LAB University of Applied Sciences

KOMPONENTTIKIRJASTON SUUNNITTELU JA KEHITTÄMINEN

LAB-AMMATTIKORKEAKOULU
Tieto- ja viestintätekniikka, insinööri
Ohjelmistotekniikka
Syksy 2020
Julia Kuisma

Tiivistelmä

Tekijä(t)	Julkaisun laji	Valmistumisaika
Kuisma, Julia	Opinnäytetyö, AMK	Syksy 2020
	Sivumäärä	
	36	

Työn nimi

Komponenttikirjaston suunnittelu ja kehittäminen

Tutkinto

Insinööri, Tieto- ja viestintätekniikka (AMK)

Tiivistelmä

Opinnäytetyön tavoitteena oli suunnitella ja kehittää komponenttikirjasto WordPress-ympäristössä. Komponenttikirjasto pilotoitiin käyttäen kahta näkökulmaa, joita ovat kehittäjän ja asiakkaan näkökulma. Kirjaston lisäksi suunniteltiin levitystapa, jolla levitetään komponentit www-sivustoille.

Komponenttikirjasto toteutettiin työnä lahtelaiselle ohjelmistoyritykselle Tammi Digital Oy:lle. Yritys kehittää www-sivustoja ja tarjoaa ylläpitopalveluita sivustoille. Komponenttikirjaston tarkoituksena on helpottaa uusien sivustojen kehitystä, sillä sivuston sisältö voidaan rakentaa käyttäen komponentteja.

Komponenttikirjaston komponenteiksi valittiin yrityksen kehittämillä nettisivuilla usein toistuvat osat, joita olivat Herokuva, Otsikko ja sisältö, Kuvakaruselli, Yhteystiedot, Artikkelinosto sekä Lomake. Komponenttien template-tiedostojen lisäksi luotiin ACF-kenttäryhmä komponenteille.

Työn tuloksena saatiin komponenttikirjasto toteutettua halutuilla sivun osilla. Lisäksi komponentit saatiin pilotoitua käyttäen valittuja näkökulmia. Kirjaston levitystavan suunnittelu jäi keskeneräiseksi, sillä projektista loppui aika.

Asiasanat

WordPress, Komponentti, Teema

Abstract

Author(s)	Type of publication	Published
Kuisma, Julia	Bachelor's thesis	Autumn 2020
	Number of pages	
	36	

Title of publication

Designing and developing a component library

Name of Degree

Bachelor of Engineering, Information and Communications Technology

Abstract

The goal of the thesis was to design and develop a component library in a WordPress environment. The component library was piloted with two points of view, which were the developer's and the customer's. In addition to the library, a method to distribute the components to websites was designed.

The component library was made for Tammi Digital Oy, a software company from Lahti. The company develops websites and offers administration services for them. The purpose of the component library is to ease the development of new websites by building the content of the website using components.

Frequently used parts from websites that were developed by the company were chosen as the components for the component library. These components were Hero image, Title and content, Image slider, Contact info, Article and Form. In addition to the components' template-files, an ACF field group was created for the components.

As a result of the thesis, the component library with all wanted parts of pages was achieved. In addition, the components were piloted using the chosen points of view. Designing the distribution method was unfinished because the project ran out of time.

Keywords

WordPress, Component, Theme

SISÄLLYS

1 JOHDANTO

WordPress on ilmainen CMS-järjestelmä (Content Management System), jolla voidaan luoda monenlaisia www-sivustoja. WordPressin vahvuuksiin kuuluu sen muokattavuus: ulkonäön muokkaaminen teemoilla sekä toiminnallisuuksien lisääminen lisäosilla. Sisällön hallinta on WordPressissä helppoa ja selkeää sen hallintasivujen ja monien ominaisuuksien avulla. WordPress on myös helposti indeksoitava hakukoneille, jotta sivusto on löydettävissä mahdollisimman helposti. Lisäksi WordPressissä on hyvä tietoturvallisuus, jolla voidaan esimerkiksi rajata kirjautumista IP-osoitteen avulla, sekä tuki monille erilaisille mediatiedostomuodoille.

Tämän opinnäytetyön tavoitteena on komponenttikirjaston suunnittelu ja kehittäminen WordPress-ympäristössä. Kirjaston komponentit ovat toteutettu käyttäen WordPressin lisäosaa ACF eli Advanced Custom Fields ja luoden template-tiedostoja sivuston teemaan. Komponentteja käytettäessä käytetään uudelleen sivuston osia, jolloin sisällön luominen on helpompaa kuin käytettäessä sivukohtaista sivupohjia. Lisäksi sisältö voi olla monipuolisempaa kuin perinteisen WordPress-sivun, sillä komponenteilla toiminnallisuuksien luominen ei ole niin rajallista. Komponenttikirjasto on toteutettu työnä lahtelaiselle ohjelmistoalan yritykselle Tammi Digital Oy:lle. Yrityksen toimialaan kuuluu www-palveluiden kehitys ja ylläpito sekä ohjelmistokehitys. Opinnäytetyössä keskitytään komponenttien suunnitteluun sekä testaukseen, jossa otetaan huomioon niin kehittäjän kuin asiakkaan näkökulma.

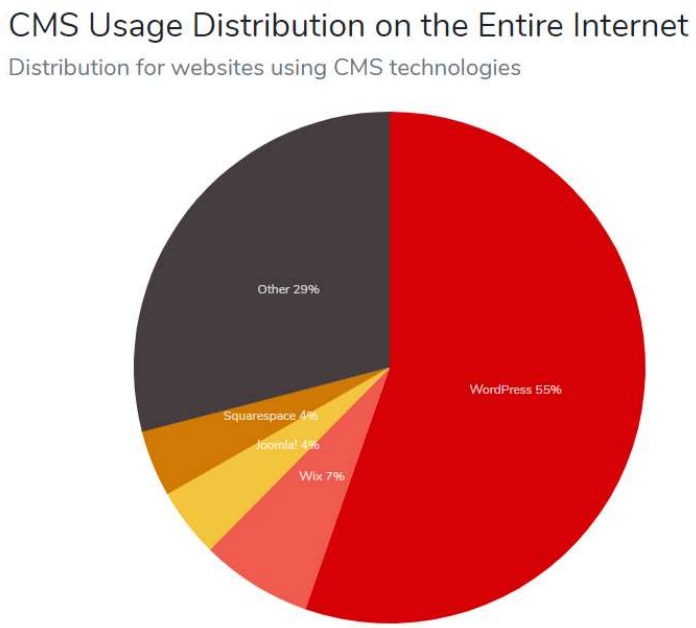
Opinnäytetyön tavoitteena on suunnitella ja pilotoida komponenttikirjasto, joka on toteutettu WordPress-ympäristössä. Kirjaston komponenteiksi on valittu yrityksen kehittämillä www-sivuilla usein toistuvat osat, joita ovat Herokuva, Otsikko ja sisältö, Kuvakaruselli, Yhteystiedot, Artikkelinosto sekä Lomake. Komponenttien suunnittelussa otetaan huomioon uudelleenkäytettävyys ja pidetään komponenttien sisäinen logiikka mahdollisimman yksinkertaisena. Komponenttien testaukseen kuuluu sivunmuokkausnäkymän testaus sekä ulkonäön testaus sivulla. Komponenttien ehdollista logiikkaa ja kenttien arvoja testataan sivunmuokkauksessa, kun taas ulkonäössä otetaan huomioon enemmän sivuston HTML-asettelua ja tyylimäärittelyjä.

Tässä opinnäytetyössä käydään läpi WordPressiä julkaisualustana. Ensin kerrotaan WordPressin ominaisuuksista, asennuksesta sekä tietokannasta. Sen jälkeen kerrotaan lisäosien, teemojen ja komponenttien rakenteesta sekä sisällöstä. Käytäntöosuudessa käydään läpi komponenttikirjaston suunnittelua, toteutusta sekä pilotointia. Pilotoinnin jälkeen suunnitellaan kirjaston levitystapa ja lopuksi reflektoidaan projektin eri osa-alueita yhteenvedossa.

2 WORDPRESS

2.1 Ominaisuudet

WordPress on avoimen lähdekoodin sovellus, jolla kehitetään ja hallitaan www-sivustoja, kuten verkkokauppoja ja blogeja. WordPress kuuluu GPLv2-lisenssin alle, johon kuuluu oikeus käyttää, muokata ja jakaa muokattua versiota ohjelmasta. Vuonna 2003 Mike Little ja Matt Mullenweg loivat oman version ”b2/cafelog”-julkaisujärjestelmästä, sillä oli tarvetta yksilölliselle julkaisujärjestelmälle. Nykyään WordPressiä käyttävät alustana 38 prosenttia kaikista www-sivustoista ja 55 prosenttia kaikista CMS-järjestelmää käyttävistä sivuista (kts. alla oleva kuvio 1.). (WordPress.org a.) WordPressin kehitykseen on myös mahdollista osallistua eri osa-alueilla, joita ovat esimerkiksi WordPress-ydin, suunnittelu ja saavutettavuus (WordPress.org b).



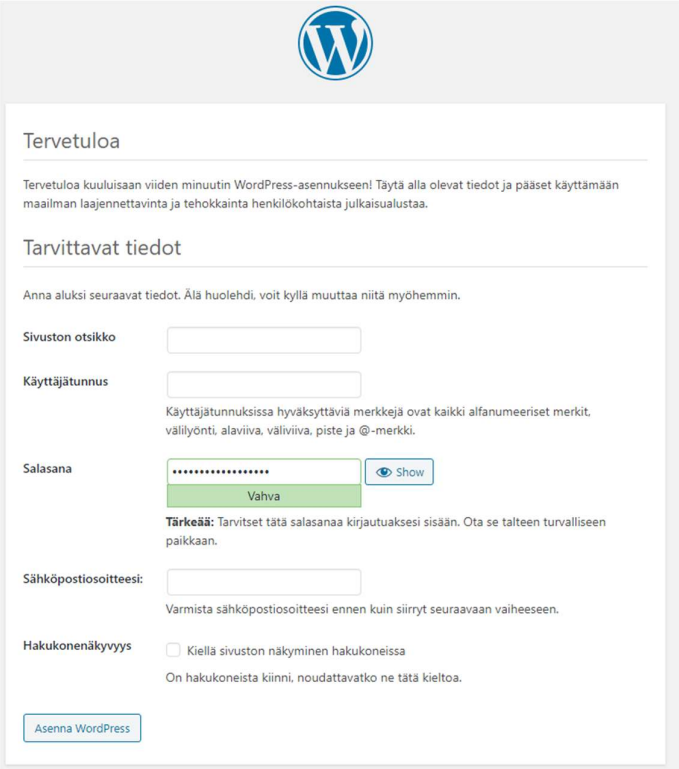
Kuvio 1. CMS-järjestelmien jakauma (Azmi 2020)

WordPressin ominaisuuksiin kuuluu joustavuus, jolla tarkoitetaan sitä, että se soveltuu useamman tyyppisen sivun julkaisuun. WordPressillä voidaan esimerkiksi julkaista henkilökohtaisia blogeja, uutissivuja tai portfolioita. Sisällön julkaisuun on käytössä erilaisia työkaluja, kuten luonnosten luominen, julkaisun ajoittaminen ja sivun tarkistaminen, joilla tehdään sisällön hallinnasta helpompaa. WordPressissä on käyttäjien hallinta, jolla määritetään käyttäjille erilaisia rooleja, kuten ylläpitäjä, kirjoittaja sekä tilaaja, ja rajoitetaan pääsyä tiettyihin ominaisuuksiin. WordPress on käännetty useammalle eri kielelle, sillä yhteisöön on luotu käännösryhmiä, jotka pyrkivät kääntämään WordPressin mahdollisimman

monelle kielelle. Kehittäjille on myös tarjolla erilaisia ominaisuuksia, kuten lisäosat ja teemat, jolla voidaan laajentaa toiminnallisuuksia ja muokata ulkonäköä. WordPressillä voidaan myös luoda muokattua sisältöä käyttäen kustomoituja julkaisuja sekä taksonomioita. (WordPress.org c.)

