jamk.fi

Nettimoton REST-rajapinnan integrointi WordPress-sivustoon

Otto Savolainen

Opinnäytetyö
Maaliskuu 2020
Tekniikan ala
Insinööri (AMK), tieto- ja viestintätekniikka

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Savolainen, Otto	Opinnäytetyö, AMK	Maaliskuu 2020
	Sivumäärä 45	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi

Nettimoton REST-rajapinnan integrointi WordPress-sivustoon

Tutkinto-ohjelma

Insinööri (AMK), tieto- ja viestintätekniikka

Työn ohjaaja(t)

Pasi Manninen, Kari Niemi

Toimeksiantaja(t)

Trimedia Oy

Tiivistelmä

Opinnäytetyön toimeksiantajana toimi digitoimisto Trimedia Oy, joka tekee pääasiassa verkkosivustoja WordPress-julkaisujärjestelmällä. Tavoitteena oli integroida Nettimoton REST-rajapinta WordPress-sivustoon ja näin mahdollistaa haettujen ajoneuvojen myynti-ilmoitusten näyttäminen asiakkaan verkkosivuilla yksinkertaisemmin, kuin näyttämällä ne ulkoisesta palvelusta.

Työssä selvitettiin, miten Nettimoton rajapinta voidaan integroida WordPressin kanssa ja miten rajapinnasta haettua dataa voidaan käsitellä ja hyödyntää. Tutkittiin myös sitä, miten haettu ajoneuvodata voidaan tallentaa WordPressiin Advanced Custom Fieldsin avulla. Opinnäytetyössä tutkittiin lisäksi WordPress-lisäosakehityksen vaiheita ja kuinka rajapinta voitaisiin integroida lisäosana toteutuksen uudelleenkäytön helpottamiseksi.

Alun toteutuksessa käytettiin teemaa, johon muokattiin tarvittavat rajapintakutsut sekä ajoneuvojen tallennus, mutta aikataulun mahdollistamana tästä kehitettiin WordPress-lisäosa. Lisäksi tehtiin sivupohjia helpottamaan ajoneuvojen näyttämistä ja hakemista.

Tuloksena saatiin helposti käyttöönotettava WordPress-lisäosa, jolla voidaan hakea Nettimotosta yrityksen myymät ajoneuvot. Lisäosan lisäksi WP:n teemaan luotiin sivupohjia ajoneuvoilmoitusten selailua, hakemista, lajittelua ja näyttämistä varten. Näiden muokkaaminen onnistuu pienellä vaivalla, joten toteutuksen hyödyntäminen on helppoa myös tulevissa asiakasprojekteissa.

Avainsanat (asiasanat)

WordPress, Nettimoto, rajapinta, integrointi, lisäosa

Muut tiedot (Salassa pidettävät liitteet)

jamk.fi

Description

Author(s)	Type of publication	Date
Savolainen, Otto	Bachelor's thesis	March 2020
		Language of publication: Finnish
	Number of pages 45	Permission for web publication: x

Title of publication

Integrating Nettimoto's REST API to WordPress site

Degree programme

Information and Communication Technologies

Supervisor(s)

Manninen, Pasi; Niemi, Kari

Assigned by

Trimedia Oy

Abstract

The thesis was assigned by Trimedia Oy which builds mainly websites with WordPress content management system. The main object was to integrate Nettimoto's REST-API into WordPress site and get vehicle advertisements at a customer's website easier than if they were shown from a third-party service.

The work studied how the interface can be integrated into WordPress and how the data gathered from the interface can be used. It was explored how the retrieved vehicle data can be saved using Advanced Custom Fields plugin. The thesis also studied WordPress plugin development and how the interface could be integrated as a plugin to help with the reusability of the implementation.

The implementation was at first created by creating necessary interface calls and vehicle saving procedure to the theme. The schedule allowed to develop this further to a WordPress plugin. The page templates were also created to help with the search and display of vehicle advertisements.

The results of the development were an easy to deploy WordPress plugin which can get company's for sale vehicles from Nettimoto's interface. In addition to the plugin, the page templates were also made to let users browse, search, sort and show vehicles. These are easily modified to suit different projects, which makes the implementation more future proof.

Keywords/tags (subjects)

WordPress, Nettimoto, interface, integration, plugin

Miscellaneous (Confidential information)

Sisältö

Kuviot

Sanasto

AJAX Asynchronous JavaScript And XML on tekniikka, jolla verkkosivu voi kommunikoida palvelimen kanssa ilman sivun uudelleen lataamista.

Base64Url on enkoodausalgoritmi, joka muuntaa mitä tahansa merkkejä verkko-osoitteeseen sopivaan muotoon.

CSS Cascading Style Sheet on verkkosivuilla käytettävien tyylien tiedostomuoto.

JSON JavaScript Object Notation on nimi-/arvopareista muodostuva tiedostomuoto, jota on helppo lukea, kirjoittaa ja parsia koneellisesti.

JWT JSON Web Token on avoimen standardin toteutus, joka mahdollistaa tiedon välittämisen JSON-objektina kahden osapuolen välillä.

iframe HTML-elementti, jolla voidaan upottaa toinen verkkosivu HTML-dokumenttiin.

MySQL Relaatiotietokantaohjelmisto, joka käyttää SQL-kyselykieltä hakujen, muutosten ja lisäyksien toteuttamiseen tietokannasta

OAuth 2 Yleinen tunnistautumisprotokolla, jolla voidaan saada pääsy jonnekin verkkopalveluun.

PHP Hypertext PreProcessor on palvelimilla ajettava ohjelmointikieli, jolla tuotetaan ensisijaisesti dynaamisia verkkosivustoja.

REST Representational state transfer on järjestelmäarkkitehtuuri verkkosovellusten luontiin.

WordPress Suosittu verkkosivujen sisällönhallintajärjestelmä

WYSIWYG What You See Is What You Get on yleinen muokkain sisällönhallintajärjestelmissä, johon voidaan mm. kirjoittaa ja lisätä kuvia.

1 Työn lähtökohdat

1.1 Taustaa

Useissa verkkosivuprojekteissa on tarvetta näyttää sivustolla yrityksen myymiä ajoneuvoja. Ajoneuvot on tallennettu usein johonkin ulkoiseen järjestelmään, jossa niiden hallinta tapahtuu, mutta usein niitä näytetään omilla sivuilla esimerkiksi vain kankean iframe-elementin avulla. Tämä on tietysti vaivaton ratkaisu, mutta se ei jätä juuri mitään mahdollisuuksia muokata ulkoasua, sillä iframen CSS-määritykset ladataan ulkopuolisesta palvelusta. Olisikin paljon kätevämpää, jos ajoneuvot haettaisiin WordPressin omasta tietokannasta ja niitä voitaisiin käsitellä kuten mitä tahansa muutakin WP:n sisältötyyppiä.

Tilannetta parantamaan lähdettiin kehittämään järjestelmää, jonka avulla tämän ajoneuvohaun saisi automatisoitua ja integroitua WordPressin kanssa. Yritysten käyttämiä myytävien ajoneuvojen hallintajärjestelmiä on useita, mutta Nettix:n Nettiauto, Nettimoto, Nettikone ja Nettivene ovat erittäin suosittuja ja tarjoavat hyvin dokumentoidun rajapinnan tähän tarkoitukseen.

Tähän työhön valikoituikin monista vaihtoehdoista Nettimoto, koska yhden uuden verkkosivustoprojektin seurauksena ilmeni tarvetta hakea tälle sivustolle juuri Nettimoton ajoneuvoja. Nettix-rajapinnat ovat kuitenkin keskenään varsin samanlaisia, joten nyt toteutettu lisäosa saadaan toimimaan muidenkin saman perheen rajapintojen kanssa verrattain pienillä muutoksilla.

Nettimoto on yksi monista Nettix Oy:n markkinapaikoista, ja se on Otava-konserniin kuuluvan Otava Markkinapaikat -liiketoimintayksikön tytäryhtiö (Markkinapaikat n.d.; Mikä on Nettix? n.d.). Se on tarkoitettu niin yksityisille myyjille kuin yrityksillekin, ja Nettimoton kautta myydäänkin vuosittain yli 100 000 ajoneuvoa (Moottoripyörät, mönkijät ja moottorikelkat n.d.).

1.2 Tilaaja

Työn toimeksiantaja oli digitoimisto Trimedia Oy. Trimedia tekee ensisijaisesti verkkosivustoja WordPress-julkaisujärjestelmällä kaiken kokoisille yrityksille. Sen toimipisteet sijaitsevat Jyväskylässä ja Espoossa.

1.3 Tavoitteet

Tässä työssä tutustuttiin Nettimoton yrityksille tarjolla olevaan rajapintaan ja toteutettiin kehittämistyönä WordPress-julkaisujärjestelmälle lisäosa, jonka avulla voidaan hakea palveluun tallennettuja ajoneuvoja mille tahansa WordPressiä käyttävälle sivustolle.

Lisäosan avulla voidaan vähentää eri asiakasprojekteihin kuluvaa työmäärää muuttamalla ajoneuvojen hakeminen automaattiseksi lisäosaksi. Tehtäväksi jää enää muokata sivuston teemaan ajoneuvosivujen ulkoasu sekä mahdolliset haku- ja lajittelutoiminnot niin haluttaessa. Etuna on lisäksi myös se, että työvaiheista putoavat pois hankalat rajapintakutsut ja JSON-datan parsiminen, eli taittajalta ei vaadita mitään erikoisosaamista näiltä osa-alueilta ja hän suoriutuu urakasta todennäköisemmin nopeammin.

2 Käytetyt teknologiat

2.1 WordPress

2.1.1 Yleistä

Yhä useammat verkkosivustot käyttävät jotain sisällönhallintajärjestelmää sisällön tuotantoon ja hallintaan. Ne mahdollistavat useiden henkilöiden osallistumisen verkkosivujen sisällön luomiseen, muokkaamiseen ja julkaisuun. Järjestelmien etuna on verkkopohjaisena mahdollisuus työskennellä mistä käsin tahansa ja useiden käyttäjäroolien luominen eri tarkoituksiin. Lisäksi käyttäjiltä ei vaadita ohjelmointitaitoja, sillä

graafisen käyttöliittymän ja esimerkiksi helppokäyttöisen WYSIWYG-muokkaimen ansiosta sisällön tuotanto on helppoa. (Content Management System n.d.)

Sisällönhallintajärjestelmiä on useita, mutta suosituin on WordPress: miljoonasta liikenteen perusteella suosituimmasta sivustosta, joilla on käytössä jokin sisällönhallintajärjestelmä, WordPressin osuus on 48 % (CMS Usage Distribution in the Top 1 Million Sites n.d.) Jopa yli 35 % kaikista maailman verkkosivuista käyttää alustanaan WordPressiä (About Features n.d.)

WordPressin kehityksen aloittivat vuonna 2003 B2/cafelog uutis- ja blogityökalun pohjalta Mike Little ja Matt Mullenweg. Se on rakennettu PHP:llä, ja tietokantana toimii MySQL. WordPress on avointa lähdekoodia GPLv2-lisenssin mukaisesti. (Our Story n.d.) Tämä tarkoittaa sitä, että WordPressiä voidaan käyttää mihin tarkoitukseen tahansa, sen toimintaa voidaan tutkia ja muuttaa halutulla tavalla, sitä voidaan levittää sellaisenaan tai muokattuna vapaasti (Bill of rights n.d.) WordPressin tavoitteena on olla helppokäyttöinen käyttäjille ja julkaisijoille, mutta myös tarjota riittävästi ominaisuuksia kehittäjille.

2.1.2 Ominaisuudet

WordPress tarjoaa laajasti ominaisuuksia erilaisten sivustojen tarpeisiin. Valmiiden toimintojen lisäksi se on täysin muokattavissa ja lisätoimintoja on mahdollista hankkia lukuisilla lisäosilla.

Julkaisutyökalut

Julkaisutyökalut sisältävät kattavat työkalut helpottamaan sisällön tuotantoa ja hallintaa. Näihin lukeutuvat esimerkiksi luonnokset, ajastettu julkaisumahdollisuus ja versionhallinta. Lisäksi artikkeleita ja sivuja voi suojata esimerkiksi salasanalla, jos niistä ei haluta julkisia. (Features n.d.)

Käyttäjät ja niiden hallinta

Käyttäjien hallinta mahdollistaa eri roolien asettamisen eri ihmisille. Näin esimerkiksi uutisten kirjoittajille ei tarvitse antaa pääsyä muokkaamaan sivuston asetuksia, vaan

sitä varten voi olla erillinen ylläpitotili. Sivuston lukijat voivat myös rekisteröityä käyttäjiksi ja esimerkiksi kirjoittaa kommentteja artikkeleihin. (Mt.)

Mediatoiminnot

Erilaista mediasisältöä varten WordPressistä löytyvät toiminnot kuvien ja muun median hallintaa varten. Kuviin voi lisätä tekstitykset ja niistä voi luoda gallerioita sivuille upotettavaksi. Niille voi myös tehdä yksinkertaisia muokkaustoimenpiteitä, kuten esimerkiksi rajaus, koon muuttaminen ja kierto. (Mt.)

Teemat

Teemojen avulla voidaan muuttaa täysin sivuston ulkoasu ja vaikuttaa sen sisältöön sekä järjestykseen. Niitä voidaan asentaa suoraan WP:n ohjauspaneelin kautta tai vaihtoehtoisesti teemoja voidaan ladata suoraan palvelimelle WordPressin teemakansioon. (Mt.)

