ilmentymälle.
- Globaali responsiivinen CSS - Tämä koodi sijoitetaan tiedostoon mt-bb-modules/mt-date/css/frontend.responsive.css. Tämän tiedoston koodi määrittää myös yleisen tyylityksen kaikille moduulin ilmentymille, mutta se astuu voimaan vasta teittyjen näytönleveyksien tullessa voimaan, jos responsiivinen asettelu on otettu käyttöön Beaver Builderin asetuksissa. Myös näytönleveydet voidaan määrittää Beaver Builderin asetuksissa.
- Ilmentymän CSS - Tämä koodi sijoitetaan tiedostoon mt-bb-modules/mt-date/includes/frontend.css.php. Tämä koodi lataa tyylin erikseen jokaiselle moduulin ilmentymälle.

Tässä esimerkissä käytämme Ilmentymän CSS-tyylitystä. Luodaan tiedosto frontend.css.php yllä mainittuun kansioon ja lisätään sinne CSS-koodi: (Module CSS, verkkoaineisto.)

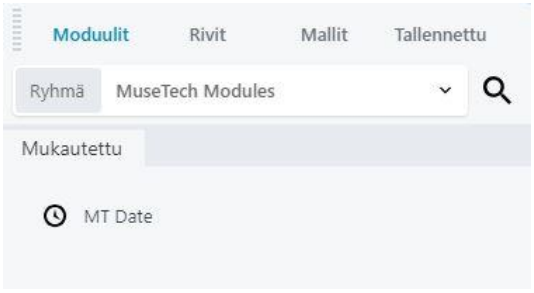
```
.fl-node-<?php echo $id; ?> {
    font-family: <?php echo $settings->font_style['family'] ?>;
    font-weight: <?php echo $settings->font_style['weight'] ?>;
    color: #<?php echo $settings->text_color; ?>;
    background-color: #<?php echo $settings->background_color; ?>;
}
```

Esimerkkikoodi 5. Yllä oleva CSS-tyylitys määrittyy jokaiselle moduulin ilmentymälle erikseen (huomaa CSS-luokan nimen dynaaminen generointi $id muuttujan perusteella). Tämä tarkoittaa sitä, että jokaisen moduulin ilmentymän asetuksia voidaan muokata ilman, että muut ilmentymät muuttuvat. Kuten esimerkkikoodissa 4. käytämme Beaver Builderin luomaa $settings-muuttujaa, jolla saamme ulkonäön päivittymään realiaikasesti, kun moduulin valikon asetuksia muutetaan.

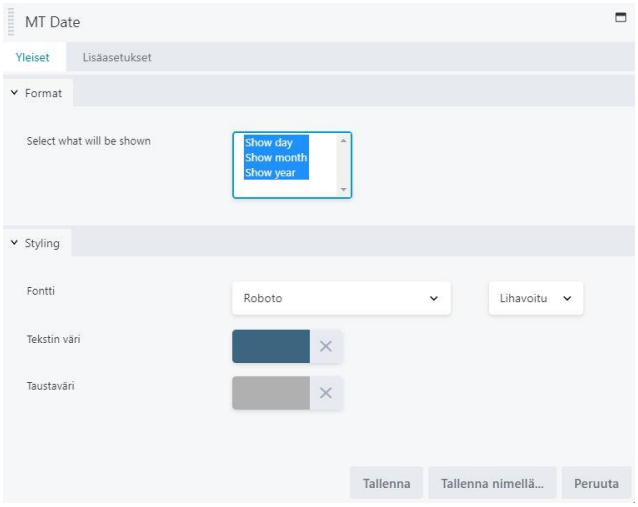
Lopputuloksena saatiin Beaver Builderin moduulivalikkoon uusi ryhmä ”MuseTech Modules”, jonka alta löytyy moduuli MT Date. Käyttäjä voi raahata moduulin sivustolle samalla tavalla kuin muitakin moduuleja ja muokata sen toimintoja ja ulkonäköä moduulivalikossa.



Kuva 1. Moduulien ryhmänäkymä, jossa näkyy esimerkkikoodissa 2 määritelty ryhmä ”MuseTech Modules”.



Kuva 2. Moduulin asetusvalikko, jossa näkyy esimerkkikoodissa 2 määritelty nimi sekä Beaver Builderin valmiiksi sisältämä ikoni ”clock.svg”.



Kuva 3. Moduulin asetusvalikon ulkonäkö esimerkkikoodissa 3 määritellyn assosiatiivisen taulukon perusteella.

Kuva 4. Moduulin tuottama sisältö sivulla kuvan 3 asetusten valinnoilla.

Omien moduulien luonti mahdollistaa tehokkaan tavan lisätä ominaisuuksia sivustolle ja tehdä niistä käyttäjäystävällisiä ylläpidon kannalta, koska moduulin asetusvalikon valintojen muokkaaminen ei vaadi ohjelmointitaitoa. Kun moduulit tehdään omaan lisäosaan, on ne myös helppo kopioida sivustolta toiselle, ja sivuston päivittäminen onnistuu WordPressin automaattisen päivitystoiminnon kautta ilman, että lisäosa poistuu. Tämä antaa mahdollisuuden luoda käyttäjälle moduuleja, joita voidaan tarjota esimerkiksi pakettina helpottamaan sivuston ylläpitoa ja täten vähentämään ylläpitokustannuksia. Luotu moduuli vaatii luonnollisesti sen, että itse Beaver Builder -lisäosa on jo asennettu sivulle. Jos lisäosaa ei ole mahdollista hankkia, kannattaa harkita moduulin toimintojen toteuttamista omalla WordPress-lisäosalla.

Eräs lisämoduuleja sisältävä paketti on PowerPack, joka antaa käyttäjälle kymmeniä lisämoduuleja. Joitakin esimerkkejä lisämoduuleista ovat:

- Sosiaalisen mediatilin integraatiot (Facebook moduulit, Instagram moduulit), jotka tuovat tilin sivun moduuliin iframe-elementillä.
- Row Separators – rivien ylä- ja alapuolelle asetettavat erottimet, jotka luovat svg vektorigrafiikalla rivien väliin osiot korostamaan rivien visuaalista ilmettä.
- Row Gradient – voidaan helposti Beaver Builderin moduulin asetusvalikosta lisätä ja muokata liukuväriä rivin taustalle tai taustan peitteeksi, esim. kuvan päälle.
- Advanced Menu – Mahdollistaa oman valikon lisäämisen sivustolle (esim. Ylärivin valikon WordPressin oletusvalikon sijaan), joka antaa useita muokkausmahdollisuuksia, kuten koko näytön peitteen valikon taustaksi.

Beaver Builder antaa myös mahdollisuuden luoda shortcodeja lisäosaan tallennetuille osioille. Shortcodella voidaan viitata seuraaviin Beaver Builderin osiin (Create Beaver Builder shortcodes, verkkoaineisto):

- Artikkelit
- Sivut
- Mukautetut artikkelityypit

- Tallennetut Beaver Builder mallit
- Tallennetut rivit ja moduulit

Näitä osia voidaan lisätä kaikille sivuston osille, joihin muitakin WordPressin shortcodeja voidaan käyttää. Beaver Builderillä luodun shortcoden ei siis tarvitse olla sivuston osassa, joka on rakennettu Beaver Builderillä, kunhan itse lisäosa on asennettu ja aktivoitu WordPressin hallintapaneelissa (Use shortcodes in your layouts, verkkoaineisto). Tässä osiossa käydään läpi tallennetun rivin ja moduulin lisääminen sivustolle shortcodella.