2.2 Asennus

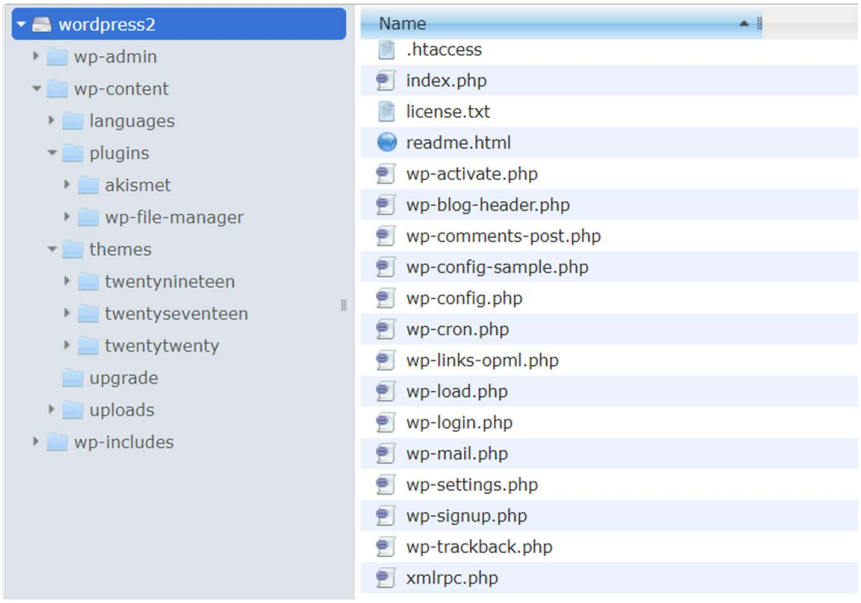
WordPressin asennuksella on tietyt vaatimukset palvelimelle, joihin kuuluvat esimerkiksi tietokannan ja PHP-ohjelmointikielen versio. Palvelimeksi suositellaan Apache- tai Nginx-palvelinta, joissa on mukana ”mod_rewrite”-moduuli, ja tukea HTTPS-protokollalle. HTTPS-protokollan tuki on suositeltavaa, sillä se luo salatun yhteyden selaimen ja palvelimen välille (Mullenweg 2016). Palvelimen tulisi tukea ainakin PHP-versiota 7.4 ja tietokantana MySQL-versiota 5.6 tai MariaDB-versiota 10.1. Vanhemmalla versiolla, kuten PHP-versio 5.6:lla, on mahdollista käyttää WordPressiä, mutta niiden käyttö ei ole suositeltua loppuneen kehityksen ja mahdollisien haavoittuvuuksien takia. (WordPress.org d.)



Kuva 2. WordPress asennus

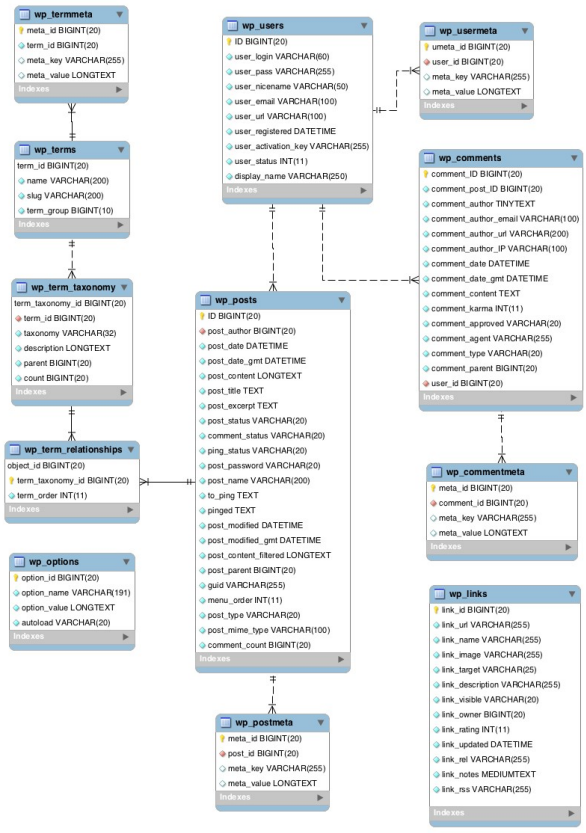
WordPressin asennukseen kuuluu viisi eri vaihetta, joista ensimmäinen on WordPress paketin lataus ja purkaminen. Purkamisen jälkeen paketti tulee siirtää palvelimelle. Sen jälkeen palvelimelle tulee luoda tietokanta WordPress-asennukselle. Tämän jälkeen

alustetaan wp-config.php tiedostoon tietokannan nimi, käyttäjä ja salasana. Viimeiseksi ajetaan WordPressin asennus selaimessa, johon kuuluu esimerkiksi kielivalinta, pääkäyttäjän luominen ja hakukonenäkyvyyden valinta. Yläpuolella olevassa kuvassa 2. on WordPress-asennuksen vaihe, jossa luodaan www-sivuston otsikko ja pääkäyttäjä. (HostGator 2020.)



Kuva 3. WordPress-asennuksen tiedostorakenne

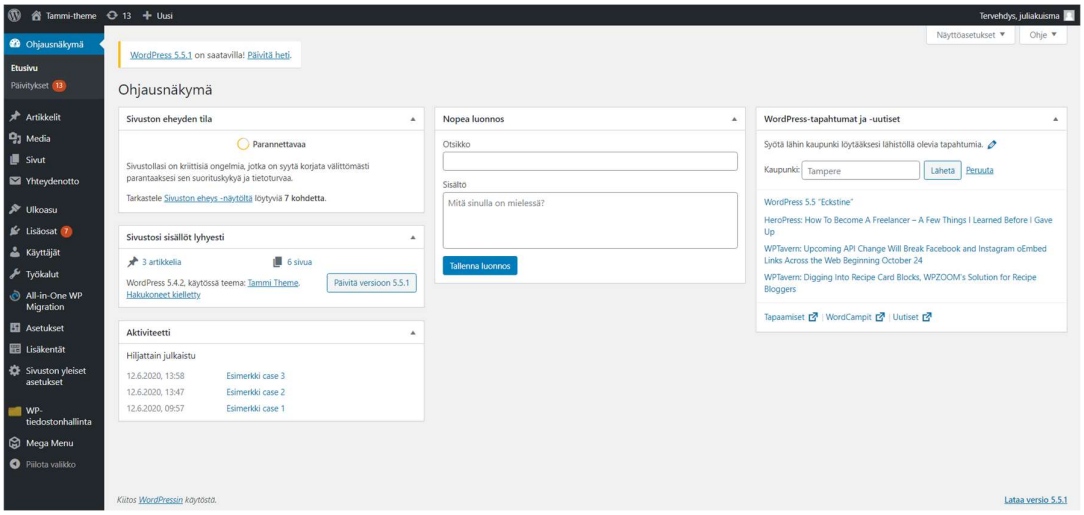
WordPress-asennuksen aikana luodaan sivuston tiedostohakemisto ja tietokanta. Yläpuolella olevassa kuvassa 3. on WordPress-asennuksen tiedostorakenne avattuna WordPress-lisäosan File Manager avulla. Kuvassa vasemmalla on kansioiden rakenne ja oikealla ylimmän kansion sisältö. Tiedostohakemisto koostuu pääasiassa WordPressin ytimen tiedostoista, konfigurointitiedostoista sekä kolmesta kansiosta: wp-admin, wp-content ja wp-includes. Kansiosta wp-admin löytyvät WordPressin hallintatyökalut, joilla esimerkiksi luodaan WordPressin hallintapaneeli sekä hallitaan käyttäjiä. Kansion tärkein tiedosto on admin.php, jolla toteutetaan kriittisiä toimintoja, kuten tietokantaan yhdistäminen ja käyttäjien hallintaoikeuksien tarkistaminen. Käyttäjien luoma sisältö, kuten teemat ja lisäosat, löytyvät kansiosta wp-content. Lisäksi kansioon sijoitetaan käyttäjien lataamat kuvat ja muut mediat kansioon ”uploads”. Wp-includes kansiosta löytyvät WordPressin luokkien tiedostot sekä ytimen tiedostot, joilla mahdollistetaan erilaisia toiminnallisuuksia. Konfigurointitiedostoihin kuuluvat wp-config.php ja .htaccess. Wp-config.php tiedosto sisältää WordPressin asetukset, joilla voidaan muokata tietokannan asetuksia ja esimerkiksi kasvattaa sivuston muistirajoitusta. Tiedostolla .htaccess hallitaan tiedostojen pääsyoikeuksia ja ikilinkkien rakenteita. (Hughes 2017.)



Kuvio 4. Standardin WordPress-tietokannan rakenne (WordPress.org e)

Standardin asennuksen aikana luodaan tietokanta, johon WordPress luo tarvittavat tietokantataulut. WordPress-asennuksen aikana valitaan tietokannan etuliite, joka on oletuksena wp. Yllä olevassa kuviossa 4. on visuaalinen näkemys tietokannan taulujen kentistä ja suhteista. WordPress luo kaksitoista tietokantataulua, joista jokainen sisältää oleellista tietoa niin eri toiminnallisuuksista kuin eri osista sivua. Esimerkiksi wp_posts-taulu pitää sisällään kaikki artikkelit, sivut ja navigointivalikon kohteet, ja sillä on useampi eri kenttä, kuten artikkelin kirjoittaja, otsikko, päiväys ja tila. (WordPress.org e.)

Tietokannasta löytyy kolmea eri suhdetta, joilla taulut ovat yhteydessä toisiinsa: yksi yhteen, yksi moneen ja monta moneen. Yksi yhteen -suhteella tarkoitetaan sitä, että taulun tietue on suhteessa toiseen tietueeseen, kuten post ID ja post content. Jotta saadaan aikaiseksi monimutkaisempia yhteyksiä, käytetään yksi moneen ja monta moneen -suhteita. Yksi moneen -suhteella voidaan yhdistää kaksi taulua käyttäen uniikkia kenttää, kuten post_id, jolla voidaan yhdistää taulut wp_posts ja wp_comments. Monta moneen -suhteella voidaan yhdistää tauluja molempiin suuntiin, kuten yhdellä artikkelilla voi olla usea termi ja yksi termi voi olla monella artikkelilla. (McCollin 2014.)



Kuva 5. WordPressin hallintapaneeli

WordPressin hallintapaneeli koostuu neljästä osiosta, joita ovat työkalurivi, päänavigaatio, työalue ja alatunniste (footer) (kts. yllä oleva kuva 5.). Työkalurivistä löytyvät useat toiminnot, kuten päivitykset ja uuden artikkelin, median, sivun tai käyttäjän lisäys. Työkalurivi on näkyvissä hallintapuolella ja myös sivuston kaikkien sivujen yläreunassa, silloin kun käyttäjä on kirjautuneena. Päänavigaatiosta löytyvät kaikki hallintatoiminnot, kuten ulkoasun muokkaus tai sivuston asetukset. Toimintojen alavalikot avautuvat navigaation oikealle puolelle, kun kursorin asettaa niiden päälle (hover). Päävalikko voidaan piilottaa valikon alareunasta löytyvällä painikkeella. Työalue pitää sisällään kyseisen hallintasivun toiminnot. Esimerkiksi etusivulla työalueesta löytyy toiminnot nopea luonnos, sivuston sisältö sekä eheys, historia ja tulevat WordPress-tapahtumat sekä -uutiset. Alatunnisteosiosta löytyy WordPress-asennuksen versio. (WordPress.org f.)

