jamk.fi

Automaattinen sisällönhaku WordPress-julkaisujärjestelmään

Tuomas Jakonen

Opinnäytetyö
Joulukuu 2016
Tekniikan ja liikenteen ala
Insinööri (AMK), mediatekniikan tutkinto-ohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Jakonen, Tuomas	Opinnäytetyö, AMK	11.12.2016
	Sivumäärä 27	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi
Automaattinen sisällönhaku WordPress-julkaisujärjestelmään

Tutkinto-ohjelma
Meditekniikka

Työn ohjaaja(t)
Manninen, Pasi

Toimeksiantaja(t)
MEOM Oy

Tiivistelmä

Opinnäytetyön aiheena oli toteuttaa asiakkaalle uusi lisäosa WordPress-julkaisujärjestelmälle, jonka avulla Twitteristä haettaisiin asiakkaan uusimmat twiitit osaksi uutta verkkosivustoa. Toteutuksen teknisenä vaatimuksena oli saada twiitit jäsennettyä osaksi sivuille tulevaa blogia. Tämän lisäksi blogissa olevat artikkelit oli pystyttävä järjestämään julkaisupäivämäärän mukaan. Opinnäytetyö toteutettiin soveltavana tutkimuksena, jonka lopputuotteena syntyi automaattisen sisällönhaun toiminnallisuus eli lisäosa WordPress-alustalle.

Opinnäytetyössä tarkasteltiin WordPressin tarjoamia teknisiä mahdollisuuksia sovelluskehittäjän näkökulmasta, automaattisen sisällönhaun taustaa sekä millaisia valmiita ratkaisuita WordPressillä on tarjota asiakkaille.

Koska twiitit tuli jäsentää osaksi blogia, oli twiiteistä kannattavinta tehdä omia artikkeleitaan WordPressin artikkeleiden sekaan. Sisällön hakemiseen toteutettiin lisäosa, joka hakee uusimmat twiitit, muokkaa haettujen twiittien datan muotoa, jotta WordPress osaisi käsitellä dataa oikein, sekä tallentaa muunnetun datan JSON-tiedostoon palvelimella määritettyyn sijaintiin. Haussa käytettiin Twitterin luomaa REST-rajapintaa sekä WordPressin ytimessä olevaa omaa REST-rajapintaa.

Jatkokehityksenä lisäosasta voisi tehdä useamman järjestelmän integraation. Tällöin lisäosasta tulisi monikäyttöinen ja sen voisi myös julkaista WordPressin lisäosakirjastossa. Useamman järjestelmän integraation yhteydessä tulisi tehdä lisäosalle oma asetussivu, jonka kautta lisäosan käyttäjäavaimien hallinta olisi helpompaa. Lisäosaa julkistettaessa tulisi myös tarve keskittyä entistä enemmän lisäosan tietoturvaan, jotta käyttäjän syöttämät tunnukset eivät joutuisi vääriin käsiin.

Avainsanat (asiasanat)
WordPress, sisällönhaku, REST, sovelluskehitys

Muut tiedot

jamk.fi

Description

Author(s)	Type of publication	Date
Jakonen, Tuomas	Bachelor's thesis	11.12.2016
		Language of publication: Finnish
	Number of pages: 27	Permission for web publication: x

Title of publication

Automatic content transfer to WordPress content management system

Degree programme

Media engineering

Supervisor(s)

Manninen, Pasi

Assigned by

MEOM Oy

Abstract

The topic of the thesis was to create a new WordPress plugin for the customer. With the plugin you should be able to get the latest tweets and add them as a part of a new website. A technical requirement of the plugin was to get tweets parsed to the blog on the website. The posts on the blog should also be sortable by the date of publishion. Bachelor's thesis was produced as an applied research, which end product was an automatic content search plugin for WordPress.

Matters reviewed in the thesis were technical possibilities offered by WordPress from a developers point of view, the basics of automatic content transfer and the range of ready-made technical possibilities WordPress has to offer to clients.

Because the tweets had to be parsed to the blog, the tweets had to be converted as posts in WordPress. In order to transfer the data, a plugin was created that transfers the newest tweets, converts tweets to the right formation so WordPress can handle the transferred data correctly, and saves the converted data to a JSON file, that has its location set on the plugin. Twitter's REST API and WordPress' REST API were used in the transfer.

As a further development, there is a possibility to make the plugin as an integration of several different systems. That would make the plugin more versatile and it would be possible to publish it in WordPress' plugin collection. Creating an integration for several different systems a settings page should be created to easily maintain every system's access tokens. When publishing the plugin the security of plugin should be taken care of so the access tokens would be safe.

Keywords/tags (subjects)

WordPress, content search, REST, software development

Miscellaneous

Sisältö

Kuviot

Taulukot

Termistö

CMS

CMS eli Content Management system eli sisällönhallintajärjestelmä. Yleisnimitys tietojärjestelmälle, joka palvelee organisaation sisällönhallintaa sen sijaan, että olisi keskittynyt ainoastaan johonkin yksittäiseen osa-alueeseen.

CURL

CURL on PHP:n yksi luokkakirjasto, jonka avulla voidaan suorittaa HTTP-kutsuja sekä lähettää dataa HTTP-protokollalla. Kirjasto tukee muun muassa palautettavan datan noutoa, useita eri prosesseja yhtäaikaisesti sekä osaa noutaa saadusta datasta tiettyjen alueiden data

GPLv2

GPL on vapaiden ohjelmistojen julkaisemiseen tarkoitettu lisenssi, joka antaa kenelle tahansa oikeuden käyttää, kopioida, muuttaa ja jakaa edelleen ohjelmia ja niiden lähdekoodia. Lisäksi lisenssi takaa, että nämä vapaudet säilyvät myös GPL-koodiin pohjautuvissa muunnelluissa teoksissa. Mikäli GPL-ohjelmaa tai sen muunnelmaa levitetään edelleen, lähdekoodi on julkaistava samalla lisenssillä eikä ohjelman käytölle tai levitykselle saa asettaa lisärajoituksia.

Toisen version isoimman muutoksen mukaan jakelija ei saa lainkaan jakaa GPL-ohjelmistoa, jos se olisi mahdollista ainoastaan toisten käyttäjien vapauksia rajoittamalla.

JSON

JSON (lyhenne sanoista JavaScript Object Notation) on yksinkertainen avoimen standardin tiedostomuoto tiedonvälitykseen. Nimestään ja JavaScript-perustastaan huolimatta JSON on JavaScriptistä riippumaton.

Julkaisujärjestelmä

Yksi mahdollisista sisällönhallintajärjestelmistä.

MySQL

MySQL on relaatiotietokantaohjelmisto, jonka avulla tallennetaan kaikki käytettävä data verkkototeutuksesta.

REST

REST (Representational State Transfer) on HTTP-protokollaan perustuva arkkitehtuurimalli ohjelmointirajapintojen toteuttamiseen.

Twitter

Yhteisö- ja mikroblogipalvelu, jonka käyttäjät pystyvät lukemaan toisten käyttäjien 140 merkin mittaisia päivityksiä.

WordPress

WordPress on avoimen lähdekoodin julkaisujärjestelmä, jonka perustana toimii PHP-ohjelmointikieli ja MySQL-tietokanta. Maailman suosituin julkaisujärjestelmä.

WP Cron

WP Cron on WordPressin funktio, jolla pystytään ajastamaan haluttuja toimintoja halutun aikamääreen jälkeen.