Luodaan ensin Beaver Builderillä sivun alareunaan tuleva rivi. Tallennetaan rivi Beaver Builderillä valitsemalla rivin asetuksista ”Tallenna nimellä...”-painike ja annetaan tallennetulle riville nimi ”Footer”. Rivi löytyy nyt WordPressin hallintapaneelista kohdasta Beaver Builder -> Tallennetut rivit -> Footer. Tämä on Beaver Builderin luoma mukautettu artikkelityyppi, jolla on muidenkin artikkelityyppien tapaan ID-numero. ID-numeron voi nähdä menemällä edellä mainittuun artikkeliin WordPressin hallintapaneelissa, jonka jälkeen ID-numeron voi nähdä sivun osoitteessa post-hakuparametrin kokonaislukuarvona (Get the slug or OD for a shortcode, verkkoaineisto). Tässä tapauksessa sivun URLissa lukee post=”154”, eli tallennetun rivin ID on 154.

Koska WordPress ei oletuksena käännä shortcodeja vimpaimissa, lisätään seuraava rivi lapsiteeman functions.php tiedostoon:

```
add_filter('widget_text', 'do_shortcode');
```

Esimerkkikoodi 6. Mahdollistaa shortcoden käyttämisen tekstityyppisessä vimpaimessa.

Rivi voidaan nyt lisätä sivun alareunaan esimerkiksi hallintapaneelissa kohdassa Ulkoasu -> Vimpaimet -> Sivun alareunan 1. sarake tekstivimpaimeen kirjoitetulla shortcodella.

Globaalin rivin ja moduulin tavoin edellä mainittua riviä voidaan nyt muokata yhdessä paikassa (esim. Beaver Builder -> Tallennetut rivit -> Footer -> Käynnistä Beaver Builder), jonka jälkeen muutokset näkyvät koko sivustolla.

3.2 Alateeman käyttö

Alateema (eng. Child Theme) perii yläteeman ulkonäön ja toiminnallisuuden, mutta mahdollistaa muutosten luonnin yläteemaan. Joitakin lapsiteeman hyötyjä ovat: (Child Themes, verkkoaineisto.)

- Tehdyt muutokset voidaan siirtää ja uudelleenkäyttää
- Muokkaukset voidaan pitää erillään yläteemasta
- Yläteema voidaan päivittää ilman että alateemaan määritetyt muutokset häviävät
- Yläteemaan kehitykseen ja testaamiseen kuluneen työn hyödyntäminen uudelleenkäyttämällä sen toiminnallisuuksia. (Child Themes, verkkoaineisto.)

3.3 WordPressin koukut

Jotta WordPressiin olisi helpompi liittää toimintoja, on luotu ns. kukkuja (eng. Hooks), jolla lisätoimintoja voi rekisteröidä WordPressiin ajettavaksi kun se suorittaa tiettyjä toimintoja. Tämä toimii siten, että koukku rekisteröidään WordPressiin valmiiksi määriteltyjä koukkuja käyttäen, ja WordPress tarkastaa aina näitä toimintoja ajettaessa, onko jokin lisäosa rekisteröinyt koukun ko. toimintoon.

WordPressissä on kahdentapaista koukkua (Plugin API, Verkkoaineisto.):

- Action koukku (eng. Actions) – käynnistetään tietyn WordPressin tapahtuman yhteydessä, esimerkiksi kun julkaistaan artikkeli tai vaihdetaan teemaa
- Suodatin (eng. Filter) – funktio jonka läpi annettu data käsitellään juuri ennen kun jokin toiminto suoritetaan, esimerkiksi kun dataa tuodaan tietokannasta front-end puolelle tai päinvastoin. (Plugin API, Verkkoaineisto.)

3.4 Oman lisäosan luonti

Oman lisäosan avulla voidaan muuttaa ja lisätä WordPressiin toiminnallisuuksia ilman, että tarvitsee muutta WordPressin omaa koodia. WordPressin lisäosa vaatii toimiakseen

vain yhden PHP-tiedoston. On kuitenkin suositeltavaa asettaa lisäosan tiedostot omaan kansioonsa, jotta kaikki lisäosan tiedostot tulevat olemaan yhdessä paikassa. Lisäosat asennetaan kansioon wp-content/plugins (Getting Started, verkkoaineisto).

Luodaan lisäosa ”Time plugin” lisäämällä kansio wp-content/plugins/mt-plugin ja sinne lisäosan päätiedoston mt-plugin.php, joka sisältää kommentin:

```
/**
 * Plugin Name: Time plugin
 * Plugin URI: https://www.musetech.fi
 * Description: Plugin for showing the current time on your site.
 * Version: 1.0
 * Author: MuseTech
 * Author URI: https://www.musetech.fi
 */
```

Esimerkkikoodi 7. Pluginin päätiedoston sisältämä koodi määrittelee PHP-kommenteilla pluginin tiedot, jotka näytetään WordPressin admin-paneelissa Lisäosat-osiossa.

Lisäosan päätiedoston nimeäminen samaksi kuin lisäosan kansio on yleinen käytäntö, mutta ei välttämätöntä lisäosan toimivuuden suhteen. Lisäosa suoritetaan WordPressin toimesta liittämällä se pääjärjestelmään, mutta tietoturvasyistä on hyvä estää pääsy suoraan tiedostoon ulkopuoliselta hyökkäykseltä. Tämä esto tehdään sijoittamalla jokaisen PHP-tiedoston alkuun koodi (Plugin Files, verkkoaineisto):

```
defined( 'ABSPATH' ) or die( 'No direct access allowed!' );
```

Esimerkkikoodi 8. ABSPATH-vakio on WordPressissä määritetty kansioviittaus. Yllä oleva koodi estää viittauksen PHP-tiedostoon muualta kuin WordPressin sisältä lopettamalla koodin suorituksen tiedoston alussa, jos siihen viitataan suoraan esim. menemällä osoitteeseen ... /wp-content/plugins/mt-plugin/mt-plugin.php. Tiedosto antaa ilmoituksen die()-funktion avulla, jos käyttäjä yrittää pyrkiä suoraan tiedostoon.

On myös järkevää olla ajamasta erillisiä PHP-funktioita ennen WordPressin funktioiden kutsua. Tällöin varmistutaan siitä, että ulkopuolinen osapuoli ei saa tietoonsa tiedostossa käsiteltävää tietoa ilman, että tiedosto on liitetty WordPressiin (Plugin Files, verkkoaineisto).

Seuraavaksi lisätään tiedostoon mt-plugin.php -lisäosan varsinaisen toiminnallisuuden määrittelevä koodi. Luodaan luokka MT_Plugin ja sille muutama metodi sekä luonnollisesti konstruktori (How to write a WordPress plugin, verkkoaineisto.):

```
if(!class_exists('MT_Plugin'))
{
    class MT_Plugin
    {
        public function __construct() {}

        public static function activate() {}

        public static function deactivate() {}
    }
}
```

Esimerkkikoodi 9. MT_Plugin-luokan määrittelyssä tarkistetaan ensiksi, onko luokkaa jo määritelty muualla. Jos luokkaa ei ole määritelty, luodaan se, sekä sille toistaiseksi tyhjä konstruktori. Lisäksi määritellään metodit, joita voidaan käyttää lisäosan aktivoinnin ja deaktivoinnin yhteydessä.

