MOBIILISOVELLUKSEN KEHITTÄMINEN WORDPRESS APIA HYÖDYNTÄEN

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulun opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Visamäki, syksy 2016

Ilona Huotilainen

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

TIIVISTELMÄ

VISAMÄKI
Tietojenkäsittelyn koulutusohjelma
eLearning ja multimedia

Tekijä Ilona Huotilainen **Vuosi** 2016

Työn nimi Mobiilisovelluksen kehittäminen WordPress APIa hyödyntäen

TIIVISTELMÄ

Työn toimeksiantaja oli hämeenlinnalainen yhdistys, joka ylläpitää löytöeläinkotia. Löytöeläinkotiin vastaanotetaan noin 300 eläintä vuosittain ja niiden tietoja on aiemmin ylläpidetty paperisena. Talteen otettujen eläinten tiedot halutaan julkaista kotisivuilla mahdollisimman nopeasti, jotta eläin löytäisi takaisin omaan kotiinsa. Myös kotia etsivistä eläimistä halutaan saada ihmisten näkyville kuvia ja esittelyteksti.

Työn tavoitteena oli helpottaa löytöeläinten ylläpitoa ja muokata WordPress-sisällönhallintajärjestelmällä luodut kotisivut toimeksiantajan tarpeiden mukaisiksi. Löytöeläinten tietojen lisääminen ja hallinta helpottuvat, kun käytössä on lisäksi mobiiliapplikaatio, joka on yhteydessä samaan tietokantaan kuin kotisivut. Näin saadaan lisättyä esimerkiksi talteen otettu eläin nopeasti sekä tietokantaan että kotisivuille julkisesti nähtäville.

WordPressin ohjelmointirajapintaa hyödyntäen saatiin mobiilisovelluksen käyttöön sivuston toiminnallisuudet. Tässä työssä tarvittiin tietokannan hallintaan luotuja funktioita. Kyselyt suoritettiin valmista WP REST API -lisäosaa käyttäen. Lisäksi tarvittavat muokkaukset kotisivuihin tehtiin sivustokohtaisella lisäosalla, jolla voidaan vaikuttaa sivuston toiminallisuuteen muuttamatta ohjelman ytimen rakennetta. Myös kirjoitettava lisäosa hyödynsi sisällönhallintajärjestelmän ohjelmointirajapintaa.

Mobiilisovelluksen kehittämisessä päädyttiin hybridiin sovellukseen, huolimatta siitä, että samat toiminnallisuudet oltaisiin voitu toteuttaa web-sovelluksena. Syynä oli halu saada käytännön kokemusta erilaisista mobiilisovelluksista. Ilmaisella Adobe PhoneGap -ohjelmalla luotiin hybridin sovelluksen pohja, ja sovellus pakattiin Android- ja iOS -käyttöjärjestelmille.

Työn tulos on WordPress-sisällönhallintajärjestelmällä luodut kotisivut, joihin on mahdollista lisätä omaa sisältötyyppiä sekä mobiilisovellus, jolla on yhteys kotisivupalvelimelle ja edelleen WordPressin ohjelmointirajapintaan, jolloin sovellus voi hakea ja päivittää tietoa.

Avainsanat WordPress API, WordPress lisäosat, Adobe PhoneGap, hybridi mobiilisovellus
Sivut 25 s.

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Visamäki
Degree Programme of Business Information Technology
eLearning ja multimedia

Author Ilona Huotilainen **Year** 2016

Subject of Bachelor's thesis Mobile Application development using WordPress API

ABSTRACT

The client of this thesis is a non-profit association which maintains an animal shelter. Annually the shelter receives almost 300 animals and information about these animals are maintained in paper. For missing animals to find back home as soon as possible, it is important to share information about them in public website. The client also wishes to tell in their website about the animals that are seeking a new home.

The purpose of this thesis is to ease maintaining information about the animals in the shelter and to modify WordPress based website to satisfy client's needs. With the mobile application, it is easier and faster to add animals and update news about them straight to the public website if so wanted. WordPress database is available for mobile application users.

With the WordPress application programming interface (API), it is possible for the mobile application to use the functions to run database queries, which are needed to get, update and post data to content management system. WP REST API -plugin enables these queries. Along with that WordPress site can be modified with site specific plugins, which allows to modify the site without making changes to system's core. Site specific plugins use WordPress API.

This mobile application was decided to develop as hybrid, even though the same functionalities could have been made in regular web application. This decision was made to increase knowledge about different mobile applications. The completed hybrid app is packaged with Adobe PhoneGap Build, and it is developed for Android- and iOS -systems.

Results of this thesis are website created with WordPress CMS, a possibility to use custom post types (animals in the shelter) and a mobile application that is authenticated by the web server. The mobile app can get, post and update data from the server.

Keywords WordPress API, WordPress plugins, hybrid mobile application, Adobe PhoneGap

Pages 25 p.

SISÄLLYS

1 JOHDANTO

Työn toimeksiantaja on hämeenlinnalainen löytöeläinkoti, jolla oli tarve yksinkertaiselle ratkaisulle löytöeläinten tietojen ylläpitoon. Tietojen ylläpidon digitalisoinnin ohessa nousi esille tarve uusista kotisivuista. Idea mobiilisovelluksen ja kotisivujen yhdistämisestä tuli yhdessä toimeksiantajan yhteyshenkilön, toisen työharjoittelijan sekä vapaaehtoisena toimivan viestinnän alan ammattilaisen kanssa, joka pääosin suunnitteli kotisivujen ulkoasun ja niiden sisällön. Tässä työssä ei käsitellä kotisivujen eikä niiden ulkoasun luomista, lukuun ottamatta WordPressin sivustokohtaisen lisäosan kirjoittamista sekä tietokannan, metadatan ja sisältötyyppien hyödyntämistä, joilla on oleellinen merkitys tämän mobiilisovelluksen kehityksessä.

Mobiilisovelluksen merkitys halutun tiedon lisäämisessä muuttui työtä tehdessä, kun kävi ilmi, että WordPress-sisällönhallintajärjestelmän ohjauspaneeli toimii myös mobiililaitteilla erinomaisesti, sekä laitteen kameraa ja tiedostoselainta voidaan käyttää tietoja lisätessä. Kuitenkin sovellus helpottaa tietojen ylläpitoa ja sillä voidaan suorittaa tarvittava autentikointi eli käyttäjäntodennus kotisivujen hallintapaneeliin ilman että käyttäjän tarvitsee itse kirjoittaa käyttäjätunnusta ja salasanaa. Sovelluksen käyttö on rajattu tietyille käyttäjille ja sitä ei jaeta julkiseen käyttöön.

Sovelluksen alustan luominen ja sen pakkaaminen suoritetaan PhoneGap-ohjelmalla, jolla myös alun alkaen oli tarkoitus mahdollistaa laitteen kameran käyttö valmiin lisäosan avulla. PhoneGapilla voidaan luoda hybridejä mobiilisovelluksia laitteistoriippumattomasti. Sovelluksen kaikki tarvittavat toiminnot olisi voitu toteuttaa web-sovelluksella, mutta nähtiin hyödylliseksi saada käytännön kokemusta hybridin sovelluksen luomisesta. Tässä työssä sovellus pakataan Android- ja iOS -käyttöjärjestelmille sopivaksi.

Tämän työn teoriaosuudessa käsitellään yleisesti WordPressin perustietokantarakennetta, lisäosia ja hierarkkista ryhmittelyä. Lisäksi kuvataan erilaisten mobiilisovellusten ominaisuuksia, jotta käy ilmi miksi tässä työssä päätettiin luoda hybridi mobiilisovellus. Työssä myös mainitaan, että käytettävä palvelin kotisivujen ylläpitoon on vuokrattu virtuaalitietokone, mutta se ei todellisuudessa vaikuta sisällönhallintajärjestelmän ylläpitoon juurikaan.

Käytännön osuudessa käydään läpi vaiheet, joilla saatiin muokattua WordPressin käyttöliittymää käyttäjille ja heidän tarpeilleen sopivammaksi ja yksinkertaisemmaksi. Mobiilisovelluksen kehittämisestä kerrotaan, kuinka voidaan hyödyntää WordPressin ohjelmointirajapintaa, jolla voidaan suorittaa esimerkiksi tietokantahakuja ja lähettää käskyjä hyvinkin yksinkertaisin menetelmin palvelimen ulkoisesta lähteestä, tässä tapauksessa mobiilisovelluksesta.

2 TOIMEKSIANTAJA JA TYÖN KUVAUS

Hämeenlinnalaiseen löytöeläinkotiin otetaan vastaan vuosittain lähes 300 eläintä. Tähän asti eläimistä on pidetty kirjaa ainoastaan paperisilla lomakkeilla. Sen lisäksi on tietoa jaettu julkisesti kotisivuilla ja blogissa tai yhteisöpalvelu Facebookissa, joihin on lisätty kuva löydetystä eläimestä, sen löytöpaikka ja talteenottopäivämäärä. Ongelmaksi on kuitenkin koitunut palvelujen käytön vaikeus ja se, että löytöeläinkodissa ei ole tietokonetta ja tietojen lisäys on monesti unohtunut. Myös eläimen tietojen katseleminen myöhemmin on ollut hankalaa ja toisinaan ihmiset ovat kyselleet jo kodin löytäneestä eläimestä, jonka kuva kuitenkin on edelleen ollut näkyvissä kotisivuilla. Osa eläimistä ei etsi aktiivisesti kotia, esimerkiksi sairauden tai oletettavien pitovaikeuksien vuoksi, nämä ovat ”kummeja”.

Viestintään ja löytöeläinkotia ylläpitävän yhdistyksen toimintaan keskittynyt työharjoittelija oli sitä mieltä, että kotisivujen teko ja niiden käyttö vastedes olisi olennainen osa löytöeläimistä tiedottamista. Ilmoituksia ei voida ainoastaan tehdä Facebookissa, koska kaikilla ihmisillä ei ole sinne pääsyä. Voidaan kuitenkin olettaa, että jokainen ihminen voi käydä katsomassa ilmoitukset kotisivuilta.