1 Työn lähtökohdat

1.1 Taustaa ja toimeksiantaja

Opinnäytetyön tilaajana toimi MEOM Oy, joka on nopeasti kasvava jyväskyläläinen digitaalinen mainostoimisto. Yrityksen pääliiketoiminta on internet-sivujen tuottaminen. Muita liiketoiminta-alueita ovat muun muassa brändäys, markkinointi, sovelluskehitys, käyttöliittymäsuunnittelu sekä videotuotanto. MEOM työllistää tällä hetkellä vakituisesti 12 työntekijää, joiden lisäksi yrityksen kirjoilla on noin kymmenkunta freelanceria ja harjoittelijaa.

MEOMin tuottamat internet-sivut toteutetaan lähes poikkeuksetta hyödyntäen WordPress-julkaisujärjestelmää. WordPress miellettiin vielä muutama vuosi sitten pelkästään blogi-tyyppiseksi julkaisujärjestelmäksi, mutta viime vuosien aikana WordPressistä on tullut monipuolinen julkaisujärjestelmä kaiken kokoisille ja tyyppisille sivuille. WordPress-julkaisujen osuus kaikista maailman internet-sivuista on jo lähes 27 %. (Usage of content management systems for websites 2016)

1.2 Tehtävä ja tavoitteet

Opinnäytetyön tavoitteena oli toteuttaa lisäosa, joka mahdollistaa ulkoisen palvelun sisällön käyttämisen WordPressissä. Työ toteutettiin osana tilaajan asiakasprojektia. Tarpeelliseksi palveluksi asiakas koki ainoastaan Twitterin, joten muita palveluita ei haluttu tässä vaiheessa ottaa mukaan kehitystyöhön. Työlle määritettyjen vaatimusten takia oman lisäosan tuottaminen koettiin välttämättömäksi, koska muut valmiit lisäosat eivät tarjonneet sopivia ominaisuuksia. Lisäksi tavoitteena oli kartuttaa tietämystä lisäosan tuottamisesta WordPressille sekä yleisistä hyvistä käytänteistä WordPressin kehityksessä.

Opinnäytetyö toteutettiin soveltavana tutkimuksena, jonka lopputuotteena syntyi automaattisen sisällönhaun toiminnallisuus eli lisäosa WordPress-alustalle.

2 WordPress

2.1 Yleistä

WordPress on ilmainen avoimeen lähdekoodiin perustuva julkaisu- ja sisällönhallintajärjestelmä, joka pohjautuu PHP-ohjelmointikieleen ja MySQL-tietokantaan. WordPress on alun perin b2/cafelog–julkaisujärjestelmän haarauma, jonka pohjalta ensimmäinen versio WordPressistä julkaistiin 27. toukokuuta 2003. Perustajajäseninä toimivat Matt Mullenweg sekä Mike Little. WordPress on lisensoitu käyttäen Free Software Foundationin GPLv2–lisenssiä. (WordPress Codex: History 2016)

Kaikista verkkosivuista noin 46 % on tuotettu jotakin sisällönhallintajärjestelmää (CMS, Content Management System) käyttäen. WordPress on selvästi suosituin CMS-alusta kattaen lähes 59 % kaikista CMS-alustoilla tuotetuista sivuista. (Usage of content management systems for websites 2016)

2.2 Tärkeimmät ominaisuudet

2.2.1 Teemat

Yksi iso osa verkkosivustoja on sivuston ulkoasu. WordPress-sivuston ulkoasusta vastaa käytössä oleva teema. Teeman tarkoituksena on esittää haluttu sisältö loppukäyttäjälle halutulla tavalla. Koska teema ainoastaan esittää halutun sisällön käyttäjälle, teeman muokkaaminen ei muuta WordPressin sisältöjä millään tavalla.

Teeman voi tehdä alusta alkaen itse, muokata jotain valmista teemaa omiin tarpeisiin sopivaksi (ns. luurankoteema, skeleton theme), tehdä valmiin teeman päälle joitain muutoksia (ns. lapsiteema, child theme) tai käyttää kokonaan valmista teemaa sivuston pohjana. Ilmaisia teemoja on tarjolla yksinomaan WordPressin teemakansiossa yli 4000 kappaletta, minkä lisäksi tarjolla on noin 100 kappaletta kaupallisia teemoja. (WordPress: Theme directory 2016)

2.2.2 Lisäosat

Sivustoa luodessa on mahdollista lisätä erilaisia toiminnallisuuksia lisäosien (plugin) avulla. Lisäosilla pystytään esimerkiksi luomaan tapahtumakalentereita, näyttämään käyttäjän sosiaalisen median sisältöjä sekä parantamaan sivuston tietoturvaa.

WordPressin sivuilta saatavien lisäosien määrä on yli 47 000 kappaletta (WordPress: Plugin directory 2016). Tämän lisäksi on tarjolla paljon erilaisia maksullisia lisäosia, jotka ovat saatavilla esimerkiksi lisäosan/kehittäjän omilta sivuilta.

2.2.3 Ohjelmointirajanpinta

WordPress tarjoaa ohjelmistokehittäjälle kattavan ohjelmointirajapinnan, joka voidaan jakaa useampaan eri rajapintaan. Jokainen rajapinta pitää sisällään omat funktionsa, ja jokainen rajapinta tuo jonkin tietyn toiminnallisuuden sovellukseen. Kaikki erilliset rajapinnat luovat niin kutsutun WordPress-rajapinnan. (WordPress APIs 2016)

2.3 Kehitys WordPressille

2.3.1 Yleistä kehityksestä

WordPressin kehitystyö voi tarkoittaa sekä teeman että lisäosan kehittämistä. Tämä opinnäytetyö keskittyy ainoastaan uuden toiminnon eli lisäosan kehittämiseen.

Kuten luvussa 2.2.3 mainittiin, WordPress tarjoaa ohjelmistokehittäjälle ohjelmointirajapinnan, jonka avulla lisäosan on mahdollista toimia WordPress-ytimen rinnalla. Seuraavissa luvuissa puran ohjelmointirajapinnan pienempiin osiin.

2.3.2 Plugin API

Plugin API on rajapinta, joka mahdollistaa toiminnallisuuden liittämisen WordPress-ytimen rinnalle. Toimintojen liittäminen tapahtuu koukuilla (hooks), joita on kahta eri tyyppiä. Joko koukut on sijoitettu WordPressin ytimeen tai koukut on sijoitettu lisäosaan, jolloin ytimessä olevat koukut voivat tarttua lisäosassa oleviin koukkuihin.

WordPressiä suoritettaessa koukut käydään läpi ja tarkistetaan, onko niihin rekisteröity funktioita. Koukkujen rekisteröidyt funktiot suoritetaan välittömästi niiden löytöhetkellä ja ytimen suorittamista jatketaan koukun funktion suorittamisen jälkeen. Koukkuihin voidaan kiinnittyä joko toiminta- (actions) tai suodatinkoukuilla (filters). (Plugin API 2016)

Actions eli toimintakoukut

Toimintakoukuilla (action hooks) muokataan WordPressin toiminnallisuutta ja ne suoritetaan yleensä jonkin toiminnon seurauksena. Toimintakoukku muodostetaan luomalla ensin funktio lisäosatiedostoon, joka on tarkoitus suorittaa, kun haluttu WordPress-tapahtuma (event) suoritetaan. Tämän jälkeen kiinnitetään luotu funktio haluttuun tapahtumaan käyttäen add_action()-funktiota, jonka ensimmäiseksi parametriksi annetaan koukku ja toiseksi suoritettava funktio.

Kolmanneksi parametriksi voidaan antaa järjestysluku, joka määrittää, missä järjestyksessä funktioita suoritetaan. Pienempi luku kuvaa tärkeää funktiota ja suurempi luku kuvaa vähemmän tärkeätä funktiota. Neljäs parametri määrittää, kuinka monta parametria suoritettava funktio voi maksimissaan sisältää. (Function reference / add_action 2016.)