Seuraavaksi luodaan luokan määrittelyn jälkeen koodi, joka määrittää lisäosan aktivointiin ja deaktivointiin liittyvät WordPress-koukut, luo MT_Plugin-luokasta ilmentymän sekä lisää asetukset-linkin lisäosat-valikkoon lisäosan nimen alle (How to write a WordPress plugin, verkkoaineisto.):

```
if(class_exists('MT_Plugin')) {
    register_activation_hook(__FILE__, array('MT_Plugin', 'activate'));
    register_deactivation_hook(__FILE__, array('MT_Plugin', 'deactivate'));

    $mt_plugin = new MT_Plugin();

    if(isset($mt_plugin)) {
        $plugin = plugin_basename(__FILE__);
        add_filter("plugin_action_links_$plugin", 'plugin_settings_link');

        function plugin_settings_link($links) {
            $settings_link = '<a href="options-gen-
eral.php?page=mt_plugin">Settings</a>';
            array_unshift($links, $settings_link);
            return $links;
        }
    }
}
```

Esimerkkikoodi 10. Koodi tarkistaa ensin, onko MT_Plugin-luokka määritelty. Jos luokka on määritelty, rekisteröidään ensin lisäosan aktivoinnin ja deaktivoinnin koukut viittamalla MT_Plugin luokan metodeihin activate() sekä deactivate(). Nämä metodit määriteltiin esimerkkikoodissa 9, mutta ne jätettiin tyhjäksi. Tämän jälkeen luodaan luokasta MT_Plugin-ilmentymä ja annetaan sille asetusvalikon linkki lisäosat-sivulle plugin_settings_link()-funktion avulla. Tätä kutsutaan WordPressin koukun suodattimella, joka antaa funktiolle taulukon niistä linkeistä, jotka näkyvät lisäosat-sivulla. Tätä taulukkoa muuttamalla ja palauttamalla se saadaan lisättyä asetukset-sivun linkki.

Asetukset-linkki ei kuitenkaan vielä toimi, koska lisäosan asetussivua ei ole vielä luotu. Lisätään seuraavaksi MT_Plugin-luokan konstruktoriin seuraavat rivit (How to write a WordPress plugin, verkkoaineisto.):

```
add_action('admin_init', array(&$this, 'admin_init'));
add_action('admin_menu', array(&$this, 'add_menu'));
```

Esimerkkikoodi 11. Annetaan add_action()-koukkujen avulla WordPressille tieto siitä, mitä MT_Plugin-luokan ilmentymän metodeja kutsutaan hallintapaneelin latautuessa sekä asetusvalikon alustusvaiiheessa.

Luodaan esimerkkikoodissa 11 annetut MT_Plugin-luokan metodit (How to write a WordPress plugin, verkkoaineisto.):

```
public function admin_init() {
	$this->init_settings();
}
public function init_settings() {
	register_setting('mt_plugin_group', 'setting_show_hours');
	register_setting('mt_plugin_group', 'setting_show_minutes');
	register_setting('mt_plugin_group', 'setting_show_seconds');
}
```

Esimerkkikoodi 12. Määritellän MT_Plugin-luokan metodit admin_init(), joka kutsuu luokan ilmentymän metodia init_settings(). Admin_init() toimii WordPressin koukun avulla, jota käytettiin esimerkkikoodissa 11. init_settings() käyttää WordPressin funktiota register_setting() luomaan lisäosalle asetukset, joiden ryhmät ja nimet määritellään funktiolle annetuilla parametreilla.

Luodaan lisäosan kansioon kansio templates ja sen sisään tiedosto settings.php. Tämän tiedoston sisään tullaan jatkossa tekemään asetussivun valikko. Asetetaan nyt tämä tiedosto lisäosan valikoksi. Lisätään MT_Plugin luokan metodit (How to write a WordPress plugin, verkkoaineisto.):

```
public function add_menu() {
	add_options_page('MT Plugin Template Settings', 'Time Plugin', 'man-
age_options', 'mt_plugin', array(&$this, 'plugin_settings_page'));
}

public function plugin_settings_page() {
	if(!current_user_can('manage_options')) {
		wp_die(('You do not have sufficient permissions to access this
page.'));
	}
	 include(sprintf("%s/templates/settings.php", dirname(__FILE__)));
}
```

Esimerkkikoodi 13. Luodaan metodi add_menu(), johon viitattiin esimerkkikoodin 11 WordPress koukussa. Add_options_page() on funktio, joka ottaa parametreiksi asetusvalikon sivun otsikon, asetusvalikon nimenä käytettävän tekstin, käyttäjältä vaadittavan käyttöoikeuden, asetusvalikon polkutunnuksen ja lopulta callback funktion, jota kutsutaan asetussivun näyttämiseen. Callback funktio määritellän add_menu() funktion jälkeen, joka ensin tarkastaa onko käyttäjällä vaadittavat käyttöoikeudet. Jos käyttäjällä ei ole 'manage_options' oikeuksia, ei funktioa ajeta loppuun, vaan näytettään virheilmoitus. Muuten näytetään settings.php-tiedostossa muotoiltu valikko.

Luodaan valikko settings.php tiedoston sisään. Valikon asetusten ja sen ulkonäön määritys on hyvä pitää erillään toisistaan, jotta lisäosan rakenne säilyy selkeämpänä (How to write a WordPress plugin, verkkoaineisto.).

```
<?php
defined( 'ABSPATH' ) or die( 'No direct access allowed!' );
?>

<div class="wrap">
    <h2>Time Plugin</h2>
    <p>Select time format:</p>
    <form method="post" action="options.php">
        <?php @settings_fields('mt_plugin_group'); ?>
        <?php @do_settings_fields('mt_plugin_group'); ?>

        <table class="form-table">
            <tr valign="top">
                <th scope="row"><label for="setting_show_hours">Show
hours</label></th>
                <td><input type="checkbox" name="setting_show_hours" id="set-
ting_show_hours" value="value1" <?php checked('value1' == get_option('set-
ting_show_hours')); ?>/></td>
            </tr>
            <tr valign="top">
                <th scope="row"><label for="setting_show_minutes">Show
minutes</label></th>
                <td><input type="checkbox" name="setting_show_minutes"
id="setting_show_minutes" value="value2" <?php checked('value2' == get_op-
tion('setting_show_minutes')); ?>/></td>
            </tr>
            <tr valign="top">
                <th scope="row"><label for="setting_show_seconds">Show sec-
onds</label></th>
                <td><input type="checkbox" name="setting_show_seconds"
id="setting_show_seconds" value="value3" <?php checked('value3' == get_op-
tion('setting_show_seconds')); ?>/></td>
            </tr>
        </table>

        <input type="submit" class="button-primary" value="Save Changes" />
    </form>
</div>
```

Esimerkkikoodi 14. Lisäosan asetussivun näyttävä koodi. HTML-lomake käyttää method="post" toimintoa tallentaakseen lomakkeen tiedot options.php-tiedostoon, jonka avulla WordPress tallentaa valitut asetukset tietokantaan. Lomakkeen sisällä käytetään settings_field() - sekä do_settings_fields() -funktioita

määrittämään, mitä asetuksia lomakkeessa näytetään. Ne vastaanottavat tavallisesti 2 parametria, mutta niiden käyttö onnistuu yhteyn parametrin avulla. Asetukset haetaan get_option() -funktiolla valintaruutuihin siten, että niiden arvot tarkastetaan ja jos arvo on sama kuin input-elementin value="" arvo, niin input elementtiin asetetaan ominaisuus ”checked”. Tiedoston alussa tarkistetaan ABSPATH-vakio, jotta tiedostoon ei voida viitata WordPressin ulkopuolelta