Lisäosat

Lisäosat ovat kätevä tapa lisätä toiminnallisuutta verkkosivustolle. Ne voivat olla esimerkiksi keskustelufoorumeita, kuvagallerioita, käännöstyökaluja tai vaikkapa hakukoneoptimoijia. Lisäosia voidaan asentaa samaan tapaan kuin teemojakin WP:n lisäosakaupasta tai lataamalla oma. (Mt.)

2.2 WordPress kehittäjille

2.2.1 Esittely

WordPress voidaan jakaa kolmeen suureen osaan: ytimeen, teemoihin ja lisäosiin. Ydin sisältää perustoiminnallisuuden, ja sen muokkaamista ei suositella, koska WordPressin päivittyessä tehdyt muutokset ylikirjoitettaisiin. (Introduction to Plugin Development n.d.) Teemoilla puolestaan vaikutetaan siihen, mitä sivustolla näkyy ja miltä se näyttää. Lisäosat ovat suositeltava ja erittäin monipuolinen keino lisätä melkein mitä tahansa ominaisuuksia WP-sivustolle ja näin laajentaa sen toiminnallisuutta.

2.2.2 Lisäosat

Lisäosat voivat olla pieniä ja sisältää tarvittaessa vain yhden PHP-tiedoston tai suuria, useita kansioita ja tiedostoja sisältäviä kokonaisuuksia. Ne voivat olla muun muassa CSS-, JavaScript- ja PHP-tiedostoja. (Introduction to Plugin Development n.d.)

Yksinkertaisimmillaan lisäosa siis koostuu yhdestä tiedostosta ja erityisestä otsikkokommenttiosiosta. Aluksi kannattaa tehdä kansio ja nimetä se lisäosan mukaan. Kansio sijoitetaan WordPressin oletuskonfiguraatiossa wp-content/plugins/-sijaintiin ja sinne luodaan PHP-tiedosto, esimerkiksi lisaosan-nimi.php, johon tehdään otsikkokommentti:

```
/**
* Plugin Name: Nettix-haku
* Description: Lisäosa, joka hakee Nettix-rajapinnasta
ajoneuvot ja tallentaa ne Wordpressiin ACF:n avulla.
* Version: 1.0
* Author: Otto Savolainen
*/
```

Kommenttiin voi lisätä paljon erilaista tietoa, mutta vain lisäosan nimi on pakollinen. (Plugin Basics n.d.) Tämän jälkeen lisäosa löytyy WordPressin ohjauspaneelista Lisäosat-kohdan alta, ja se voidaan ottaa käyttöön.

Hyviä käytänteitä

Lisäosan laajuuden mukaan sen kehityksessä on paljon huomioitavaa, mutta noudattamalla hyväksi todettuja käytäntöjä koodi pysyy siistinä ja toimivana nyt sekä myös tulevaisuudessa (Best Practices n.d.).

Nimeämisristiriidat voivat aiheuttaa ongelmia, ja niitä voi syntyä, jos esimerkiksi toinen lisäosa käyttää samaa muuttujan tai funktion nimeä. Tätä voi ehkäistä määrittelemällä muuttujat funktioiden sisällä tai laittamalla jonkin uniikin etuliitteen kaikkiin muuttujiin, funktioihin jne. Tähän voi käyttää vaikkapa oman lisäosan nimeä. (Avoid Naming Collisions n.d.)

Tiedostorakenteen pitäisi noudattaa mallia, jossa lisäosakansion juuressa on vain lisaosan-nimi.php-tiedosto sekä mahdollisesti uninstall.php. Kaikki muut tiedostot pitäisi olla lajiteltuina omiin alikansioihinsa aina kun mahdollista. Tämä auttaa lisäosan parissa työskenteleviä. (File Organization n.d.)

Koukut

WordPress sisältää useita koukkuja (hooks), joiden avulla voidaan esimerkiksi ajaa funktioita tietyn tapahtuman kohdalla. Koukkuja on kahta tyyppiä: toimintoja (actions) ja suodattimia (filters). Toimintoja voidaan käyttää, jos halutaan lisätä tai muuttaa WordPressin toiminnallisuutta ja suodattimia puolestaan käytetään sisällön muuttamiseen, kun sitä ladataan ja näytään verkkosivuston käyttäjälle. (Hooks: Actions and Filters n.d.)

Erityisesti lisäosia varten on kolme hyödyllistä koukkua, joiden avulla voidaan suorittaa toimenpiteitä oikea-aikaisesti. register_activation_hook() ajetaan aktivoitaessa lisäosa, eli siihen on hyvä kytkeä esimerkiksi oletusasetusten luonti. register_deactivation_hook() on esimerkiksi väliaikaisen tiedon poistamista varten, sillä se aktivoituu kytkettäessä lisäosa pois käytöstä. Lisäosan poiston yhteydessä ajetaan register_uninstall_hook() ja sillä voidaan esimerkiksi poistaa tietokantaan luodut taulut. (Basic Hooks n.d.)

Toiminnot

WordPress sisältää toimintoja (actions), joiden avulla omia funktioita voidaan suorittaa halutussa vaiheessa. Toiminto voidaan lisätä luomalla ensin kutsuttava funktio ja lisäämällä se WP:n add_action()-funktioon:

```
add_action('nettimoto_import', 'importMotos');
```

Sille annetaan parametreiksi haluttu koukku ja tämän perään itse kutsuttava funktio. (Actions n.d.)

Toiminnoille voidaan antaa myös edellä mainittujen kahden pakollisen parametrin, koukun ja kutsuttavan funktion, lisäksi kaksi lisäparametriä: prioriteetin ($priority) ja

hyväksytyt argumentit ($accepted_args). Molemmat näistä ovat kokonaislukuja (int) ja ne voidaan laittaa pilkuilla erotettuna kutsuttavan funktion perään. (Additional Parameters n.d.)

Prioriteetti määrittää missä järjestyksessä kutsuttavat funktiot ajetaan. Jos esimerkiksi on toimintoja, joissa init-koukulla ajetaan useita funktioita, tähän järjestykseen voidaan vaikuttaa prioriteettiarvolla. Jos arvoa ei aseteta, sen oletus on 10. Pienemmät luvut ovat järjestyksessä ensin, ja suuremmat ajetaan viimeiseksi:

```
add_action('init', 'run_me_early', 9);
add_action('init', 'run_me_normal'); // default value of
10 is used since a priority wasn't specified
add_action('init', 'run_me_late', 11);
```

Mikäli useilla funktioilla on sama prioriteetti, se kumpi on rekisteröity ensin, ajetaan ensin. (Priority n.d.)

Järjestyksen lisäksi add_action():lle voidaan vielä antaa neljänneksi parametriksi välitettävien argumenttien määrä. Tällöin koukun suoritusvaiheessa voidaan välittää funktion käyttämät parametrit kutsuttavalle funktiolle, mikäli niin halutaan. Koko toiminto voisi olla siis esimerkiksi tällainen: add_action(”init”, ”ajettava_funktio”, 10, 2). (Number of Arguments n.d.)

Suodattimet

Suodattimet ovat toimintojen tapaan koukkuja, ja ne mahdollistavat muiden funktioiden datan muokkaamisen. Suodattimien pitäisi toimia eristetysti, ja niiden ei tulisi vaikuttaa globaaleihin muuttujiin. (Filters n.d.) Niiden käyttöä varten tulee luoda ensiksi kutsuttava funktio ja koukku, johon funktio liitetään. Tätä varten WordPressissä on add_filter()-funktio. Sille annetaan käytettävä suodatin, joka määrittää, milloin suoritus tapahtuu, ja itse kutsuttava funktio:

```
function wporg_filter_title($title) {
   return 'The ' . $title . ' was filtered';
}
add_filter('the_title', 'wporg_filter_title');
```

Suodattimille on olemassa myös samat parametrit kuin toiminnoille, eli prioriteetti ja hyväksytyt argumentit. Ne toimivat samalla tavalla kuin toimintojen vastaavat. (Mt.)

2.2.3 Teemat

WordPress-teema muuttaa verkkosivuston ulkoasun ja sen rakenteen. Teemoja vaihtamalla voidaan siis vaikuttaa siihen, mitä sivuston vierailija näkee. Valmiita teemoja voidaan ladata lisäosien tapaan wordpress.orgin omasta palvelusta. (What is a Theme? n.d.)

Teemat vaativat vähintään index.php- ja style.css-tiedostot. Usein ne sisältävät kuitenkin paljon muutakin, esimerkiksi PHP-, CSS-, JavaScript-, kieli-, teksti- ja kuvatiedostoja. (Required files n.d.)

Teeman hierarkia

WordPress-teemat ovat modulaarisia ja koostuvat osista, joita on useita eri tarkoituksiin. Näitä osia ovat pohjatiedostot (template files), sivupohjat (page templates) ja pohjien osat (template partials). (Template Files n.d.) Lopullisen, näytettävän verkkosivun HTML muodostuu näistä osista. WordPress etsii kulloinkin käytettävää osaa teeman hierarkian perusteella, kunnes se löytää sopivan. Jos sopivaa tiedostoa ei löydy, se käyttää teemasta löytyvää index.php:ta. (The Template File Hierarchy n.d.)

Esimerkiksi jos sivun osoite on ”http://example.com/blog/” ja vierailija klikkaa linkkiä kategoriasivulle, jonka osoite on ”http://example.com/blog/category/your-cat/”, WordPress etsii teemakansiosta pohjatiedostoa, joka täsmää kategorian ID:hen. (Examples n.d.) Etsinnän vaiheet menevät seuraavasti:

1. WP etsii teemakansiosta pohjatiedostoa, joka täsmää kategorian polkutunnukseen. Jos se on esimerkiksi ”kissa”, niin WordPress etsii tiedostoa ”category-kissa.php”.
2. Mikäli tätä ei löydy ja kategorian ID on 4, niin WordPress yrittää etsiä category-4.php-tiedostoa.
3. Jos tätäkään ei löydy, niin WP: hakee pelkkää category.php-tiedostoa.

4. Mikäli kategoriatiedostoja ei löydy, niin WordPress yrittää löytää arkistosivun archive.php.
5. Viimeisenä oljenkortena se palaa index.php:hen, jos mitään muuta ei löytynyt.

(Mt.)

Pohjatiedoston löytymisen jälkeen näytettävä sivu rakentuu sen perusteella. Tiedostossa voi olla kuitenkin ”template tageja”, joiden perusteella sivulle haetaan vielä sisältöä muista tiedostoista. Nämä ovat funktioita, jotka toimivat PHP:n includen tapaan, eli ne sisällyttävät tiedoston sisällön siihen kohtaan missä ovat. Yleisimmin käytettyjä ovat esimerkiksi get_header() ja get_footer(). Ne hakevat ylä- ja alatunnisteen sisältävät header.php- ja footer.php-tiedostot. Lisäksi sisällön mukaan sivu saattaa käyttää pohjien osia näyttämään esimerkiksi sivupalkkia tai muuta vastaavaa osaa. (What is a Template Tag? n.d.)

Template Tag -funktioiden ansiosta samoja asioita ei tarvitse tehdä monta kertaa, vaan useilla sivuilla näkyvät elementit voidaan tehdä vain yhteen tiedostoon ja sisällyttää kaikkialle missä niitä tarvitaan. Tämä vähentää työmäärää, mutta myös selkeyttää teeman tiedostojen rakennetta.

2.2.4 Sisältötyypit

WordPress sisältää oletuksena viisi sisältötyyppiä. Nämä eivät usein kuitenkaan riitä, joten onkin hyvä, että niitä voidaan tehdä helposti lisää. Esimerkiksi tässä työssä Nettimotosta tuoduille ajonevoille tehtiin oma ”moto”-sisältötyyppi, jollaisina ne tallennettiin. Jokainen luotu sisältötyyppi saa oman valikkonsa WordPressin ohjauspaneeliin, josta sitä voidaan hallita. Yksittäinen sisältötyypin ”artikkeli” on oma sivunsa yksilöllisellä ositteella. Sisältötyyppien rekisteröintiin käytetään register_post_type()-funktiota:

```
function wporg_custom_post_type() {
    register_post_type('wporg_product',
        array(
            'labels' => array(
                'name' => __('Products'),
                'singular_name' => __('Product'),
            ),
            'public' => true,
            'has_archive' => true,
        )
    );
}
add_action('init', 'wporg_custom_post_type');
```

Omille sisältötyypeille voidaan luoda sivupohjat, joita käytetään automaattisesti. Niiden nimeksi tulee antaa ”single-sisaltotyypin_nimi.php” ja ”archive-sisaltotyypin_nimi.php”. (Custom Post Type Templates n.d.) Sisältö näytetään tavallisella WP:n kyselyllä vain laittamalla WP_Queryn argumentteihin sisältötyyppi ja sen nimi (Querying by Post Type n.d):

```
$args = [
    'post_type' => 'product',
    'posts_per_page' => 10,
];
$loop = new WP_Query($args);
while ($loop->have_posts()) {
    $loop->the_post(); ?>
    <div class="entry-content">
        <?php the_title(); ?>
        <?php the_content(); ?>
    </div><?php
}
```

2.3 Advanced Custom Fields -lisäosa

2.3.1 Esittely

Advanced Custom Fields on Elliot Condonin kehittämä WordPressille saatava lisäosa. Sen tarkoituksena on laajentaa WP:n sisältötyyppien tiedon tallennusmahdollisuuksia tarjoamalla erilaisia lisäkenttiä monenlaiselle datalle ja tarjota näille selkeä käyttöliittymä. (Condon n.d.; What is ACF? n.d.)