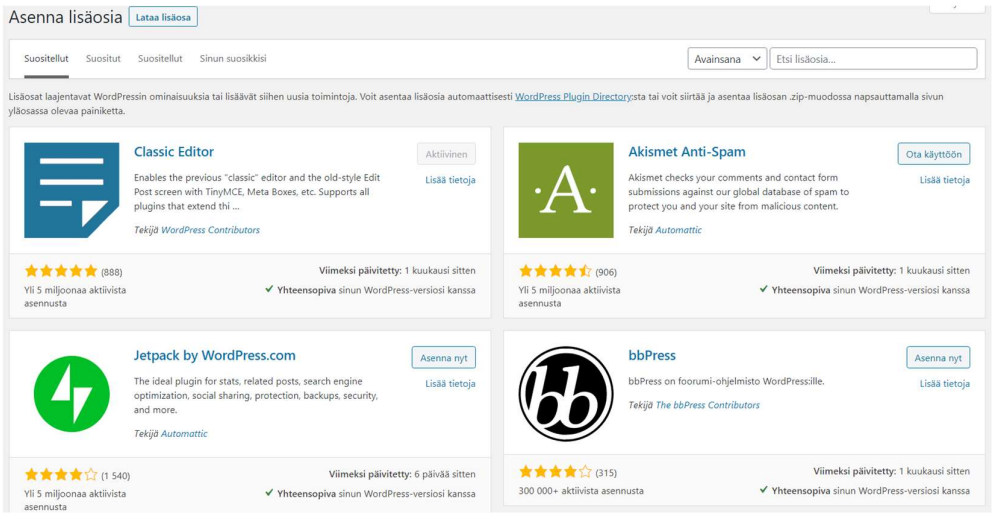
2.3 Lisäosa

WordPress lisäosat ovat tiedostoja, joilla luodaan uusia tai parannetaan jo olemassa olevia ominaisuuksia. Lisäosat on kirjoitettu PHP-ohjelmointikielellä ja ne koostuvat pääosin funktioista. WordPressiin löytyy tuhansia ilmaisia lisäosia, joilla voidaan lisätä ominaisuuksia sivuille, kirjoittamatta itse PHP-ohjelmaa. Alapuolella olevassa kuvassa 6. on esimerkki lisäosan ”Hello Dolly” PHP-lähdekoodista. Yleisimmät lisäosat jakautuvat tiettyihin kategorioihin, joita ovat tiedon tuonti ja vienti, SEO (Search Engine Optimization) eli hakukoneoptimointi sekä tietoturva. (WordPress.org g.)

```php
// This just echoes the chosen line, we'll position it later.
function hello_dolly() {
    $chosen = hello_dolly_get_lyric();
    $lang   = '';
    if ( 'en_' !== substr( get_user_locale(), 0, 3 ) ) {
        $lang = ' lang="en"';
    }

    printf(
        '<p id="dolly"><span class="screen-reader-text">%s </span><span dir="ltr"%s>%s</span></p>',
        __( 'Quote from Hello Dolly song, by Jerry Herman:' ),
        $lang,
        $chosen
    );
}

// Now we set that function up to execute when the admin_notices action is called.
add_action( 'admin_notices', 'hello_dolly' );
```

Kuva 6. Hello Dolly -lisäosan PHP-tiedosto

Lisäosat sijaitsevat wp-content-kansion alla olevassa kansiossa ”plugins”. Lisäosan kansio ja PHP-tiedosto tulisi nimetä lisäosan mukaan pienellä kirjoitettuna, kuten ”esimerkkilisaosa”. Lisäosatiedoston alussa tulisi olla lisäosan yläviitekommentti. Kommentti on muotoiltu PHP-osio, jonka sisään voidaan sijoittaa tietoa lisäosasta, kuten lisäosan nimi, kirjoittaja, versio ja lisenssi, mutta sen tulisi vähintään sisältää lisäosan nimi. Yläviitekommentti sijoitetaan vain lisäosan yhteen tiedostoon, vaikka sillä olisi useampi PHP-tiedosto. (WordPress.org h.)



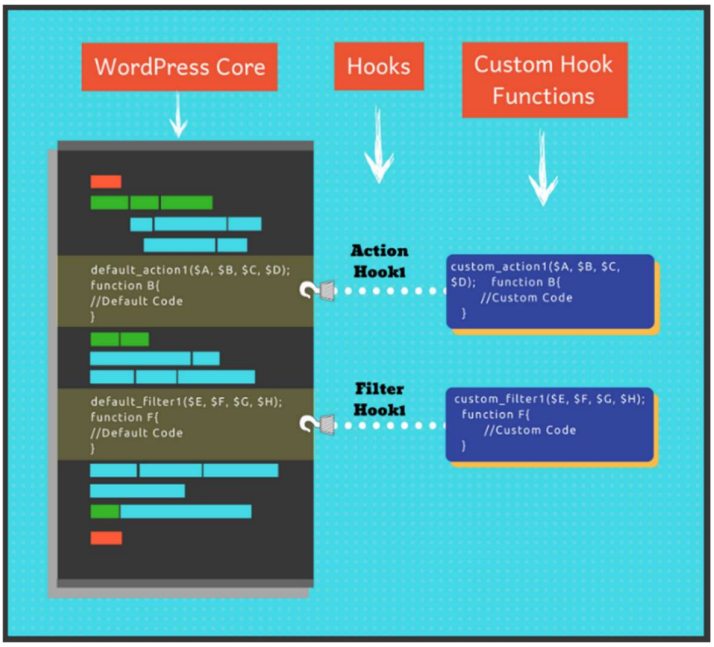
Kuva 7. Uusien lisäosien lisääminen

Lisäosia voidaan asentaa kolmella tavalla: lisäosien lisäys käyttäen WordPressin sisäänrakennettua hakemistoa (kts. yllä oleva kuva 7.), lataamalla WordPressiin lisäosan pakattu kansio tai lataamalla lisäosan tiedostot suoraan palvelimelle. WordPress-hakemistossa lisäosista löytyvät seuraavat tiedot: nimi, kuvaus, arvostelut, aktiivisten asennuksien määrä, viimeisin päivitys ja yhteensopivuus. Yhteensopivuudella tarkoitetaan sitä, että

onko kyseinen lisäosa sopiva WordPress-ytimen version kanssa. Lisäosa voidaan lisätä pakattuna kansiona esimerkiksi silloin, kun se on ladattu erilliseltä sivulta. Lisäosien tiedostojen latausta suoraan palvelimelle voidaan käyttää silloin, kun esimerkiksi palvelin on konfiguroitu siten, että lisäosien asennus on kiellettyä. Suoraan palvelimelle latauksessa on tosin riskinä se, että lisäosa ei ole yhteensopiva WordPress-version kanssa tai se voi sisältää haitallisia ohjelmia, jos sen julkaisija on epäluotettava. Tällöin on suositeltavaa luoda varmuuskopio sivusta ennen lisäosien siirtoa palvelimelle. (WordPress.org g.)

2.3.1 Lisäosan koukku

WordPressin koukuilla (hooks) voidaan muokata tai vaikuttaa tiettyyn osaan WordPress-ytimen koodista (kts. alla oleva kuva 8.). Koukuilla teemat ja lisäosat voivat olla vuorovaikutuksessa WordPressin ytimen kanssa muokkaamatta sitä. Jotta koukkuja voidaan käyttää, tulee luoda takaisinkutsufunktio ja rekisteröidä koukku joko toiminto- tai suodatinkoukuksi. Toimintokoukulla voidaan lisätä tietoa tai vaikuttaa WordPressin toimintaan, kuten tietyn funktion suoritus WordPress-ytimen suorituksen aikana tai tiedon lisäys tietokantaan. Suodatinkoukuilla voidaan muokata tietoa WordPress-ytimen ajon aikana. Suodatinkoukkujen takaisinkutsufunktiolle välitetään muuttuja, jota se muokkaa ja sen jälkeen palauttaa. Ero näiden kahden välillä on se, että toimintokoukku tekee toiminnon ja ei palauta takaisinkutsufunktiolle mitään, kun suodatinkoukku ottaa tiedon, muokkaa sitä ja palauttaa muokatun tiedon funktiolle. Lisäksi voidaan kehittää omia koukkuja, joilla voidaan muokata ja laajentaa lisäosaa tai teemaa. (WordPress.org i.)

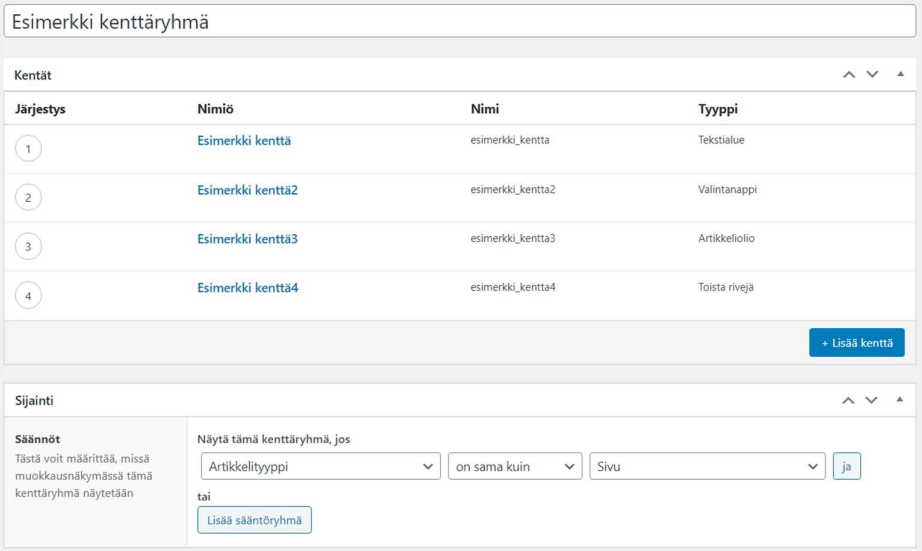


Kuva 8. WordPress-koukku (Azhar 2020)

WordPressin rajapinnasta löytyy funktioita, joilla koukut voivat tehdä erilaisia toimintoja. Funktioita ovat esimerkiksi lisäys, toiminnon tekeminen ja poistaminen. Koukun lisäyksessä luodaan ensimmäiseksi PHP-funktio, joka ajetaan tietyn tilanteen tapahtuessa. Sitten lisätään tämä funktio käyttäen add_action-funktiota. Koukun lisäyksessä välitetään tarvittavat parametrit, joita ovat koukun nimi, funktion tai suodattimen nimi, prioriteetti ja hyväksytyt argumentit. Koukun nimen tuottaa WordPress ja se kertoo esimerkiksi, minkä tapahtuman yhteydessä toteutetaan toiminto tai milloin suodatin otetaan käyttöön. Prioriteetti ja hyväksytyt argumentit ovat valinnaisia kokonaislukumuuttujia, joilla voidaan määrittää koukun suorituksen prioriteetti ja valinnaisten argumenttien määrän. Lopuksi lisätään PHP-funktio lisäosan tietoihin ja aktivoidaan se. Funktion nimen tulisi olla uniikki, sillä PHP ei hyväksy useaa samannimistä funktiota. Samannimisyyttä voidaan välttää laittamalla funktioiden nimen eteen tietty etuliite, joka voi esimerkiksi olla lyhenne lisäosan kirjoittajasta tai lisäosan nimestä. Toinen tapa on lisätä funktiot luokan sisään, jolloin funktion kutsun yhteydessä on luokan nimi. (WordPress.org j.)