Luodaan templates-kansioon tiedosto mt-time.php, joka määrittää lisäosan ulkonäön sivustolla (How to write a WordPress plugin, verkkoaineisto.):

```
<?php
defined( 'ABSPATH' ) or die( 'No direct access allowed!' );
?>

<div class="wrap">
    <?php
    if(get_option('setting_show_hours')){
        echo date( 'H', current_time( 'timestamp', 0)) .":";
    }
    if(get_option('setting_show_minutes')){
        echo date( 'i', current_time( 'timestamp', 0)) .":";
    }
    if(get_option('setting_show_seconds')){
        echo date( 's', current_time( 'timestamp', 0));
    }
    ?>
</div>
```

Esimerkkikoodi 15. Koodi tulostaa ne tiedot, jotka on valittuna lisäosan asetusvalikossa. Tiedoston alussa tarkistetaan ABSPATH-vakio, jotta tiedostoon ei voida viitata WordPressin ulkopuolelta.

Lopuksi tuodaan tiedoston mt-time.php-sisältö lisäosan päätiedostoon ja määritellään shortcode, jonka avulla lisäosan voi asettaa sivustolle. Lisätään MT_Plugin-luokan metodi:

```
public function renderTime() {
     ob_start();
     include(sprintf("%s/templates/mt-time.php", dirname(__FILE__)));
     $output = ob_get_clean();

     return $output;
}
```

Esimerkkikoodi 16. Metodi, joka palauttaa mt-time.php-sisällön merkkijonona siten, että sen sisältämä PHP-koodi on käännetty. Tällöin merkkijonossa näkyy PHP-koodin tuloste eikä itse koodi, koska client-puoli ei käännä PHP-koodia.

Shortcoden määrittely lisätään MT_Plugin-luokan konstruktoriin:

```
add_shortcode('mt_time', array($this, 'renderTime'));
```

Esimerkkikoodi 17. Luodaan shortcode [mt_time] WordPressin add_scortcode() funktiolla, jossa kutsutaan esimerkkikoodissa 16 esitettyä metodia renderTime(). Huomataan, että toinen parametri on taulukko, jonka ensimmäinen alkio on MT_Plugin-luokan ilmentymä ja toinen alkio on itse metodi. Tämä on kirjoitettava siksi, että funktio renderTime() on MT_Plugin-luokan metodi, eikä siihen voida viitata suoraan luokan ulkopuolelta ilman luokan ilmentymän viittausta.

Lisäosan luonti mahdollistaa lisätoiminnallisuuden luonnin ilman, että suoraan muokataan WordPressin omaa koodia. Tällöin järjestelmästä saadaan helpommin ylläpidettävä, sillä lisäosan koodi ei häviä WordPressiä päivitettäessä ja debuggausvaiheessa voidaan helposti aktivoida/deaktivoida kirjoitettu koodi, joka helpottaa virheen löytämistä. Lisäksi toiminnallisuus on modulaarista, koska lisäosan kansio on nyt helposti kopioitavissa toiseen WordPress-asennukseen, joka myös mahdollistaa tehokkaamman koodin uudelleenkäytön. Käyttäjälle on myös täten mahdollista tarjota lisätoiminnallisuuksia ilman, että niiden muokkaamminen vaatii ohjelmointitaitoa, kun käytetään lisäosan asetusvalikkoa. Edellä esitetty esimerkki on toki käyttötapauksena erittäin suppea toiminnallisuuksien suhteen. Jos siis kyseessä olisi ainoastaan esitettyjen toiminnallisuuksien lisääminen sivustolle, olisi paljon helpompaa käyttää suoraan shortcode-menetelmää esim. lapsiteeman functions.php-tiedostossa. Toisaalta kun toiminnallisuuksien määrä kasvaa ja monimutkaistuu, niin tekee myös kirjoitettu koodi, jolloin jossain vaiheessa on hyvä harkita toimintojen siirtäminen lisäosaan.

3.5 WordPress REST API

WordPressin REST API julkaistiin virallistesti WordPress-versiossa 4.7.0 (Changelog, verkkoaineisto). Tämän ansiosta voidaan client-puolen koodilla hakea JSON-dataa suoraan tietokannasta (ja lähettää JSON-dataa tietokantaan), joka mahdollistaa uuden tavan rakentaa WordPress-verkkosivuja. Koska REST APIa on mahdollista käyttää myös WordPerss-asennuksen ulkopuoleta, voidaan verkkoaplikaatioiden lisäksi rakentaa jopa mobiiliaplikaatioita perustuen WordPress REST APIin (REST API Handbook, verkkoaineisto).

3.5.1 WordPress REST APIn rakenne

WordPress REST API määrittää reittejä (eng. route), jotka yhdistävät tietyn URIn eri HTTP-metodeihin. Tietyn HTTP-metodi yhdistämistä tiettyyn URIin kutsutaan päätepisteeksi (eng. endpoint). Esimerkiksi URLin https://musetech.fi/wp-json/v2/posts kutsuminen GET-metodilla on päätepiste, joka palauttaa vastauksena sivuston WordPress-artikkelit JSON-datana client-puolelle (Routes & Endpoints, verkkoaineisto).

WordPress REST APIn pyyntö voidaan tehdä joko HTTP pyynnön kautta tai sisäisesti PHP-koodin avulla. Kun lähetetään HTTP-pyyntö olemassa olevaan reittiin, WordPress luo automaattisesti WP_REST_Request-olion ilmentymän, jota käytetään pyynnössä käytetyn datan tallentamiseen ja välittämiseen (Requests, verkkoaineisto).

WordPress REST APIn vastaukset ovat järjestelmän palauttama tieto (esim. tietokannasta palautettava JSON-data tai virhekoodi), jota voidaan käsitellä WP_REST_Response-luokan avulla (Responses, verkkoaineisto).

WordPress REST APIn Skeema määrittää päätepisteen käsittelemän datan rakenteen ja edistää myös järjestelmän tieturvaa mahdollistamalla APIlle lähetetyn pyynnön validoinnin (Schema, verkkoaineisto).

WordPress REST API yhdistää em. osat Controller-luokkien avulla, jotka määrittävät reittien ja päätepisteiden rekisteröinnin, pyynnön käsittelyn ja vastauksen luonnin sekä skeeman käytön (Controller Classes, verkkoaineisto).

3.5.2 WordPress REST APIn käyttö

Client-puolelta WordPress REST APIlle lähetetty pyyntö voidaan tehdä esimerkiksi seuraavasti:

```
const ROOT_URL = 'https://www.musetech.fi/wp-json/wp/v2/';

jQuery.ajax({
    method: "GET",
    url: ROOT_URL + 'posts'
}).done(function(data) {
    console.log(data);
}).fail(function() {
```

```
    alert("artikkeleita ei voitu hakea!");
}).always(function() {
    console.log( "pyyntö suoritettu!" );
});
```

Esimerkkikoodi 18. WordPress REST APIlle lähetetty asynkroninen HTTP GET -pyyntö, joka on suoritettu JavaScriptin jQuery-kirjaston ajax() funktiolla. HTTP-metodi määritetään ”method”-attribuutilla ja osoite ”url”-attribuutilla.