2.3.2 Lisäkentät

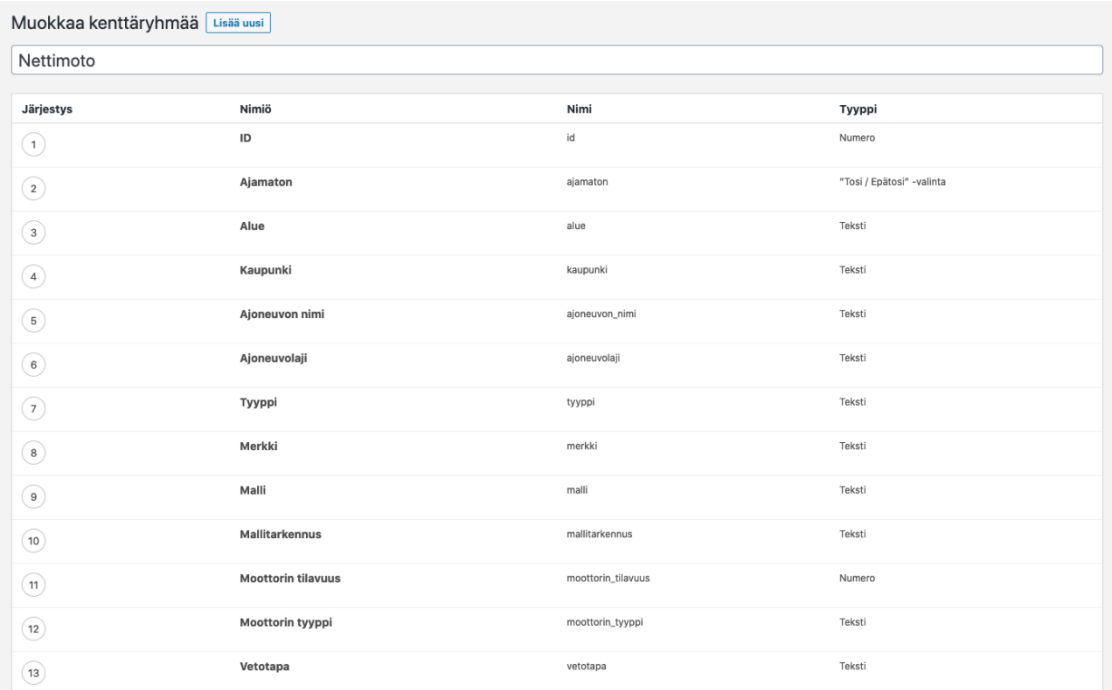
WordPressin perusasennuksen mukana tulee kaksi sisältötyyppiä: Artikkelit (Posts) ja Sivut (Pages). Nämä mahdollistavat sisällön tuottamisen ohjauspaneelista käsin ja sen näyttämisen sivuston puolella. Usein pelkästään sivun otsikko ja sisältö eivät kuitenkaan riitä, vaan tarvetta olisi tallentaa muutakin dataa. Tätä varten WordPressistä löytyvät lisäkentät (Custom Fields), joiden avulla voidaan tallentaa erilaisia metatietoja sisältötyyppeihin. (What is a custom field? n.d.)

Käyttäjä voisi luoda esimerkiksi Kirjat-sisältötyypin ja sille kirjailija- ja lajityyppi-lisäkentät. Nämä kentät kulkevat sisältötyypin mukana, eli jos sisältötyyppiin luodaan uusi sivu, samoihin lisäkenttiin voidaan taas syöttää halutut arvot, ja ne tallentuvat sivun metatiedoiksi. Metatiedot tallennetaan avain-/arvopareina ja niitä voidaan näyttää sivustolla esimerkiksi get_post_meta()-funktion avulla (Custom Fields n.d.). Funktiolle annetaan Post ID ja tarvittaessa halutun kentän avain, jolloin se palauttaa avainta vastaavan lisäkentän arvon. Jos avaimen jättää tyhjäksi, funktio palauttaa taulukon kaikista sivun kentistä. (get_post_meta() n.d.)

Advanced Custom Fields laajentaa tätä WordPressin toiminnallisuutta ja tarjoaa monipuolisen käyttöliittymän lisäkenttien luontiin ja hallinnointiin. Valmiita kenttätyyppejä on lähes kolmekymmentä erilaista, ja ne mahdollistavat helposti hyvinkin monenlaisen datan tallentamisen ja näyttämisen. Kenttien lisäksi ACF sisältää suuren joukon funktioita, joilla voidaan esimerkiksi näyttää, päivittää, lisätä tai poistaa kenttiä. (What is ACF? n.d.)

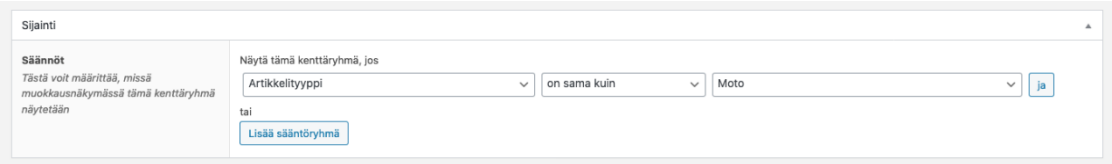
2.3.3 Lisäosan käyttö

Lisäkenttien luonti ACF:llä tapahtuu siten, että aluksi luodaan kenttäryhmä ja sille sopivat kentät. Nettimoton lisäosaa varten tarvittiin vain yksi kenttäryhmä, joka sisälsi kentät kaikista ajoneuvojen ominaisuuksista, jotka haluttiin tallentaa. (Ks. kuvio 1.) Ryhmiä kannattaa tilanteen mukaan tehdä useita, koska se mahdollistaa vain relevanttien kenttien näyttämisen ja kenttien järjestyksen vaihtamisen raahaamalla sivun muokkausnäkymässä.



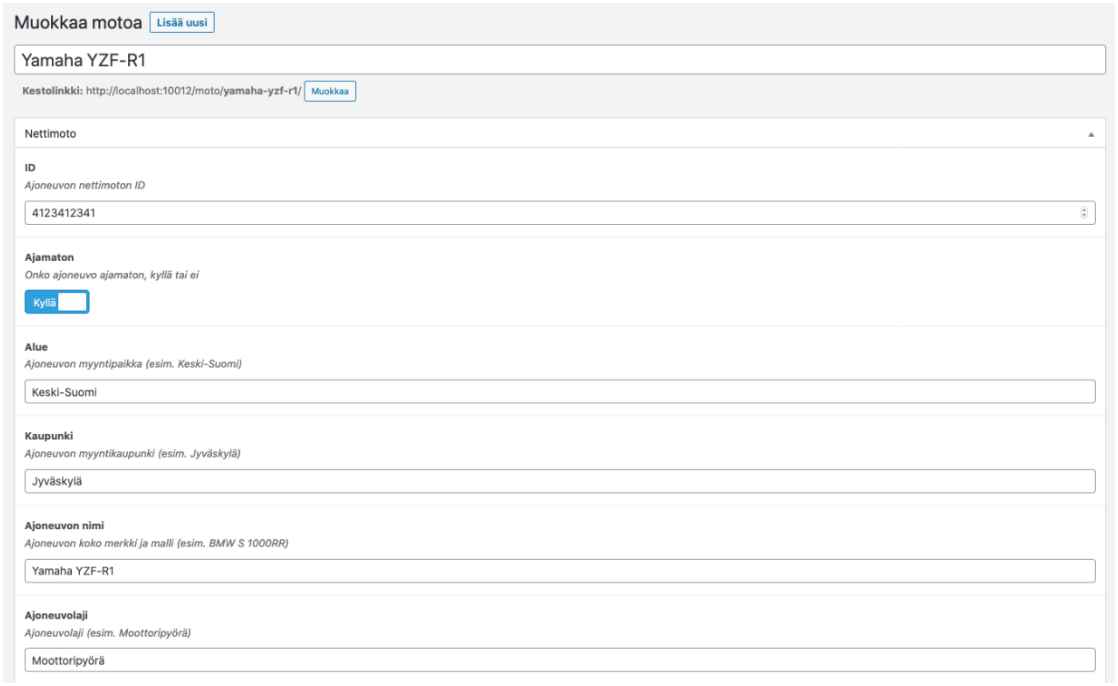
Kuvio 1. Osa Nettimoton ajoneuvoja varten luoduista ACF-kentistä

Koska eri kenttäryhmät ovat eri tarkoituksia varten, on hyödyllistä asettaa ehtoja niiden näkyvyydelle. Tämän voi tehdä ryhmän alapuolelta löytyvästä asetusvalikosta. Kenttäryhmän voi esimerkiksi asettaa näkymään vain tietyn sisältötyypin muokkausnäkymässä. (Ks. kuvio 2.) Sääntöjä voi tarvittaessa luoda enemmänkin, mikä mahdollistaa monimutkaistenkin ehtojen hyödyntämisen.



Kuvio 2. Lisäkenttäryhmän näkyvyysasetukset

Muokkaaminen tapahtuu kenttäryhmän näkyvyysehtojen mukaisesti sivun muokkausnäkymässä (ks. kuvio 3). Eri kenttätyypit ovat tietysti erilaisia, mutta yksinkertaisemmillaan tavallinen tekstikenttä on vain laatikko, johon voi kirjoittaa. Nettimoto-kenttäryhmässä on numero-, tosi/epätosi-, teksti-, tekstialue-, kuva- ja toista rivejä -kenttiä.



Kuvio 3. Sisällön syöttö ACF-kenttiin.

Sivustolla kenttien sisällön näyttäminen riippuu sen tyypistä, mutta yleensä riittää the_field()-funktio, jolle annetaan kentän nimi heittomerkkien sisällä, ja se tulostaa kentän sisällön. Mikäli käytössä on kuitenkin toista rivejä -kenttä, silloin täytyy käyttää the_sub_field()-funktiota, koska varsinainen sisältö on kentän sisällä olevissa kentissä. Lisäksi tarvitaan PHP:n while-silmukkaa käymään läpi mahdollisesti useat rivit dataa:

```
// check if the repeater field has rows of data
if( have_rows('repeater_field_name') ):

    // loop through the rows of data
    while ( have_rows('repeater_field_name') ) : the_row();

        // display a sub field value
        the_sub_field('sub_field_name');

    endwhile;

else :
    // no rows found
endif;
```

2.3.4 Tiivistelmä

Advanced Custom Fieldsin monipuolisuus aukeaa helpommin sitä vähän jo käyttäneelle, mutta kokonaisuudessaan se tarjoaa kätevän tavan luoda verkkosivuja, joita voidaan muokata myös ulkoasun osalta WP:n ohjauspaneelista. Tavallisesti jos halutaan muuttaa muutakin kuin sisältöä, niin se täytyisi tehdä koodiin. ACF:llä voidaan sivun rakennusvaiheessa luoda tulevaisuudessa mahdollisesti muutettaville asioille kentät, jotka toimivat ikään kuin sivuston asetuksina joihin sivuston ylläpitäjällä on pääsy. Näin sivuston omistaja voi jälkikäteen vaihtaa esimerkiksi eri elementtien järjestystä, väriä tai vaikkapa poistaa jonkun rakenteen käytöstä normaalin tekstisisällön muokkaamisen lisäksi.

2.4 FacetWP-lisäosa

2.4.1 Esittely

Ajoneuvojen hakutoimintoja varten projektiin otettiin käyttöön myös FacetWP-lisäosa. Se mahdollistaa monipuolisen datan lajittelun ja hakemisen ilman sivuston uudelleen lataamista ja näyttää vain relevantteja suodatinvaihtoehtoja saatavilla olevan datan perusteella. Tämä nopeuttaa asioiden hakemista, koska käyttäjä näkee reaaliajassa hakutulokset valituilla suodattimilla.

Lisäosa on suhteellisen vaivaton ottaa käyttöön, koska se osaa käyttää laajasti olemassa olevaa dataa. FacetWP voi käyttää esimerkiksi eri sisältötyyppejä, lisäkenttiä ja taksonomioita (Use your existing data n.d.). Se toimii myös WordPressin arkistosivuilla suoraan. Käytettävissä on useita erilaisia HTML-elementtejä suodattamiseen. Näihin lukeutuvat esimerkiksi valintaruutu, liukusäätimiä, täydentävä hakukenttä ja pudotusvalikko.