Käytännössä tämän työn tarkoituksena on päivittää löytöeläinkodin kotisivut nykyaikaisiksi, luoda sinne luokittelut löytöeläimille (erikseen talteen otetut, kummit ja uutta kotia etsivät), sekä omat sivut katoamisilmoituksille, tapahtumille ja uutisille, sekä kaikkien edellä mainittujen tietojen lisäämiselle lomakkeet. Tärkeänä osana on tehdä yksinkertainen käyttöliittymä, jota on mahdollista käyttää myös mobiililaitteilla.

Toimeksiantajalle vuokrattiin palvelin kotisivujen ylläpitoon Shellit.org-palveluntarjoajalta. Virtuaalipalvelimelle otetaan etäyhteys SSH-asiakasohjelmalla, Putty-emulaattorilla. SSH eli Secure Shell on salattuun tietoliikenteeseen tarkoitettu protokolla. Putty on avoimen lähdekoodin ohjelma, joka toimii eri käyttöjärjestelmäalustojen välillä. Työaseman ja palvelimen välisen yhteyden luomisessa käytetään porttia 22. Portti 22 on yleisesti varattu SSH-liikenteelle.

Palvelimen verkkonimi eli domain tilattiin Suomen viestintävirastolta. Domain määrittelee palvelimelle halutun nimen, tällöin sivuston haku ei tapahdu IP-numerosarjalla vaan tekstinä. 5. syyskuuta 2016 alkaen fi-verkkotunnusten rekisteröintiä ei enää tehdä viestintäviraston vaan erillisen verkkotunnusvälittäjän kautta. Viestintävirasto kuitenkin ylläpitää edelleen fi-verkkotunnusrekisteriä, ja niiden sivuilta voi hakea erikseen tunnuksen välittäjää. (Viestintävirasto.)

3 WORDPRESS-SISÄLLÖNHALLINTAJÄRJESTELMÄ

WordPress on ilmainen avoimen lähdekoodin sisällönhallintajärjestelmä, jonka rakenne perustuu PHP-skriptikieleen ja MySQL-tietokantaan. Sisällönhallintajärjestelmällä tarkoitetaan tietokoneohjelmaa, jonka avulla voidaan luoda ja hallita digitaalista sisältöä. Web-sivujen ylläpitoon tarkoitettujen sisällönhallintajärjestelmien huomattavin ominaisuus on erillinen

käyttöliittymä, jolla voidaan suorittaa front end -toimintoja graafisesti. Front endilla tarkoitetaan sellaisia toimintoja, jotka ovat sivuston tai sovelluksen käyttäjälle käytettävissä. Sisällönhallintajärjestelmä huolehtii tietokantayhteyksistä ja usein myös suurimmalta osin tietoturvasta.

WordPressin ensimmäinen versio julkaistiin maaliskuussa 2003. Vaikka alun alkaen WordPress tunnettiin lähinnä kevyenä blogialustana, sen käyttötarkoitus on sittemmin laajentunut huomattavasti. Se toimii kymmenien miljoonien sivustojen alustana. Uusin järjestelmän versio 4.7. tullaan julkaisemaan joulukuussa 2016. Yksi syy järjestelmän suosioon on varmasti sen helppokäyttöisyys ja maksuttomuus. Peruskäyttäjän ei tarvitse tuntea PHP-kieltä, eikä edes HTML-kieltä, sillä käyttäjä voi ladata valmiita lisäosia ja teemoja parantaakseen sivustonsa toimivuutta. Kaikki tehtävät sivujen luomisesta ja muokkaamisesta ulkoasun valintaan voidaan tehdä graafisella alustalla. Kuitenkin WordPressillä voidaan ylläpitää hyvinkin monimutkaisia ja raskaita, persoonallisia sivustoja. (WordPress.org.)

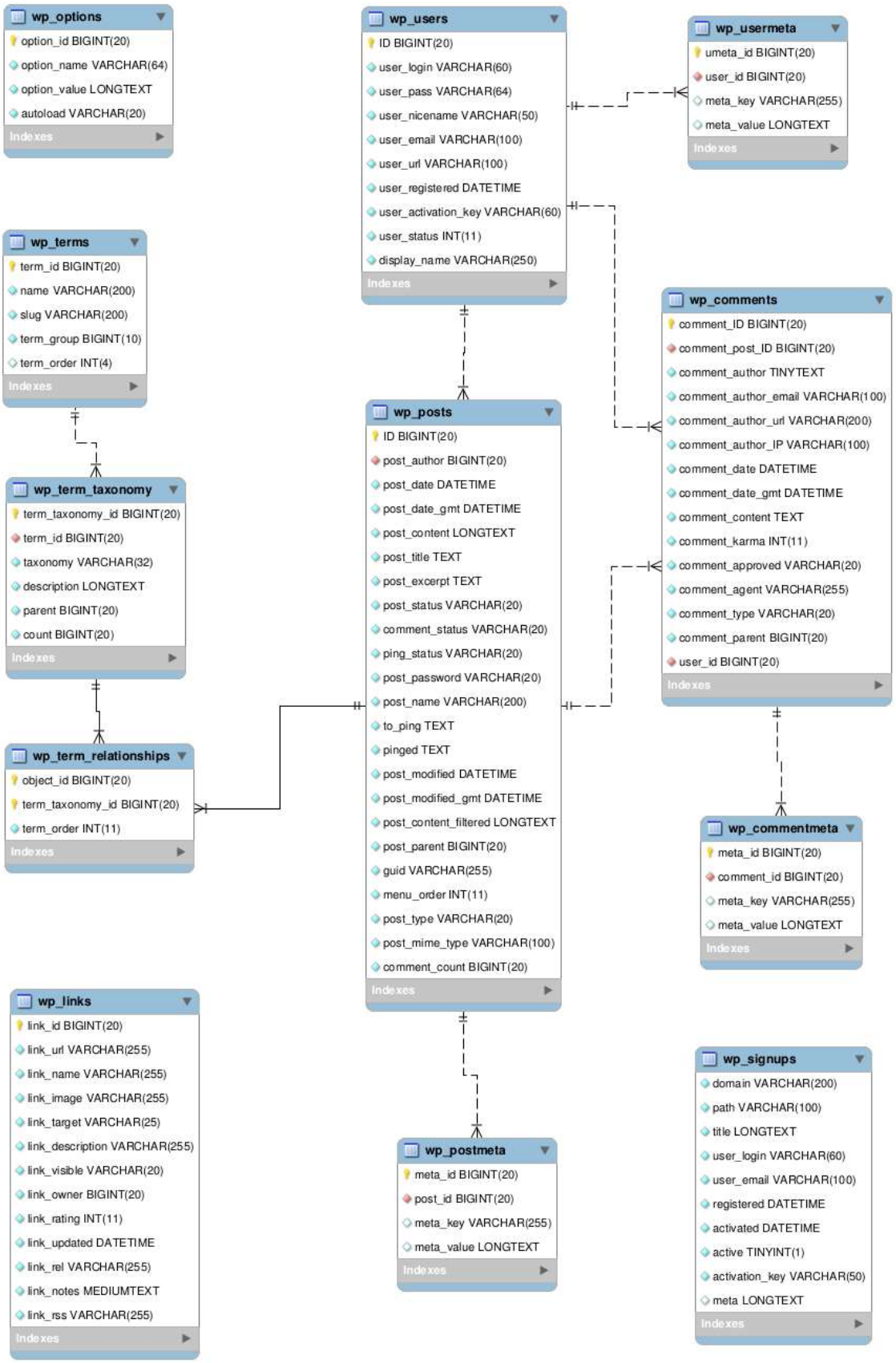
WordPress-sisällönhallintajärjestelmää voi käyttää millä tahansa tietokoneen käyttöjärjestelmällä. Vähimmäisohjelmistovaatimukset ovat PHP 5.6 ja MySQL 5.6 tai MariaDB 10.0. Wordpress.org-sivusto suosittelee web-palvelinohjelmistoksi Apachea tai Nginxia, kuitenkin mikä tahansa PHP:ta ja MySQL:aa tukeva ohjelmisto käy. (WordPress.org.)

WordPressin paikallinen asennus mahdollistaa lisäosien käytön mutta siitä on myös olemassa web-pohjainen hallintajärjestelmä. Se toimii osoitteessa www.wordpress.com, eikä vaadi asennuksia vaan ainoastaan rekisteröitymisen sivustolle. Tällöin järjestelmän käyttö on kuitenkin melko rajallista, ja sopii lähinnä erilaisille julkaisu- ja blogi-sivustoille. WWW-sisällönhallintajärjestelmällä luodulle sivustolle on myös mahdollista liittää oma olemassa oleva Domain-nimi eli laitteen yksilöivä verkkotunnus, mutta se vaatii maksullisen käyttäjätilin avaamista.

3.1 WordPressin tietokantarakenne, sisältötyypit ja metadata

WordPressin tietokanta vaatii toimiakseen vähintään MySQL:n version 5.6. tai MariaDB:n version 10.0. (WordPress.org.)

Sisällönhallintajärjestelmän asennusvaiheessa luodaan oma tietokanta WordPressille. WordPress luo tietokantaan automaattisesti tarvitsemansa taulut. (Kuva 1.)



Kuva 1. WordPress-sisällönhallintajärjestelmän tietokantarakenne (Codex.wordpress.org.)

Kuvassa 1 näkyvät WordPressin perusasennuksen yhteydessä luodut tietokannan taulut ja niiden relaatiot. Tässä taulujen relaatiot ovat yhden suhde moneen. Esimerkiksi wp_posts-tauluun lisätty rivi voi sisältää useita kommentteja wp_comments-taulusta, mutta yksi tietty ID-numerolla yksilöity kommentti voi olla vain yhdessä postauksessa. (Codex.wordpress.org.)

Wp_posts -taulu sisältää kaiken sivustolle lisätyn datan ja kaikki tietokannan rivit ovat yksilöity juoksevalla ID-numerolla. Posts-taulu on yhteydessä wp_users-tauluun, joka sisältää käyttäjätunnukset sekä yhdistää käyttäjät ja julkaisut toisiinsa.