Käytettävän toimintakoukun muodostama toiminta suoritetaan tai sen tuottama sisältö tulostuu siihen kohtaan dokumenttia, jossa koukkua kutsutaan. Alla esimerkkinä tuotetun lisäosan toimintakoukusta.

```
add_action( 'rest_api_init', function() {
   register_rest_route( 'twitter/', 'list', array(
      'methods' => 'GET',
      'callback' => 'twitter_post_loader'
   )   );
}, 10 );
```

Filters eli suodatinkoukut

Suodatinkoukut (filter hooks) manipuloivat jo olemassa olevaa sisältöä. Koukkuun lisätään haluttu suodatin (filter), joka käsittelee datan ja suorittaa siihen määritetyt muutokset. (Function reference / add_filter 2016)

Suodattimilla pystytään esimerkiksi muuttamaan tulostettavaa sisältöä. Alla on esimerkki, jossa piilotetaan teeman ja lisäosien automaattiset päivitykset, jolla pyritään estämään päivityksissä tulevat mahdolliset virheet.

```
add_filter( 'auto_update_plugin', '__return_false' );
add_filter( 'auto_update_theme', '__return_false' );
```

2.3.3 Settings API

Jos halutaan luoda käyttäjäystävällinen lisäosa, sen tulisi sisältää asetussivu, jonka avulla loppukäyttäjä pystyy hallitsemaan lisäosan toimintaa omalla sivustollaan. Asetussivun luominen onnistuu WordPressin Settings-rajapinnalla, jonka avulla pystytään luomaan standardin mukainen asetussivu. Asetussivulle voidaan esimerkiksi asettaa kentät eri palveluiden käyttöavaimille tai mahdollisuus muuttaa loppukäyttäjälle näkyvän osan tyylejä. (Settings API 2016)

Jos lisäosa tulee omaan käyttöön, asetussivun tekeminen ei ole pakollista vaan normaalisti asetussivun kautta syötettävät avaimet voidaan syöttää myös osaksi lisäosan tiedostoja.

2.3.4 Options API

Options API on yksinkertainen ja standardisoitu tapa tallentaa asetuksiin liittyvää dataa tietokantaan. Rajapinta mahdollistaa asetustietojen luomisen, lukemisen, muokkaamisen ja poistamisen. Kaikki säilötty data löytyy wp_options–tietokantataulusta. (Options API 2016)

3 Automaattinen sisällönhaku

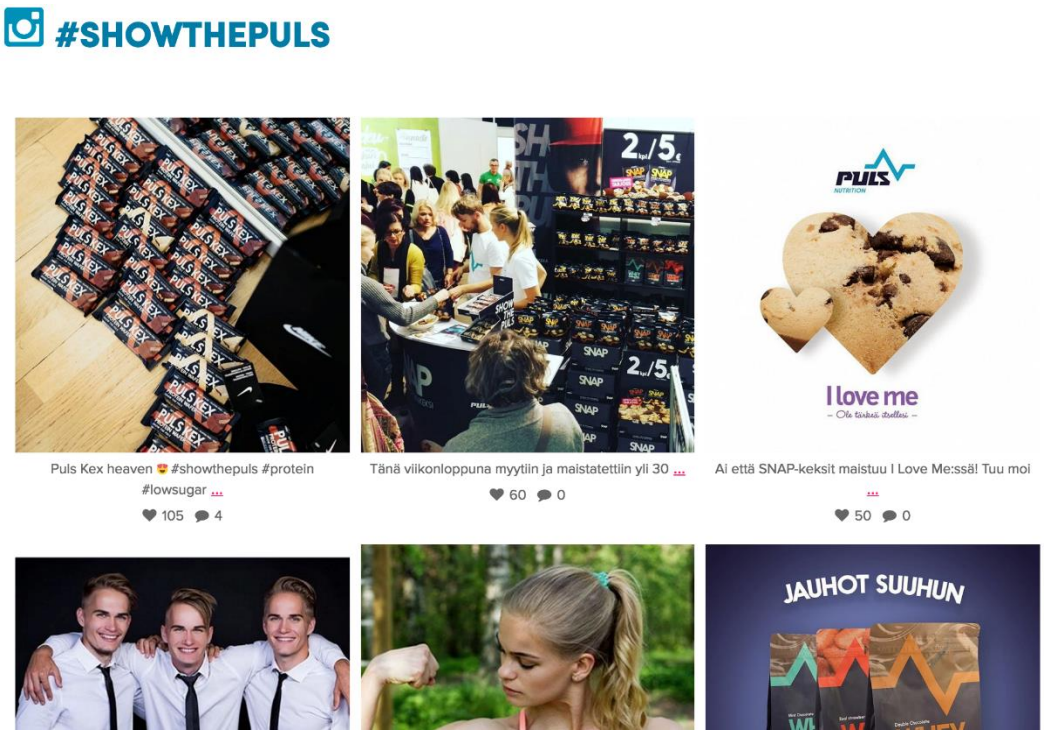
3.1 Taustaa

Sosiaalisen median käyttö on viime vuosien aikana kasvanut räjähdysmäisesti. Kyseinen trendi näkyy myös verkkototeutuksissa, kun lähestulkoon jokaisella verkkosivulla on upotettuna jonkinlainen sosiaalisen median moduuli.

Yleisimpiä liitännäisiä ovat sosiaalisen median syötteet (feed), johon haetaan automaattisesti esimerkiksi uusimmat twiitit, Facebook-julkaisut tai kuvat Instagramista. Tällä tavalla loppukäyttäjälle on mahdollista tarjota läheisempi kosketuspinta esimerkiksi yrityksen toimintaan ja imagoon.

3.2 Valmiit lisäosat sisällönhaulle

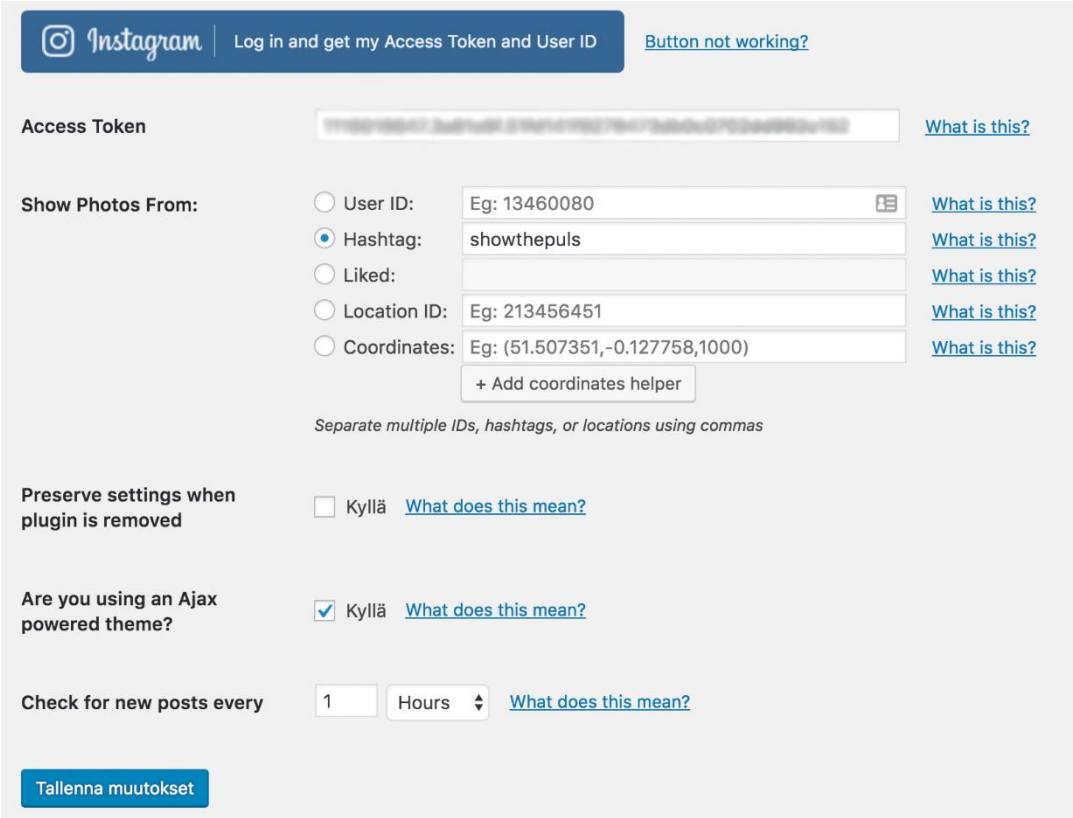
WordPress-yhteisö tarjoaa useita erilaisia automaattisen sisällönhaun lisäosia sivustojen käyttöön. Useimmat liittävät ainoastaan tietyn median julkaisut sivuston käytettäväksi. Lisäosat ovat yleensä helppoja ottaa käyttöön, mikä madaltaa kynnystä käyttää niitä, jos lisäosan toiminnallisuus on tarpeeksi kattava omaan tarkoitukseen. Kuviossa 1 on esitelty, millaisen lopputuloksen valmiilla lisäosalla voi saada aikaan.