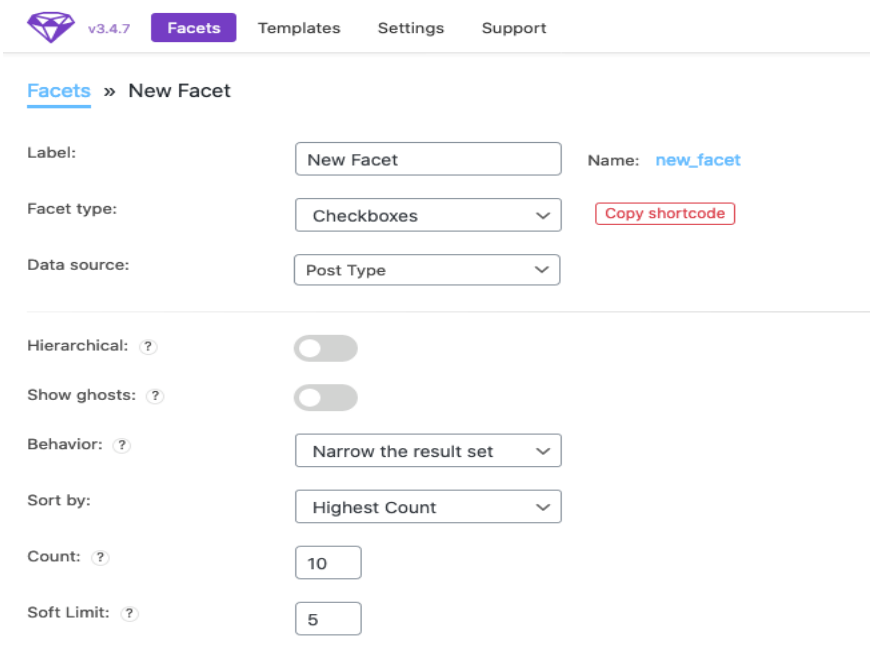
2.4.2 Käyttö

Yksinkertaisuudessaan ensiksi luodaan oikeaa tyyppiä olevat fasetit käyttötarkoituksen mukaan ja nämä lisätään sivustolle haluttuun paikkaan. Tämän voi tehdä joko ly-

hytkoodilla tai facetwp_display(”facet”, ”fasetin_nimi”) PHP-funktion avulla. Funktiota käytetään echon kanssa, jolloin se tulostaa valmiin HTML-elementin tähän kohtaan:

```
<div>
    <h2>Suodattimet</h2>
    <div>
        <?php echo facetwp_display( 'facet',
        'moto_ajoneuvolaji' ); ?>
    </div>
    <div>
        <?php echo facetwp_display( 'facet', 'moto_tyyppi'
        ); ?>
    </div>
    <div>
        <?php echo facetwp_display( 'facet', 'moto_merkki'
        ); ?>
    </div>
</div>
```

Fasettien luonti onnistuu WordPressin ohjauspaneelista asetuksien alta löytyvästä valikosta. Sille voidaan antaa nimi, tyyppi, datan lähde (esim. lisäkenttä) sekä runsaasti muita tyyppiin liittyviä asetuksia. (Ks. kuvio 4.) Tämän jälkeen täytyy vielä tallentaa asetukset oikean ylänurkan painikkeella, jotta muutokset tulevat voimaan. Lisäksi kannattaa ajaa FacetWP:n tietokannan indeksointi, sillä tämä nopeuttaa hakuprosessia.



Kuvio 4. Uuden Fasetin luonti.

Tässä työssä tarvittiin hakutoimintoa pelkästään moto-sisältötyypin arkistosivulle, joten muita mahdollisia vaihtoehtoja ei käydä erikseen läpi. Arkistoon toteutetaan itse fasettien näyttäminen normaalisti PHP-funktion avulla. Hakutuloksien näyttäminen vaatii WordPressin WP_Queryn käärimisen esimerkiksi div-elementtiin, jolle on annettu facetwp-template-luokka:

```
<div class="facetwp-template">
    <?php
    $motot_args = [
         'post_type' => 'moto',
         'post_status' => 'publish',
         'posts_per_page'=> 10,
         'order' => 'ASC',
         'facetwp' => true,
    ];
    $query = new WP_Query($motot_args); ?>

    <?php if ( $query->have_posts() ) : ?>

         <div class="w-full px-8">

            <?php while ( $query->have_posts() ) : $query-
            >the_post(); ?>

              <?php get_template_part( 'template-
              parts/content', 'motolist' ); ?>

              <?php endwhile; ?>

         </div>

         <div class="pagination mt-10 text-center">
              <?php echo facetwp_display( 'pager' ); ?>
         </div>

    <?php endif; ?>

    <?php wp_reset_postdata(); ?>
</div>
```

Lisäksi WP_Query:lle tarvitaan argumentteihin ”facetwp” => ”true”.

Mikäli tuloksia on paljon ja niistä kertyy useampia sivuja, niin FacetWP tarjoaa sivunumerointitoiminnon. Sen saa käyttöön facetwp_display(”pager”)-funktiolla. Funktio täytyy sijoittaa WP_Queryn endwhilen ja endifin väliin.

2.5 Nettimoton REST-rajapinnan esittely

2.5.1 Tietoa

Nettix REST API -rajapinnan avulla yritykset voivat siirtää liikkeen ajoneuvojen tiedot omille verkkosivuilleen. Lisäksi rajapinta mahdollistaa ajoneuvoilmoitusten luonnin, muokkaamisen, julkaisun ja poistamisen. (Helpoin tapa hallita ilmoitusdataa n.d.)

2.5.2 REST API

Representational state transfer eli REST, on järjestelmäarkkitehtuuri web-palveluiden luontiin. Sitä käyttävät palvelut mahdollistavat pääsyn tekstimuotoisiin verkkoresurs-seihin käyttämällä HTTP-metodeita. Se sisältää 6-vaiheisen ohjeistuksen, jonka rajoit-teita jokaisen käyttöliittymän tulee noudattaa ollakseen ”RESTful”. (What is REST n.d.)

1. Asiakaspalvelin (Client-server) – Eriyttämällä käyttöliittymä tietokannasta voi-daan parantaa sen siirrettävyyttä eri alustojen välillä. Tämä mahdollistaa pal-velimen paremman skaalautumisen yksinkertaistamalla sen komponentteja.
2. Tilaton (Stateless) – Kaikkien asiakkaan palvelimelle lähettämien pyyntöjen tulee sisältää tarvittava tieto, jota vaaditaan kyseisen pyynnön ymmärtämi-seen. Näin session tila pidetään vain asiakkaalla.
3. Välimuistitettava (Cacheable) – Välimuistirajoitteet edellyttävät vastauksen sisältämän datan merkitsemistä joko suoraan, tai epäsuorasti joko välimuistiin soveltuvaksi tai ei. Mikäli vastaus voidaan tallentaa välimuistiin, niin asiak-kaan välimuisti saa oikeuden käyttää vastausta myöhemminkin samanlaisiin pyyntöihin.
4. Yhtenäinen käyttöliittymä (Uniform interface) – REST määritellään neljällä käyttöliittymän rajoitteella. Ne ovat resurssien tunnistaminen, resurssien hyö-dyntäminen esitysten kautta, havainnollistavat viestit ja hypermedia sovelluk-sen tilan taustalla.

5. Kerrostettu järjestelmä (Layered system) – Monikerroksinen arkkitehtuuri mahdollistaa sen muodostamisen hierarkkisesti kerroksista rajoittamalla arkkitehtuurin komponenttien käyttäytymistä vain omalle kerrokselleen. Näin komponentit näkevät vain sen kerroksen, millä ne toimivat.
6. Koodia lennossa (vaihtoehtoinen) (Code on demand) – REST mahdollistaa asiakaspään toiminnallisuuden laajentamisen lataamalla ja suorittamalla koodia sovelluksissa tai funktioissa.

(Mt.)

2.5.3 OAuth 2 ja tunnistautuminen

OAuth 2 on suosittu protokolla tunnistautumiseen. Se mahdollistaa kolmannen osapuolen sovelluksen pääsyn HTTP-palveluun joko sen omistajan puolesta järjestämällä hyväksytyn kommunikoinnin resurssin omistajan ja HTTP-palvelun välillä, tai sallimalla kolmannen osapuolen sovelluksen pääsyn käsiksi palveluun itsekseen. (Abstract 9.2012.)

Nettix tarjoaa tunnistautumista varten oman Authentication API:n, joka käyttää OAuth 2 -protokollaa. Tunnistautumisrajapinta mahdollistaa tunnistautumisen käyttäjänimen ja salasanan avulla, uniikin sähköpostin ja salasanan avulla, client ID:n ja client secretin avulla, sekä Facebookin tai Googlen avulla. (NettiX authentication API n.d.)

Sovellukset käyttävät tunnistautumiseen client_credentials-menetelmää, jossa tunnistautuminen tapahtuu client ID:n ja client secretin avulla. Tunnistautuminen tapahtuu lähettämälle palvelimelle client_credentials-tunnistautumistyyppi, client ID ja client secret:

```
curl -d grant_type=client_credentials -d
client_id=test_client -d client_secret=$SECRET \
  https://auth.nettix.fi/oauth2/token
```

(Aaron Parecki n.d.; NettiX authentication API n.d.)

Rajapinta palauttaa RS256-enkoodatun JSON Web Token -access tokenin (NettiX authentication API. N.d). Access token koostuu kolmesta Base64Url-enkoodatusta osasta, jotka on erotettu toisistaan pisteillä:

```
eyJ0eXAiOiJKV1QiLCJhbGci.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZ
SI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWUsImp0aSI6ImI2MTUwZTE0LTd
iZjgtNDJkNC.ytlQXehCbXz5V6csMbePZulT
```

(What is the JSON Web Token structure? n.d.)

Ensimmäinen osa on nimeltään header, ja se sisältää tokenin tyypin ja sen algoritmin JSON-muodossa:

```
{
   "typ": "JWT",
   "alg": "RS256"
}
```

(Mt.)

Header enkoodataan Base64Url:lla ja se muodostaa tokenin ensimmäisen osan. Payload on toinen osa ja se sisältää käyttäjän datan. Tähän lukeutuu esimerkiksi ID, client ID, myöntämis- ja vanhenemisajat, sekä käyttöoikeudet:

```
{
   "id": "wxbz820r6109ca39a521b4ck9c099",
   "jti": " wxbz820r6109ca39a521b4ck9c099",
   "iss": "",
   "aud": "client_id",
   "sub": null,
   "exp": 1251019428,
   "iat": 1251019428,
   "token_type": "bearer",
   "scope": "read write"
}
```

(Aaron Parecki n.d.; NettiX authentication API n.d.)

Kolmas ja viimeinen osa JWT:stä on sen allekirjoitus (signature) ja se sisältää Base64Url enkoodatun headerin ja payloadin pisteellä erotettuna sekä julkisen avaimen ja client secretin. Näitä käytetään varmistamaan, että access token on aito ja sen lähettäjä on se, kuka väittääkin olevansa. (Aaron Parecki N.d.)

Access tokenia käytetään sovelluksissa välittämällä se palvelimelle ” X-Access-Token”-headerina. Se on voimassa 24 tuntia, jonka jälkeen täytyy hakea uusi token. (NettiX authentication API n.d.)

2.5.4 Hakeminen rajapinnasta

Nettimoton rajapinta mahdollistaa ajoneuvojen myynti-ilmoitusten hakemisen lisäksi muun muassa niiden tietojen päivittämisen sekä uusien ilmoitusten luonnin tai niiden poistamisen. Myynti-ilmoituksia voi lisäksi hakea kattavasti eri ominaisuuksien perusteella. Omien myyjien tiedot ovat myös haettavissa erikseen ja tämän avulla voidaan hakea esimerkiksi vain tietyn liikkeen ajoneuvot. (Nettimoto REST API n.d.)

Haku tapahtuu valitsemalla haettava asia ja sen polku rajapinnassa. Polku laitetaan osoitteen loppuun /bike/:n jälkeen. Esimerkiksi pyörien tyyppi voitaisiin hakea lähettämällä kutsu GET-metodilla osoitteeseen

```
https://palvelin.fi/rest/bike/options/bikeType
```

Onnistuneessa kutsussa vastauksena tulee JSON-objekti, joka sisältää kutsua vastaavat tiedot:

```
[
    {
        "id": 2,
        "fi": "Moottoripyörä",
        "en": "Motorcycle"
    },
    {
        "id": 5,
        "fi": "Moped",
        "en": "Mopo"
    }
]
```

Mahdollisessa virhetilanteessa vastaus sisältää virhekoodin:

```
{
    "status": 404,
    "error": "Resource not found: /rest/bike/nothing"
}
```

(Errors n.d.)

3 Työn toteutus

3.1 Projektin aloitus ja suunnittelu

Nettimoton rajapinnan integroinnille WordPressiin ilmeni tarvetta, kun erään verkkosivuston uudistamisen yhteydessä toivottiin myytävien ajoneuvojen hakemista sivustolle. Vastaavia tarpeita oli ollut aiemminkin, ja epäilemättä niitä voisi tulla tulevaisuudessa lisää, joten tällaisesta integraatiosta voisi olla pidemmällä aikajänteellä selvää hyötyä.

Aluksi ei ollut vielä selvää, mitä rajapintaa tultaisiin käyttämään, joten selvitystyö aloitettiin Nettiautosta. Suunnitteluvaiheessa tutkittiin Nettiauton lisäksi eri autokauppiaiden verkkosivustoja ja kartoitettiin oleellisia ominaisuuksia, joita rajapinnasta tallennettaisiin. Nettiauton ajoneuvoilmoituksia varten luotiin uusi sisältötyyppi niiden tallentamista varten. Ilmoituksille tehtiin sopivat lisäkentät Advanced Custom Fieldsillä ja ne asetettiin näkymään vain autoille, käyttämällä ACF:n kenttäryhmän asetuksia.

Testaamista varten ajoneuvoja luotiin paikallisesti suoraan WordPressin ohjauspaneelista. Näin jo tässä vaiheessa oli mahdollista rakentaa teemaan autojen haku-, lajittelu- ja näyttötoimintoja, vaikka rajapinnasta ei ollut mitään tietämystä.

Työnkuvan selvittyä tarkemmin kävi ilmi, että Nettiauton sijaan tarvittaisiinkin Nettimoton rajapintaa. Näinpä alettiin kartoittaa sen toimintaa ja myös sitä, miten olemassa olevaa Nettiautoa varten tehtyä toteutusta voitaisiin hyödyntää motojen kanssa.

Pääpiirteissään valmiista toteutuksesta voitiin hyödyntää kaikki toiminnallisuus pienehköin muutoksin. Erilaisten ajoneuvotyyppien luonnolliset erot toivat muutoksia tallennettaviin tietoihin ja näiden nimiin. WordPressiin autoja varten luotu sisältötyyppi vaihdettiin motojen mukaan eri nimiseksi ja Advanced Custom Fields -lisäkentät muutettiin vastaamaan moottoripyörien, mopojen, mönkijöiden ja muiden motojen ominaisuuksia.