2.3.2 ACF

ACF eli Advanced Custom Fields on lisäosa, jolla voidaan lisätä uusia kenttiä WordPressin hallintasivuihin, kuten artikkeleihin, käyttäjiin ja mediaan. Kenttien arvoja voidaan näyttää käyttäen funktioita the_field ja get_field. Lisäksi on mahdollista käydä läpi kenttiä, joilla on useampi rivi, käyttäen funktioita have_rows tai get_row. Kentät voidaan jakaa kenttäryhmiin, joilla voidaan määritellä sääntöjä, jotka esimerkiksi määrittelevät, millä sivuilla kentät ovat näkyvissä (kts. alla oleva kuva 9.). (Advanced Custom Fields 2020a.) Lisäosasta on kahta eri versiota: ilmainen ja maksullinen PRO-versio. Näissä versioissa on eroina esimerkiksi erityyppisten kenttien määrä. Alla olevassa kuvassa 9. on esimerkki siitä, miten neljä kenttää on lisätty kenttäryhmään ”Esimerkki kenttäryhmä”.

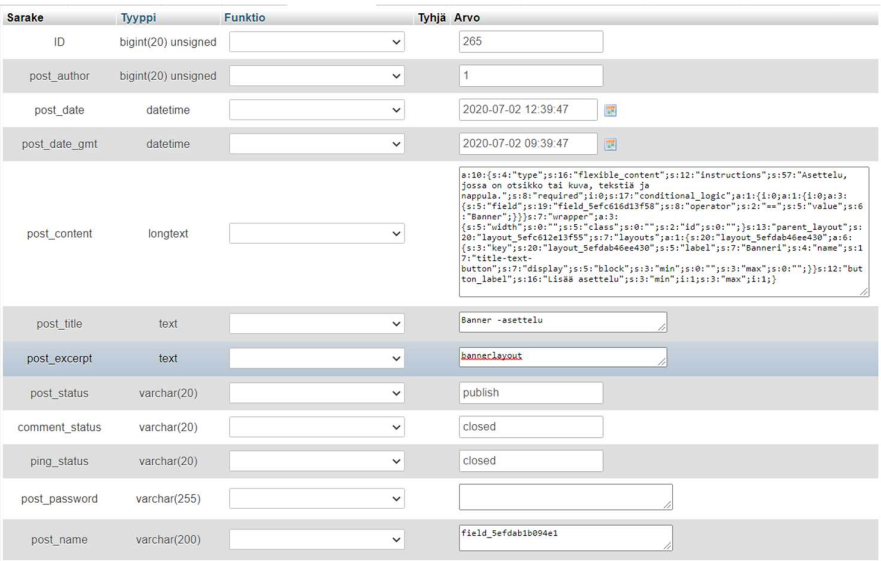


Kuva 7. ACF-kenttäryhmä

ACF-kenttien tyyppejä on määrältään 29 ja ne on jaoteltu kuuteen kategoriaan: perus-, sisältö-, valinta-, relationaaliset, jQuery- ja asettelukentät. Peruskenttien kategoriaan kuuluvat kentät, kuten liukusäädin, tekstikenttä ja tekstialue, joilla voidaan tehdä numerovalitsin tiettyjen raja-arvojen välille ja lisätä sivulle tekstikappaleita tai yksittäisiä lauseita. Sisältöön kuuluvat kentät, joilla lisätään tiedostoja, kuten kuvia, tai median muotoja, kuten galleria- tai oEmbed-kenttä. oEmbed-kentällä upotetaan esimerkiksi videoita tai eri sosiaalisen median julkaisuja. Lisäksi sisältökenttään kuuluu Wysiwyg-tekstieditori, jolla luodaan kenttä, jolle määritellään sekä tekstisisältö ja sen tyylittely että mediasisältö. Valintakentillä tehdään interaktiivisia valintavaihtoehtoja käyttäen painikkeita, kuten painikeryhmä tai radiopainike. Valintakentillä voidaan myös luoda pudotusvalikko tai tosi/epätosi valinta. Relationaalisilla kentillä voidaan hakea WordPress-sivu, artikkeli tai kustomoitu sivutyyppi, kuten sivulinkki tai sivuolio. Kentillä voidaan myös valita käyttäjä tai taksonomia. Kenttätyypillä jQuery voidaan lisätä esimerkiksi päivä-, väri- tai aikavalitsin, ja Google Maps -kartta sivulle. Viimeisimpään kategoriaan kuuluvat asettelukentät, joilla voidaan luoda kenttien asetteluja. Näihin kuuluvat esimerkiksi joustava sisältö ja toistokentät. (Advanced Custom Fields 2020b.)

Kuva 8. Esimerkki kenttäryhmän ulkoasu

ACF-kentät sijoittuvat sivulla päätekstieditorin alle. ACF kentät ryhmittäytyvät kenttäryhmien mukaan ja niiden järjestystä voidaan muokata kenttäryhmäikkunan nuolipainikkeiden avulla. Yläpuolella olevassa kuvassa 10. on esimerkkikomponentin ulkoasu sivun muokkaustilassa. Kenttien ulkoasu riippuu niiden tyypistä, kuten kuvassa on kentille valittu tyypeiksi tekstialue, valintanappi, artikkeliolio ja rivien toisto, jonka sisällä on tekstikenttä. Kenttien asettelua voidaan muuttaa siten, että ovatko ne pysty- tai vaakasuunnassa, ja onko niiden asettelu taulukon, lohkon vai rivin mallinen. Useamman rivin kentille voidaan asettaa minimi- ja maksimiarvo siitä, kuinka monta riviä niillä on. Kenttiin voidaan lisätä ehdollista logiikka, jolla voidaan asettaa ehto, kuten kentän arvo, jolla määritellään, onko jokin toinen kenttä näkyvissä.



Kuva 9. ACF-Kenttä tietokannassa

ACF-kenttäryhmän asettelu sekä kenttien arvot sijaitsevat WordPressin tietokannassa. Kenttien tiedot, johon kuuluu esimerkiksi kenttien tyypit, kuvaukset ja asettelut, sijaitsevat tietokannassa taulussa wp_posts. Yllä olevassa kuvassa 11. on Banneri asettelun tiedot wp_posts-taulussa. Tauluun tallennetaan kaikki komponentin tiedot, kuten ehdollinen logiikka ja komponentin nimi, jotta WordPressin sivunmuokkauksessa voidaan luoda haluttu ACF-kenttärakenne. ACF-kenttien arvot sijaitsevat taulussa wp_postmeta. Taulussa on kaksi eri saraketta, joita ovat meta_key ja meta_value. Sarakkeet toimivat arvopareina ja niillä merkitään kentän avainta ja arvoa. Avain voi olla esimerkiksi komponentin nimitys, kuten ”components_content_1_teksti_palstat” ja arvona voi toimia kentälle syötetty teksti tai muu ACF-kentän arvo.

2.4 Teema

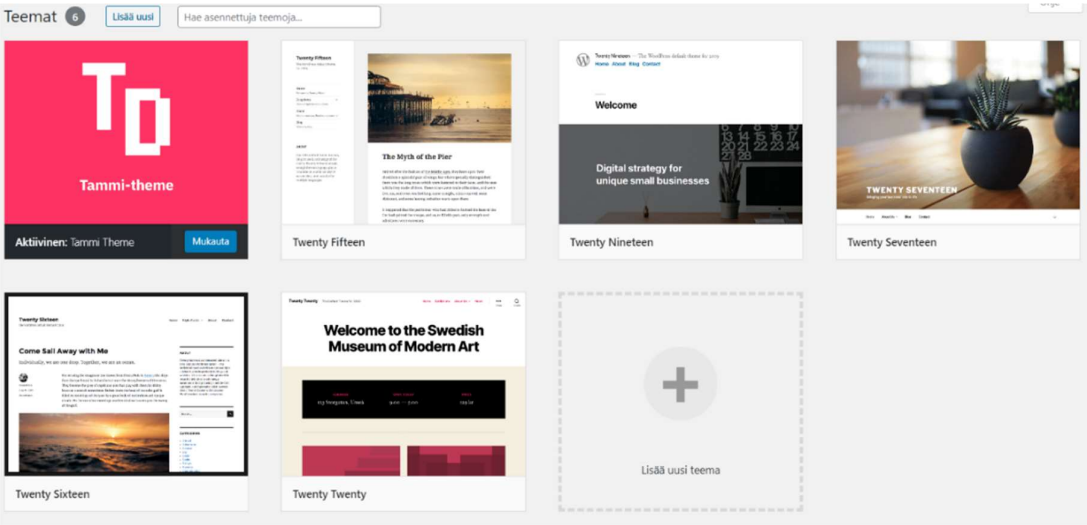
WordPress-teema on tiedostoryhmä, jolla muokataan sivun ulkoasua. Teemalla määritellään selaimelle sivuston sisältö ja kerrotaan, miltä sen tulisi näyttää. Teemoilla voidaan esimerkiksi käyttää erilaisia ulkoasuja, ja voidaan määrittää, millä laitteilla sivun sisältö on näkyvä. (WordPress.org k.) Teema on erillään WordPressin järjestelmän tiedoista. Kun järjestelmään tulee suuria päivityksiä, niin ne eivät vaikuta ulkoasuun. (WordPress.org l.)

Teeman tiedostot sijoittuvat teeman kansioon wp-content-kansion sisään. Teema vähimmillään koostuu kahdesta tiedostosta, jotka ovat index.php ja style.css. Index.php-tiedostolla määritellään sivun oletussivupohja, jota käytetään oletuksena kaikilla sivuilla. Style.css-tiedostoon sijoitetaan sivun tyylimäärittelyt. Teemat tyypillisesti sisältävät kolmea eri tyypin tiedostoja, joita ovat tyyli-, template- ja valinnaiset tiedostot. Template-tiedostot ovat rakennuspalikoita, jotka sisältävät HTML- ja CSS-määrittelyjä. Yleisiä palikoita ovat header ja footer, joita kutsutaan WordPressin sisään rakennetuilla funktioilla, get_header ja get_footer. Valinnaisiin tiedostoihin kuuluvat tiedostot, kuten functions.php. (WordPress.org m.)

Teeman toiminnallisuudet, kuten funktiot ja kirjastojen lisäykset, sijoitetaan functions.php-tiedostoon. Tiedostossa voidaan kutsua PHP:n natiiveja, WordPressin tai käyttäjän luomia funktioita. Tiedosto itsessään muistuttaa lisäosaa, mutta niillä on eroja. Esimerkiksi functions.php-tiedosto vaikuttaa vain teemaan, jonka alla se sijaitsee, ja on käytössä vain silloin, kun kyseinen teema on aktiivinen. Lisäksi tiedostoon ei tarvita uniikkia yläviitekommenttia. Functions.php-tiedostolla voidaan käyttää esimerkiksi koukkuja, joilla vaikutetaan WordPressin toimintaan. Lisäksi tiedostolla voidaan lisätä ominaisuuksia sivulle käyttäen WordPressin sisäänrakennettuja funktioita, kuten uuden kuvakoon lisäys käyttäen add_image_size -funktiota. (WordPress.org n.)

Kuva 10. WordPressin template-hierarkia (Hayes 2017)

Template-tiedostot ovat uudelleenkäytettäviä osia, joilla luodaan WordPress-sivuston ulkonäkö. Template-tiedostolla määritellään sivujen osia, joita voidaan käyttää kaikilla sivuilla, kuten header.php ja footer.php, tai ne voivat olla sivukohtaisia, kuten front-page.php. WordPressissä on tiedostojen hierarkia, joka määrittää, mitä tiedostoa sivu käyttää. (kts. yllä oleva kuva 12.) Hierarkiassa on kolmen eri tason tiedostoja: ensisijainen, toissijainen ja muuttuja-template. Ensisijainen tiedosto kuuluu hierarkian huipulle ja sitä käytetään silloin, kun teemasta ei löydy tarkemmin määriteltyä tiedostoa. Näitä tiedostoja ovat esimerkiksi index.php, single.php ja archive.php. Toissijaisiin tiedostoihin kuuluvat tiedostot, joiden sijainti on jo tarkemmin määritelty, kuten author.php, single-post.php tai category.php. Muuttuja-template määritelmä on kaikista tarkin, sillä sen tiedostonimeen määritellään esimerkiksi ID tai artikkelin tyyppi, kuten author-$id.php tai single-$post-type.php. (WordPress.org o.)