Kuvio 1. Esimerkki valmiilla lisäosalla tuotetusta Instagram-syötteestä

Jotta sisältöä saadaan haettua lisäosan toimesta, tulee lisäosalle antaa oikeudet sisällön hakemiseen. Tunnistautuminen tapahtuu käyttöoikeustunnuksella (access token), joka sisältää käyttäjästä kaiken tarvittavan tiedon, jotta palvelu pystyy tunnistamaan käyttäjän oikeaksi henkilöksi. Kuviossa 2 on esimerkki, miten Smash Balloon -

yrityksen tuottama Instagram Feed Pro -lisäosa hoitaa käyttäjän käyttöoikeustunnusten hankinnan. Käyttäjä pystyy hakemaan tarvitsemansa tunnukset ainoastaan nappia painamalla.



Kuvio 2. Instagram Feed Pro -lisäosan käyttöoikeustunnusten hakeminen

3.3 Valmiiden lisäosien ongelmat

3.3.1 Sisällön saatavuus

Jotta haettu sisältö on varmasti saatavilla, se on tallennettava tavalla tai toisella. Yleensä lisäosat tallentavat tiedot omaan tietokantatauluun. Kuviossa 2 esitelty Instagram-lisäosaan määritetään, kuinka usein lisäosa hakee uusimmat julkaisut määritellystä lähteestä. Samalla määritetty aikaväli toimii välimuistina (cache) haetulle sisällölle. Kun lisäosa suorittaa uuden sisällön haun, poistaa se samalla vanhat sisällöt välimuistista ja lisää uudet sisällöt vanhojen tilalle. Jos sisältöjä ei esimerkiksi vanhentuneen käyttöoikeustunnuksen takia pystytä hakemaan, lisäosa ei sillä hetkellä pysty näyttämään ainuttakaan kuvaa vanhoista sisällöistä.

3.3.2 Muokattavuus

Valmiiden lisäosien muokattavuus on monesti hankalaa ja muokattavia kohteita ontodella niukasti. Muokattavia kohteita ovat yleensä lisäosan väriteema ja sisällön näyttämiseen liittyvät asetukset, esimerkiksi ladattavien kuvien määrä yhdellä kertaa.

3.4 REST

3.4.1 Yleistä

REST on arkkitehtuurityyli, joka tulee sanoista Representational State Transfer. REST esiteltiin Roy Fieldingin väitöskirjassa ”Architectural styles and the design of network-based software architectures” vuonna 2000. Monesti REST mielletään pelkäksi rajapinnaksi, mutta yleisenä arkkitehtuurityylinä se on alun perin suunniteltu käytettäväksi hajautetussa (hypermedia)järjestelmässä. Yleisin esimerkki REST-arkkitehtuurityyliin perustuvasta järjestelmästä on WWW (World Wide Web). (Fielding 2000, 76-106.)

Jos halutaan luoda pääsy internetpalvelun sisältämään dataan, palvelun datan ja palvelulle tulevien kysymysten väliin rakennetaan REST-rajapinta, joka huolehtii palvelulle tulevien kyselyiden ja välitettävän datan kommunikoinnista.

3.4.2 Toiminta

REST-rajapinnan toiminta perustuu resursseihin (resource) ja se käyttää tiedonvälitykseen HTTP-protokollaa. Resurssit yksilöidään URI:n (Uniform Resource Idnetifier) avulla. URI:n alaluokka on yleisesti tunnettu URL (Uniform Resource Locator). REST-rajapinta hyödyntää HTTP-toimintoja taulukon 1 mukaisesti.

Taulukko 1. HTTP-toiminnot REST-arkkitehtuurissa

Toiminto	Kuvaus
GET	Pyytää URI:n mukaisen resurssin
POST	Lähettää resurssin palvelimelle, päivittää resurssin URI:n määrittelemässä kohteessa.
PUT	Lähettää resurssin palvelimelle tallennettavaksi URI:n määrittelemään sijaintiin.
DELETE	Poistaa URI:n määrittelemän resurssin.
HEAD	Pyytää URI:n määrittämän resurssin metadataan.

3.4.3 REST WordPressissä

Versiossa 4.4 WordPress julkaisi oman REST-rajapintansa virallisena osana ydintään. Aiemmin ainoastaan lisäosamuodossa ollut toiminnallisuus integroitiin keskeisimmiltä osin WordPressin ytimeen, jotta kehittäjät pystyivät aloittamaan rajapinnan hyödyntämisen lisäosissa ja teemoissa. (WP REST API 2016.)

Taulukossa 1 mainitut HTTP-toiminnot ovat myös käytössä WordPressin rajapinnassa.

4 Automaattisen sisällönhaun lisäosan tuottaminen

4.1 Työn vaatimukset

Tavoitteena oli tuottaa lisäosa, jonka avulla haetaan automaattisesti viimeisimmät twiitit asiakkaan Twitter-tililtä ja yhdistetään twiitit osaksi blogin artikkeleita. Haettavat twiitit tuli pystyä esittämään muiden artikkeleiden seassa päivämäärän mukaan järjesteltynä. Lisäksi twiiteillä tulisi olla oma arkistosivu, jonne kootaan kaikki haettavat twiitit.

4.2 Suunnittelu

4.2.1 Alustaminen

Suunnittelun alkuvaiheessa kävi selväksi, että vaatimusmäärittelyssä asetetut rajoitteet sulkisivat muutamia toteutustapoja pois saman tien. Suurin haaste oli sisällyttää haetut twiitit osaksi blogia. Kyseiseen ongelmaan oli kaksi vaihtoehtoa:

1. Tallennetaan JSON-tiedostomuodossa haetut twiitit omaan tietokantatauluun, minkä jälkeen twiitit jäsennetään osaksi artikkeleiden hakua blogi-sivulla.
2. Haetaan twiitit JSON-tiedostomuodossa, minkä jälkeen tallennetaan twiitit WordPressin posts-tietokantatauluun artikkeleina, jolloin twiittejä ei erikseen tarvitse jäsentää osaksi blogi-sivua.