Muutosten seurauksena myös teemaan tehdyt haku- ja lajittelutoiminnot vaativat muutoksia nimeämisiin, jotta ne vastaisivat uusia arvoja ja osaisivat käsitellä erilaisia ominaisuuksia.

3.2 Tekninen toteutus

3.2.1 Sisältötyypin ja ACF-kenttien luominen

Ajoneuvot haluttiin tallentaa WordPressiin, jotta niitä ei tarvitsisi hakea rajapinnasta jatkuvasti. Lisäksi ajoneuvojen hakuun ja lajitteluun voitaisiin käyttää WP:n omaa toiminnallisuutta eikä Nettimoton tarjoamia toimintoja. Tämä yksinkertaistaa toteutusta ja mahdollistaa tulevaisuudessa sen muokkaamisen sivuston tarpeisiin sopivaksi ilman laajaa tietämystä rajapinnan toiminnasta. Lisäksi tallentamalla ajoneuvot sivuston tietokantaan ne säilyvät näkyvissä, vaikka rajapintayhteydessä olisikin jotain ongelmia.

Tallennuksen mahdollistamiseksi ajoneuvoja varten rekisteröitiin uusi moto-sisältötyyppi:

```
register_cpt_moto() {
    $labels = array(
         'name' => 'Motot',
         'singular_name' => 'Moto',
         'add_new' => 'Lisää uusi',
         'add_new_item' => 'Lisää uusi moto',
         'edit_item' => 'Muokkaa motoa',
         'new_item' => 'Uusi moto',
         'all_items' => 'Kaikki motot',
         'view_item' => 'Katsele moto',
         'search_items' => 'Etsi motoja',
         'not_found' => 'Motoja ei löytynyt',
         'not_found_in_trash' => 'Motoja ei löytynyt
         roskakorista',
         'parent_item_colon' => '',
         'menu_name' => 'Motot'
    );
    $args = array(
         'labels' => $labels,
         'public' => true,
         'publicly_queryable' => true,
         'show_ui' => true,
         'show_in_menu' => true,
         'query_var' => true,
         'rewrite' => array( 'slug' => 'moto' ),
         'capability_type' => 'page',
         'has_archive' => true,
         'hierarchical'=> false,
         'menu_position' => null,
         'menu_icon' => 'dashicons-pressthis',
         'supports' => array( 'title', 'editor', 'author',
         'custom-fields')
    );
    register_post_type( 'moto', $args );
}
```

$labels-muuttuja on taulukko, jossa on sisältötyyppiin liittyvät ohjauspaneelin tekstit. Lisäksi tarvitaan toinen taulukko ($args), joka sisältää sisältötyypin ominaisuudet. Näin ajoneuvojen ominaisuuksia varten voidaan tehdä lisäkentät ja rajoittaa niiden näkyvyys vain motoihin.

Advanced Custom Fieldsin lisäkenttiä varten selvitettiin, mitä ominaisuuksia ajoneuvoista haluttiin tallentaa. Näihin valikoitui ID, ajamaton vai ei, alue, kaupunki, ajoneuvolaji, tyyppi, merkki, malli, mallitarkenne, moottorin tilavuus, moottorin tyyppi, vetotapa, vaihteisto, hinta, vuosimalli, kilometrit, tunnit, rekisterinumero, kuvaus, tarvikkeet ja kuvat. (Ks. Liite 1.)

Lisäkentät luotiin PHP-koodin avulla tallentamalla se erilliseen nettix-acf.php-tiedostoonsa. Tämän etuna on se, että tarvittavat lisäkentät luodaan automaattisesti. Jos

toteutusta siirretään eri sivustoille, niin kenttiä ei tarvitse luoda joka kerta manuaalisesti uudelleen ohjauspaneelista, vaan ne luodaan automaattisesti WP:n latauduttua.

PHP-koodin luontia varten lisäkentät kannattaa kuitenkin luoda ohjauspaneelissa graafisen käyttöliittymän avulla ja sen jälkeen käyttää ACF:n ”Vie kenttäryhmiä” -työkalua, joka generoi automaattisesti tarvittavan koodin.

3.2.2 Ajoneuvojen hakeminen rajapinnasta

Rajapintaa varten luotiin PHP:llä Nettix-luokka, jolle annettiin ominaisuuksiksi access token ($access_token), client ID ($username), client secret ($secret), avaintiedosto ($keyfile), tunnistautumispalvelin ($authserver), palvelin ($server) ja rajapinnan tyyppi ($mode). Luokalle tehtiin kolme funktiota: get_token(), validate_token() ja ads().

get_token():in avulla tarkistetaan, onko tallennettua access tokenia olemassa tai onko se yli 24 tuntia vanha. Mikäli sitä ei löydy tai se on vanhentunut, haetaan uusi ja tallennetaan se tiedostoon. Access tokenin hakeminen:

```
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, rtrim($this->authserver,
'/') . '/oauth2/token');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS,
http_build_query($params));
$results = curl_exec($ch);
```

Saatu access token on JSON Web Token -muodossa. get_token() palauttaa myös access tokenin seuraavalle validate_token()-funktiolle.

Access token on allekirjoitettu yksityisellä RSA-avaimella, jonka julkisen avaintiedoston validate_token() hakee. Tällä varmistetaan, että access tokeniin voidaan luottaa ja näin sallia ohjelmalle pääsy rajapintaan.

ads()-funktio kutsuu get_toke():nia ja saa tätä kautta access tokenin käyttöönsä. Se välitetään ”X-Access-Token”-headerina rajapinnalle:

```
$headers = [
    'Content-Type: application/x-www-form-urlencoded;
    charset=utf-8',
    'X-Access-Token: ' . $token,
];
```

Rajapinnasta haetaan palvelinta ja rajapinnan tyyppiä vastaavat myyjät:

```
$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, rtrim($this->server, '/') .
'/rest/' . $this->mode . '/my-dealers');
curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
curl_setopt($ch, CURLOPT_HTTPHEADER, $headers);
$dealers = curl_exec($ch);
curl_close($ch);
```

Myyjät ovat JSON-muodossa ja ne dekoodataan json_decode():n avulla. Tämä taulukko käydään läpi foreachilla ja siitä etsitään asetettua $username:a vastaava myyjä. Tältä myyjältä haetaan rajapinnasta ”adsUrl”, joka on verkko-osoite myytäviin ajoneuvoihin:

```
if (!empty($dealers)) {
    $dealers = json_decode($dealers);
    foreach ($dealers as $dealer) {
         if (!empty($dealer->username == $this->username))
         {
              $ch = curl_init();
              curl_setopt($ch, CURLOPT_URL,   $dealer-
              >adsUrl);
              curl_setopt($ch,   CURLOPT_RETURNTRANSFER,
              1);
              curl_setopt($ch, CURLOPT_HTTPHEADER,
              $headers);
              $ads = curl_exec($ch);
              curl_close($ch);
              if (!empty($ads)) {
                   return $ads;
              }
         }
    }
}
```

Ajoneuvoilmoitukset tulevat JSON-muodossa, josta ne voidaan sitten tallentaa WordPressiin.

3.2.3 Ajoneuvodatan tallentaminen WordPressiin

Rajapinnasta haetut ajoneuvot ovat aluksi JSON-muotoisena merkkijonona, mutta PHP:n json_decode()-funktio muuttaa sen taulukoksi, joka sisältää kaikki myyjän ajoneuvoilmoitukset omina objekteinaan:


```
[
    {
        "status":"forsale",
        "bikeType":{
            "id":17,
            "fi":"Skootteri",
            "en":"Scooter"
        },
        "bodyType":null,
        "make":{
            "id":29,
            "name":"Honda",
            "mostPopular":true
        },
        "model":{
            "id":2963,
            "name":"SCV"
        }
    },
    {
        "adType":"forsale",
        "bodyType":{
            "id":49,
            "fi":"Crossim\u00f6nkij\u00e4",
            "en":"ATV motocross"
        },
        "make":{
            "id":782,
            "name":"Crossmax"
        },
        "model":{
            "id":7178,
            "name":"110"
        },
        "deliveryCost":120
    }
]
```


Tästä käydään läpi yksittäiset ajoneuvot foreach-silmukan avulla, jonka seurauksena kukin ajoneuvo säilötään $ad-muuttujana. Niinpä ajoneuvojen ominaisuuksiin päästään käsiksi viittaamalla niihin näin:

```
$ad->make->name
```

Tämä on ajoneuvon mallinimi, eli yllä olevan esimerkkitulosteen mukaan ensimmäisen ajoneuvon kohdalla ”SCV”, tai toisen kohdalla ”Crossmax”.

WordPressiin tallentamisen yhteydessä on huomioitava jo mahdollisesti olemassa olevien ajoneuvojen poistaminen, että samat ajoneuvot eivät tallennu moneen kertaan. Tämä onnistuu sijoittamalla sopiva SQL-kysely WP:n tietokantaan juuri ennen ajoneuvojen tallentamista. Kaikki moto-sisältötyypin ajoneuvot voidaan poistaa seuraavasti:

```
global $wpdb;
    $wpdb->query("DELETE a,b,c
         FROM {$wpdb->prefix}posts a
         LEFT JOIN {$wpdb->prefix}term_relationships b ON
         (a.ID = b.object_id)
         LEFT JOIN {$wpdb->prefix}postmeta c ON (a.ID =
         c.post_id)
         WHERE a.post_type = 'moto';"
    );
```

Kun vanhat ajoneuvot on poistettu, voidaan niiden tilalle tallentaa uudet. Tallennukseen käytetään WordPressin wp_insert_post()-funktiota, jolle annetaan taulukkona haluttu sivun nimi, sisältötyyppi ja sisällön tila:

```
$name = $ad->make->name . ' - ' . $ad->model->name . ' ' .
$ad->modelType->name . ' - ' . $ad->id;
$post_id = wp_insert_post([
    'post_title' => $name,
    'post_type' => 'moto',
    'post_status' => 'publish',
]);
```

Jokainen ajoneuvo on WP:ssä oma ”sivunsa” ja niillä on siten myös omat verkko-osoitteensa. Osoitteen loppuosa muodostuu post_titlen perusteella, joten siihen haluttiin tallentaa selkeyden vuoksi ajoneuvon merkki, malli, mallitarkenne ja ID. Esimerkiksi KTM 125 Dukelle, jonka ID on 12345 muodostuisi osoitteen polkutunnisteeksi (slug) ktm-125-duke-12345.

Tämän jälkeen voidaan tallentaa tiedot ACF-kenttiin update_field()-funktion avulla. Halutut ajoneuvojen ominaisuudet laitetaan taulukkoon, jossa ensiksi on ACF-kentän nimi ja =>-merkki, sekä viittaus haluttuun $ad-muuttujan ominaisuuteen:

```
$meta = [
    'id' => $ad->id,
    'ajamaton' => $ad->undrivenVehicle,
    'alue' => $ad->region->fi,
    'kaupunki' => $ad->town->fi,
    'ajoneuvolaji' => $ad->bikeType->fi,
    'tyyppi' => $ad->bodyType->fi,
    'merkki' => $ad->make->name,
    'malli' => $ad->model->name,
    'hinta' => $ad->price,
];

// tallentaa arvot acf-kenttiin
foreach ($meta as $key => $value) {
    update_field($key, $value, $post_id);
}
```

PHP:n foreach-silmukalla voidaan tallentaa kenttien arvot käymällä läpi jokainen taulukon rivi avain-/arvopareina. update_field()-funktiolle annetaan ensiksi ACF-kentän nimi, kentälle tuleva arvo, sekä nykyisen ajoneuvosivun ID.

Haun automatisoimiseksi hyödynnettiin WordPressin Cronia, jolla voidaan ajastaa haluttuja toimintoja. Tätä varten luotiin uusi toiminto add_action()-funktiolla. Sille annetaan ensin koukku, johon tartutaan, ja kutsuttava funktio. Koukku on tässä tapauksessa ajastus, joka suoritetaan kerran vuorokaudessa:

```
if ( ! wp_next_scheduled( 'nettimoto_import' ) ) {
    wp_schedule_event( time(), 'daily', 'nettimoto_import'
    );
}
add_action( 'nettimoto_import', 'importMotos' );
```

3.2.4 Ajoneuvojen haku, lajittelu ja näyttäminen sivustolla

WordPressiin tallennetut ajoneuvot voidaan näyttää sivustolla samoin, kuin mikä tahansa muukin sisältötyyppi. Työtä varten luotiin arkisto, joka näyttää kaikki ajoneuvot listana. Näytettäviä ajoneuvoja voidaan myös suodattaa ja järjestää esimerkiksi hinnan mukaan. WordPressissä arkistosivut ovat mallia archive-sisältötyyppi.php ja ne sijoitetaan teeman juureen. Tässä tapauksessa teemaan luotiin siis archive-moto.php.

Arkistosivun perustoiminnallisuus on vain ajoneuvojen haku ja tätä varten vaaditaan käytännössä uusi WP_Query-luokka ja sille argumenttina moto-sisältötyypin nimi. If-

ehtolauseella voidaan tarkistaa, onko tietokannassa yhtään motoa kutsumalla WP_Queryn have_posts()-funktiota. Tulokset käydään läpi while-silmukalla asettamalla sen ehdoksi sama have_posts()-funktio. Silmukan sisään laitetaan vielä the_post()-funktio, joka tarkistaa, että silmukka on käynnistynyt ja pitää huolen, että while-silmukan jokaisella kierroksella siirrytään seuraavaan ajoneuvoon. Tällöin whilen sisällä voidaan tulostaa tarvittavat ajoneuvojen ominaisuudet ja haluttu ulkoasu HTML:n avulla ja näin kaikki ajoneuvoilmoitukset tulostuvat sivulle.