Kuva 11. WordPressin teemavalikko

Teemojen lisäys toimii samalla tavalla kuin lisäosien lisäys: teemoja voidaan lisätä WordPressin teemahakemistosta (kts. yllä oleva kuva 13.), lataamalla WordPressiin pakatun tiedoston tai siirtämällä tiedostot suoraan palvelimen ”themes”-kansioon. Teemoja voidaan muokata WordPressin sisältä käyttäen mukautussivua. Mukautussivun valikosta voidaan esimerkiksi muokata sivun identiteettiä, värejä, valikoita ja kotisivun asetuksia. Sivuston identiteetillä voidaan vaikuttaa sivun logoon, otsikkoon, kuvaukseen ja kuvakkeeseen. Sivuston valikoita voidaan luoda lisää ja vaihtaa niiden sijaintia mukautussivulla. Kotisivun asetuksilla voidaan vaihtaa oletussivu kotisivulle ja artikkeleille. Kotisivuksi voidaan myös laittaa joko staattinen sivu tai uusin artikkeli. Lisäksi mukautussivulle voidaan lisätä myös omaa CSS-koodia.

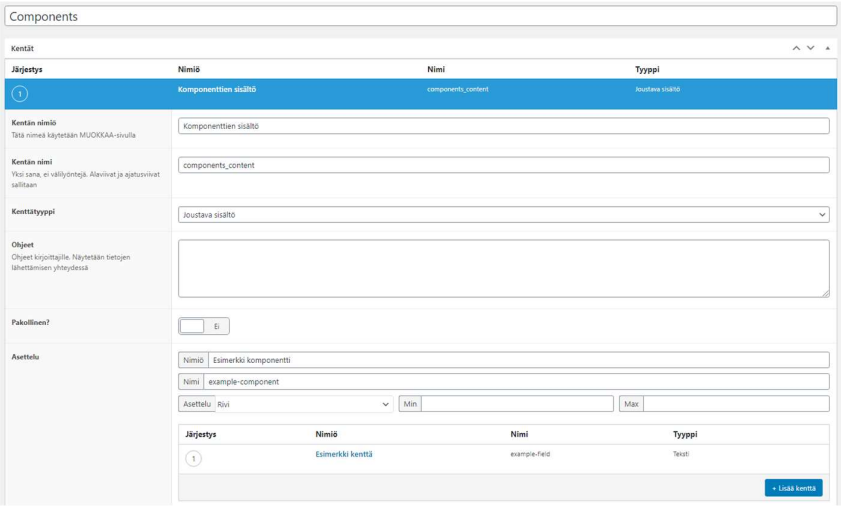
2.5 Komponentti

Komponentti on uudelleenkäytettävä sivuston osa, jolla voidaan lisätä tietty toiminnallisuus sivulle. Toiminnallisuuksia voivat olla esimerkiksi Kuvakaruselli tai upotettu video. WordPressissä komponenteilla tarkoitetaan template-osia, jotka löytyvät teeman sisältä components-kansiosta. Komponentilla voi olla oma tyylimäärittely komponentin tiedostossa, jos tyylimäärittely on komponenttikohtaista. Komponentit ovat valmiita palikoita, joilla muodostetaan sivun sisältö. Komponentteja hallitaan ACF-kentillä, joilla voidaan lisätä komponentteja, vaihtaa komponenttien järjestystä tai poistaa komponentteja. ACF-kentillä välitetään tietoa komponenteille, kuten värivalitsimella voidaan välittää komponentille sen taustan väri.

```php
<?php
/*
* Component name: Example component
* Author: Author name
* Date: 2020-06-23
* Description: Example component
*/
?>
<style>
    /* Custom styles */
</style>

<?php // if component has multiple layouts
if (have_rows('layout-name')) :
    while (have_rows('layout-name')) : the_row(); ?>

        <section class="component-name">
            <!-- Bootstrap layout -->
            <div class="row">
                <!-- component -->
            </div>
        </section>

        <?php }
    endwhile;
endif;?>
```

Kuva 12. Esimerkkikomponentin PHP-tiedosto

Komponentit koostuvat pääosin PHP-tiedostosta sekä ACF-kentästä. PHP-tiedostossa määritellään komponentin ulkoasu, HTML-, PHP- ja komponenttikohtainen CSS-koodi. Yläpuolella olevassa kuvassa 14. on esimerkkikomponentin PHP-tiedosto. Komponentin tiedoston alkuun olisi hyvä lisätä kommentoituna komponentin tiedot, joita ovat komponentin nimi, kehittäjän nimi, päiväys ja komponentin kuvaus. Kommentoidun osuuden jälkeen sijoitetaan komponentille kuuluva tyylielementti. ACF-kentän käsittely sijoitetaan silmukkarakenteen ympärille, jossa testataan, onko kyseisellä ACF-kentällä rivejä. Silmukkarakenteen jälkeen voidaan sijoittaa komponentin sisäinen rakenne HTML-koodiin. Komponentin HTML-elementtiin on hyvä laittaa luokaksi komponentin nimi, jolloin samojen komponenttien tyyliä voidaan muokata tarvittaessa teeman CSS-tiedostossa.



Kuva 13. Esimerkkikomponentin ACF-kenttäryhmä

ACF kentällä määritellään komponentille kuuluvat kentät. ACF-kenttäryhmän nimeksi laitetaan ”Components” ja sen sisälle luodaan joustava sisältö -kenttä, jonka nimeksi laitetaan esimerkiksi Komponenttien sisältö. Yläpuolella olevassa kuvassa 15. on luotu kenttäryhmä esimerkkikomponentille. Komponentteja lisätään kenttäryhmään luomalla uusia asetteluja. Asetteluille määritellään niiden nimiö, nimi, asettelun ulkonäkö ja kentät. Asettelun nimiöllä tarkoitetaan komponentin nimeä, joka näkyy vain sivunmuokkausnäkymässä. Asettelun nimeä käytetään, kun viitataan komponenttiin template-tiedostossa. Komponentin PHP-tiedoston nimen tulisi sisältää komponentin asettelun nimen, kuten esimerkki komponentin PHP-tiedoston nimi tulisi olla ”component-example-component.php”.

```php
<?php

ini_set('display_errors', 1);
ini_set('display_startup_errors', 1);
error_reporting(E_ALL);

if (have_rows('components_content')) {
    while (have_rows('components_content')) {
        the_row();

        get_template_part('components/component', get_row_layout());
    }
}

?>
```

Kuva 14. Router.php-tiedosto

Komponenttien tiedostoon ohjaus tapahtuu router.php-tiedostossa (kts. yläpuolella oleva kuva 16.). Joustava sisältö -kentän sisältö tarkistetaan silmukkarakenteessa have_rows -funktiolla kuvassa rivillä 7 ja 8. Sisältö asetetaan rivimuuttujaan rivillä 9, jos kentällä on rivejä. Kun komponentin asettelu on valittu aktiiviseksi sivun muokkauksessa, niin router.php ohjaa asettelun oikeaan tiedostoon käyttämällä funktiota, get_template_part. Funktiolle välitetään parametreina tiedoston sijainti, johon kuuluu tiedoston etuliite ja nimi, käyttäen funktiota get_row_layout. Funktiolla get_template_part voidaan hakea template-osa useaan kertaan, jolloin samaa komponenttia voidaan käyttää sivulla useaan otteeseen. Router.php-tiedostoon on lisätty virheiden käsittelyä, jotta mahdolliset ongelmat olisi helpompi huomata.

Komponenttien kehityksessä on otettava huomioon koodin uudelleenkäytettävyys ja se, että komponentit jaetaan tarpeeksi pieniin osiin. Pienillä komponenteilla varmistetaan se, ettei komponentin sisäinen rakenne ole liian monimutkaista. Komponenttien tulisi olla myös yhteneviä muiden komponenttien kanssa, jotta niiden käyttäminen ja uusien luominen olisi mahdollisimman helppoa. Komponenttien kehityksessä olisi myös hyvä noudattaa samankaltaista syntaksia esimerkiksi kommenttien ja HTML-rakenteen kanssa.

2.6 Kirjaston levitystavat

Komponenttikirjastoa on mahdollista jakaa käyttäjille kolmella eri tavalla: WordPress-lisäosana, tiedostojen siirtona teemaan ja valmiina teemana, jossa komponentit tulevat mukana. Komponenttikirjaston levityksessä tulee ottaa huomioon esimerkiksi se, että voiko kirjaston lisätä niin valmiille kuin kehityksessä oleville sivuille. Komponenttien levityksessä pitäisi myös minimoida mahdolliset virheet, joita sen levitykseen mahtaa liittyä. Komponenttikirjaston levitystavan vertailussa on otettu huomioon vain nämä kolme tapaa.

Komponenttikirjaston levitys lisäosalla mahdollistaa kirjaston jakamisen kaikille WordPress-sivuille, joille voidaan asentaa lisäosia. Komponenttien lisäys teemaan tapahtuu lisäosan asennuksen avulla, jolloin käyttäjän itse ei tarvitse siirrellä tiedostoja teeman kansioon. Käyttäjän vastuulle tulee komponenttikirjaston lisäys niin sivuston oletussivulle, kuin tiettyjen kirjastojen lisäys functions.php-tiedostoon. Lisäosan kehitys vaatii eniten työtä kehittäjältä, mutta voi olla käyttäjälle helppokäyttöisin.

Kirjastoa voidaan myös jakaa käsin siirtämällä tiedostoja teeman kansioihin. Kirjasto voidaan lisätä tällä tavalla sivulle, jolla on jo valmis teema. Käsin tiedostojen siirrossa voi tapahtua virheitä, kuten tiedoston siirtäminen väärään kansioon tai jonkin tiedoston puuttuminen. Tiedostojen siirrossa voi tulla myös ongelmia, jos esimerkiksi teeman kansioon ei ole käyttöoikeutta. Tiedostojen lisäksi kirjasto tulee lisätä oletussivulle ja functions.php-tiedostoon.

Komponenttikirjastoa voidaan myös jakaa pohjateemana, jossa komponenttien tiedostot ovat valmiina teemassa. Tämä levitystapa on mahdollinen vain uusille sivuille, sillä uuden teeman tuominen sivulle korvaa aiemman teeman. Lisäksi pohjateema voi olla hieman rajoittava alusta teeman kehitykseen, sillä teeman pohja, kuten HTML-rakenne tai CSS-määritykset, on jo valmiiksi määritelty.