Toinen vaihtoehto osoittautui nopeasti paremmaksi ideaksi. Kun twiitit muutetaan artikkelimuotoon, sillä saavutetaan muun muassa seuraavia hyötyjä:

- Twiitit ovat automaattisesti osana artikkelihakuja
- Twiiteille tulee samat tyylit kuin muillekin artikkeleille
- Twiitit voidaan lokeroida osaksi uutta kategoriaa, jolloin twiittien näyttäminen yhtenä listana tapahtuu samalla tiedostolla kuin muidenkin kategorioiden sisältöjen

Jokainen edellä mainittu hyöty vähentäisi uuden koodin kirjoittamista. Arvioitu ajallinen hyöty oli useamman tunnin luokkaa heti luontivaiheessa. Lisäksi twiittien hallittavuus helpottuisi huomattavasti, kun twiitit ovat nähtävissä sivuston hallintapaneelin kautta. Mahdollisten virheiden sattuessa sisällön korjaaminen olisi täten helppoa ja nopeaa.

4.2.2 Sisällön hakeminen Twitteristä

Twitter on luonut sovelluskehittäjien käyttöön REST API -rajapinnan, jonka avulla pystytään hakemaan, luomaan, päivittämään ja poistamaan twiittejä Twitterin hyväksymiltä tileiltä. Projektia varten oli siis tarve hankkia oikeudet asiakkaan Twitter-tilin muokkaamiseen.

Twitterin REST API -rajapinnan käyttämiseen oli myös pari erilaista tapaa. PHP-ohjelmointikielessä keskustelu REST-rajapinnan kanssa hoidetaan yleensä CURL-metodeja käyttäen. WordPressin tapauksessa keskustelu pystyttiin toteuttamaan kätevästi WordPressin oman rajapinnan kautta.

4.2.3 Haetun sisällön liittäminen WordPressiin

Haetun sisällön liittämiseen osaksi WordPressiä oli kaksi eri tapaa. Joko lisäosaan lisättäisiin toiminto, joka muuttaa jokaisen haetun twiitin uudeksi artikkeliksi tai käytettäisiin erillistä lisäosaa twiittien muuttamiseksi artikkeleiksi.

Monessa aiemmassa asiakasprojektissa on täytynyt noutaa artikkelit vanhalta sivulta ja liittää ne osaksi uutta tuotettua sivustoa. WP All Import Pro on maksullinen lisäosa, joka tarjoaa kyseiseen tehtävään sopivat toiminnot. Yksi lisäosan toiminnoista mahdollistaa JSON-tiedoston lukemisen, jonka jälkeen käyttäjä pystyy määrittämään, miten lisäosa jäsentää tiedoston uusiksi artikkeleiksi. Lisäosa mahdollistaa myös toimintojen ajastamisen WP Cronin avulla, jolloin Twitterin asettamat hakurajat saataisiin toteutettua ilman suurempia ongelmia.

4.3 Lisäosan tuottaminen

4.3.1 Työn alustaminen

Lisäosan tuottaminen aloitettiin lisäosan nimeämisellä. Nimen tulisi olla tarpeeksi lyhyt ja samalla tarpeeksi informatiivinen, jotta mahdollisen julkaisun jälkeen lisäosa olisi helposti löydettävissä. Koska tarkoituksena oli noutaa uusimpia twiittejä, annettiin lisäosalle nimeksi Twitter Post Loader.

Lisäosan koodi löytyy kokonaisuudessaan liitteestä 1. Seuraavissa osissa käydään läpi yksityiskohtaisesti lisäosan eri vaiheet.

4.3.2 REST API -kutsu

Lisäosa koostuu kahdesta osasta, REST API -rajapintakutsusta sekä kutsussa käytettävästä twiittien jäsentämiseen tarkoitetusta funktiosta.

Lisäosan rajapintakutsu toteutettiin toimintokoukun (action hook) avulla.

```
add_action( ’rest_api_init’, function() {}, 10 );
```

Alussa add_action-funktiolla alustetaan toimintokoukun käyttäminen. Funktion ensimmäinen parametri rest_api_init tarttuu WordPressin ytimessä olevaan koukkuun (hook). Toiseksi parametriksi lisätään koukussa suoritettava funktio. Kolmas parametri kertoo WordPressile kuinka tärkeästä toiminnosta on kyse. Mitä pienempi numero, sen tärkeämpi toiminto. 10 on oletusparametri jokaiselle toiminnolle, jota ei tässä tapauksessa ollut tarvetta muuttaa.

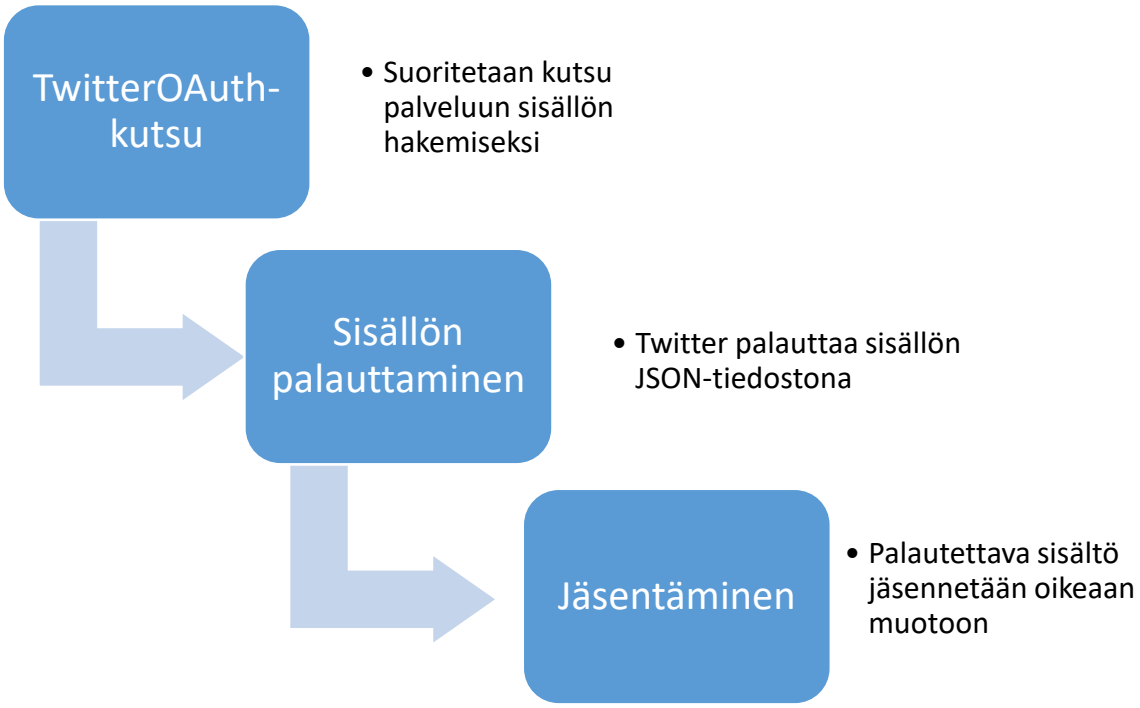
```
add_action( ’rest_api_init’, function() {
   register_rest_route( ’twitter/’, ’list’, array(
      ’methods’ => ’GET’,
      ’callback’ => ’twitter_post_loader’
   ) );
}, 10 );
```

Toimintokoukun alustamisen jälkeen toteutettiin suoritettava funktio. Register_rest_route-funktion avulla haettava sisältö saadaan ohjattua haluttuun paikkaan. Ensimmäinen parametri kertoo WordPressille, minne koukun avulla haettava data tallentuu. Tässä tapauksessa tallennukselle määritetty reitti olisi muotoa www.example.com/wp-json/twitter/. Alun www.example.com/wp-json/ on WordPressin käyttämä osoite kaikille REST-rajapinnan sisällöille.