Arkistosivulla ei ole tarvetta tulostaa kaikkia ominaisuuksia, joten siihen rajattiin otsikko, kuva, hinta, osa kuvauksesta ja linkki ilmoitukseen. Koko hakuprosessi voisi olla siis yksinkertaistettuna esimerkiksi seuraavanlainen:

```
// WP_Queryn asetukset
$args = array(
    'post_type' => array( 'moto' ),
    'posts_per_page' => -1
);

// Kysely
$the_query = new WP_Query( $args );

// Silmukka
if ( $the_query->have_posts() ) {
    echo '<ul>';
    while ( $the_query->have_posts() ) {
         $the_query->the_post();
         echo '<li>' . get_the_title() . '</li>';
    }
    echo '</ul>';
} else {
    // motoja ei löytynyt
}
// Palauta alkuperäinen Post Data
wp_reset_postdata();
```

the_post():in käyttämisessä on huomioitavaa palauttaa alkuperäinen post data kutsumalla kyselyn jälkeen wp_reset_postdata()-funktiota. Tämä palauttaa template tagit takaisin pääkyselyn (main query) käyttöön. Template tageja käytetään hakemaan sisältöä tietokannasta.

Yllä olevan kyselyn lisäksi arkistoon tehtiin toinen ajoneuvoilmoitusten haku, joka suoritetaan, mikäli käyttäjä lajittelee tulokset tai muuttaa niiden määrää sivua koh-

den. Tätä varten sivulle luotiin HTML-lomake ja sen sisään pudotusvalikot vaihtoehtoja varten. Lomakkeen toiminnoksi asetettiin arkistosivun osoite ja sen metodiksi get, koska haluttiin, että parametrit näkyvät selaimen osoiterivillä.

Lomakkeen lähetyksen jälkeen ehtolause tarkastaa, että lomake on lähetetty:

```
if ( isset($_GET['sort']) ) {
```

Tämän jälkeen voidaan yksinkertaisesti tarkastaa asetetut lajitteluparametrit hakemalla ne URL:sta PHP:n $_GET:llä:

```
if ($_GET['sort-options'] == 'aakkosjarjestys') {
```

ja välittää ne muuttujien avulla WP_Queryn asetuksiin. Kyselyä voidaan muokata lukuisilla tavoilla ja esimerkiksi tulosten järjestäminen aakkosjärjestykseen onnistuu ”order” => ”ASC” -koodilla. Käänteinen aakkosjärjestys saadaan aikaan vaihtamalla ASC:n tilalle DESC.

WordPress käyttää sisältötyypin yksittäisten artikkeleiden näyttämiseen singlesivuja, joten motoja varten tehtiin single-moto.php-tiedosto teeman juureen. Sen sisälle voidaan laittaa tarvittava PHP- ja HTML-koodi mitä vaaditaan halutun lopputuloksen saamiseksi.

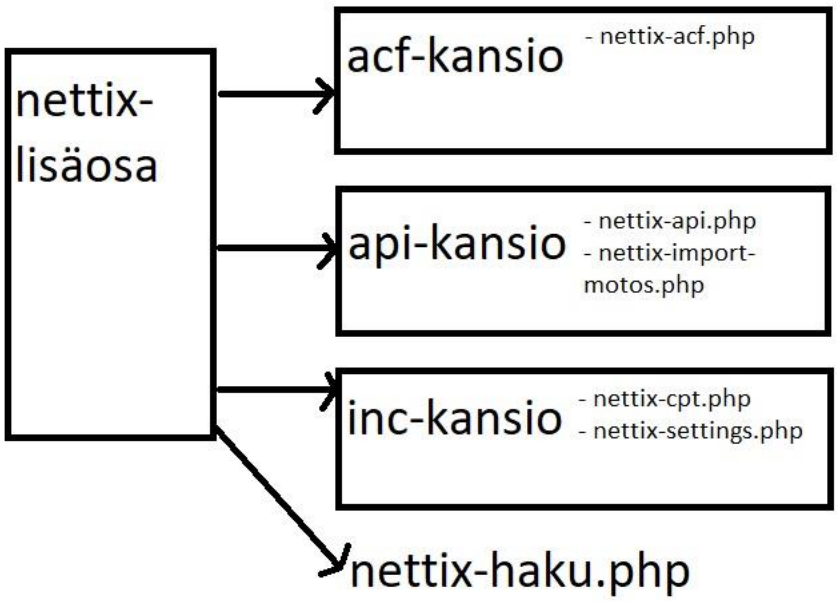
Singlesivulle lisättiin tarvittava HTML-rakenne ja WP_Queryn while -silmukka. Sen sisälle tehtiin sivulla näytettävän datan tulostus ja muotoilu. Data haetaan ACF-kentistä, joten se voidaan tulostaa the_field()-funktiolla. Sisäkkäisten kenttien tapauksessa vaaditaan the_sub_field():n käyttöä. Esimerkiksi kuvat on tallennettu näin, koska niiden määrää ei voi tietää ennakkoon.

3.2.5 WordPress-lisäosan tekeminen

Aluksi tavoitteena oli vain integroida rajapinta WP:n teeman kanssa, mutta aikaa jäi, joten toteutusta päätettiin muokata lisäosaksi. Moni asia pysyi samana, mutta siitä huolimatta lisäosa vaati joitakin oleellisia muutoksia toteutukseen.

Tiedostorakennetta täytyi muuttaa, koska teemojen toiminta poikkeaa lisäosista. Teemat toimivat ennalta määritellyn hierarkian mukaan, mutta lisäosan rakenne voidaan päättää itse. Lisäksi lisäosan asentamista ja poistamista varten täytyi tehdä omia toimintojaan. Lisäosan asetussivu vaati myös toiminnallisuutta, jota ei teemaan tehty.

Lisäosaa varten luotiin sen juurikansioon kolme kansiota ja nettix-haku.php-tiedosto. (Ks. kuvio 5.) Se sisältää pakollisen otsikkokommentin ja lisäosan aktivointiin ja poistoon liittyvät toiminnot. Acf-kansion tiedosto vastaa lisäkenttien luonnista ja se ajetaan lisäosan aktivoiduttua. Api-kansio sisältää kaksi tiedostoa, jotka vastaavat datan hakemisesta rajapinnasta ja sen tallentamisesta ACF-kenttiin. Inc-kansio sisältää moto-sisältötyypin luonnin ja lisäosan asetussivun toiminnot sisältävän nettix-settings.php-tiedoston.



Kuvio 5. Lisäosan tiedostorakenne

Osa toiminnallisuudesta päätettiin kuitenkin jättää teemaan. Ajoneuvojen näyttäminen on parempi hoitaa teemassa, koska se toteutetaan eri tavoilla eri sivustoilla. Lisäosan ideana on sisältää toiminnallisuutta, joka voidaan lisätä sellaisenaan eri sivus-

toille, joten ajoneuvojen näyttäminen on parempi jättää sen ulkopuolelle. Lisäosa sisältää siis sisältötyypin ja lisäkenttien luomisen, ajoneuvojen hakemisen rajapinnasta sekä niiden tallentamisen lisäkenttien avulla sisältötyyppiin.

Lisäosan juuressa oleva nettix-haku.php-tiedosto on lisäosan päätiedosto, koska se sisältää erityisen WordPressin vaatiman kommentin, joka tarvitaan jokaiseen lisäosaan. Kommentin lisäksi tiedostossa on metodit lisäosan aktivoimista, käyttöä ja poistamista varten.

Kun lisäosa aktivoidaan, niin suoritetaan tarkistus onko Advanced Custom Fields asennettu. Jos ei ole, niin siitä näytetään ilmoitus ja aktivointi ei onnistu. Jos ACF on asennettu, niin lisäosa aktivoidaan onnistuneesti.

Aktivoinnin jälkeen suoritetaan nettix-settings.php, nettix-cpt.php- ja nettix-acf.php-tiedostot, jotka luovat asetussivun, sisältötyypin ja lisäkentät motoja varten. Tämän jälkeen haetaan ajoneuvot rajapinnasta nettix-api.php-tiedostossa ja tallennetaan ne motoihin nettix-import-motos.php:n myötä. Import-motos-tiedostossa luodaan myös uusi ajastettu cron-tapahtuma ajoneuvohaulle ja sen mukaisesti haku suoritetaan jatkossa aina kerran vuorokaudessa.

Asetuksia varten luotiin oma sivunsa motojen alle. Tähän käytettiin admin_menu-koukkua, joka suoritetaan ennen kuin ohjauspaneelin valikko ladataan. Siihen kiinnitettiin add_submenu_page(), joka sisältää asetussivun luontiin vaadittavat asetukset:

```
add_submenu_page(
    'edit.php?post_type=moto',
    'Nettix-haku',
    'Asetukset',
    'manage_options',
    'nettix_haku',
    'nettix_options_page',
);
```

Tämän jälkeen suoritetaan asetusten rekisteröinti. Sitä varten tehtiin uusi toiminto, joka ajetaan ylläpitosivun tai skriptin latautuessa. Tätä varten WordPressissä on admin_init-koukku. Siihen kiinnitettiin register_setting()-funktio. Sille annettiin asetusryhmän nimi ja asetusten nimi:

```
register_setting('plugin_page', 'nettix_settings');
```

Rekisteröitävät asetukset luodaan samassa yhteydessä. Uuden alueen voi luoda asetuksiin add_settings_field():n avulla:

```
add_settings_field(
    'nettix_text_field_0',
    __( 'Username (client_id)', 'nettix' ),
    'nettix_text_field_0_render',
    'pluginPage',
    'nettix_pluginPage_section'
);
```

Sille annetaan kentän id, otsikkoteksti, funktio, joka tulostaa alueen sisällön, sivun tunniste ja osion nimi. Alueen sisällön tulostus onnistuu luomalla add_settings_field():ssä käytetyn kolmannen muuttujan niminen funktio, joka tulostaa halutun sisällön:

```
function nettix_text_field_0_render() {

    $options = get_option( 'nettix_settings' );
    ?>
    <input type='text'
    name='nettix_settings[nettix_text_field_0]'
    value='<?php echo $options['nettix_text_field_0']; ?>'>
    <?php

}
```

Asetussivun osiot saadaan näkyviin setting_field()- ja do_settings_sections()-funktioilla. Niille annetaan parametreiksi sivun tunniste, eli tässä tapauksessa pluginPage. Lomaketta varten käytettiin vielä submit_button():ia, joka tulostaa valmiin lomakkeen lähetyspainikkeen. Tämä toteutus käyttää WordPressin Settings API:a ja tallentaa asetukset tietokantaan. Tallennettujen arvojen käyttö onnistuu $options-muuttujan kautta seuraavasti: $options['nettix_text_field_0'].

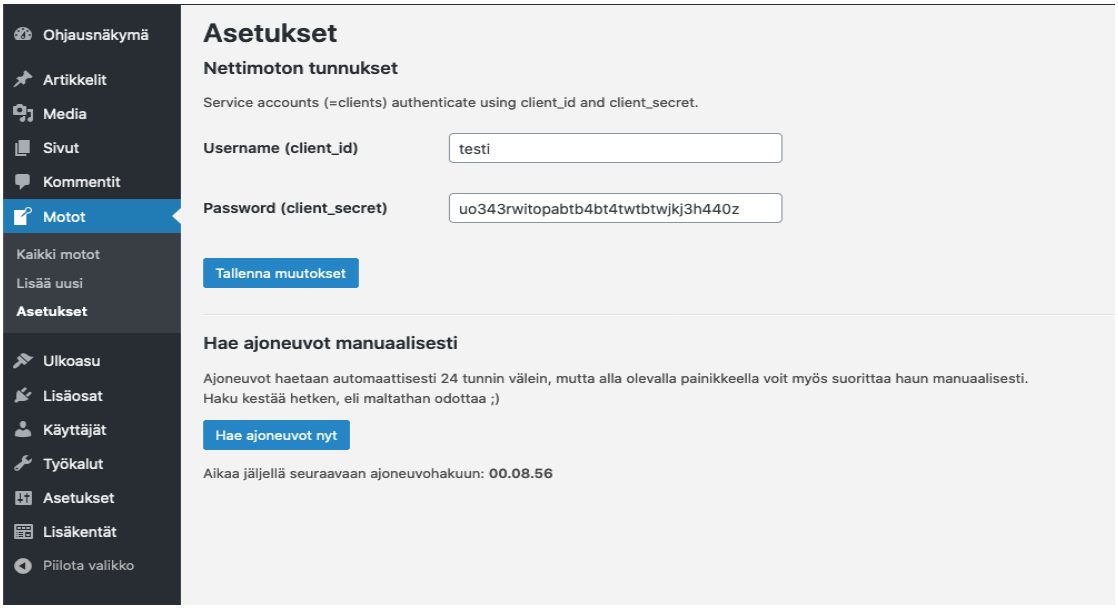
Lisäosan poiston yhteydessä suoritetaan cron-tapahtuman poisto ja asetusvalikon tekstikenttien tyhjennys. Nämä funktiot on kiinnitetty register_deactivation_hook()-funktioon, joka ajetaan automaattisesti lisäosan aktivointia poistettaessa. Sille annetaan tiedostopolku, jossa kutsuttava funktio sijaitsee sekä funktio, jota kutsutaan:

```
register_deactivation_hook(__FILE__, 'remove_cron_event');
register_deactivation_hook(__FILE__, 'remove_settings');
```

4 Tulokset

4.1 Lisäosa

Oleellisimpana saavutettuna tuloksena saatiin WordPressin lisäosa, joka voidaan asentaa helposti kaikille tarvittaville sivustoille. Lisäosa luo asennuksen yhteydessä uuden ”moto”-nimisen sisältötyypin, ajoneuvojen ominaisuuksia vastaavat lisäkentät sekä hakee ja tallentaa Nettimoton rajapinnasta käyttäjän myynnissä olevat ajoneuvot näihin kenttiin. Lisäosalla on myös oma asetusvalikkonsa, josta voidaan asettaa käyttäjätunnukset rajapintaa varten, sekä hakea ajoneuvot manuaalisesti. (Ks. kuvio 6.) Automaattinen haku tapahtuu kerran vuorokaudessa.