3 KOMPONENTTIKIRJASTO

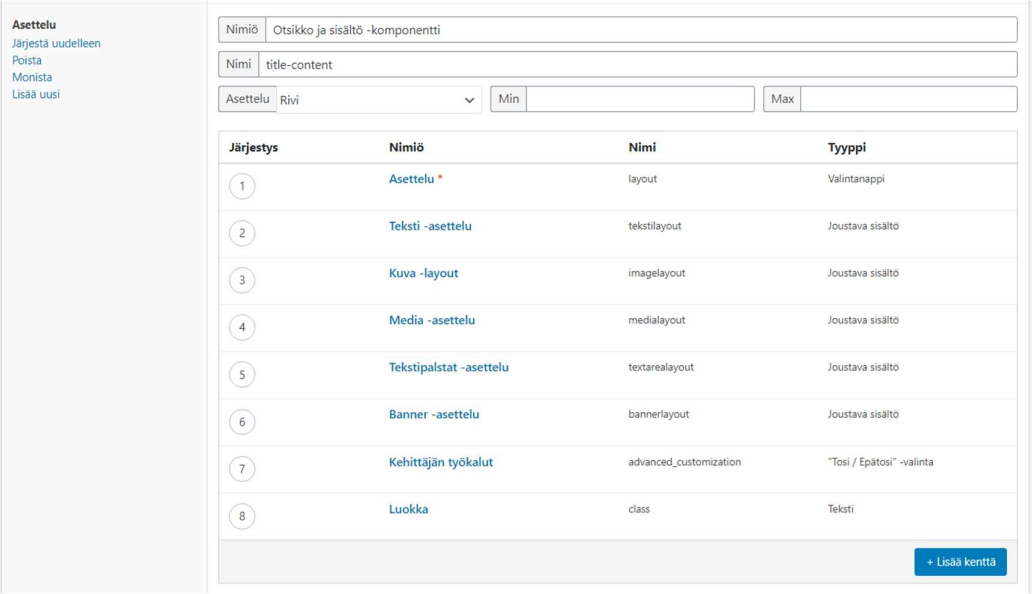
3.1 Komponenttien suunnittelu ja kehitys

Komponenttikirjaston tarkoitus on helpottaa uusien ominaisuuksien lisäystä sivuille käyttäen valmiita komponentteja. Alla olevassa kuvassa 17. on luotu sivun asettelu käyttäen komponentteja Otsikko ja sisältö sekä Herokuva. Komponentit valittiin tarkastelemalla ominaisuuksia ja toiminnallisuuksia olemassa olevilta sivuilta ja valitsemalla, mitkä näistä ominaisuuksista ovat tärkeimmät. Komponenteiksi valittiin Herokuva, Otsikko ja sisältö, Kuvakaruselli, Yhteystiedot, Artikkelinosto sekä Lomake. Komponenteissa pyrittiin ottamaan huomioon kustomoitavuus ja sopivuus erilaisille sivuille.

Kuva 15. Esimerkkisivu luotu komponenteilla

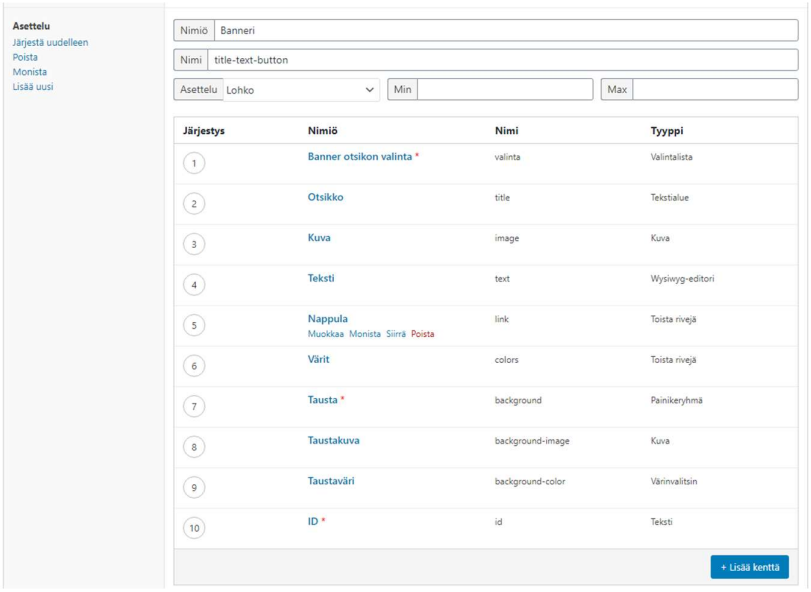
3.1.1 ACF-kentät

Komponentteja varten luotiin kenttäryhmä Components, jonka sisällä on kenttä, Komponenttien sisältö, joka on tyypiltään ”joustava sisältö”. Jokaiselle komponentille luodaan oma asettelu joustavaan sisältöön. Komponenteille asetetaan nimi, nimiö ja valitut kentät. Tietyille komponenteille on luotu mahdollisuus erilaisiin asetteluihin, kuten komponentille Otsikko ja sisältö. Komponentin asettelun valintaan käytetään kenttää valintanappi, jotta pystytään käyttämään oikeaa tiedostoa komponentin luomiseen. Komponenttien sijaintiin valitaan sivu sekä artikkeli, jotta komponentit ovat saatavilla molemmilla artikkelityypeillä.



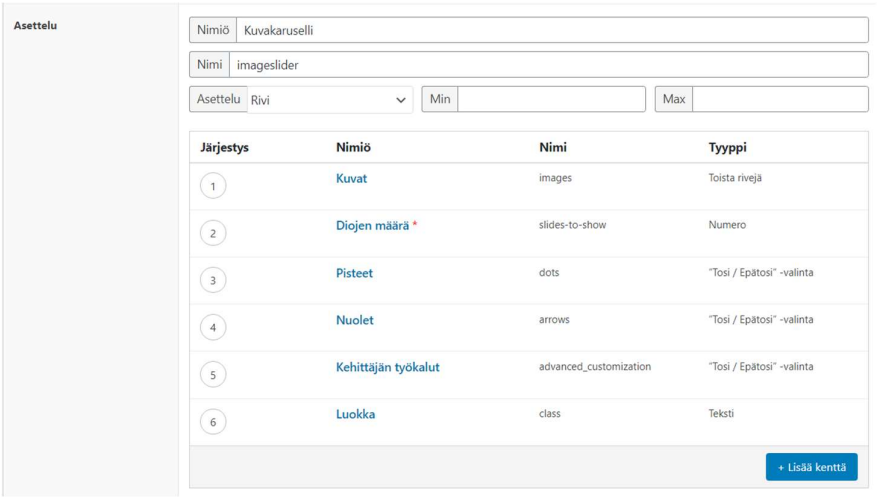
Kuva 16. Otsikko ja sisältö -komponentti

Komponenttien asetteluun vaikuttaa komponenttien monimutkaisuus. Esimerkiksi Herokuva-komponentilla on kenttänä vain kuva, joka palauttaa valitun kuvan, kun the_field funktiota kutsutaan. Yllä olevassa kuvassa 18. on Otsikko ja sisältö -komponentin asettelu, joka on monimutkaisempi kuin Herokuvan asettelu. Otsikko ja sisältö -komponentti pitää sisällään viisi eri asettelua, joita ovat teksti, kuva, media, tekstipalstat ja banneri. Asetteluilla valitaan kyseisen komponentin ulkonäkö ja sisältö. Esimerkiksi Teksti ja Tekstipalstat -asettelut eroavat ulkonäkönsä puolesta, sillä Teksti-asettelussa tekstikappale on yhtenäinen alue, kun tekstipalstoilla voidaan luoda monta tekstikappaletta vierekkäin. Asettelun valintaan käytetään valintanappia Asettelu, jonka arvon valitseminen on pakollista. Valintanappiin on yhdistetty ehdollista logiikkaa, jolla voidaan näyttää vain tietty asettelu muokkaussivulla. Asetteluiden tyypiksi on valittu joustava sisältö. Komponenttien sisältä löytyy totuusarvovalinta, Kehittäjien työkalut, jolla voidaan asettaa komponentille ylimääräinen luokka. Luokka sijoitetaan komponentin ympäröivään HTML-elementtiin.



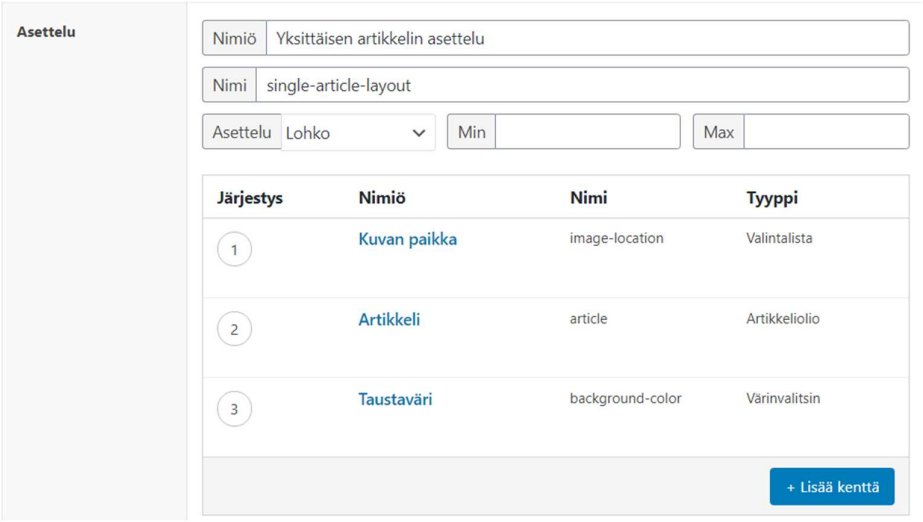
Kuva 17. Banneri-asettelu

Komponenteilla voi olla sisäisiä valintoja, joilla vaikutetaan komponentin ulkonäköön. Esimerkiksi Otsikko ja sisältö -komponentin sisäisellä asettelulla Banneri (kts. yllä oleva kuva 19.) on laaja määrä muokkausvaihtoehtoja. Komponentin kentillä voidaan vaikuttaa komponentin ulkonäköön valinnoilla, kuten otsikon valinta, jossa vaihtoehtoina ovat kuva tai otsikko, tai CSS-määrityksiin, kuten värit, joilla voidaan vaihtaa tekstin, taustan tai nappulan värejä. Komponentin ulkonäön valinnoilla voidaan tehdä komponentista kustomoitavampi, sillä komponentin ulkonäkö ei ole sidottu vain yhteen asetteluun. Lisäksi muokattava CSS-koodi mahdollistaa komponentin kustomoinnin laajemmin siten, ettei erillisiä tyylimäärityksiä tarvitse laittaa style.css tiedostoon. Banneri asetteluun on tarvittavaa sijoittaa yksilöivä tunniste jokaiselle bannerille, jotta CSS-määritykset koskevat vain yhtä banneria kerrallaan.



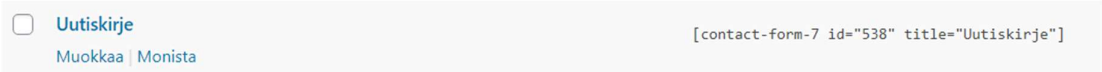
Kuva 18. Kuvakaruselli-komponentti

Komponenttien kentillä voidaan määrittää myös komponentin sisäisiä asetuksia. Esimerkiksi yläpuolella olevassa kuvassa 20. on Kuvakaruselli-komponentti, jonka kentillä määritellään karusellin asetukset. Kuvakarusellin luomiseen käytetään jQuery:lla luotua Slick-karusellia, jonka on kehittänyt Ken Wheeler. Slick-karuselli toimii JavaScript-kielellä, mutta sen asetuksia voidaan muokata käyttäen jQuery-kirjastoa. Kuvakaruselli-komponentilla asetetaan Slick-karusellin asetuksia käyttäen kenttiä, joita ovat diojen määrä, pisteet ja nuolet. Slick-karusellissa on paljon erilaisia asetuksia, joilla voidaan muokata karusellin ulkonäköä ja toiminnallisuuksia, kuten karusellille voidaan määrittää, että se automaattisesti vaihtaa dioja tietyn ajan kuluessa. Komponenttiin on otettu vain Slick-karusellin tärkeimmät ja yleisimmät ominaisuudet, joita tarvitaan karusellin luomiseen.