Post-sanalla viitataan ”postaukseen” eli blogijulkaisuun. Nykyään WordPressissä voidaan luoda viittä eri tyyppistä postausta:

- Post eli ”postaus”, luotu sivu tai artikkeli
- Page kuten Post, mutta voidaan luoda omalla sivupohjalla, ja sivut voidaan järjestää hierarkkisesti
- Attachment eli liite, esimerkiksi kuva tai jokin muu tiedosto
- Revision eli luonnosversio postauksesta tai sivusta
- Navigation menu eli sivustolle valitun teeman navigaatiomenuun lisätty data.

Näitä kutsutaan ”post typeiksi”, ja ne kaikki tallennetaan tietokannan wp_posts-tauluun. Erottavana tekijänä on rivin sarake ”post_type”, joka kertoo, minkälaista dataa tullaan käsittelemään. Kuvaavampi sana post typelle olisi sisältötyyppi. Näille sisältötyypeille on jo valmiiksi olemassa linkit hallintapaneelin käyttöliittymässä, lomakkeet uusien lisäämiseen ja olemassa olevien muokkaukseen, yksittäisen postauksen oma tulostussivu sekä niin kutsuttu arkistosivu, johon listataan kaikki saman sisältötyypin postaukset. (Codex.wordpress.org.)

ID	post_title	post_name	post_type
11	INFO	info	page
12	Info	11-revision-v1	revision

Kuva 2. Wp-posts_taulun post-type -sarake

Kuvassa 2 näkyy tietokantaan lisätty Page -post typea eli sisältötyppiä oleva sivu ja sen revisio.

WordPressissä on myös mahdollista luoda ”Custom Post Typeja” eli omia sisältötyyppejä. Omien sisältötyyppien luomiseen on olemassa valmiita lisäosia, tai sen luomiseen käytettävät funktiot voi kirjoittaa itse. Luvussa 5 kerrotaan lisää oman lisäosan ja sisältötyyppien kirjoittamisesta.

WordPressin tietokantaan voidaan oletuksena ladata neljää erityyppistä dataa: postauksia, käyttäjiä, kommentteja sekä linkkejä, joille kaikille on myös tietokannassa omat taulunsa. Muuhun paitsi linkeiksi luokiteltavaan dataan

voidaan myös syöttää metadataa. Metadatalla tarkoitetaan tietoa, joka määrittelee ja kuvailee sitä dataa, johon se sisältyy, esimerkiksi blogipostauksen julkaisupäivämäärä. Kuvassa 1 näkyy wp_post_meta-taulu, jolla on relaatio wp_posts-tauluun, joka sisältää käytännössä kaikki sisällönhallintajärjestelmään ladatun tai luodun sisällön. Edelleen taulujen relaatio on yhden suhde moneen, eli yksi postaus voi olla yhdistetty useampaan metadata-taulun riviin. Tämä tarkoittaa sitä, että yhteen postaukseen voidaan lisätä useampi metadatakenttä. (Codex.wordpress.org.)

Metadata-taulu sisältää aina nämä neljä saraketta:

- meta_id, metadatan yksilöivä numerotunnus
- post_id, postauksen yksilöivä numerotunnus, tällä metadata yhdistetään haluttuun postaukseen
- meta_key, metadatan avain
- meta_value, metadatan arvo.

Metadataa hallitaan avain/arvo-pareina. Avain (key) on metadataelementin nimi ja sen arvo (value) näkyy siihen liitetyssä postauksessa.

Käytännössä tietokantaan lisättyyn riviin voidaan määritellä joitakin omia arvoja syöttämällä sille metadataa. Nämä arvot voivat poiketa WordPressin oletusarvoista. Esimerkiksi blogipostaukseen voidaan lisätä kenttä ”Paikkakunta”, johon käyttäjä voi valita arvoksi ”Hämeenlinna”. ”Paikkakunta” on metadatan avain ja ”Hämeenlinna” sen arvo. Tietokantahakuja voidaan tehdä metadatan perusteella.

WordPressissä on wpdb-niminen luokka, joka sisältää funktioita tietokannan hallintaan. Wpdb-luokan metodeja ei tule kutsua suoraan, vaan niitä tulisi käyttää $wpdb-objektin kautta. $wpdb on yhdistetty WordPressin tietokantaan, joten sitä voi käyttää suoraan.

Alla on yksinkertainen esimerkki tietokantakyselystä $wpdb-objektilla PHP-tiedostossa:

```
global $wpdb;
$hakutulos = $wpdb->get_results(SELECT * FROM  'wp_posts'
WHERE 'post_type' = 'page' );
```

3.2 WordPressin lisäosat

WordPressin lisäosalla eli pluginilla voidaan ohjata sisällönhallintajärjestelmän toimintaa ilman järjestelmän ytimen muokkausta. Lisäosat ovat pieniä PHP-kielellä kirjoitettuja ohjelmia, joilla voidaan muokata järjestelmän alkuperäistä toimintaa ja lisätä siihen ominaisuuksia. Valmiita lisäosia on jo tehty lähes jokaiseen mahdolliseen tarpeeseen, pienistä ulkoasumuutoksista kokonaisiin verkkokauppoihin. Ne voivat olla ilmaisia tai maksullisia. Julkaistuja lisäosia voi hakea WordPressin kotisivuilta. Lisäosia voi kirjoittaa myös itse, eikä sitä ole pakko jakaa julkiseen käyttöön. Tällaista lisäosaa kutsutaan sivustokohtaiseksi. (Codex.wordpress.org.)

On parempi vaihtoehto kirjoittaa sivustokohtaiset muutokset kokonaan omaan tiedostoon eli lisäosaan kuin suoraan järjestelmän functions.php-tiedostoon. Tällöin ne eivät ole sivustolle valitun teeman alaisia, ja esimerkiksi teemaa päivittäessä tehdyt muutokset eivät ole vaarassa kadota.

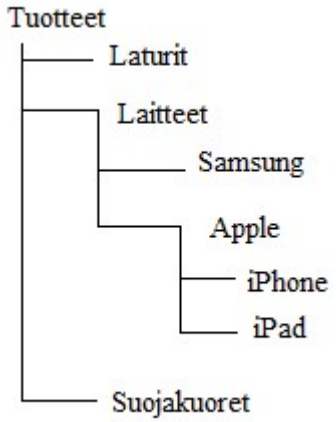
Oman lisäosan kirjoittaminen vaatii tuntemusta PHP-kielestä, tarvittaessa Ajaxista sekä Javascriptistä. Pääasiassa lisäosan funktiot on järkevää kirjoittaa PHP:llä, koska kyseessä on palvelinpuolen ohjelmointi. WordPress mahdollistaa kuitenkin myös Ajaxin ja Javascriptin käytön, esimerkiksi siihen tarkoitukseen tehtyjen lisäosien avulla. (Codex.wordpress.org.)

3.3 Hierarkkinen ryhmittely

Valmiiksi käytettäviä ryhmittelyjä WordPressissä ovat Category sekä Tags. Categoryyn voidaan luoda omia termejä, johon postaukset voidaan ryhmitellä. Se ei kuitenkaan ole hierarkkinen. Tageja eli asiasanoja voidaan lisätä yhteen julkaisuun useita. Asiasanoista tehdään linkki, jota klikkaamalla voidaan nähdä kaikki sillä merkityt julkaisut.

Järjestelmään voidaan kuitenkin luoda myös omia kategorioita, joilla voidaan luokitella ja ryhmitellä sisältötyyppejä, myös itse luotuja. Itse luodun kategorian voi tehdä hierarkkiseksi ja silloin sitä kutsutaan taksonomiaksi.

Hyvä esimerkki itse luodusta taksonomiasta ja sisältötyypistä on verkkokauppa. Aluksi luodaan verkkokaupalle oma ryhmittely, joka halutaan hierarkkiseksi, kuten kuvassa 3.



Kuva 3. Esimerkki hierarkkisesta kategoriasta

WordPressissä kuvan 3 mukainen ryhmittely voidaan toteuttaa kustomoidulla taksonomialla. Luodaan uusi ryhmittely ja nimetään se ”Tuotteet”. Sille on luotu alaryhmä ”Laturit”, ”Laitteet” ja ”Suojakuoret”. Laitteille on vielä luotu alaryhmät ”Samsung” ja ”Apple” ja niin edelleen. Kun järjestelmään lisätään tuotteita, ne halutaan luoda omalla sisältötyypillä ”tuotteet”. Kun tuotteita lisätään järjestelmään itse luodulla lomakkeella, ne voidaan asettaa haluttuihin kategorioihin. Tämä helpottaa huomattavasti sisällön selaamista ja lisättyjen tuotteiden hallintaa. Kustomoidun taksonomian tekemiseen on olemassa valmiita lisäosia tai sen voi kirjoittaa itse.

4 MOBIILISOVELLUS

Mobiilisovellus on ohjelmisto, joka on suunniteltu toimimaan erilaisilla mobiililaitteilla, kuten älypuhelimilla ja tablettitietokoneilla.

Mobiilisovelluksia on kolmea erilaista. Natiivi, web-sovellus sekä hybridisovellus. Natiivilla sovelluksella tarkoitetaan ohjelmaa, joka on kirjoitettu tietylle mobiilialustalle tai laitteelle käyttäen kyseisen alustan tukemaa ohjelmointikieltä. Näitä alustoja ovat esimerkiksi Android ja iOS, jotka ovat yleisimmät mobiililaitteiden käyttöjärjestelmät. Android-sovelluksen voi kirjoittaa käyttäen Java-ohjelmointikieltä, ja iOS-alustalle Swift-kielellä. Natiivi sovellus asennetaan suoraan laitteeseen, ja sovelluksen kehittäjän tulee kirjoittaa jokaiselle alustalle sopiva ohjelma. Tämäntyyppinen sovellus mahdollistaa mobiililaitteen laitteiston ja muiden siihen asennettujen sovellusten käytön, kuten kameran, GPS-paikannuksen ja tiedostoselaimen. Laitteen ei tarvitse välttämättä olla yhteydessä Internetiin toimiakseen. (Nngroup.com.)