Esimerkkikoodin 18 pyyntö palauttaa JSON-objektin:

```
(10) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]
     0: {id: 1794, date: "2019-03-19T15:12:12", date_gmt: "2019-03-
     19T13:12:12", guid: {…}, modified: "2019-04-02T13:19:31", …}
     1: {id: 1766, date: "2019-02-20T10:56:33", date_gmt: "2019-02-
     20T08:56:33", guid: {…}, modified: "2019-04-02T12:16:23", …}
     2: {id: 1516, date: "2019-01-09T10:54:15", date_gmt: "2019-01-
     09T08:54:15", guid: {…}, modified: "2019-04-02T12:20:57", …}
     3: {id: 1470, date: "2019-01-08T15:08:00", date_gmt: "2019-01-
     08T13:08:00", guid: {…}, modified: "2019-04-02T13:19:52", …}
     4: {id: 1345, date: "2018-11-20T12:42:32", date_gmt: "2018-11-
     20T10:42:32", guid: {…}, modified: "2019-04-02T13:20:01", …}
     5: {id: 1313, date: "2018-11-18T16:57:18", date_gmt: "2018-11-
     18T14:57:18", guid: {…}, modified: "2019-04-02T13:20:22", …}
     6: {id: 1294, date: "2018-10-25T13:01:21", date_gmt: "2018-10-
     25T10:01:21", guid: {…}, modified: "2019-04-02T13:21:59", …}
     7: {id: 1302, date: "2018-04-30T12:00:51", date_gmt: "2018-04-
     30T09:00:51", guid: {…}, modified: "2019-04-02T13:22:39", …}
     8: {id: 1304, date: "2018-02-01T12:00:57", date_gmt: "2018-02-
     01T10:00:57", guid: {…}, modified: "2019-04-04T11:19:42", …}
     9: {id: 1306, date: "2018-01-06T12:00:09", date_gmt: "2018-01-
     06T10:00:09", guid: {…}, modified: "2019-04-02T13:26:04", …}
     length: 10
```

Esimerkkikoodi 19. WordPress REST APIn palauttama JSON-objekti, joka sisältää kaikki sivuston 10 artikkelia taulukossa sekä length-muuttujan, jonka sisältämä kokonaisluku kertoo taulukon alkioiden määrän.

Artikkelin dataan voidaan nyt viitata yksinkertaisesti tyyliin:

```
data[0].date
```

Esimerkkikoodi 20. Esimerkkikoodin 19 ensimmäisen artikkelin päivämäärä-tietoon viittaus, joka käyttää esimerkkikoodissa 18 nimettyä data-muuttujaa done() funktion callback-funktiossa.

Esimerkkikoodin 19 JSON-objekti sisältää muunmuassa categories-taulkon, jonka alkiot viittaavat artikkelin kategorioiden id-alkioon kokonaisluvuilla. Jos asetetaan esimerkkikoodiin 18 ”url: ROOT_URL + ’categories’”, saadaan kaikki kategoriat JSON-taulukkona.

```
0: {id: 25, count: 10, description: "", link: "https://www.musetech.fi/category/news/
", name: "news", …}
1: {id: 21, count: 0, description: "", link: "https://www.musetech.fi/category/refere
nssit-extra/", name: "referenssit-extra", …}
2: {id: 1, count: 0, description: "", link: "https://www.musetech.fi/category/uncateg
orized-fi/", name: "Uncategorized @fi", …}
```

Esimerkkikoodi 21. WordPress REST APIn palauttama Categories-taulukko, jonka jokainen alkio sisältää yhden kategorian tiedot.

Kategoriaobjektin ”name”-arvo antaa kategorian nimen. Tällä tavoin voidaan suorittaa vertailu siitä, mikä post-objekti kuuluu minkäkin nimiseen kategoriaan.

Edellä mainituilla tavoin voidaan rakentaa client-puolella toimiva järjestelmä, joka näyttää haluttujen kategorioiden artikkelit. Tällöin ylläpito voidaan suorittaa WordPressin hallintapaneelissa ilman ohjelmointitaitoa esimerkiksi siten, että käyttäjä valitsee mitkä artikkelit kuuluvat näytettävään kategoriaan. Tällöin WordPress toimii eräänlaisena tiedonhallintajärjestelmänä ja client-puoli voidaan erottaa jopa täysin WordPress-alustasta.

3.6 Shortcode

Sivujen sisällön kirjoittaminen suoraan HTML-muodossa voi joskus tehdä sivustosta vaikeasti ylläpidettävän. Jos esimerkiksi kuvagalleriaa ylläpidettäisiin täten, vaatisi uuden kuvan lisääminen galleriaan sen, että kun kuva on ensin ladattu WordPressin mediakirjastoon (jolloin se ladataan tietokantaan), sen URLiin viitattaisiin HTML-koodissa ja siihen asetettaisiin sopivat luokat ja atribuutit jne. Jos kuva vaihdettaisiin toiseen mediakirjastossa, olisi aina HTML-koodissa oleva URL päivitettävä käsin, jotta sisältö pysyisi ajan tasalla.

Tällainen tapaus on esimerkki siitä, miksi WordPress on versiosta 2.5 eteenpäin alkanut tukemaan shortcode-järjestelmää. Yllä olevan ongelman ratkaisemiseksi WordPress sisältää valmiiksi sisäänrakennetun gallery-shortcoden. Kirjoittamalla shortcoden [gallery] sivulle WordPress lataa suoraan sivulle liitetyt kuvat listana, jonka HTML on muotoa: (The Shortcode API, verkkoaineisto.)

```
<div id="gallery-1" class="gallery galleryid-7 gallery-columns-3 gallery-size-
thumbnail">
```

```
    <dl class="gallery-item">
        <dt class="gallery-icon landscape">
            <a href="kuvan 1 linkki URL"><img width="150" height="150"
src="kuvan 1 URL" class="attachment-thumbnail size-thumbnail" alt=""></a>
        </dt></dl>
    <dl class="gallery-item">
        <dt class="gallery-icon portrait">
            <a href="kuvan 2 linkki URL"><img width="150" height="150"
src="kuvan 2 URL" class="attachment-thumbnail size-thumbnail" alt=""></a>
        </dt></dl>
</div>
```

Esimerkkikoodi 22. WordPressin luoma HTML-koodi, joka korvaa esimerkkikoodi 1 olevan merkkijonon sivua ladattaessa.

Sivulle liitettyjä kuvia voi hallita mediakirjastossa. Jos sivulle liitetään nyt mediakirjastoon ladattu kuva, WordPress lataa kuvan suoraan galleriaan samalla tavalla kuin esimekkikoodissa 2 on merkitty.