Toinen parametri kertoo funktiolle tiedoston, jonne haettava data tallennetaan. Koska tarkoituksena on hakea listaus kaikista twiiteistä, asetettiin tiedostonimeksi ”list”. Tiedoston osoite olisi siis www.example.com/wp-json/twitter/list/. Kolmantena parametrina annetaan joukko asetuksia, joilla määritetään käytettävä REST-toiminto sekä suoritettava funktio. Ainoastaan REST-toiminto on pakollinen, mutta tässä tapauksessa syötetään myös funktio, joka jäsentää haettavan datan. Koska tarkoituksena on ainoastaan hakea dataa Twitteristä, toiminnoksi asetetaan GET.

4.3.3 Suoritettava funktio

Suoritettavan funktion rakenne koostuu kolmesta osasta: suoritettavasta kutsusta, palautettavasta sisällöstä sekä palautetun sisällön jäsentämisestä. Kuviossa 3 on esitetty kyseinen rakenne.



Kuvio 3. Suoritettavan funktion rakenne.

Funktion aluksi alustetaan tyhjä jono, jonne lopuksi lisätään jäsennetty data kokonaisuudessaan. Lisäksi toiseksi muuttujaksi on määritetty kokonaisluku, joka rajoittaa haettavien twiittien määrän 25 kappaleeseen yhtä kertaa kohden. Rajoitus on asetettu täsmälleen kyseiseen lukuun, koska Twitter ei salli haettavan yhdellä kertaa enempää dataa.

```
$allTweets = [];
$count = 25;
```

Lukumäärän jälkeen määritetään neljä muuttujaa, jotka sisältävät Twitterin luodut kehittäjätunnukset. Ilman toimivia tunnuksia kutsut eivät pääse läpi ja haettavaa sisältöä ei ole mahdollista saada. Tietoturvasyistä tunnukset on jätetty näyttämättä.

```
$consumerkey = **;
$consumersecret = **;
$accesstoken = **;
```

```
$accesstokensecret = **;
```

Tunnusten jälkeen luodaan yhteys Twitterin palvelimelle. TwitterOAuth-funktio suorittaa sovelluksen tunnistuksen edellä mainittujen tunnusten avulla. Tunnistautumista käytetään seuraavan kohdan GET-kutsussa, jossa suoritetaan halutun datan haku palvelimelta. Kutsussa haetaan twiittejä tietyn käyttäjän aikajanalta. Lisämääreenä loppuun on lisätty aiemmin asetettu haettavien twiittien määrä.

```
$connection = new TwitterOAuth($consumerkey, $consumersecret, $accesstoken, $accesstokensecret);
$twitdata_tag[0] = $connection->get('https://api.twitter.com/1.1/statuses/user_timeline.json?screen_name=**&count='. $count);
```

Kun haku on suoritettu, suoritetaan kahden foreach-silmukan avulla jäsennys, jolla saadaan oikea data JSON-tiedostosta käyttöön. Jälkimmäisen foreach-silmukan sisällä on ehtolause (if), joka tarkistaa, onko twiitti tyhjä vai ei. Ainoastaan twiitit, joilla havaitaan jotain sisältöä oikeassa solmussa (node), otetaan mukaan tallennettavaan sisältöön.

```
foreach($twitdata_tag as $tagdata) {
  foreach ($tagdata as $tweet) {
    if ($tweet->text != NULL) { // Ei tulosteta tyhjiä tweettejä
    } <!—endif -->
  } <!—endforeach -->
} <!—endforeach -->
```

Edellä mainitun ehtolauseen jälkeen aloitetaan sisällön jäsentäminen tallennusta varten. Ensin määritetään twiitille kustomoitu linkki, joka ohjaa käyttäjän twiittiä klikatessa asiakkaan Twitter-tiliin ja aukaisee klikatun twiitin uuteen popup-ikkunaan. Linkin kohteeksi määritetään asiakkaan tili ja loppuun lisätään määrite /status, joka tarkentaa linkin kohteen yksittäiseksi twiitiksi. Lopuksi lisätään twiitin ID, jonka avulla saadaan aukaistua oikea twiitti Twitterin puolella.

```
$tweet->link = 'https://www.twitter.com/**/status/'. $tweet->id;
```

Twiitin osoitteen jälkeen tarkistetaan, onko kyseistä twiittiä vielä luotu JSON-tiedostoon. Tarkistus suoritetaan twiitin ID:n perusteella. Jos twiittiä ei löydy, lisätään alussa määritettyyn jonoon uusi twiitti sisältöineen. Tämä tarkistus estää samojen twiittien tallentamisen moneen kertaan.

```
if( ! isset($allTweets[$tweet->id]) ) $allT-weets[$tweet->id] = $tweet;
```

Twiitin olemassaolon tarkistuksen jälkeen jäsennetään twiitille oikeanlainen päivämäärä. Twitter tallentaa päivämäärän palvelimelleen sellaisessa muodossa, että sitä on suoraan vaikea lähteä jäsentämään järkevään muotoon. Päivämäärä tuleekin ensin muuttaa toiseen muotoon, jonka jälkeen sen avulla pystytään luomaan uusi aikaleima, joka on muodossa pp.kk.vvvv. Lisäksi aikavyöhykkeeksi asetetaan sillä hetkellä Suomessa käytössä oleva aikavyöhyke, jotta twiittien julkaisuajankohdat eivät mahdollisesti vaihtelisi käyttäjän mukaan.

```
$pt = date_parse_from_format( "D M d H:i:s O Y", $tweet->created_at );
$date = $pt['year'] .'-'. $pt['month'] .'-'. $pt['day'] .'T'. $pt['hour'] .':'. $pt['minute'] .':'. $pt['second'];
$date = date_create($date, timezone_open("Europe/Helsinki"));
$parse = date_format( $date, 'd.m.Y' );
$tweet->date = $parse;
```

Kun molemmat foreach-silmukat on käyty läpi, lisätään kaikki twiitit alussa määritettyyn jonoon, joka aivan lopuksi palautetaan lisäosan alussa olleelle register_rest_route-funktiolle, joka tallentaa tuodun datan samassa funktiossa määritettyyn kohteeseen.

```
$allTweets = array_values($allTweets);
return $allTweets;
```

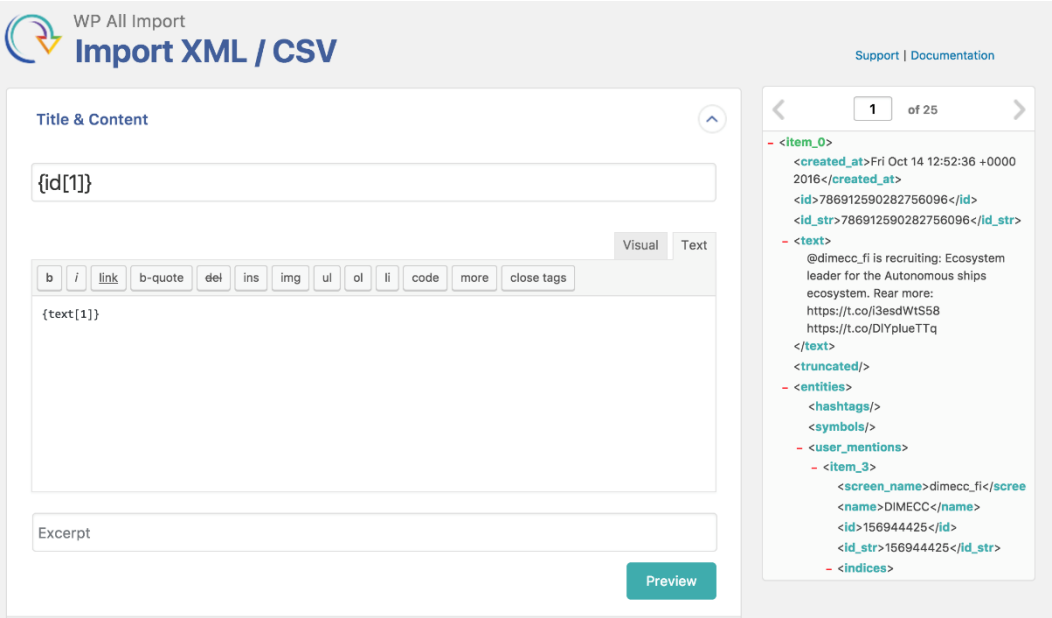
4.4 Sisällön tallentaminen WordPressiin

Sisällön tallentaminen päätettiin aiemmin tehdä erillisellä lisäosalla. WP All Import Pro:n avulla pystytään ajastamaan sisällönhaku tietystä palvelimen URL:sta. Kyseisen lisäosan avulla säästetään paljon aikaa koodaamiselta sekä helpotetaan sisällönhallintaa huomattavasti.