Sovelluksen kirjoittamiseen käytettäviä kehitysalustoja on paljon eritasoisia ja -hintaisia.

4.1 Hybridi ja web-sovellus

Hybridi sovellus yhdistää sekä natiivin sovelluksen että web-sovelluksen ominaisuudet. Käytännössä se on HTML5-sovellus, joka on luotu natiivin sovelluksen sisään. Hybridi sovellus mahdollistaa laitteen kameran, tiedostoselaimen ja muiden paikallisten ominaisuuksien käytön lisäksi esimerkiksi yhteyden muihin webpalveluihin. Hybridi sovellus voi haluttujen toimintojen puitteissa toimia ilman yhteyttä Internetiin. Hybridi sovellus asennetaan laitteeseen. (Nngroup.com.)

Web-mobiilisovellus toimii jokaisella käyttöjärjestelmäalustalla. Sitä ei asenneta suoraan laitteeseen, vaan se toimii laitteen selaimen kautta, joten laitteen tulee olla yhteydessä Internetiin. Web-mobiilisovellus on kirjoitettu käyttäen web-ohjelmoinnin työkaluja, HTML-kielen lisäksi esimerkiksi Javascriptiä ja Ajaxia. (Nngroup.com.)

4.2 Adobe PhoneGap

Nykyisin Adoben omistama PhoneGap on yksi suosituimmista avoimen lähdekoodin laitteistoriippumattomista kehitysalustoista hybridien ja web-mobiilisovellusten kirjoittamiseen. Siitä on olemassa komentorivipohjainen versio sekä työpöytäsovellus. Tällä hetkellä PhoneGapin työpöytäsovellus on vielä kehitysvaiheessa, mutta kuitenkin käytettävissä Mac ja Windows -käyttöjärjestelmille. Sen voi ladata osoitteesta http://phonegap.com. Komentorivipohjaisen alustan (CLI) voi asentaa käyttäen esimerkiksi npm-paketinhallintaohjelmaa. (PhoneGap.com.)

PhoneGapin Developer-sovelluksen voi asentaa mille tahansa mobiililaitteelle, ja sillä voidaan testata kehitettävää sovellusta. Se näyttää kirjoitetta-

vaan sovellukseen tehtävät muutokset nopeasti ja helposti eri laitteilla. Developer-ohjelman käyttö vaatii ainoastaan, että yhdistettävät laitteet (työasema ja mobiililaite) ovat samassa verkossa. Kehitettävää sovellusta voi myös testata Chrome-selaimelle asennettavan emulaattori-lisäosan kanssa. Lisäosan voi ladata osoitteesta http://emulate.phonegap.com. (PhoneGap.com.)

PhoneGapilla voidaan luoda laitteistoriippumattomia sovelluksia käyttäen web-kehitystyökaluja HTML, CSS ja Javascript- kieliä, ja hybridillä sovelluksella saadaan myös käyttöön laitteiston ominaisuuksia. PhoneGapilla voidaan ladata valmiita lisäosia mobiilisovellukseen, joilla mahdollistavat esimerkiksi kameran käytön. (PhoneGap.com.)

4.3 jQuery Mobile

jQuery Mobile on tekniikka mobiilisovellusten kehittämiseen. Se perustuu tavallisiin web-kehityksessä käytettäviin menetelmiin, kuten HTML5- ja CSS-kieliin. Se ei siis ole oma kielensä, vaan helpotettu tapa kirjoittaa jo olemassa olevia menetelmiä käyttäen, toisin sanoen se on yksi Javascript-kirjasto. jQuery Mobilen käyttö vaatii lisäksi osaamista HTML-, CSS- ja jQuery-tekniikoista. jQuery Mobile -kirjaston saa käyttöön linkittämällä sen kirjaston osoitteen käytettävän dokumentin head-tagiin. Kirjaston voi myös asentaa paikallisesti sovelluksen sisään.

Alla olevassa esimerkissä näkyy html-tiedoston head-tagiin linkitetyt jQuery-kirjastot:

```
<head>
<!-- jQuery tyylitiedosto -->
<link rel="stylesheet" href="https://code.jquery.com/mo-
bile/1.4.5/jquery.mobile-1.4.5.min.css">
<!-- jQuery kirjasto -->
<script src="https://code.jquery.com/jquery-1.11.3.min.js">
</script>
<!-- jQuery Mobile kirjasto -->
<script src="https://code.jquery.com/mobile/1.4.5/jquery.mo-
bile-1.4.5.min.js"> </script>
</head>
```

Suurin osa suosituimpien mobiililaitteiden alustoista tukee jQuery Mobilea, esimerkiksi iOS, Android, Windows Phone sekä Symbia.

5 WORDPRESS-KOTISIVUJEN MUOKKAUS LISÄOSALLA

Koska toimeksiantajalle tehtiin arkistojen digitalisoinnin lisäksi uudet kotisivut, päätettiin että on vaivattominta käyttää yhteistä tietokantaa niiden hallintaan. Toimeksiantajan toiveena oli käyttää ilmaisia ohjelmia ja niiden hallinnan tulisi toimia pääosin suomen kielellä. WordPress-sisällönhallintajärjestelmään päädyttiin muun muassa sen suuren suosion ja muokattavuuden vuoksi.

Toimeksiantajan tärkeimmät vaatimukset ilmaisten ohjelmien käytön lisäksi olivat käyttöliittymän yksinkertaisuus ja selkeys, sekä se, että niiden tulisi toimia mobiililaitteilla. Mobiililaitteella tulisi myös olla käytettävissä laitteen kamera, jotta saadaan liitettyä kuva talteen otetusta löytöeläimestä.

Kaikki WordPressiin tehtävät muutokset tehdään ensiksi paikalliselle kopiolle, jonka jälkeen ne voidaan lisätä palvelimelle asennettuun oikeaan versioon.

5.1 WordPressin lisäosan kirjoittaminen

Koska työssä on tärkeänä osana yksinkertaisen käyttöliittymän teko ja tietojen lisääminen helposti sekä tietokantaan että kotisivuille julkisesti nähtäville, päätettiin tehdä kokonaan oma lisäosa tietojen lisäämistä varten. Samalla lisäosalla voidaan muokata sisällönhallintajärjestelmän hallintasivujen ulkoasua selkeämmäksi ja luoda jokaiselle tietotyypille oma lisäys- ja muokkauslomake.

Oman lisäosan kirjoittamisen voi aloittaa luomalla palvelimen hakemistoon /wordpress/wp-content/plugins/ uuden kansion ja sinne PHP-tiedoston. PHP-tiedoston nimi ei ole lisäosan nimi. Tiedoston alkuun lisätään kommenteilla tiedot lisäosasta:

```
<? php

/*
Plugin name: Oma Lisäosa
Description: Sivustokohtaiset muutokset kotisivuille

*/
?>
```

5.2 Oman sisältötyypin ja taksonomian luonti

Aluksi tarkoituksena oli tehdä omat taulut tietokantaan, koska WordPress mahdollistaa myös niiden käytön sivustolla. Kuitenkin yksinkertaisemmaksi vaihtoehdoksi osoittautui luoda oma sisältötyyppi järjestelmään.

Oman sisältötyypin ryhmittely tehtiin luomalla taksonomia omaan sivustokohtaiseen lisäosaan:

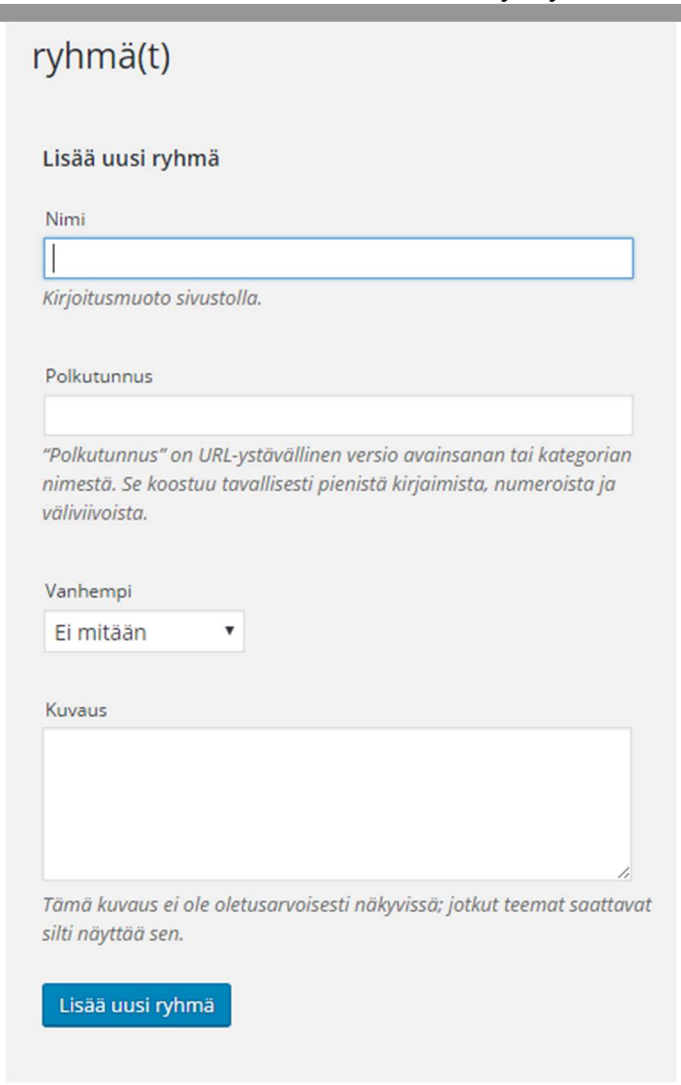
```
add_action('init', 'create_topics', 0 );

function luo_taxonomia() {
```

```
//Käytöliittymä
 $labels = array(
    'name' => _x( 'ryhmä(t)', 'taxonomy general name' ),
    'singular_name' => __( 'Ryhmä', 'taxonomy singular name'
),
    'search_items' =>  __( 'Etsi ryhmää' ),
    'all_items' => __( 'Kaikki ryhmät' ),
    'parent_item' => __( 'Ryhmän Parent ' ),
    'parent_item_colon' => __( 'Parent:' ),
    'edit_item' => __( 'Muokkaa ryhmittelyä' ),
    'update_item' => __( 'Päivitä ryhmittely' ),
    'add_new_item' => __( 'Lisää uusi ryhmä'),
    'new_item_name' => __( 'Uuden ryhmän nimi'),
    'menu_name' => __( 'Löytöeläinten ryhmittely' ),
  );

//Ryhmien rekisteröinti

  register_taxonomy('taxonomia',array('post'), array(
    'hierarchical' => true,
    'labels' => $labels,
    'show_ui' => true,
    'show_admin_column' => true,
    'query_var' => true,
 'show_in_rest' =>true,
    'rewrite' => array( 'slug' => 'ryhma' )

  ));

}
```

Koodiesimerkki 1. Oman ryhmittelyn luonti PHP:llä (Wpbeginner.com.)