Esimerkkikoodi 21:stä voi huomata, että WordPress luo oletuksena monia luokkia HTML-koodiin, jotka vaikuttavat gallerian ulkonäköön. WordPressin shortcodeen voi myös lisätä attribuutteja, joilla sivulle ladatun sisällön ominaisuuksia voi muuttaa. Esimerkiksi, jos esimerkkikoodi 21 kuvan kokoa halutaan suurentaa, voidaan shortcodeen lisätä size=”medium” attribuutti (The Shortcode API, verkkoaineisto):

```
[gallery size=”medium”]
```

Esimerkkikoodi 23. Kuvagallerian luova shortcode, jonka kuvien kokoa on muokattu WordPressin oletuskokojen avulla

3.7 Mukautettu shortcode

WordPressissä oleva Shortcode API mahdollistaa myös omien mukautettujen shortcode-merkkijonojen luonnin. Shortcoden toiminnallisuus määrittellään PHP-funktiolla:

```
function vuosiluku_shortcode() {
  $vuosi = date('Y');
  return $vuosi;
}
```

Esimerkkikoodi 24. Mukautetun shortcoden toiminnallisuuden määrittely

Shortcode määritellään add_shortcode() -funktiolla WordPressissä antamalla siihen viittavaa merkkijono ja yhdistämällä tämä sen toiminnallisuuden määrittelevään funktioon (The Shortcode API, verkkoaineisto):

```
add_shortcode('vuosi', 'vuosiluku_shortcode');
```

Esimerkkikoodi 25. Mukautetun shortcoden määrittely.

Automaattisen vuosiluvun saa nyt lisättyä sivulle shortcodella:

```
[vuosi]
```

Esimerkkikoodi 26. Shortcode, joka korvataan automaattisesti päivittyvällä vuosiluvulla, kun sivu ladataan.

Edellä olevat koodirivit mahdollistavat vuosiluvun automaattisen päivittämisen esimerkiksi sivun alareunassa (eng. footer) ja siten vuosiluvun päivittäminen ei vaadi ylläpitoa.

Mukautettu shortcode mahdollistaa nopeamman automatisoidun toiminnon luonnin ja lisäyksen sivustolle, sillä toiminto ei vaadi oman lisäosan luomista. Oman shortcoden luonti sovueltuu siksi varsinkin pienien toiminnallisuuksien lisäämiseen sivustolle. Huono puoli on, että toiminnallisuuden muuttaminen vaatii ohjelmointitaitoa, eli shortcode ei sovellu käyttäjälle, joka haluaa mahdollisimman helposti päivittää sivustoa visuaalisen käyttöliittymän avulla. Jos halutaan tarjota toiminto, jota käyttäjä voi itse päivittää, on oman lisäosan luonti suositeltavaa, sillä tämä mahdollistaa helpommin graafisen asetusvalikon luonnin WordPressin hallintapaneelissa.

4 Lisäosat

4.1 WPML

WPML (WordPress Multilingual Plugin) mahdollistaa monikielisten sivustojen rakentamisen ja ylläpidon.

Asennuksen jälkeen lisäosa on alustettava. Alustus vaatii aluksi määrittämään sivuston nykyisen kielen. Tämän jälkeen valitaan kielet, johon sivusto halutaan käänttää, sekä joitakin kielivalikkoon liittyviä ominaisuuksia. Lopuksi lisäosa on rekisteröitävä, jotta voidaan saada päivityksiä lisäosalle. Rekisteröintiä varten sivuston URL on ensin annettava WPML-sivuston omalla tilillä, jolloin saadaan rekisteröintiavain ja tämän jälkeen rekisteröintiavain annetaan lisäosan alustuksen viimeisessä vaiheessa.

WPML sisältää kaksi versiota, Multitlingual Blogin sekä Multilingual CMS:n, joista jälkimmäisellä on useampia toimintoja. Tässä työssä käydään läpi Multilingual CMS -tilityyppiin liittyviä toimintoja (jotkut saattavat sisältyä myös Multilingual Blog tilityyppiin). Joitakin lisäosaan liittyviä osia ovat: (WPML Core and Add-on Plugins.)

- WPML Multilingual CMS – lisäosan ydin, joka mahdollistaa sivun käännöksen.
- WPML String Translation – mahdollistaa merkkijonojen käännöksen suoraan WordPressin hallintapaneelista ilman .mo-käännöstiedostoja. Mahdollistaa myös sivujen ja artikkeleiden ulkopuolella olevan tekstin käännöksen, kuten sivun kuvaustekstin sekä SEOon liityvän tekstin.
- WPML Translation Management – antaa ylläpitäjälle mahdollisuuden hallita sivun käännösprosessia antamalla eri käyttäjille käännöstöitä ja valvomalla käännöksen valmistumista.
- WPML Media Translation – mahdollistaa eri kuvagallerioiden ylläpidon erikseen eri kielillä.
- WPML Sticky Links – auttaa pitämään monikielisen sivuston linkkien osoitteet ajan tasalla, jotta sivustolle ei synny rikkinäisiä linkkejä, kun monikielisyyttä ylläpidetään.
- WPML CMS Navigation – lisää sivustolle monikielisyyteen liittyviä valikoita, joita voidaan asettaa sivulle helpottamaan navigointia.
- Contact Form 7 Multilingual – mahdollistaa Contact Form 7 -lisäosalla tehtyjen yhteydenottolomakkeiden käännöksen ilman, että lomaketta tarvitsee luoda uudestaan eri kielille. (WPML Core and Add-on Plugins.)

4.2 SASS-lisäosa

WP-SCSS-lisäosa mahdollistaa Sass-tyylityksen käytön perinteisen CSS-tyylityksen sijaan helposti asennettuna WordPressiin lisäosan muodossa. Lisäosan asennuksen jälkeen on asetusvalikossa annettava suhteelliset polut (alkaen aktiivisen teeman polusta) sekä .scss-tiedostojen kansioon että käännettyjen .css-tiedostojen kansioon.

Lisäosa sisältää myös asetuksia liittyen, mihin muotoon .scss-tiedostot käännetään, sekä asetuksen, joka lisää käännetyn CSS-tyylin sivun headeriin automaattisesti. Jos tyyli lisätään sivulle automaattisesti, tulevat tyylityksen muutokset voimaan pelkästään tallentamalla validi .scss-tiedosto edellä annettuun kansioon. Tämä mahdollistaa hyvin tehokkaan tyylitysten luomisen perinteiseen CSS-tyylitykseen verrattuna. Tässä työssä emme käy läpi Sass-tyylityksen syntaksia, mutta alla on annettu esimerkki .scss-tiedostoon kirjoitetusta koodista sekä lisäosan kääntämästä .css-koodista.

```
$red: #ff0000;
$blue: #425caa;
$black: #1e1e1e;

tr {
    border-color: $blue;
    td {
        border-color: $red;
        span {
            color: $black;
        }
    }
}
```

Esimerkkikoodi 27. Sass-tyylityksen määritys muuttujilla ja sisäkkäisten elementtien määrittelyllä.

```
tr {
  border-color: #425caa;
}
tr td {
  border-color: #f00;
}
tr td span {
  color: #1e1e1e;
}
```

Esimerkkikoodi 28. WP-SCSS-lisäosan esimerkkikoodi 27:n perusteella kääntämä CSS-koodi.

4.3 Tietoturvallisuutta tehostavat lisäosat

4.3.1 WPS Hide Login

WordPressiin kirjaudutaan oletuksena menemällä osoitteeseen ”wordpresssivusto.fi/wp-admin” tai ”wordpresssivusto/wp-login.php”. Tämä on erittäin suuri tietoturvariski, jos kirjautumissivua ei suojata mitenkään. Yksi tapa vähentää mahdollisia hyökkäyksiä kirjautumissivulle on vaihtaa URL johonkin toiseen vaikeammin arvattavaan. WPS Hide Login -lisäosan avulla tämä onnistuu helposti. Lisäosan asennuksen jälkeen voidaan

vaihtaa kirjautumissivun URL-kohdasta Asetukset -> Yleinen. Tämän lisäksi voidaan valita, mille sivulle ohjataan käyttäjä, joka ei ole kirjautunut sisään, mikäli tämä yrittää kirjautua sisään osoitteessa /wp-login.php.