Kuva 19. Artikkelinoston asettelu

Komponenttien kentillä voidaan myös välittää komponentille niin WordPress-sivuolio, kuin ajettava PHP-lyhytkoodi. Komponentilla Artikkelinosto voidaan nostaa sivulle yksi tai useampi artikkeli riippuen asettelusta, jonka kenttänä on valintanappi Asettelu. Asettelun mukaan valitaan joko yksittäinen artikkeli (kts. yläpuolella oleva kuva 21.) tai asettelun sisältä löytyy ”toista rivejä” -kenttä, jolle voidaan valita vähintään kaksi ja enintään neljä artikkelioliota. Komponentin asettelulla valitaan myös kuvan tai otsikon paikka, jolla vaikutetaan komponentin ulkonäköön. Yksittäiselle artikkelille voidaan asettaa myös CSS-määrittelynä taustaväri, jos on valittu tietty kuvan paikka. Lomake-komponentille välitetään PHP-lyhytkoodi, jonka kentän arvona on tekstikenttä. Lomakkeen luomiseen käytetään lisäosaa Contact Form 7, jolla luodaan lomakkeita lisäosan omalla editorilla. Lomakkeen luonnin jälkeen sen tiedoista löytyy lyhytkoodi, kuten alla olevassa kuvassa 22. Lomake komponentilta löytyy myös useampi asettelu, joita ovat Tekstikenttä ja lomake, Kuva ja lomake, Kapea lomake, Leveä lomake sekä uutiskirje.



Kuva 20. Contact Form 7 -lyhytkoodi

3.1.2 Kehitetyt komponentit

Komponenttikirjaston käyttöön vaaditaan tiettyjen kirjastojen ja tyylitiedostojen lisäystä functions.php-tiedostoon. Näitä skriptejä ovat esimerkiksi Bootstrap, jota käytetään sivuston ulkonäköön liittyvässä asettelussa, jQuery ja Slick. Lisäksi Kuvakaruselli-komponenttia varten tulee lisätä functions.php-tiedostoon uusia kuvakokoja, joita käytetään esimerkiksi silloin, kun käyttäjän laite on mobiililaite. Komponenttikirjaston komponentteihin tarvitaan Advanced Custom Fields pro -lisäosa, jotta voidaan käyttää joustava sisältö -kenttää. Lisäksi tiettyihin komponentteihin, kuten lomakkeelle, vaaditaan lisäosia, jotta komponentit toimivat oikein.

```php
<?php
// Layout value
$layout = get_sub_field('layout');

if ($layout == "Kuva") {
    require('title-content/component-title-image.php');
}

else if ($layout == "Teksti") {
    require('title-content/component-title-text.php');
}

else if ($layout == "Media") {
    require('title-content/component-title-video.php');
}

else if ($layout == "Tekstipalstat") {
    require('title-content/component-title-textareas.php');
}

else if ($layout == "Banner") {
    require('title-content/component-title-text-button.php');
}
?>
```

Kuva 21. Komponentin asettelun testaus

Komponentit, joilla on useampi eri asettelu, vaativat komponentin tiedostossa testausta siitä, että mikä asettelu on aktiivinen. Yläpuolella olevassa kuvassa 23. testataan komponentin Otsikko ja sisältö -asettelua, ja asettelun mukaan otetaan oikea tiedosto mukaan teemaan. Komponentin asettelun testaukseen käytetään kenttää Asettelu, jonka arvo sijoitetaan tiedoston alussa muuttujaan. Komponentin asetteluiden tiedostot ovat selkeyden vuoksi sijoitettu kansioon, jonka nimeksi on asetettu komponentin nimi, esimerkiksi Otsikko ja sisältö -komponentin asetteluiden tiedostot löytyvät kansiosta ”title-content”. Komponentin asettelut haetaan käyttäen funktiota require, jotta voidaan hakea tarvittaessa sama asettelu uudelleen samalla sivulla. Tiedostojen hakuun voidaan käyttää myös funktiota, get_template_part, jonka käyttöä suositellaan silloin kun haetaan teeman tiedostoja.

```css
<style>
    .slider-container {
        width: 100vw; /* make it 100% of the viewport width (vw) */
        margin-left: calc((100% - 100vw) / 2);
    }
    .slider-container img {
        margin: 10px;
    }
    .slick-prev {
        left: 10px;
        z-index: 1;
    }
    .slick-next {
        right: 10px;
    }
</style>
```

Kuva 22. Kuvakaruselli-komponentin tyylimääritykset

Komponenttitiedoston alkuun sijoitetaan kommenttiin komponentin tiedot. Komponentti kohtaiset CSS-tyylimääritykset sijoitetaan kommentin jälkeen tiedostoon style-elementtiin. Yläpuolella olevassa kuvassa 24. on komponentin Kuvakaruselli tyylimääritykset. Tyylimäärityksiin kuuluu esimerkiksi Kuvakarusellin leveyden määrittäminen sekä eteenpäin ja

taaksepäin nuolien muokkaaminen. Kuvakarusellin CSS-määritykset vaikuttavat vain tähän komponenttiin, jolloin tyylimäärityksien sijainti on komponentin tiedostossa. Jos komponentin ulkoasua halutaan muokata, niin silloin käytetään komponentille annettavaa luokkaa ”Kehittäjien työkalut”-kentän kautta tai käytetään komponentin ympäröivään HTML-elementtiin sijoitettua luokkaa. Esimerkiksi Herokuva komponentin ympäröivän HTML-elementin luokka on nimellä ”hero-image”, ja se sijaitsee section-elementissä. Jälkikäteen kehitetyt CSS-tyylimääritykset sijoitetaan teeman style.css-tiedostoon.

```
<style>
    .banner {
        width: 100vw;
        margin-left: calc((100% - 100vw) / 2);
    }
    .banner-container {
        padding: 40px 0;
        margin-bottom: 20px;
    }
    .banner-text {
        width: 50%;
        margin: auto;
        text-align: center;
    }
    .banner-image {
        margin: auto;
    }
    <?php echo $currentID; ?> .link_button {
        padding: 10px 20px;
        background-color: <?php echo $buttonBackground; ?>;
        color: <?php echo $buttonText; ?>;
    }
    <?php echo $currentID; ?> .link_button:hover {
        text-decoration: none;
        background-color: <?php echo $buttonBgHover; ?>;
        color: <?php echo $buttonTextHover; ?>;
    }
    @media only screen and (max-width: 600px) {
        .banner-text {
            width: 90%;
        }
    }
</style>
```

Kuva 23. Banneri-asettelun CSS-määritykset.

Komponentin sisäisiin tyyliasetuksiin voidaan sijoittaa myös kenttien arvoja. Asettelussa Banneri määritellä CSS-tyylimääritykset esimerkiksi komponentin väreihin käyttäen kenttien arvoja. Banneri asettelun alta löytyy Värit-kenttä, joka on tyypiltään ”toista rivejä” -kenttä. Väreistä löytyy viisi eri valintaa, jonka kenttien arvot ovat värivalitsimia. Värivalitsimilla valitaan värit tekstille ja painikkeelle, jolle valitaan taustan ja tekstin värit sekä värit myös silloin, kun painikkeen päällä pidetään kursoria (hover). Väriarvot välitetään komponentille värivalitsimen kautta, joka palauttaa arvokseen värin Hex-koodina. Esimerkiksi valkoisen värin Hex-koodi on #FFFFFF. Bannerin värit asetetaan oikeisiin kohtiin style-elementissä ja käytetään PHP-funktiota echo tulostamaan värin Hex-koodi. Tyylimäärityksiä ennen on käyty Värit-kenttä läpi silmukkarakenteessa ja asetettu arvot muuttujiin. Tyylimääritykset, jotka asetetaan käyttäen kenttien arvoja, voidaan sijoittaa myös suoraan HTML-elementteihin käyttäen parametriä style (kts. yllä oleva kuva 25.). Banneri-asettelun

CSS-määrityksiin käytetään ID-kenttää, jotta Bannerin CSS-määritykset vaikuttavat vain yhteen banneriin.

```
22  if( have_rows('personlayout') ): ?>
23      <?php while( have_rows('personlayout') ): the_row();
24      if (get_row_layout() == 'person-info') { ?>
25
26      <section class="person-info <?php if ($class) { echo $class; } ?>">
27          <?php if (get_sub_field('valinta')) { ?>
28              <div class="row">
29                  <div class="col-12">
30                      <h1><?php the_sub_field('title'); ?></h1>
31                  </div>
32              </div>
33          <?php }
34          $imagechoice = get_sub_field('personimagechoice');
35          $titlechoice = get_sub_field('persontitle'); ?>
36
37          <div class="row">
38              <?php while(have_rows('persons')) {
39                  the_row(); ?>
40                  <div class="col-lg-3 col-md-6 col-sm-12 person">
41                      <?php if ($imagechoice) { ?>
42                          <img src="<?php the_sub_field('image'); ?>" alt="">
43                      <?php } ?>
44                      <p><?php the_sub_field('name'); ?></p>
45                      <?php if ($titlechoice) { ?>
46                          <p><?php the_sub_field('persontitle'); ?></p>
47                      <?php } ?>
48                      <a href="mailto:<?php the_sub_field('email'); ?>"><?php the_sub_field('email'); ?></a><br>
49                      <a href="tel:<?php the_sub_field('tel-number'); ?>"><?php the_sub_field('tel-number'); ?></a>
50                  </div>
51              <?php } ?>
52          </div>
53      </section>
54      <?php }
55      endwhile;
56  endif;
```

Kuva 24. Komponentin sisältö

Komponentin sisältö rakentuu muutamasta osasta: komponentin sisällön testaus, tietyn asettelun testaus, HTML-rakenne ja ACF-kenttien sijoitus HTML-rakenteen sisään. Komponentin sisältöä testataan yllä olevassa kuvassa 26. riveillä 22 ja 23. Sisällön testaukseen käytetään PHP:n silmukkarakennetta, jossa ensin testataan, onko sisällöllä rivejä ja sitten asetetaan rivi muuttujaan sisältö, niin kauan kun rivejä on. Rivillä 24 testataan, onko valittu asettelu aktiivisena. Näiden PHP-silmukoiden jälkeen luodaan HTML-rakenne. Komponenteissa ympäröidään sisältö section-elementillä. Elementille annetaan komponentin luokka ja asetetaan valinnainen luokka, jos ”Kehittäjien työkalut” on valittu ja luokka kenttä on täytetty. HTML-rakenteeseen on käytetty Bootstrap-asettelua, jolla tehdään sisällöstä responsiivinen ja skaalautuva niin työpöytä- kuin mobiilikokoon. Bootstrap-asettelulla voidaan luoda rivejä ja sarakkeita, joilla määritellään sivun asettelu. Sarakkeen arvoksi voidaan asettaa ruudun koko, joka voi olla iso, keskikokoinen tai pieni, ja sarakkeiden määrä välillä yksi ja kaksitoista. ACF-kentät sijoitetaan HTML-rakenteen sisään niiden oikeille paikoille. Kenttien arvot saadaan käyttäen funktioita get_sub_field, jolla haetaan kentän arvo, ja the_sub_field, jolla tulostetaan kentän arvo. Esimerkiksi rivien välillä 41 ja 49 asetetaan kenttien arvot HTML-elementteihin käyttäen sijoitukseen PHP-lauseita.

```
<!-- Slick init -->
<script type="text/javascript">
    jQuery(document).ready(function(){
        jQuery('.slider-container').slick({
            dots: <?php echo json_encode(get_sub_field('dots')); ?>,
            arrows: <?php echo json_encode(get_sub_field('arrows')); ?>,
            infinite: true,
            centerMode: <?php echo json_encode($centerMode); ?>,
            slidesToShow: <?php echo get_sub_field('slides-to-show'); ?>
        });
    });
</script>
```

Kuva 25. jQuery:n asetukset

Kenttien arvoja voidaan käyttää myös JavaScriptin sisällä. Kuvakaruselli-komponentin kentät, kuten nuolet ja pisteet, ovat totuusarvokenttiä, joilla muutetaan Kuvakarusellin konfiguraatiota. Slick-kuvakarusellin konfiguraatiota muokataan JavaScript-elementin sisällä olevalla jQuery-funktiolla yllä olevassa kuvassa 27. Funktiolla kutsutaan tiettyä HTML-elementtiä käyttäen luokkaa ”slider-container”. Sitten kutsutaan slick-karusellin funktiota, jolle välitetään Kuvakarusellin konfiguraatio. ACF-kenttien arvot sijoitetaan konfiguraatioon käyttäen PHP-lausetta. Totuusarvokenttien arvot tulee muuttaa JSON-muotoon, jotta ne toimivat JavaScriptin sisällä. Konfiguraatioon voidaan sijoittaa muitakin Slick-karusellin määrityksiä, mutta komponenttiin on valittu vain tärkeimmät asetukset. Kuvakarusellia varten tulee teemaan lisätä Slick-kansio, joka sisältää tarvittavat tiedostot karusellin toimimiseen.