Funktiolla add_action liitetään create-topics-funktio WordPressin init-tapahtumaan. Käytännössä tällä tarkoitetaan sitä, että funktio suoritetaan, kun järjestelmä on latautunut mutta ennen kuin sivuston headerit on lähetetty. Init-tapahtuman sijaan voi myös käyttää wp_loaded-funktiota, jolloin kutsuttava funktio suoritetaan, kun käyttäjälle on latautunut WordPress, kaikki lisäosat sekä teema ja sen asetukset. Tässä tapauksessa on käytetty init-funktiota, koska taksonomialle on luotu myös käyttöliittymä. Lisäksi register_taxonomy-funktion käytössä saattaa ilmetä ongelmia, jollei sitä kutsuta init-tapahtumassa. (Wpbeginner.com.)

Funktio create_topics() suorittaa varsinaisen taksonomian luonnin. Ensin luodaan käyttöliittymä, jolla voidaan lisätä ja muokata taksonomian termejä. Tässä luotu graafinen käyttöliittymä ryhmien luomiseen näkyy kuvassa 4.



Kuva 4. Kustomoidun taksonomian lisäyslomake

Register_taxonomy-funktiossa määritellään ryhmittelyn asetukset:

register_taxonomy($taxonomy, $object_type, $args);
$taxonomy – taxonomian nimi
$object_type – sisältötyyppi jota käytetään ryhmittelyssä
$args – taulukko tietokantaan syötettävistä parametreista.

Ryhmittelyn nimeksi on annettu ”taxonomia” ja se on yhdistetty post-sisältötyyppeihin. Sen voisi esimerkiksi yhdistää users-sisältötyyppiin post-tyypin sijaan. Taulukossa määritellään tietokannan sarakkeisiin asetukset. Se toimii hierarkkisesti ja käyttöliittymä on määritelty näkyväksi. Tiedot tallentuvat tietokannan terms-tauluihin. ”Show_in_rest”-arvo tulee määrittää ”trueksi”, jotta tätä taksonomiaa voidaan käyttää WP REST API -lisäosalla (Luku 6.2).

Oman sisältötyypin luominen tapahtuu samalla logiikalla kuin taksonomian. Ensin luodaan käyttöliittymä. Yhtenä parametrina taulukkoon on annettu ”ultra”, joka on sivustolle asetetun teeman nimi. Käyttöliittymään haetaan suoraan teemassa määritellyt tyyliasetukset. Tämä toimii useimpien

teemojen kanssa. Jotkut teemat myös suorittavat suoraan kielen käännöksen.

Tietokantaan tallennettavat tiedot voi syöttää taulukkona:

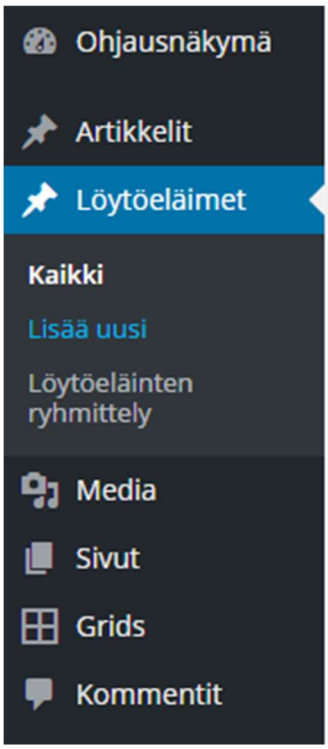
```
add_action( 'init', 'oma_post_type', 0 );

function oma_post_type() {

// Käyttöliittymä
    $labels = array(
'name'                => _x( 'Löytöeläimet', 'Post Type General Name', 'ultra' ),
'singular_name'       => _x( 'Löytöeläin', 'Post Type Singular Name', 'ultra' ),
'menu_name'           => __( 'Löytöeläimet', 'ultra' ),
'parent_item_colon'   => __( 'Parent', 'ultra' ),
'all_items'           => __( 'Kaikki', 'ultra' ),
'view_item'           => __( 'Näytä', 'ultra' ),
'add_new_item'        => __( 'Lisää uusi', 'ultra' ),
'add_new'             => __( 'Lisää uusi', 'ultra' ),
'edit_item'           => __( 'Muokkaa tietoja', 'ultra' ),
'update_item'         => __( 'Päivitä', 'ultra' ),
'search_items'        => __( 'Etsi', 'ultra' ),
'not_found'           => __( 'Ei löytynyt', 'ultra' ),
'not_found_in_trash'  => __( 'Ei löytynyt roskakorista', 'ultra' ),
    );
    $args = array(
        'label'                 => __( 'löytöeläimet', 'ultra' ),
        'description'           => __( 'löytöeläimet', 'ultra' ),
        'labels'                => $labels,
        // Mitka sarakkeet sallittuja editorissa
        'supports'              => array( 'title', 'editor', 'author', 'thumbnail', 'comments', 'revisions', 'custom-fields', ),

        'rewrite' => array("slug" => "löytöeläimet"),

        // Linkitetaan aiemmin luotuun custom taxonomyyn
        'taxonomies'            => array( 'taxonomia' ),

        'hierarchical'          => false,
        'public'                => true,
        'show_ui'               => true,
        'show_in_menu'          => true,
        'show_in_nav_menus'     => true,
        'show_in_admin_bar'     => true,
        'menu_position'         => 5,
        'can_export'            => true,
        'has_archive'           => true,
        'exclude_from_search'   => false,
        'publicly_queryable'    => true,
        'show_in_rest'          => true,
        'capability_type'       => 'page',

    );

    // Rekisteröinti
    register_post_type( 'löytöeläimet', $args );
```

```
}
```

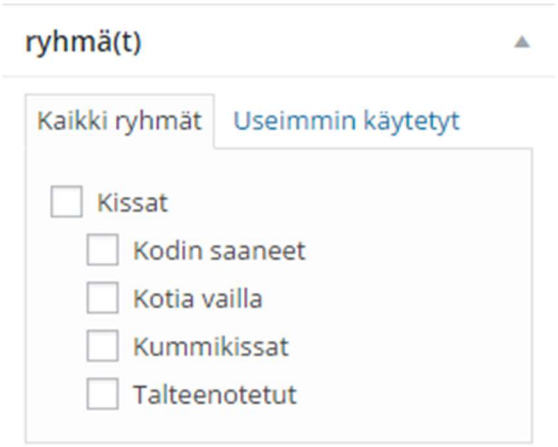
Koodiesimerkki 2. Oman sisältötyypin rekisteröinti (Wpbeginner.com.)

Funktiossa myös liitetään luotu sisältötyyppi aiemmin luotuun ryhmittelyyn. Sisältötyypin rekisteröinti tehdään register_post_type()-funktiolla. Myös oma sisältötyyppi halutaan käytettäväksi WP REST API -lisäosalla (Luku 6.2).

Kuvassa 5 näkyy luodun sisältötyypin linkki WordPressin ohjausnäkymän paneelissa ja kuvassa 6 ryhmittelyn näkymä lisäyslomakkeessa.



Kuva 5. Luotu sisältötyyppi ohjausnäkymässä



Kuva 6. Hierarkkisen ryhmittelyn näkymä lisäyslomakkeessa

Alla on esimerkki oman sisältötyypin ja tiettyyn taksonomian ryhmään lisättyjen haku tietokannasta ja niiden tulostus HTML-tiedostoon PHP:llä:

```
<?php
 query_posts( array(
 'post_type' => 'löytöeläimet',
 'taxonomia' => 'talteenotetut',
 'posts_per_page' => -1,
 'post_status' => 'publish',
 'orderby' => 'title',
 'order' => 'asc'
) );
if ( have_posts() ) :
while ( have_posts() ) : the_post();

?>

<div class="tulostus">

    <h3 class="talteenotettu_title">
        <a href="<?php the_permalink(); ?>" title="<?php
the_title_attribute(); ?>"><?php the_title(); ?></a>
    </h3>

    <a href="<?php the_permalink(); ?>" class="talteen-
otettu_kuva"><?php the_post_thumbnail('medium'); ?></a>

</div>
```

Työssä tehtiin myös oma sisältötyyppi tapahtumille ja uutisille.

5.3 Metadatan lisääminen sisältötyypille

Luodulle sisältötyypille halutaan lisätä metadataa, kuten löytöeläimen löytöpaikka ja saapumispäivämäärä. Lisättävän metadatan lomakkeet ja tallennusfunktiot kirjoitetaan lisäosan funktiot.php-tiedostoon.

Metadatan lisäämisessä otetaan huomioon tarve yksinkertaiselle käyttöliittymälle, joten aluksi luodaan halutun sisältötyypin lisäyslomakkeelle ”metabox”, johon voidaan laittaa muun muassa otsikko ja tekstikenttä. Metabox tulostetaan halutun sisältötyypin lisäyslomakkeessa, sen voi määritellä kuten tavallisessa HTML-lomakkeessa esimerkiksi teksti

Metaboxit lisätään add_meta_box()-funktiolla:

add_meta_box(string $id, string $title, callable $callback, string|array|WP_Screen $screen = null, string $context = 'advanced', string $priority = 'default', array $callback_args = null)
(Codex.wordpress.org.)