4.3.2 Limit Login Attempts Reloaded

WordPress antaa oletuksena äärettömän määrän kertoja yrittää kirjautua sisään, joka mahdollistaa Brute Force -hyökkäyksen kirjautumissivulle. Limit Login Attempts Reloaded -lisäosa mahdollistaa kirajutumisen yrityskertojen rajaamisen annettuun määrään. Jos käyttäjä ei onnistu kirjautumisessa annettujen kertojen jälkeen, estetään käyttäjää (IP osoitetetta) kirjautumasta sisään annetun ajan. Jos käyttäjä ei vieläkään onnistu kirjautumaan sisään annettujen estämiskertojen jälkeen, voidaan käyttäjän IP-osoite estää vielä pidemmäksi aikaa. Edellä mainitut yrityskerrat sekä estojen ajat ovat valittavissa lisäosan asennuksen jälkeen lisäosan asetusvalikosta kohdasta Asetukset -> Limit Login Attempts. Asetusvalikossa voidaan myös lisätä IP-osoitteita Whitelist/Blacklist-listoille, sekä määrittää IP-osoitteille vielä WordPress-käyttäjänimet, jotka listaan lisätään.

4.4 Contact Form 7

Melkein jokaisella verkkosivustolla on yleensä jonkinlainen yhteydenottolomake. Contact Form 7 on lisäosa, joka mahdollistaa yhteydenottolomakkeen helpon luonnin, mutta ei samalla estä ylläpitäjältä mahdollisuutta tehdä mukautettuja ratkaisuja. Lisäosalle löytyy myös laajennuksia, joiden avulla lisäosalle voidaan saada lisätoiminnallisuuksia. Luodaan lisäosalla yksinkertainen yhteydenottolomake.

Asennuksen jälkeen WordPressin hallintapaneeliin ilmestyy Yhteydenotto-osio, josta pääsee näkemään kaikki Contact Form 7 -lisäosalla luodut lomakkeet listana. Lomakkeet ovat mukautettuja artikkelityyppejä, ja lisäosa luo niille shortcoden, jolla ne voidaan lisätä sivustolle. Luotua lomaketta voi muokata HTML-muodossa käyttämällä lisäosan luomia lomaketunnisteita (eng. form-tag) lomakkeen kenttien lisäämiseen. Lomaketunnisteet korvataan lisäosan generoimalla HTML-koodilla käännöksen yhteydessä (Editing Form Template, verkkoaiensto). Yksinkertaien yhteydenottolomake voidaan luoda lisäosan Lomake-tekstikentässä esimerkiksi seuraavasti:

```
<label> Nimi (pakollinen)
    [text* your-name] </label>

<label> Sähköposti (pakollinen)
    [email* your-email] </label>

<label> Puhelinnumero
[tel your-phone] </label>

<label> Aihe
    [text your-subject] </label>

<label> Viesti
    [textarea your-message] </label>

[submit "Lähetä"]
```

Esimerkkikoodi 29. Contact Form 7 -lisäosassa on määritelty yhteydenottolomake. Lomake kirjoitetaan HTML-muodossa käyttämällä lisäosan lomaketunnisteita. Lomaketunnisteen ensimmäinen osa määrittää kentän sisällön tyypin, joka vaikuttaa muun muassa lomakkeen validointitoimintoihin. Tähdellä määritelty tyyppi kertoo validaattorille, että kenttä on pakollinen. Toiseksi kenttään on annettu sähköpostielementti, jolla kentän sisältöön voidaan jatkossa viitata. Submit-painikkeen tunnisteen toinen osa on yksinkertaisesti painikkeen teksti. Lomaketunnisteeseen voidaan vielä lisätä id tai luokka tyyliin id:id_nimi ja class:luokan_nimi.

Lomakkeen luonnin jälkeen voidaan määritellä, miten lähetetty tieto näkyy sähköpostissa. Tällöin lomakkeesta on noudettava kenttien sisältö, jonka voi tehdä käyttämällä edellä mainittuja sähköpostielementtejä. Esimerkkikoodin 29 lähettämän sähköpostin viesti voidaan muotoilla esimerkiksi seuraavasti:

```
Lähettäjä: [your-name]
Aihe: [your-subject]
Puhelinnumero: [your-phone]

Viesti:
[your-message]
```

Esimerkkikoodi 30. Esimerkkikoodissa 29 määritetyn yhteydenottolomakkeen lähettämä viesti määritellään tässä pelkällä tekstisisällöllä. Lisäosan lähettämässä Viesti-kentässä voidaan myös valita, halutaanko viestissä näyttää rivit, joiden sähköpostielementit ovat tyhjiä.

Tämän jälkeen voidaan muokata joitakin lisäosan valmiiksi määrittelemiä ilmoituksia, kuten lähetyksen virheilmoitukset tai lomakkeen validointiin liittyvät virheilmoitukset. Lopuksi voidaan määrittää joitakin lisäasetuksia. Lisäasetukset kirjoitetaan koodipätkillä, kuten ”subscribers_only: true”, joka estää sivustolle kirjautumattomia käyttäjiä lähettämästä viestiä lomakkeella. Toinen käytännöllinen lisäasetus sivuston kehitysvaiheessa on ”demo_mode: on”, joka estää varsianisen viestin lähetyksen

lomakkeella, mutta käy läpi kaikki validointiin liittyvät prosessit virheilmoituksia myöten. (Additional Settings, verkkoaineisto.)

Luotu yhteydenottolomake voidaan nyt lisätä sivustolle yksinkertaisesti shortcodella. Tässä esimerkissä luotiin lomake nimeltä ”Testilomake”, jolloin shortcode on seuraavanlainen:

```
[contact-form-7 id="240" title="Testilomake"]
```

Esimerkkikoodi 31. Lisäosan luoma shortcode, jolla edellä luotu yhteydenottolomake voidaan lisätä sivustolle.

Contact Form 7 tehostaa huomattavasti yhteydenottolomakkeen luontia. Yksinkertaisen yhteydenottolomakkeen luonti ei vaadi validointilogiikan määrittelyä, ja virheilmoituksien toiminnot ovat valmiiksi määriteltyjä lisäosassa, mikä nopeuttaa lomakkeen lisäämistä sivustolle. Lisäosassa ei ole muotoiluun liittyvää toiminnallisuutta, eli ne on tehtävä HTML/CSS-muodossa itse.

4.5 Duplicate Post

Duplicate Post -lisäosa lisää yksinkertaisen mutta tehokkaan toiminnon jokaiselle WordPress-artikkelityypille, ja se on artikkelin monistus. Lisäosan asennuksen ja aktivoinnin jälkeen artikkelin voi kopioida yksinkertaisesti valitsemalla Artikkelit-valikosta artikkelin otsikon alta ”Kopioi”, jolloin artikkelista luodaan toinen Luonnos. Artikkelia voi nyt muokata ilman, että se näkyy käyttäjälle ja sen jälkeen julkaista, kun muutokset ovat valmiit. Vaikka toiminto on hyvin yksinkertainen, voi se tehostaa sivuston ylläpitoa huomattavasti. Ylläpitäjälle voidaan nyt luoda artikkelipohjia nopeasti jo luoduista artikkeleista, joita kopioimalla ja muokkaamaalla saadaan uuden artikkelin luonnista tehokkaampaa. (Duplicate Post, verkkoaineisto.)