Tallentaminen aloitetaan kuvion 4 mukaisesti haettavan sisällön lähteen määrittelemisellä. Lähde voisi olla joko tiedosto, jo ennestään käytetty tiedosto tai tiedosto URL:n takana. Tässä tapauksessa käytetään luvussa 4.3 määritettyä URL:ia.

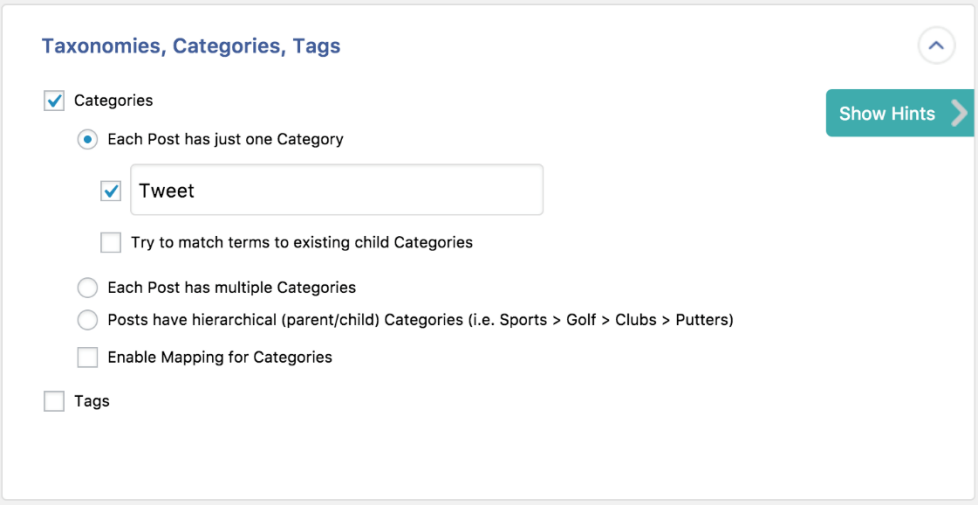
Kuvio 4. Twiitit artikkeleiksi: latauksen sijainti

Lähteen määrittelyn jälkeen asetetaan oikeat sisällöt oikeisiin paikkoihin. Kuviossa 5 määritetään artikkelin otsikko ja tuleva sisältö. Otsikoksi on valittu twiitin ID, jotta vältetään samojen otsikoiden käyttäminen useampaan kertaan. Sisältönä käytetään twiitin sisältöä.



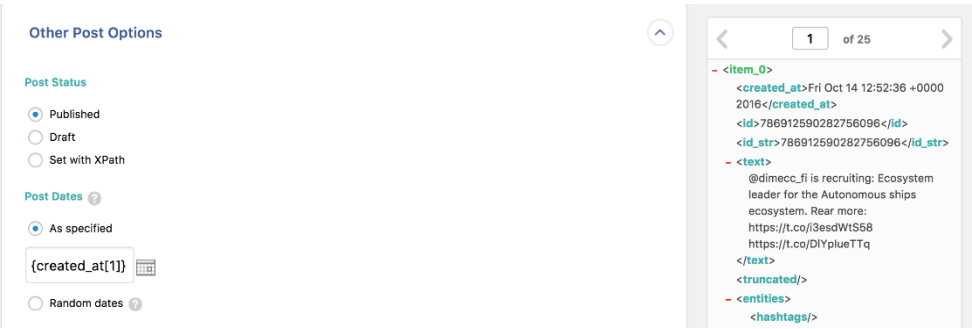
Kuvio 5. Twiitit artikkeleiksi: sisällön määrittäminen

Kuviossa 6 otsikon ja sisällön jälkeen asetetaan artikkelille kategoria, jota käytetään hyväksi artikkeleiden lajittelemiseen asiakkaan sivustolla.



Kuvio 6. Twiitit artikkeleiksi: kategorian määrittäminen

Luvussa 4.3 määritettiin twiitin käytettävä päivämäärä, joka asetetaan artikkelille kuviossa 7. Päivämäärä tulee asettaa kyseisellä tavalla, koska WordPress ei suoraan tue Twitterin päivämäärämuotoa, joten päivämäärä tulee ensin muokata oikeaan muotoon, jonka jälkeen muokattua muotoa voidaan käyttää hyväksi sisällönluonnissa.



Kuvio 7. Twiitit artikkeleiksi: päivämäärän määrittäminen

Lopuksi kuviossa 8 määritetään twiitin kustomoidut kentät, joilla ilmaistaan asiakkaan Twitter-käyttäjän nimi sekä tunnus, jotta loppukäyttäjä löytäisi asiakkaan tilin Twitteristä.

Kuvio 8. Twiitit artikkeleiksi: kustomoitujen kenttien asettaminen

5 Tulokset

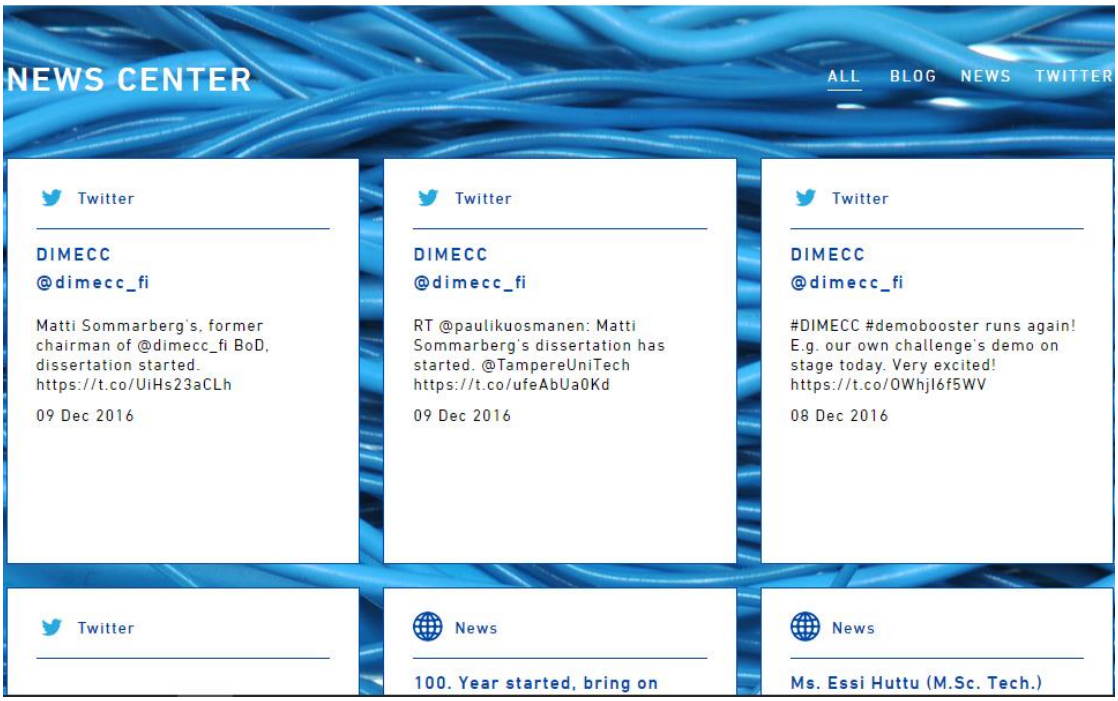
5.1 Tavoitteiden täyttyminen

Opinnäytetyön tuloksena saatiin toteutettua lisäosa, jonka avulla saadaan automaattisesti haettua asiakkaan twiitit heidän uuteen verkkosivustoon. Onnistuneen lisäosan ja karttuneen tietotaidon myötä voidaan todeta opinnäytetyön saavuttaneen asetetut tavoitteet.