Selvyyden vuoksi add_meta_box-funktio suoritetaan toisen funktion sisällä, kun halutaan luoda kaksi metadatalomaketta yhdessä:

```
function add_metaboxes() {
    add_meta_box('loytopaikka', 'Eläimen löytöpaikka', 'lo-
ytopaikka', 'löytöeläimet', 'normal', 'default');
```

```
    add_meta_box('saapumispvm', 'Saapumispäivä', 'saapu-
mispvm', 'löytöeläimet', 'normal', 'default');

}
```

Käytettävät funktiot sisältötyypin metadatan hallintaan:

- add_post_meta($post_id, $meta_key, $meta_value, $unique);
- update_post_meta($post_id, $meta_key, $meta_value, $prev_value);
- delete_post_meta($post_id, $meta_key, $meta_value);
- get_post_meta($post_id, $key, $single);

$post_id (int) – postauksen yksilöivä id-numero.
$meta_key – metadatan avain.
$meta_value – metadatan arvo.
$unique (bool) – Ei pakollinen arvo. Määritetäänkö data yksilölliseksi tietokantaan. Oletusarvo ”false”.
$prev_value (string)– Ei pakollinen arvo. Kentän vanha arvo, joka halutaan päivittää. Oletuksena tyhjä.
$key – Ei pakollinen arvo. Metadatan avain (metakey), oletusarvo ” ”.
$single (bool) – Ei pakollinen arvo. Oletusarvo ”false”. Jos arvona on ”false”, funktio palauttaa datan taulukkona, mikäli ”true”, voidaan palauttaa yksi haluttu arvo.

Alla näkyvä loytopaikka-funktio on esimerkki metadatalomakkeen tulostuksessa oman sisältötyypin lisäyslomakkeessa:

```
function loytopaikka() {

    global $post;

    // Tulostetaan lomakkeeseen metadata jos jo tallennettu
    $loytopaikka = get_post_meta($post->ID, '_loytopaikka',
true);

    // Tulostetaan lomake
    echo '<input type="text" name="_mista" value="' . $lo-
ytopaikka  . '" class="widefat" />';

}
```

Koodiesimerkki 3. Metadatalomake lisäys- ja muokkauslomakkeessa (WpTheming.com.)

Alla olevalla loopilla tallennetaan syötetty metadata:

```
    $elaimet_meta['_loytopaikka'] = $_POST['_loytopaikka'];
    $elaimet_meta['_saapumispvm'] = $_POST['_saapumispvm'];


    foreach ($elaimet_meta as $key => $value) {

        if( $post->post_type == 'revision' ) return;
// Tallentaan vain kerran

        $value = implode(',', (array)$value);
```

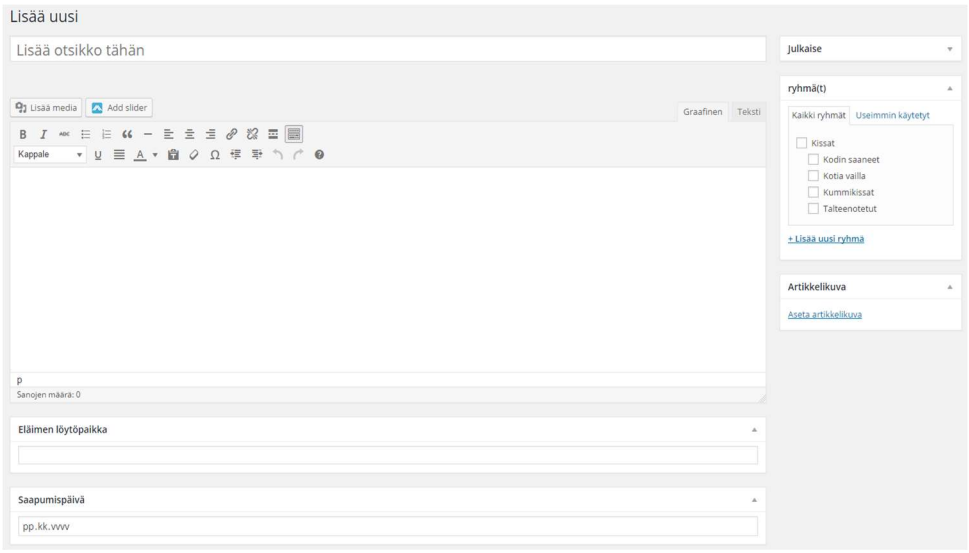
```
        if(get_post_meta($post->ID, $key, FALSE)) {
// Jos on jo syötetty, päivitetään data
            update_post_meta($post->ID, $key, $value);

        }
        // ...tai lisätään uusi
        else {
            add_post_meta($post->ID, $key, $value);
        }

        if(!$value) delete_post_meta($post->ID, $key);
// Poistetaan arvo jos kenttä tyhjä
    }
```

Koodiesimerkki 4. Metadatan käsittely (WpTheming.com.)

Kuvassa 7 näkyy metadatalomakkeet artikkelin lisäyslomakkeessa.

Kuva 7. Luodut metaboxit sivustolle

Kuvassa 8 on kokonaisnäkymä Löytöeläimet-sisältötyypin lisäyslomakkeesta.

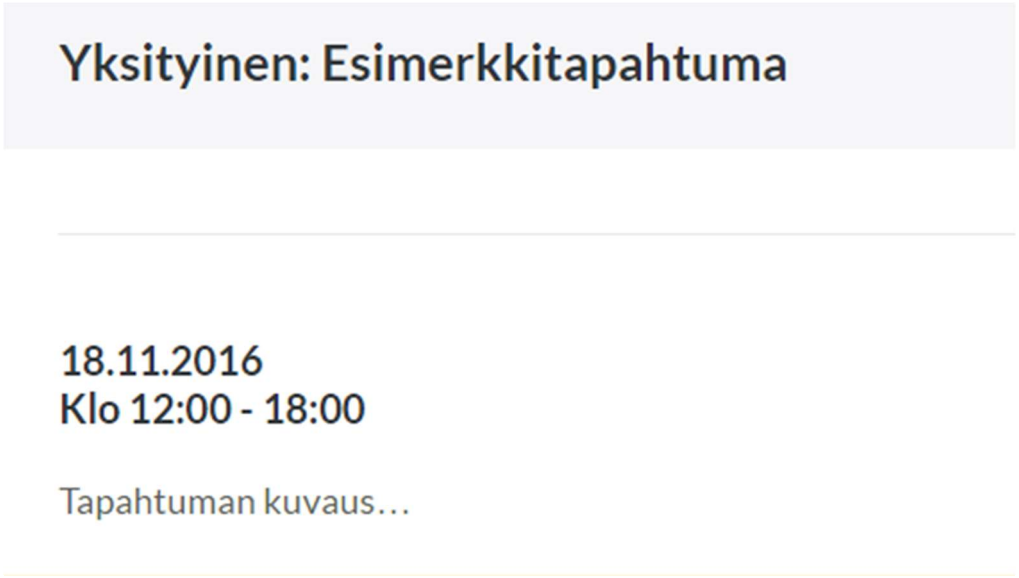


Kuva 8. Kokonaiskuva oman sisältötyypin lisäyslomakkeesta. Oikealla näkyy siihen liitetty taksonomia.

Esimerkiksi Tapahtumat-sisältötyyppien metadatan tulostus yksittäisessä julkaisussa:

```
<?php
$pvm = get_post_meta($post->ID, "_tapahtumatpvm", true);
       $alkaaKlo = get_post_meta($post->ID, "_tapahtumat-
klo1", true);
       $loppuuKlo = get_post_meta($post->ID, "_tapahtumat-
klo2", true);

echo " " . date("d.m.Y", strtotime($pvm)) . "<br>";
echo "Klo " . $alkaaKlo . " - " . $loppuuKlo;
?>
```

Tapahtumat-sisältötyypille syötettyä metadataa ovat päivämäärä sekä alku- ja loppumiskellonajat, jotka halutaan kiinteästi tulostaa jokaiseen saman sisältötyypin julkaisuun. Näin toteutettu julkaisu näyttäisi kuten kuvassa 9.



Kuva 9. Esimerkki sisältötyypin kiinteästä tulostusasettelusta

6 MOBIILISOVELLUKSEN KEHITTÄMINEN WORDPRESSIN OHJELMOINTIRAJAPINTAA HYÖDYNTÄEN

Toimeksiantajan toiveen mukaisesti kehitettävän mobiilisovelluksen tulee mahdollistaa löytöeläinten tietojen ylläpito. Tarkoituksena on helpottaa tietojen lisäämistä tietokantaan ja siitä edelleen kotisivuille julkiseksi. Suunnitelmana oli luoda hybridi mobiilisovellus, jotta myös kameran käyttö olisi mahdollista. Olennaisena ominaisuutena pidetään talteen otetun tai kotia etsivän eläimen kuvan julkaisua kotisivuilla, ja sen lisäksi kuvan ja tietojen tallentamista tietokantaan, mikäli niitä pitäisi muokata vielä myöhemmin.

Työtä tehdessä kävi kuitenkin ilmi, että WordPress-sisällönhallintajärjestelmän pääkäyttäjän hallintasivu toimii mobiililaitteilla ja mahdollistaa laitteen tiedostoselaimen ja kameran käytön uutta tietoa lisättäessä. Työn si-

sältö muuttui siten, että kehitetään sovellus WordPressin luoman tietokannan käyttöön. Sovelluksen pyynnöt palvelimelle suoritetaan Ajaxilla ja tiedot palautetaan palvelimelta käyttäen JSONia.

PhoneGapilla on erittäin helppoa ottaa natiivin sovelluksen ominaisuuksia käyttöön, lataamalla valmiita lisäosia. Lisäosia löytyy sivulta http://cordova.apache.org/plugins/.