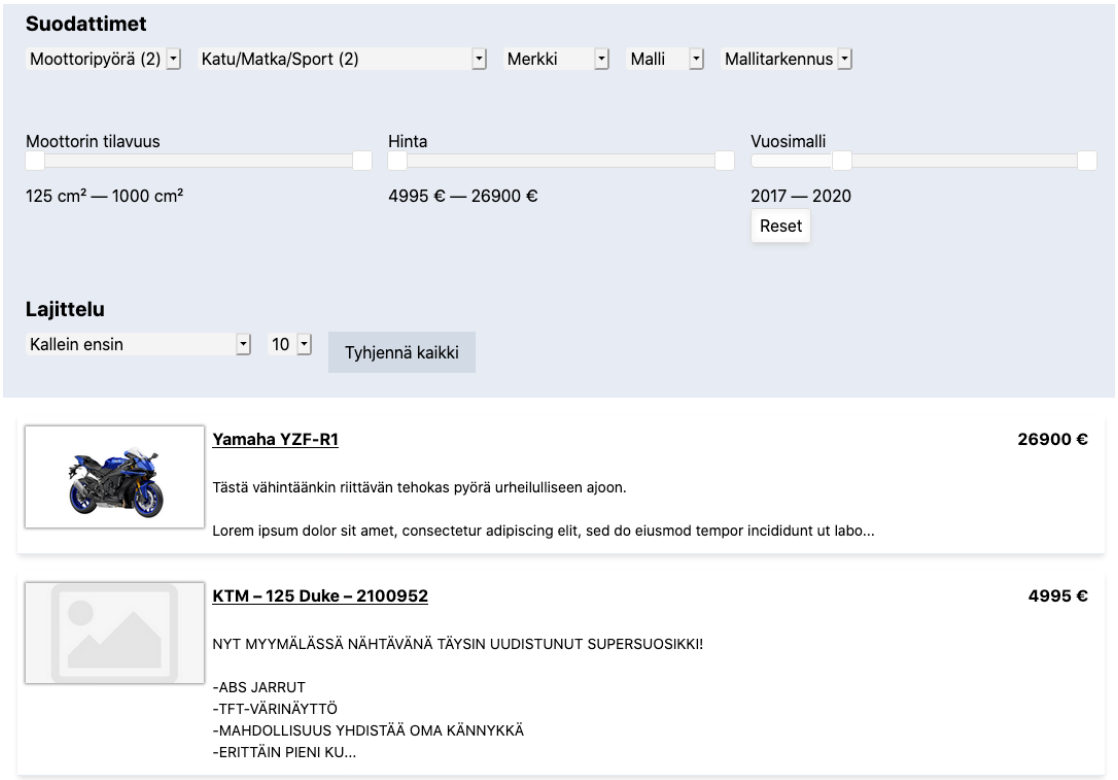


Kuvio 6. Lisäosan asetusvalikko

4.2 Sivupohjat

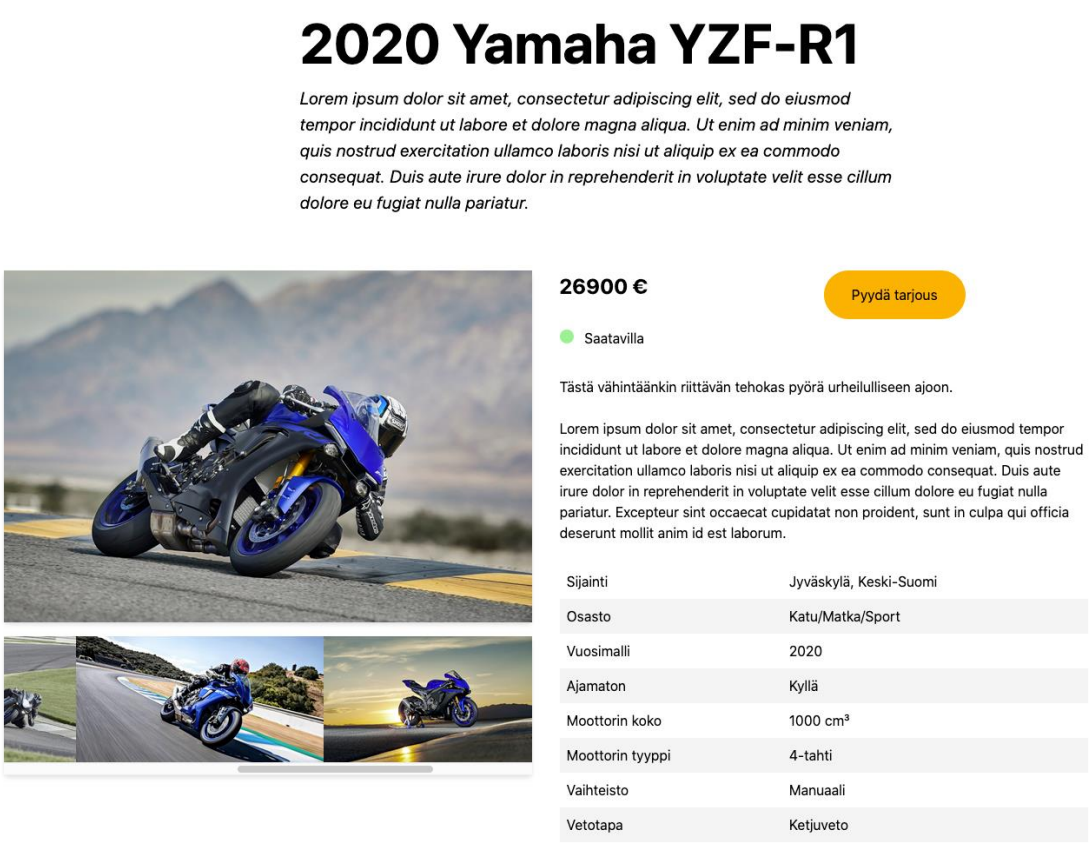
Lisäosan lisäksi kehitettiin Trimedian pohjateemaan sivupohjia ajoneuvojen näyttämistä varten. Sivupohjat sisältävät moto-sisältötyypin arkistosivun, johon listataan kaikki ajoneuvot. Pohjassa on myös mahdollisuus järjestää ajoneuvot nousevaan tai laskevaan aakkos- ja hintajärjestykseen sekä valita, montako ajoneuvoa näkyy yhdellä sivulla.

Arkistosivun lisäksi toteutettiin erillinen hakusivu ja yksittäisen ajoneuvon sivu. Hakusivulla on FacetWP-lisäosan avulla toteutettu haku, missä voidaan suodattaa haettavia ajoneuvoja ajoneuvolajin, tyypin, merkin, mallin, mallitarkenteen, moottorin tilavuuden, hinnan ja vuosimallin avulla. (Ks. kuvio 7.) Hakutulokset voidaan vielä lajitella samoin kuin arkistosivullakin. Haku käyttää AJAXia, eli sivua ei tarvitse ladata uudelleen suodattimia muutettaessa, vaan ajoneuvot haetaan dynaamisesti suodattimien kulloisenkin valinnan mukaan.



Kuvio 7. FacetWP-lisäosalla toteutettu ajoneuvohaku.

Yksittäisten ajoneuvojen näyttämistä varten tarvittiin myös oma sivupohjansa. (Ks. kuvio 8.) Sinne haetaan kaikki ajoneuvon tiedot ja esitetään ne selkeästi. Sivun ulkoasu täytyy muokata kuhunkin projektiin erikseen, mutta se on melko yksinkertaista tehdä HTML:n ja CSS:n avulla, kun sivulla näytettävä sisältö on haettu valmiiksi sivupohjaan.



Kuvio 8. Yksittäisen ajoneuvon sivu

4.3 Toteutuksen edut

Uusi toteutus helpottaa ennen kaikkea sivuston tekijöitä, mutta siitä on hyötyä muillekin osapuolille. Verkkosivujen tilaaja voi vaikuttaa enemmän lopullisen sivuston ulkoasuun, koska data voidaan muotoilla vapaasti ilman iframen asettamia rajoituksia. Lisäksi muuta toiminnallisuutta kuten haku- ja lajittelutoimintoja voidaan lisätä tarpeen mukaan. Ajoneuvojen tietoja ei tarvitse myöskään syöttää moneen kertaan eri palveluihin, sillä ne haetaan yhdestä ja samasta paikasta. Sivuston käyttäjälle hyötyä on puolestaan paremmin toimivasta ja yhtenäisestä selainkokemuksesta. Toteutus

estää myös myytyjen ajoneuvojen jäämisen roikkumaan verkkosivuille, koska ne poistuvat automaattisesti seuraavan haun yhteydessä, kun ne on poistettu myynnistä Nettimotossa.

Lisäosa nopeuttaa sivuston kehitysvaihetta, sillä osa toiminnallisuudesta on jo valmiina. Ohjelmointia tarvitaan siis vähemmän ja taittoprosessi yksinkertaistuu, kun sivulle ei tarvitse liittää kiinteitä iframe-kokonaisuuksia.

4.4 Tulevaisuuden suunnitelmat ja kehityskohteet

4.4.1 Yhteenveto

Vaikkakin rajapinta saatiin onnistuneesti integroitua WordPressiin ja paketoitua onnistuneesti lisäosaksi, niin tulevaisuudessa siintää silti useita mahdollisia kehityskohteita, joita nyt jätettiin pois aikataulusyistä. Osa kehityskohteista on vain nykytoteutuksen parannuksia, mutta osa on uusia ominaisuuksia, joista voisi olla hyötyä tulevaisuudessa erilaisten projektien kanssa.

4.4.2 Uudet ominaisuudet

Merkittävimpänä uutena ominaisuutena olisi toimiminen myös toiseen suuntaan, mikä mahdollistaisi uusien ajoneuvojen luonnin suoraan WP:n ohjauspaneelista. Tämän etuna olisi täysi sivuston hallinnan keskittäminen yhteen paikkaan, sillä ajoneuvojen luontia tai muokkausta varten ei tarvitsisi mennä erilliseen palveluun, vaan kaiken voisi hoitaa omilta verkkosivuiltaan.

Toinen parannuskohde olisi lisätä tuki myös muille Nettix-rajapinnoille, eli esimerkiksi Nettiautolle. Näin lisäosaa voisi hyödyntää myös autokauppiaiden verkkosivuilla. Käytännössä uuden rajapinnan lisääminen toteutukseen vaatisi uuden sisältötyypin autoille ja sille sopivat, ominaisuuksia vastaavat lisäkentät. Lisäosan asetusvalikkoon täytyisi vielä tehdä omat asetuksensa toista rajapintaa varten. Tämän lisäksi ohjelman suoritusjärjestystä ja logiikkaa täytyisi muuttaa siten, että ei luotaisi automaatti-

sesti molempia ajoneuvotyyppejä, vaan vasta sitten, kun käyttäjä on valinnut ne asetuksista. Tällä ehkäistäisiin turhan datan luominen WordPressiin ja tarpeeton rajapinnan kutsuminen väärästä palvelusta.

Vaihtoehtoisesti olisi mahdollista luoda erillinen lisäosansa Nettiautoa varten. Tämän etuna olisi pienempi työmäärä ja yksinkertaisempi toteutus. Olemassa olevaan lisäosaan voisi vain muokata sisältötyypin nimen ja lisäkentät vastaamaan autojen ominaisuuksia. Näin ollen varsinaista toimintaa ei tarvitsisi muuttaa, vaan nykyinen toteutus toimisi hyvin.

4.4.3 Parannukset

Uusien ominaisuuksien lisäksi käyttökokemusta voisi kehittää paremmaksi. Käyttäjää voisi informoida mahdollisista virheistä nykyistä enemmän ja näin antaa vinkkiä, mistä mahdollista vikaa voisi etsiä. Esimerkiksi jos syötetyt tunnukset ovat väärin, niin ohjauspaneeliin voisi tulla asiasta ilmoitus.

Asetusvalikkoon voisi lisätä mahdollisuuden muokata ajoneuvojen hakuaikaa halutuun kellonaikaan ja intervallin monenko tunnin välein haku suoritetaan. Nykytoteutuksessa haku tapahtuu aina 24 tunnin välein lisäosan asennushetkestä alkaen. Lisäksi haun voi suorittaa manuaalisesti milloin vain, mutta tämä ei vaikuta automatisoituun hakuun mitenkään.

5 Pohdinta

5.1 Tulokset verrattuna tavoitteisiin

Työn tavoitteena oli kehittää toteutus, jolla Nettimoto-palvelun rajapinta integroidaan WordPress-julkaisujärjestelmää käyttävän sivuston kanssa. Useat toimeksiantajan asiakkaat toivovat kotisivuilleen myytävien ajoneuvojen näyttömahdollisuutta, joten tällaiselle lisäosalle olisi tarvetta myös tulevaisuudessa. Tarkoituksena oli myös

yksinkertaistaa verkkosivustojen kehittämistä tulevaisuudessa uudelleen käytettävän lisäosan myötä. Aluksi tavoitteet olivat varsin ympäripyöreät, mutta kehitystyön edetessä aihepiiri tarkentui ja työn tahti säilyi jatkuvasti varsin hyvänä tasapainotellen tiedonhankinnan ja ohjelmoinnin välillä.