5 Johtopäätökset

WordPress antaa kehitysalustana laajat mahdollisuudet tehokkaasen sivustojen luontiin sekä ylläpitoon. Lisäosien myötä sekä luontia että ylläpitoa voidaan tehostaa entisestään.

Jos halutaan lisätä sivustolle pelkkä toiminnallisuus, on shortcode tehokas ratkaisu luoda uudelleenkäytettävä toiminnallisuus. Jos toisaalta halutaan tarjota käyttäjälle edes jonkinlainen asetusvalikko, on lisäosan luonti hyvä ratkaisu, sillä asetusvalikon lisääminen WordPressin hallintapaneeliin sujuu helposti. Jos lisäosassa käytetään shortcodea, on tällöin toki pystyttävä lisäämään pluginin luoma shortcode sivustolle, mikä voi vaatia hieman ohjelmointitaitoa. Jos halutaan tarjota ratkaisu, joka toimii ainoastaan graafisen käyttöliittymän avulla, antavat rakennustyökalut kuten Beaver Builder erittäin hyvän alustan drag & drop -tyyliselle sivun rakentamiselle sekä ylläpidolle. Lisämoduulien avulla voidaan käyttäjälle tarjota graafinen käyttöliittymä, jolloin sivuston mukautetun toiminnallisuuden muokkaaminen eivät välttämättä vaadi lainkaan ohjelmointitaitoa. WordPress REST API antaa mahdollisuuden luoda client-puoleen nojautuva ratkaisu, jonka ei välttämättä tarvitse edes sijaita samalla sivustolla kuin WordPress-asennuksen.

Lähteet

Add a module to your plugin. Verkkoaineisto. https://kb.wpbeaverbuilder.com/article/597-cmdg-02-add-a-module-to-your-plugin. Luettu 19.4.2019.

Add custom JavaScript. Verkkoaineisto. https://kb.wpbeaverbuilder.com/article/95-add-custom-javascript. Luettu 20.4.2019.

Additional Settings. Verkkoaineisto. https://contactform7.com/additional-settings/. Luettu 22.4.2019.

Changelog. Verkkoaineisto. https://developer.wordpress.org/rest-api/changelog/. Luettu 19.4.2019.

Child Themes. Verkkoaineisto. https://developer.wordpress.org/themes/advanced-topics/child-themes/#what-is-a-child-theme. Luettu 19.4.2019.

Controller Classes. Verkkoaineisto. https://developer.wordpress.org/rest-api/#controller-classes. Luettu 19.4.2019.

Create a plugin. Verkkoaineisto. https://kb.wpbeaverbuilder.com/article/596-cmdg-01-create-a-plugin. Luettu 19.4.2019.

Create Beaver Builder shortcodes. Verkkoaineisto. https://kb.wpbeaverbuilder.com/article/192-use-shortcodes-in-your-layouts. Luettu 20.4.2019.

Define module settings. Verkkoaineisto. https://kb.wpbeaverbuilder.com/article/598-cmdg-03-define-module-settings. Luettu 19.4.2019.

Duplicate Post. Verkkoaineisto. https://wordpress.org/plugins/duplicate-post/. Luettu 16.4.2019.

Editing Form Template. Verkkoaineisto. https://contactform7.com/editing-form-template/. Luettu 22.4.2019.

Features. Verkkoaineisto. https://wordpress.org/about/features/. Luettu 16.4.2019.

Gallery Shortcode. Verkkoaineisto. https://codex.wordpress.org/Gallery_Shortcode. Luettu 19.4.2019.

Get the slug or ID for a shortcode. Verkkoaineisto. https://kb.wpbeaverbuilder.com/article/191-get-the-slug-or-id-for-a-shortcode. Luettu 19.4.2019.

Getting Started. Verkkoaineisto. https://developer.wordpress.org/plugins/plugin-basics/#getting-started. Luettu 20.4.2019.

How to write a WordPress plugin. Verkkoaineisto. https://www.yaco-niello.com/blog/how-to-write-wordpress-plugin. Luettu 20.4.2019.

Limit Login Attempts Reloaded. Verkkoaineisto. https://wordpress.org/plugins/limit-login-attempts-reloaded/. Luettu 20.4.2019

Module CSS. Verkkoaineisto. https://kb.wpbeaverbuilder.com/article/601-cmdg-06-module-css. Luettu 19.4.2019.

Module HTML. Verkkoaineisto. https://kb.wpbeaverbuilder.com/article/600-cmdg-05-module-html. Luettu 19.4.2019.

Our Story. Verkkoaineisto. https://WordPress.org/about/. Luettu 16.4.2019.

Pages. Verkkoaineisto. https://codex.wordpress.org/Pages. Luettu 23.4.2019.

Plugin API. Verkkoaineisto. https://codex.wordpress.org/Plugin_API. Luettu 19.4.2019.

Plugin Files, verkkoaineisto. https://codex.wordpress.org/Writing_a_Plugin. Luettu 20.4.2019

Post Types. Verkkoaineisto. https://codex.wordpress.org/Post_Types. Luettu 16.4.2019.

Requests. Verkkoaineisto. https://developer.wordpress.org/rest-api/#requestsl. Luettu 19.4.2019.

Required files. Verkkoaineisto. https://developer.wordpress.org/themes/getting-started/what-is-a-theme/#required-files. Luettu 16.4.2019.

Requirements. Verkkoaineisto. https://wordpress.org/about/requirements/. Luettu 16.4.2019.

Responses. Verkkoaineisto. https://developer.wordpress.org/rest-api/#responses. Luettu 19.4.2019.

REST API Handbook. Verkkoaineisto. https://developer.wordpress.org/rest-api/. Luettu 20.4.2019.

Routes & endpoints. Verkkoaineisto. https://developer.wordpress.org/rest-api/#routes-endpoints. Lueattu 19.4.2019.

Schema. Verkkoaineisto. https://developer.wordpress.org/rest-api/#schema. Luettu 19.4.2019.

Taxonomies. Verkkoaineisto. https://codex.wordpress.org/Taxonomies. Luettu 19.3.2019.

The Dynamic Nature of WordPress Pages. Verkkoaineisto. https://codex.wordpress.org/Pages. Luettu 23.4.2019.

The Shortcode API. Verkkoaineisto. https://codex.wordpress.org/Shortcode_API. Luettu 19.4.2019.

To create a subpage. Verkkoaineisto. https://codex.wordpress.org/Pages. Luettu 23.4.2019.

Use shortcodes in your layouts. Verkkoaineisto. https://kb.wpbeaverbuilder.com/article/192-use-shortcodes-in-your-layouts. Luettu 20.4.2019.

What is the difference between a theme and a plugin? Verkkoaineisto. https://developer.wordpress.org/themes/getting-started/what-is-a-theme/#what-is-the-difference-between-a-theme-and-a-plugin. Luettu 20.4.2019.

What Pages Are Not. Verkkoaineisto. https://codex.wordpress.org/Pages. Luettu 23.4.2019

WPML Core and Add-on Plugins. Verkkoaineisto. https://wpml.org/documentation/wpml-core-and-add-on-plugins/. Luettu 23.4.2019.