6.1 Sovelluksen alustan luominen PhoneGapilla

Tässä työssä käytettiin PhoneGapin komentorivipohjaista käyttöliittymää ja sovelluksen testaukseen mobiililaitteilla PhoneGap Developeria, jonka voi asentaa laitteen sovelluskaupasta. jQuery Mobilea käytetään sovelluksen ulkoasun ja toimintojen kehittämiseen. Tekstieditorina Notepad++.

Komennolla *phonegap template list* tulostetaan lista alustoista, joilla sovellus voidaan luoda. (Kuva 10.)

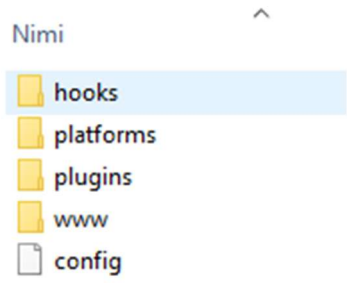
```
blank                          A blank and empty PhoneGap app.
hello-cordova                  Default hello world app for Cordova.
hello-world                    Default hello world app for PhoneGap.
jquery-mobile-starter          Starter PhoneGap project using jQuery Mobile.
phonegap-template-framework7   Starter PhoneGap project for Framework7.
```

Kuva 10. Lista PhoneGapin mallialustoista

Tässä työssä käytetään jQuery Mobilea, joten sovellus luodaan komennolla

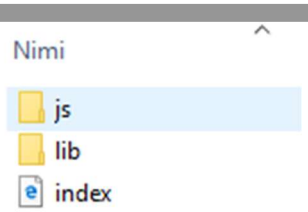
C:\> phonegap create c:\appsit\sovellus --template jquery-mobile-starter

PhoneGap luo annettuun polkuun sovelluksen nimen mukaisen kansion ja sinne seuraavat kansiot (kuva 11):



Kuva 11. PhoneGap luo sovellukselle hakemiston ja tiedostot

Luodussa www-kansiossa on index.html-tiedosto sekä Js- ja lib-alihakemistot, jotka sisältävät automaattisesti luodut Javascript-tiedostot ja jQuery Mobile -kirjaston. (Kuva 12.)



Kuva 12. PhoneGapin luomat alihakemistot ja index.hml

Sovelluksen käyttöliittymän kirjoittaminen aloitetaan www-kansiossa olevassa index.html-tiedostossa, johon PhoneGap on luomisvaiheessa jo tehnyt tarvittavat määritykset. Index.html-tiedosto on sovelluksen aloitussivu, mutta tiedostojen välillä voi siirtyä ja hakemistoon voi luoda muita tiedostoja itse. Oletuksena kaikkien sovelluksessa käytettävien tiedostojen tulisi olla sovelluksen hakemistossa ja muuten toiminta on sama kuin nettisivuja tehdessä.

Sovelluksen testiympäristö käynnistetään siirtymällä komentokehotteella sovelluksen hakemistoon ja suorittamalla käsky (kuva 13):

c:\appsit\sovellus\> phonegap serve

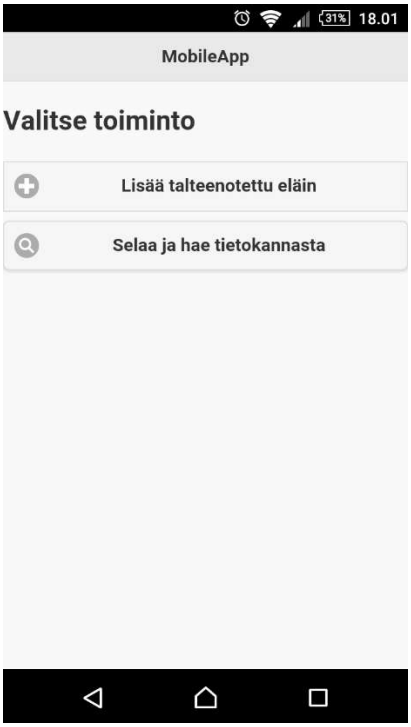
```
[phonegap] starting app server...
[phonegap] listening on 192.168.1.101:3000
[phonegap]
[phonegap] ctrl-c to stop the server
[phonegap]
```

Kuva 13. PhoneGap serve -komennolla käynnistetty sovelluksen jako

Testattavan mobiililaitteen tulee olla samassa verkossa työaseman kanssa. Kun PhoneGap Developer -sovellus käynnistetään mobiililaitteella, ensimmäiseksi syötetään komentokehotteella näkyvä palvelimen osoite ja portti. (Kuva 14.)

Kuva 14. Mobiilaitteen PhoneGap Developer -sovelluksella yhdistetään työasemalle, jossa on käynnissä mobiilisovelluksen jako

Tämän jälkeen PhoneGap Developer purkaa sovelluspaketin ja sovellus päivittyy laitteeseen sitä mukaa kun sitä muokataan ja tallennetaan työaseman tekstieditorissa. (Kuva 15.)



Kuva 15. Kuvakaappaus mobiililaitteen näytöstä ja käynnissä olevasta PhoneGap Developer -palvelusta

Kuvan 15 mukainen yksinkertainen ulkoasu on toteutettu alla olevalla HTML-kielellä jQuery Mobilen kirjastoa hyödyntäen. Tämä on sovelluksen aloitussivu, eli index.html:

```
<body>
<div data-role="header">
<h1> MobileApp </h1>
</div>

<h2> Valitse toiminto</h2>
<a href="#uusi" class="ui-btn ui-icon-plus ui-btn-icon-left"
data-transition="fade"> Lisää talteenotettu eläin </a>
<a href="#haku" class="ui-btn ui-corner-all ui-shadow ui-
icon-search ui-btn-icon-left" data-transition="fade"> Selaa
ja hae tietokannasta </a>
</div>
</body>
```

Painikkeiden ja kuvakkeiden ulkoasu haetaan automaattisesti jQuery Mobile -kirjastosta määrittelemällä sen luokka. Esimerkiksi ”ui-btn” tarkoittaa painiketta ja ”ui-icon-search” hakua symboloivaa suurennuslasikuvaketta. ”Data-transition” määrittelee siirtymisefektin sivujen välillä.

Sovelluksen hakemiston Config.xml-tiedostossa voidaan määritellä asetuksia ja metadataa sovellukselle, kuten sovelluksen kuvauksen, nimen ja ikonin. Ikoni määritellään lisäämällä tiedostoon tagi <icon src=”kuvat\icon.png” \>.

Kun sovellus on valmis, se pakataan PhoneGapin palvelulla ja paketti asennetaan mobiililaitteeseen.

6.2 WordPress API ja mobiilisovelluksen yhteys palvelimeen

WordPressin ohjelmallisuuksia ja tietokantaa voidaan käyttää hyväksi mobiilisovelluksen toiminnassa käyttäen sen API:a (Application programming interface) eli ohjelmointirajapintaa. Tässä työssä käytetään CORS-menetelmää, jolla saadaan lähetettyä palvelimelle pyyntöjä ulkopuolisesta lähteestä.

PhoneGap luo Whitelist-lisäosan automaattisesti. Whitelistiin listataan sallitut lähteet, joiden sisältöä sovellus voi hyödyntää. Siihen siis tulisi laittaa vain luotettavia lähteitä. Sovelluksen Config.xml-tiedoston Whitelist-pluginiin lisättiin Allow Access -tagiin haluttu palvelimen domain-osoite, josta tietoa halutaan ladata sovellukselle. Tällä tavalla voidaan rajoittaa pyyntöjen vastaanottamista ja lähettämistä.

WordPress-sisällönhallintajärjestelmään asennettiin valmis WP REST API -lisäosa, joka mahdollistaa HTML-pyyntöjen suorittamisen palvelimelle ulkopuolisesta lähteestä. HTML-pyyntö suoritetaan jQueryä ja Ajaxia käyttäen. (Fi.wordpress.org.)

Alla esimerkki Ajax-kutsusta käyttäen hyväksi WP REST API -lisäosaa:

```
<script>
$(document).ready(function(){
$('#tulostusbtn').click(function(){
$.ajax({
    url: 'http://palvelimeni.fi/wp-json/wp/v2/loytoelai-
met?filter[taxonomia]=talteenotetut',
    method: 'GET',
    crossDomain: true,
    success: function() {
        //onnistunut kutsu…
    }
})
});
</script>
```

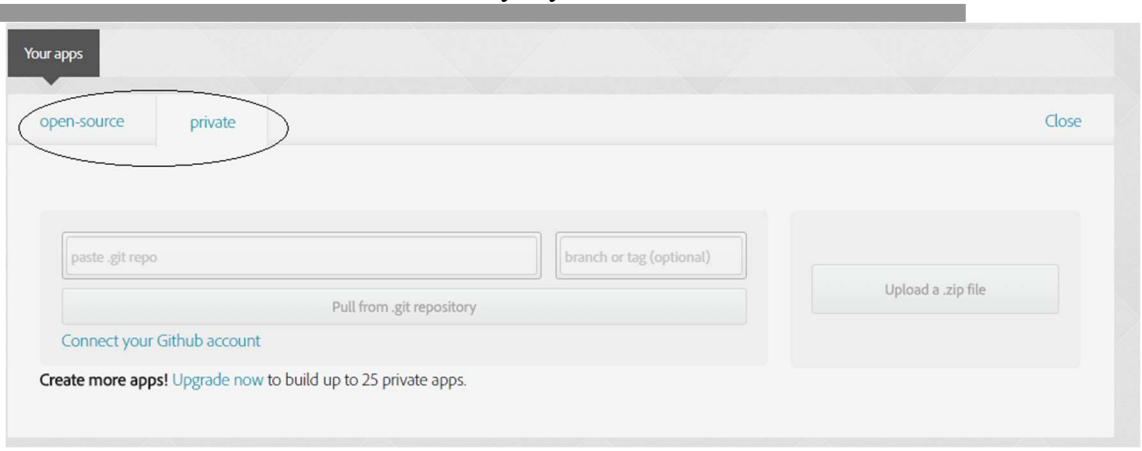
Lähetettävässä URLissa suoritetaan haku seuraavin perustein: sisältötyyppi on ”loytoelaimet”, ja ”taxonomia”-taksonomian termi ”talteenotetut”. Mikäli omaa sisältötyyppiä ja taksonomiaa luodessa ei ole määritelty ”show_in_rest”-määreen arvoksi ”true”, pyyntöä ei sallita itse luodulla sisältötyypillä tai taksonomialla.