Ajoneuvojen myynti-ilmoitukset on usein aiemmin haettu iframe-elementteinä, mutta ne kahlitsevat pitkälti sivuston kehitystä estämällä ilmoitusten tyylimuutokset sivuston puolella. Niiden etuna on helppous, koska koko ajoneuvolistaus tulee yhdessä paketissa ja se voidaan liittää sivuston HTML-rakenteeseen sellaisenaan.

Heikkoutena on tyylimuutosten hankaloitumisen lisäksi se, ettei ajoneuvoja tallenneta WordPressiin. Tästä seuraa, että niiden lajitteluun, hakemiseen, näyttämiseen tai hallintaan voida vaikuttaa, vaan toiminnallisuudet ovat täysin riippuvaisia ulkoisesta palvelusta.

Tässä työssä tehty toteutus vastaa näihin haasteisiin integroimalla rajapinta toimimaan suoraan sivuston kanssa yhdessä ja näin voidaan eliminoida kaikki edellä kuvatut ongelmat. Lisäosa hakee tunnusten syöttämisen jälkeen yrityksen myymät ajoneuvot ja tallentaa ne WordPressin tietokantaan. Sivuston taitossa voidaan keskittyä ajoneuvodatan esilletuontiin ja sen käsittelyyn halutulla tavalla.

Sivupohjat toimivat runkona, jonka pohjalta voidaan lähteä kehittämään yksilöllisiä sivuja eri asiakasprojekteissa. Niiden toiminnallisuus voidaan pitää suhteellisen ennallaan ja taittovaiheessa päästään keskittymään sivuston ulkoasuun. Tämä nopeuttaa kehitystyötä jatkossa ja tuo säästöjä molemmille osapuolille.

5.2 Haasteet ja onnistumiset

Suurimmat kompastuskivet työn aikana liittyivät tehtävänantoon. Esimerkiksi käytettävä rajapinta tarkentui Nettiautosta Nettimotoksi vasta kun sivupohjien toiminnallisuus oli jo suurelta osin valmis. Tämä rajasi aikaa rajapinnan käytön opettelulta,

koska sen käyttöönotto odotti käytettävän rajapinnan varmistusta. Toisaalta nyt aikaa jäi kehittää toteutuksen muita osa-alueita ja rajapinnan varmistumisen jälkeen se saatiin varsin nopeasti yhdistettyä jo tehtyyn työhön. Tämä osoittaa sen, miten laajojakin kokonaisuuksia voidaan tehdä lopulta varsin monella tavalla ja päästä samaan lopputulokseen kuluttamatta kuitenkaan enempää aikaa.

Lisäksi haasteita aiheutti sopivan ja yhtenäisen laajuuden säilyttäminen tietoperustan kirjoittamisessa. Käsiteltävistä aiheista on valtavasti dokumentaatiota ja materiaalia sekä ne ovat varsin monimutkaisia, joten riittävän laajan, mutta ei kuitenkaan kaiken kattavan tietoperustan kirjoittaminen on haastavaa. Työssä käsiteltyjä aiheita on pyritty kuvaamaan riittävällä laajuudella, että niiden perusteella on helpompi ymmärtää ja mahdollisesti lähteä kehittelemään WordPress-lisäosaa, teemaa tai jotain Nettix-rajapintaa hyödyntävää sovellusta.

Työssä onnistuttiin kuitenkin saavuttamaan sille asetetut tavoitteet ja lisäksi keräämään kattava tietopohja WordPressistä ja Nettimoton rajapinnasta. Toteutus vähentää työtä ja helpottaa osaltaan verkkosivustojen kehitystä. Tämä tuo osaltaan taloudellisia säästöjä.

5.3 Tulosten luotettavuus ja laatu

Tietoperustan ajankohtaisuus ja paikkansapitävyys pitäisi olla hyvällä tasolla, koska valtaosa materiaalista on hankittu virallisten ohjeistusten ja dokumentaatioiden kautta. Tämän lisäksi suurinta osaa esitellyistä aiheista ja ominaisuuksista on hyödynnetty työssä, ja ne on todettu näin myös käytännössä toimiviksi tai paikkansapitäviksi. Kehitystyön tuloksena syntyneen lisäosan toimintaa ei ole voitu käytännön syistä kokeilla rajapinnan tuotantoversiolla, mutta testirajapinnan käytön perusteella sen toiminnassa ei havaittu ongelmia.

5.4 Jatkokehitys

Tulevaisuudessa voidaan jatkokehittää lisäosaa tukemaan esimerkiksi muita Nettix-rajapintoja ja sallimaan toiminta myös toiseen suuntaan mahdollistaen myynti-ilmoitusten tekemisen WordPressistä käsin. Opinnäytetyöprosessissa hankittu tietotaito auttaa esimerkiksi WordPress-lisäosien kehittämisessä, ja helpottaa WordPressin tarjoamien lisäosatyökalujen laajamittaista hyödyntämistä.

Lähteet

Aaron Parecki. N.d. OAuth 2 Simplified. Viitattu 2.3.2020. https://aaronparecki.com/oauth-2-simplified/

About Features. N.d. Tietoa WordPressistä wordpress.org-sivustolla. Viitattu 13.2.2020. https://wordpress.org/about/features/

Abstract. 9.2012. The OAuth 2.0 Authorization Framework. Viitattu 3.3.2020. https://tools.ietf.org/pdf/rfc6749.pdf

Actions. N.d. Lisäosakehityksen käsikirja wordpress.org-sivustolla. Viitattu 17.2.2020. https://developer.wordpress.org/plugins/hooks/actions/

Additional Parameters. N.d. Lisäosakehityksen käsikirja wordpress.org-sivustolla. Viitattu 17.2.2020. https://developer.wordpress.org/plugins/hooks/actions/#additional-parameters

Avoid Naming Collisions. N.d. Lisäosakehityksen käsikirja wordpress.org-sivustolla. Viitattu 17.2.2020. https://developer.wordpress.org/plugins/plugin-basics/best-practices/#avoid-naming-collisions

Basic Hooks. N.d. Lisäosakehityksen käsikirja wordpress.org-sivustolla. Viitattu 14.2.2020. https://developer.wordpress.org/plugins/plugin-basics/#basic-hooks

Best Practices. N.d. Lisäosakehityksen käsikirja wordpress.org-sivustolla. Viitattu 17.2..2020. https://developer.wordpress.org/plugins/plugin-basics/best-practices/

Bill of rights. N.d. Tietoa WordPressistä wordpress.org-sivustolla. Viitattu 13.2.2020. https://wordpress.org/about/

CMS Usage Distribution in the Top 1 Million Sites. N.d. Built With -verkkosivusto. Viitattu 13.2.2020. https://trends.builtwith.com/cms

Content Management System. N.d. Optimization Glossary optimizely-verkkosivuilla. Viitattu 13.2.2020. https://www.optimizely.com/optimization-glossary/content-management-system/

Condon, E. N.d. Hello. Elliot Condonin kotisivu. Viitattu 12.2.2020. https://www.elliotcondon.com

Custom Fields. N.d. WordPressin tukisivusto. Viitattu 12.2.2020. https://wordpress.org/support/article/custom-fields/

Custom Post Type Templates. N.d. Lisäosakehityksen käsikirja wordpress.org-sivustolla. Viitattu 17.2.2020. https://developer.wordpress.org/plugins/post-types/working-with-custom-post-types/#custom-post-type-templates

Errors. N.d. Nettix Common guidelines to all Nettix REST APIs. Viitattu 3.3.2020. https://api-test.nettix.fi/docs/api-common.html

Examples. N.d. Teemakehityksen käsikirja wordpress.org-sivustolla. Viitattu 17.2.2020. https://developer.wordpress.org/themes/basics/template-hierarchy/#examples

Features. N.d. Tietoa WordPressistä wordpress.org-sivustolla. Viitattu 13.2.2020. https://wordpress.org/about/features/

File Organization. N.d. Lisäosakehityksen käsikirja wordpress.org-sivustolla. Viitattu 17.2.2020. https://developer.wordpress.org/plugins/plugin-basics/best-practices/#file-organization

Filters. N.d. Lisäosakehityksen käsikirja wordpress.org-sivustolla. Viitattu 17.2.2020. https://developer.wordpress.org/plugins/hooks/filters/

get_post_meta(). N.d. Koodiohjeistus WordPressin kehittäjäsivuilla. Viitattu 12.2.2020. https://developer.wordpress.org/reference/functions/get_post_meta/

Helpoin tapa hallita ilmoitusdataa. N.d. Nettix-rajapinnan verkkosivut. https://nettix.fi/yrityksille/kaikki-palvelut/nettix-rest-api/

Hooks: Actions and Filters. N.d. Lisäosakehityksen käsikirja wordpress.org-sivustolla. Viitattu 14.2.2020. https://developer.wordpress.org/plugins/plugin-basics/#hooks-actions-and-filters

Introduction to Plugin Development. N.d. Lisäosakehityksen käsikirja wordpress.org-sivustolla. Viitattu 14.2.2020. https://developer.wordpress.org/plugins/intro/

Markkinapaikat. N.d. Otava Markkinapaikkojen verkkosivu. Viitattu 28.1.2020. https://otavamarkkinapaikat.fi/markkinapaikat/

Mikä on Nettix?. N.d. Nettix:n verkkosivu. Viitattu 28.1.2020. https://nettix.fi/meista/

Moottoripyörät, mönkijät ja moottorikelkat. N.d. Nettimoto-palvelun esittely Nettix:n verkkosivuilla. Viitattu 28.1.2020. https://nettix.fi/yksityisille/kaikki-palvelut/nettimoto/

Nettimoto REST API. N.d. Nettimoton rajapinnan dokumentaatio. Viitattu 3.3.2020. https://api-test.nettix.fi/docs/bike/#/

NettiX authentication API. N.d. NettiX authentication API:n dokumentaatio. Viitattu 3.3.2020. https://auth.nettix.fi/docs/swagger-ui/index.html

Number of Arguments. N.d. Lisäosakehityksen käsikirja wordpress.org-sivustolla. Viitattu 17.2.2020. https://developer.wordpress.org/plugins/hooks/actions/#number-of-arguments

Our Story. N.d. Tietoa WordPressistä wordpress.org-sivustolla. Viitattu 13.2.2020. https://wordpress.org/about/

Plugin Basics. N.d. Lisäosakehityksen käsikirja wordpress.org-sivustolla. Viitattu 14.2.2020. https://developer.wordpress.org/plugins/plugin-basics/

Priority. N.d. Lisäosakehityksen käsikirja wordpress.org-sivustolla. Viitattu 17.2.2020. https://developer.wordpress.org/plugins/hooks/actions/#priority

Querying by Post Type. N.d. Lisäosakehityksen käsikirja wordpress.org-sivustolla. Viitattu 17.2.2020. https://developer.wordpress.org/plugins/post-types/working-with-custom-post-types/#querying-by-post-type

Required files. N.d. Teemakehityksen käsikirja wordpress.org-sivustolla. Viitattu 17.2.2020. https://developer.wordpress.org/themes/getting-started/what-is-a-theme/#required-files

Template Files. N.d. Teemakehityksen käsikirja wordpress.org-sivustolla. Viitattu 17.2.2020. https://developer.wordpress.org/themes/basics/template-files/

The Template File Hierarchy. N.d. Teemakehityksen käsikirja wordpress.org-sivustolla. Viitattu 17.2.2020. https://developer.wordpress.org/themes/basics/template-hierarchy/#the-template-file-hierarchy

Use your existing data. N.d. FacetWP:n verkkosivut. Viitattu 12.2.2020. https://facetwp.com

What is ACF? N.d. Advanced Custom Fieldsin usein kysytyt kysymykset. Viitattu 12.2.2020. https://www.advancedcustomfields.com/resources/frequently-asked-questions/

What is a custom field? N.d. Advanced Custom Fieldsin usein kysytyt kysymykset. Viitattu 12.2.2020. https://www.advancedcustomfields.com/resources/frequently-asked-questions/

What is a Template Tag? N.d. Teemakehityksen käsikirja wordpress.org-sivustolla. Viitattu 17.2.2020. https://developer.wordpress.org/themes/basics/template-tags/#what-is-a-template-tag

What is a Theme? N.d. Teemakehityksen käsikirja wordpress.org-sivustolla. Viitattu 17.2.2020. https://developer.wordpress.org/themes/getting-started/what-is-a-theme/

What is the JSON Web Token structure? N.d. Introduction to JSON Web Tokens jwt.io-verkkosivustolla. Viitattu 3.3.2020. https://jwt.io/introduction/

What is REST. N.d. REST API-rajapintojen ohjesivusto. Viitattu 3.3.2020. https://restfulapi.net

Liitteet

Liite 1. Taulukko 1. Nettimoto-lisäkentät ja niiden tyypit.

ACF-kenttä	**Kentän tyyppi**
Ajoneuvon ID	Numero
Ajamaton	Kyllä/Ei
Alue	Teksti
Kaupunki	Teksti
Ajoneuvon nimi	Teksti
Ajoneuvolaji	Teksti
Tyyppi	Teksti
Merkki	Teksti
Malli	Teksti
Mallitarkennus	Teksti
Moottorin tilavuus	Numero
Moottorin tyyppi	Teksti
Vetotapa	Teksti
Vaihteisto	Teksti
Hinta	Numero
Vuosimalli	Numero

Kilometrit	Numero
Tunnit	Numero
Rekisterinumero	Teksti
Kuvaus	Tekstialue
Tarvikkeet	Toista rivejä
Kuvat	Toista rivejä