Saavutettuun lopputulokseen oltiin sekä tilaajan että asiakkaan puolelta tyytyväisiä. Opinnäytetyön aikana opitut uudet tiedot ja taidot lisäosakehityksestä ovat arvokasta tietoa ammatillisen kehittymisen kannalta niin tekijälle kuin tilaajallekin. Opittuja tapoja pystytään hyödyntämään monipuolisesti jatkossa eteen tulevissa haasteissa.

Asiakas oli tyytyväinen toteutuksen toimintaan. He saivat haluamansa automaattisen sisällönhakijan, joka toimii monipuolisemmin kuin muut vastaavanlaiset toteutukset. Lisäosa vähentää heidän manuaalista työtään huomattavasti. Kuvioissa 8 ja 9 on nähtävissä toteutuksen lopputulos asiakkaan verkkosivulla.

Kuvio 9. Artikkeleiden haku kategorioittain



Kuvio 10. Artikkeleiden listaus päivämäärän mukaan

5.2 Jatkokehitys

Tuloksen pohjalta olisi mahdollista tehdä useamman palvelun integraatio samaan lisäosaan. Samalla tavalla pystyisi hakemaan esimerkiksi YouTube-kanavan videot osaksi verkkosivustoa.

Lisäosaa laajennettaessa sen sisältämät tunnistautumisavaimet tulisi siirtää hallintapaneelin puolelle. Kyseisellä tavalla lisäosasta voitaisiin tehdä versio, joka julkaistaisiin WordPressin omassa lisäosakansiossa. Kyseisessä tapauksessa pitäisi kiinnittää paljon huomiota lisäosan hallintapaneelin tietoturvallisuuteen, jotta syötettävät käyttäjäavaimet eivät joutuisi vääriin käsiin.

Lisäksi hallintapaneeliin voisi lisätä mahdollisuuden muokata yksittäisen julkaisun ulkoasua. Tämä tekisi lisäosasta monipuolisemman, eikä jokaisella sivustolla tarvitsisi käyttää samaa ulkoasua.

6 Pohdinta

Ennen opinnäytetyön aloittamista aiempaa kokemusta lisäosan tuottamisesta WordPress-sivustoille ei ollut. Joitain pienempiä toiminnallisuuksia on täytynyt toteuttaa, mutta puhtaan lisäosan tuottamiseen en ollut vielä päässyt käsiksi. Opinnäytetyö antoikin hyvän mahdollisuuden uuden asian oppimiseen sekä asiakkaan tarpeiden täyttämiseen. Aiemmat pienemmät toteutukset tarjosivat hyvän lähtökohdan työn toteuttamiselle.

Opinnäytetyössä käytetty REST-arkkitehtuuri ei ennestään ollut kovin tuttu. Tutkimuksen aikana tietotaito karttui arkkitehtuurin ja sitä hyödyntävien REST-rajapintojen tiedoilla ja hyvillä toimintatavoilla. Kyseisistä taidoista onkin jatkossa paljon hyötyä niin WordPressin kanssa toimiessa kuin muissakin ohjelmistoprojekteissa.

Myös WordPressin omat lisäosakäytänteet tulivat paremmin tutuiksi tutkimuksen aikana. Vaikka opinnäytetyöhöni toteutettu lisäosa on kooltaan hyvin pieni, tuli työn puitteissa tarve tarkastella lisäosakehitystä suuremmalla mittakaavalla. WordPressin

sisältämät rajapinnat tulivatkin tutuiksi, samaten erilaiset toiminta- ja suodatin-koukut sekä niiden toiminta. Kyseiset tekniikat tulevat olemaan isona apuna tulevaisuuden tehtävissä.

Lähteet

Fielding, Roy. 2000. Architectural Styles and the Design of Network-based Software Architectures. Viitattu 20.10.2016
http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf

Function Reference / add action. 2016 WordPress Codex, verkkosivusto. Viitattu 24.10.2016.
http://codex.wordpress.org/Function_Reference/add_action

Function Reference / add filter. 2016 WordPress Codex, verkkosivusto. Viitattu 24.10.2016.
https://codex.wordpress.org/Function_Reference/add_filter

Options API. 2016 WordPress Codex, verkkosivusto. Viitattu 20.10.2016.
http://codex.wordpress.org/Options_API

Plugin API. 2016 WordPress Codex, verkkosivusto. Viitattu 24.10.2016.
http://codex.wordpress.org/Plugin_API

Settings API. 2016 WordPress Codex, verkkosivusto. Viitattu 20.10.2016.
http://codex.wordpress.org/Settings_API

Usage of content management systems for websites. 2016 W3Techs, verkkosivusto. Viitattu 17.9.2016.
https://w3techs.com/technologies/overview/content_management/all

WordPress: Theme directory. 2016, verkkosivusto. Viitattu 17.9.2016.
https://wordpress.org/themes/browse/new/

WordPress: Plugin directory. 2016, verkkosivusto. Viitattu 17.9.2016
https://wordpress.org/plugins/

WordPress Codex: History. 2016, verkkosivusto. Viitattu 20.9.2016
https://codex.wordpress.org/History

WordPress Codex: WordPress APIs. 2016, verkkosivusto. Viitattu 20.9.2016
https://codex.wordpress.org/WordPress_APIs

WP REST API. 2016. Viitattu 25.10.2016.
https://wordpress.org/themes/browse/new/

Liitteet

Liite 1. Lisäosan lähdekoodi

```
add_action( ’rest_api_init’, function() {
   register_rest_route( ’twitter/’, ’list’, array(
      ’methods’ => ’GET’,
      ’callback’ => ’twitter_post_loader’
   ) );
}, 10 );
function twitter_post_loader() {
   $allTweets = [];
   $count = 25;
   $consumerkey = **;
   $consumersecret = **;
   $accesstoken = **;
   $accesstokensecret = **;
   $connection = new TwitterOAuth($consumerkey, $consumersecret, $ac-
cesstoken, $accesstokensecret);
   $twitdata_tag[0] = $connection->get('https://api.twitter.com/1.1/statu-
ses/user_timeline.json?screen_name=**&count='. $count);
   foreach($twitdata_tag as $tagdata) {
      foreach ($tagdata as $tweet) {
         if ($tweet->text != NULL) { // Ei tulosteta tyhjiä tweettejä
            $tweet->link = 'https://www.twitter.com/**/status/'. $tweet->id;
            if( ! isset($allTweets[$tweet->id]) ) $allT-weets[$tweet->id] = $tweet;
          $pt = date_parse_from_format( "D M d H:i:s O Y", $tweet-
>created_at );
           $date = $pt['year'] .'-'. $pt['month'] .'-'. $pt['day'] .'T'. $pt['hour'] .':'.
$pt['minute'] .':'. $pt['second'];
           $date = date_create($date, timezo-ne_open("Europe/Helsinki"));
           $parse = date_format( $date, 'd.m.Y' );
          $tweet->date = $parse;
        } <!—endif -->
     } <!—endforeach -->
  } <!—endforeach -->
  $allTweets = array_values($allTweets);
  return $allTweets;
} <!—end of function -->
```