WP REST API -lisäosa mahdollistaa CRUD-toimintojen suorittamisen, CRUD on lyhenne sanoista Create, Read, Update ja Delete. Get-metodin lisäksi käytettävissä olevia pyyntöjä ovat Post, Put, Delete, Head ja Options. Kaikkiin pyyntöihin, jotka muokkaavat jotain dataa, pitää luoda autentikointi jotta pyyntö voidaan suorittaa. Käytössä on Basic Authenticationin lisäksi Auth0 Authentication API. (Fi.wordpress.org.)

Sovelluksen autentikointiin käytetään 0Auth-autentikointia, joka on tietoturvaltaan luotettavampi kuin Basic Authentication. Basic Authenticationia tulisi käyttää vain luotettavassa yksityisessä verkossa. Sovelluksen autentikointia varten asennettiin lisäosa, esimerkiksi osoitteesta https://github.com/WP-API/OAuth1.

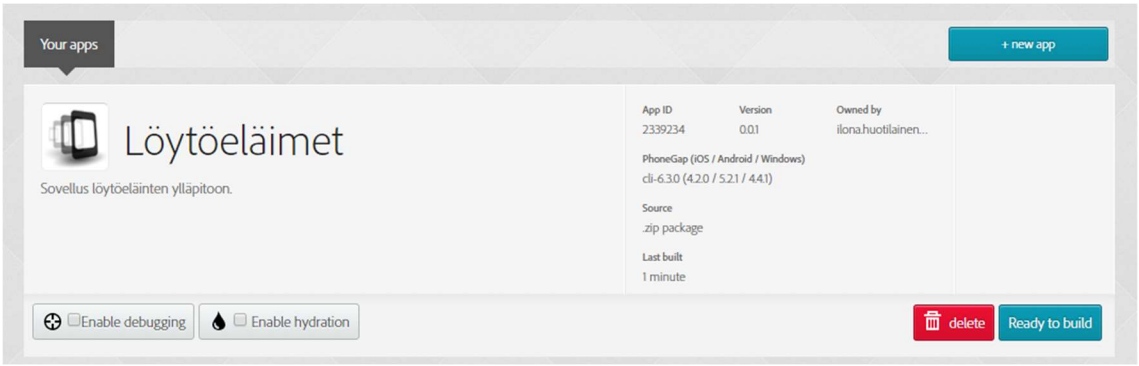
6.3 Sovelluksen pakkaus ja asennus mobiililaitteelle

Luodun sovelluksen saa pakattua helposti Adobe PhoneGapin Build -palvelulla. Kirjaudutaan PhoneGap Build -palveluun osoitteessa https://build.phonegap.com/. Kirjautuminen on ilmaista ja jos on olemassa olevat Adobe ID -tunnukset, niillä voi kirjautua. Palveluun voi ladata koko sovelluksen pakattuna .zip-muodossa tai Githubin kautta.



Kuva 16. Sovelluksen tiedostojen lataaminen PhoneGap Build-palveluun

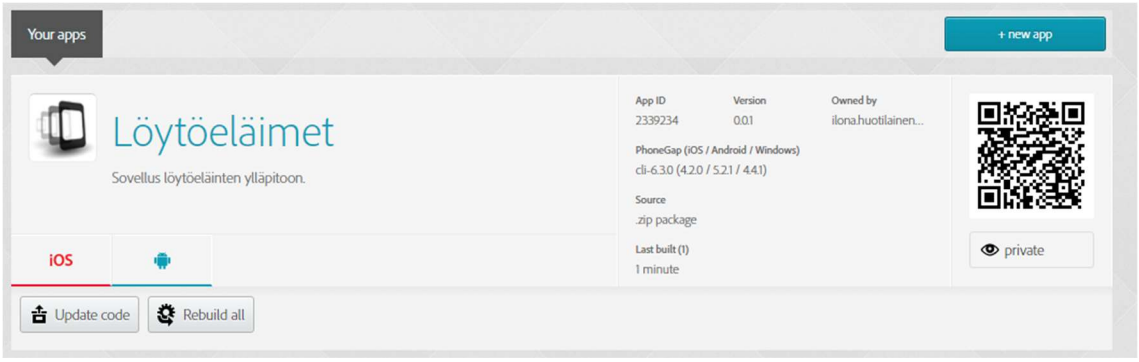
Kuvassa 16 on PhoneGap Build-palvelun Your apps -sivu, jossa voidaan ladata uusia sovelluksia pakattavaksi. Ylhäällä näkyvästä välilehdestä voi valita ladataanko sovellus julkiseksi ja kaikkien käytettäväksi (open-source) vai yksityiseksi (private). Sovelluksen kaikki tiedostot voi pakata .zip-tiedostoksi tai ladata sen palveluun GitHub-palvelun kautta.



Kuva 17. Sovellus ladattu onnistuneesti PhoneGap Build -palveluun

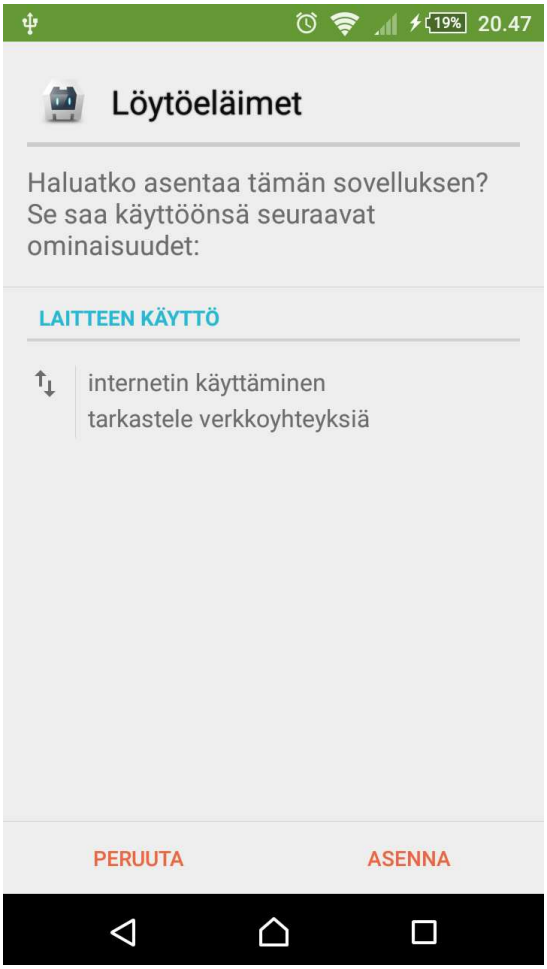
Kun tiedostot on ladattu pilveen, sovellus pakataan klikkaamalla oikealta alareunasta ”Ready to build”. (Kuva 17.)

Kun pakkaus on valmis, sen voi ladata klikkaamalla Androidin tai iOS -järjestelmän kuvaketta. Jos sovellus olisi luotu muillekin alustoille, ne näkyisivät kuten nämä. (Kuva 18.)



Kuva 18. Sovellus pakattu PhoneGap Build -palvelussa iOS- ja Android-alustoille

Kun Android-järjestelmän paketti (APK) on ladattu tietokoneelle, se siirretään USB-johdolla yhdistettynä mobiililaitteeseen. Kuvassa 19 näkyy Android-laitteen ohjattu asennustoiminto.



Kuva 19. Mobiilisovelluksen asennus Android-järjestelmään.

Mobiililaitteen asetuksista tulee sallia sovellusten asennus tuntemattomista lähteistä. Asennuksen jälkeen asetuksen voi palauttaa ennalleen.

LÄHTEET

Codex.wordpress.org. Custom Fields ja Metadata. Haettu 15.11.2016 osoitteesta:
https://codex.wordpress.org/Using_Custom_Fields

Codex.wordpress.org. Database Description. Haettu 25.10.2016 osoitteesta:
https://codex.wordpress.org/Database_Description

Codex.wordpress.org. Plugins. Haettu 12.12.2016 osoitteesta:
https://codex.wordpress.org/Writing_a_Plugin

Codex.wordpress.org. WordPress Post Types. Haettu 6.10.2016 osoitteesta:
https://codex.wordpress.org/Post_Types/

Fi.wordpress.org. Wordpress API. Haettu 15.11.2016 osoitteesta:
https://fi.wordpress.org/plugins/rest-api/

Nngroup.com. Mobile Apps. Haettu 4.10.2016 osoitteesta:
https://www.nngroup.com/articles/mobile-native-apps/

PhoneGap.com. About PhoneGap. Haettu 15.11.2016 osoitteesta:
http://phonegap.com/about/

PhoneGap.com. PhoneGap CLI. Haettu 6.10.2016 osoitteesta:
http://docs.phonegap.com/getting-started/1-install-phonegap/cli/

Viestintävirasto. Fi-verkkotunnus. Haettu 10.10.2016 osoitteesta:
https://www.viestintavirasto.fi/fiverkkotunnus.html

WordPress.org. About WordPress. Haettu 15.11.2016 osoitteesta:
https://wordpress.org/about/

WordPress.org. Requirements. Haettu 12.10.2016 osoitteesta:
https://wordpress.org/about/requirements/

Wpbeginner.com. Create custom post type. Haettu 6.10.2016 osoitteesta:
http://www.wpbeginner.com/wp-tutorials/how-to-create-custom-post-types-in-wordpress/

WpTheming.com. Metadata, Custom metaboxes. Haettu 15.11.2016 osoitteesta:
http://wptheming.com/2010/08/custom-metabox-for-post-type/

Wpbeginner.com. Custom taxonomies. Haettu 5.11.2016 osoitteesta:
http://www.wpbeginner.com/wp-tutorials/create-custom-taxonomies-wordpress/