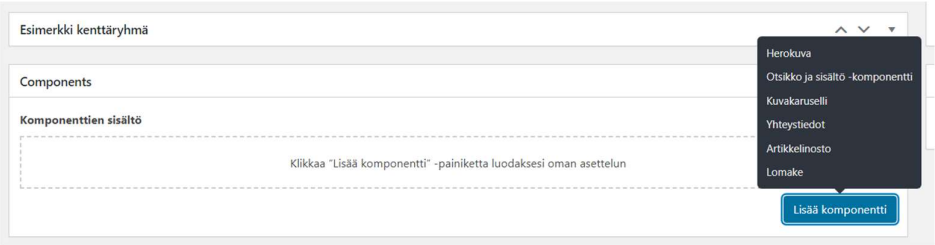
3.2 Komponenttien pilotointi

Komponenttien pilotoimisessa otetaan kaksi näkökulmaa, jolla testataan lopulliset komponentit: kehittäjän- ja asiakkaan näkökulma. Kehittäjän näkökulmalla tarkoitetaan komponenttien muokkausta ja käsittelyä sivunmuokkausnäkymässä. Asiakkaan näkökulmalla tarkoitetaan komponenttien ulkonäköä ja soveltuvuutta niin työpöytä kuin mobiilikokoon. Sivun testauksessa käytetään hyödyksi Google Chromen kehittäjän työkaluja simuloimaan näkymä mobiiliin.

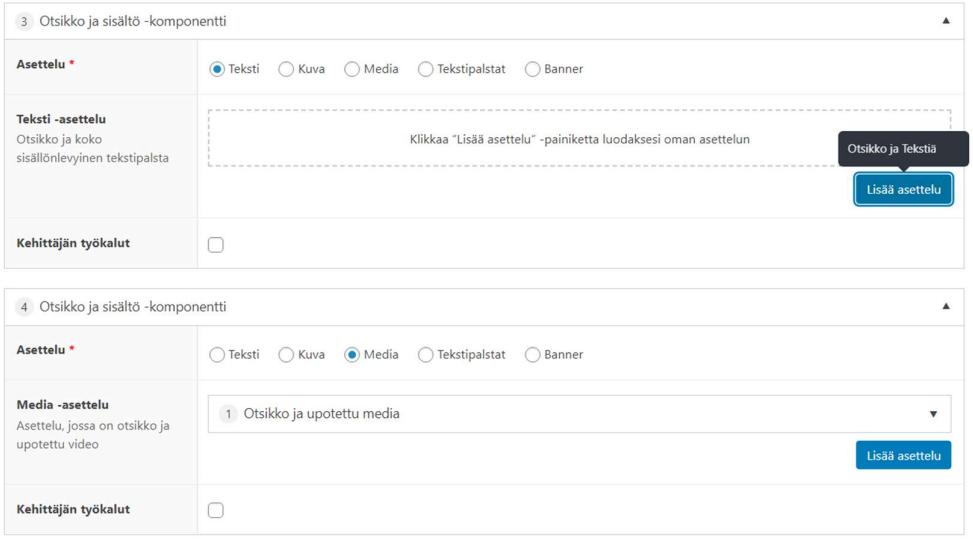
3.2.1 Testaus kehittäjän näkökulmasta

Komponenttien testauksessa otetaan huomioon WordPress sivunmuokkauksen näkymä ja sivun muokkaus, sillä sisältö rakennetaan käyttäen komponentteja. Komponentit lisätään sivun muokkauseditorin alta, kohdasta Components. Komponentit lisätään sivulle käyttämällä painiketta ”Lisää komponentti”, kuten alla olevassa kuvassa 28. on esitetty. Kaikkien komponenttien nimet nousevat valikkoon ja painamalla nimeä voidaan lisätä komponentti. Komponenttien järjestystä voidaan muuttaa raahaamalla komponentteja

ylöspäin tai alaspäin. Komponenttien nimen vasemmalla puolella on numero, jolla merkitään komponenttien järjestystä, kun ne on asetettu sivulle.

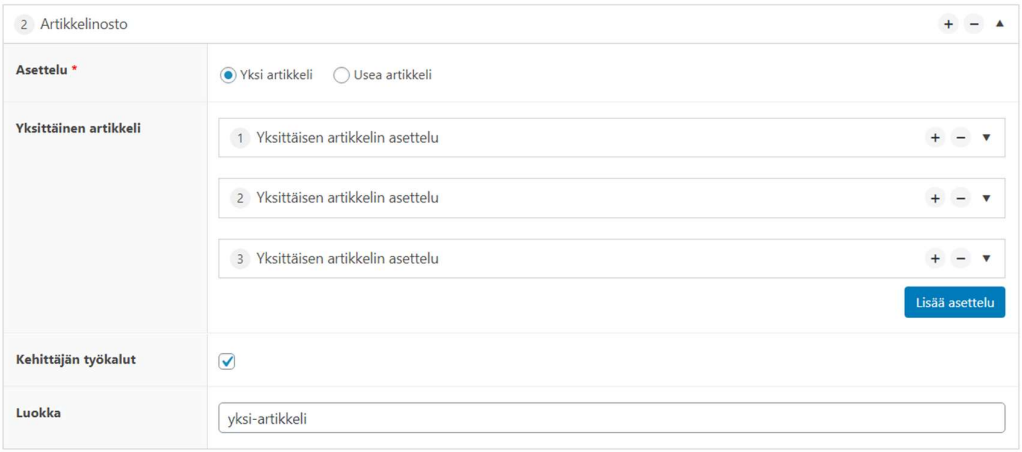


Kuva 28. Komponentin lisääminen sivulle



Kuva 26. Teksti-asettelu Otsikko ja sisältö -komponentissa

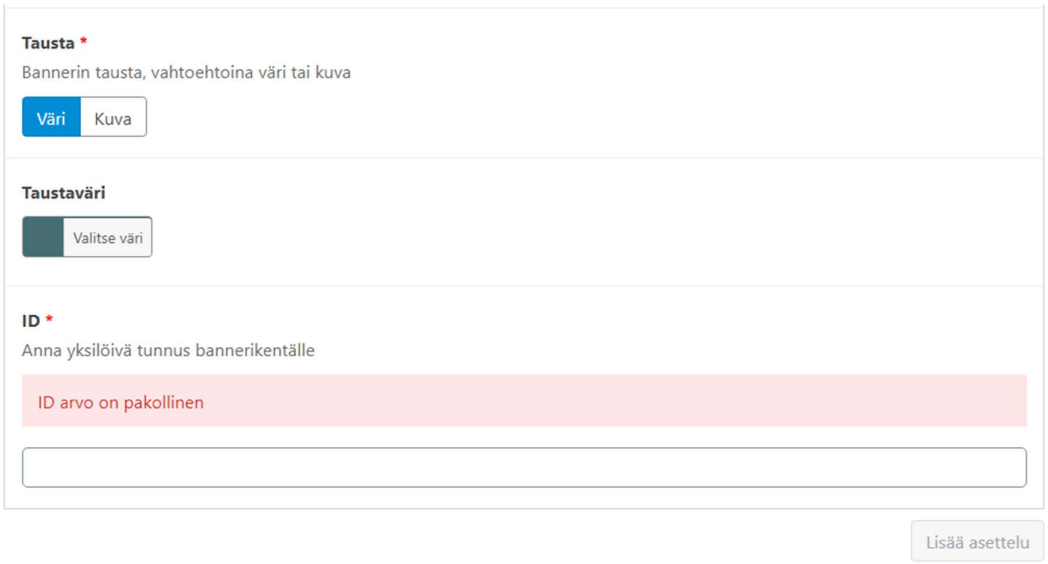
Komponenttien, kuten Otsikko ja sisältö tai Lomake, muokkauksessa pitää ottaa huomioon oikean asettelun valinta. Komponentin asettelu vaihtuu Asettelu-kentän mukaan. Yllä olevassa kuvassa 29. on Otsikko ja sisältö -komponenttia asetettu sivulle kaksi kappaletta ja niille on valittu eri asettelut. Ylemmässä komponentissa on asetteluksi valittu Teksti, kun taas toisen komponentin asettelu on Media. Asettelurivi, johon kuuluu niin asettelu itse, nimi sekä kuvaus, vaihtuvat riippuen käyttäjän valinnasta. Samaan komponenttiin voidaan lisätä sama asettelu useampaan kertaan, mutta eri asettelua varten tulee luoda erillinen komponentti.



Kuva 27. Komponentin valinnainen luokka

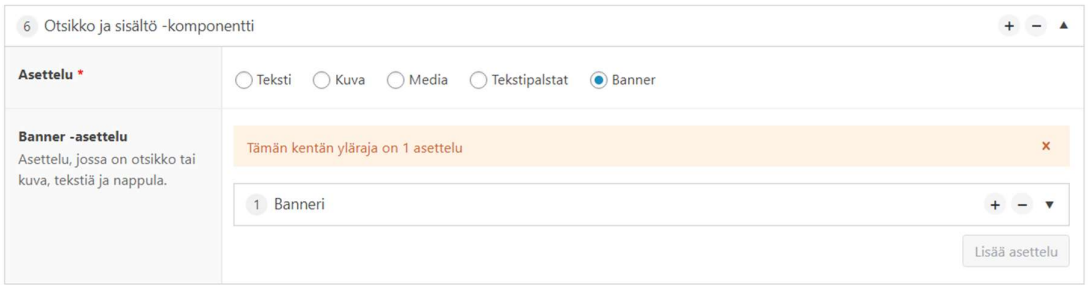
Komponentille voidaan asettaa valinnainen luokka käyttäen ”Kehittäjän työkalut” kenttää. Kenttä toimii ehdollisella logiikalla siten, että jos se on valittu, tulee kenttä ”Luokka” näkyviin. Esimerkiksi yllä olevassa kuvassa 30. on asetettu komponentille luokaksi ”yksi-artikkeli”. Luokka sijoitetaan komponentin ympäröivään elementtiin, jotta komponenttia voidaan muokata tarvittaessa esimerkiksi style.css-tiedostossa. Alla olevassa kuvassa 31. on näkymä Google Chromen kehittäjän työkaluista, joissa on valittu komponentin ympäröivä elementti aktiiviseksi. Elementin luokkana on komponentin luokan lisäksi luokka ”yksi-artikkeli”. Sama luokka sijoitetaan myös muihin asetteluihin, jos ne ovat samassa komponentin alustuksessa, kuten kuvassa 30. olevat ”Yksittäisen artikkelin asettelu” -asettelut.

Kuva 28. Google Chromen kehittäjän työkalut



Kuva 29. Kentän arvon testaus

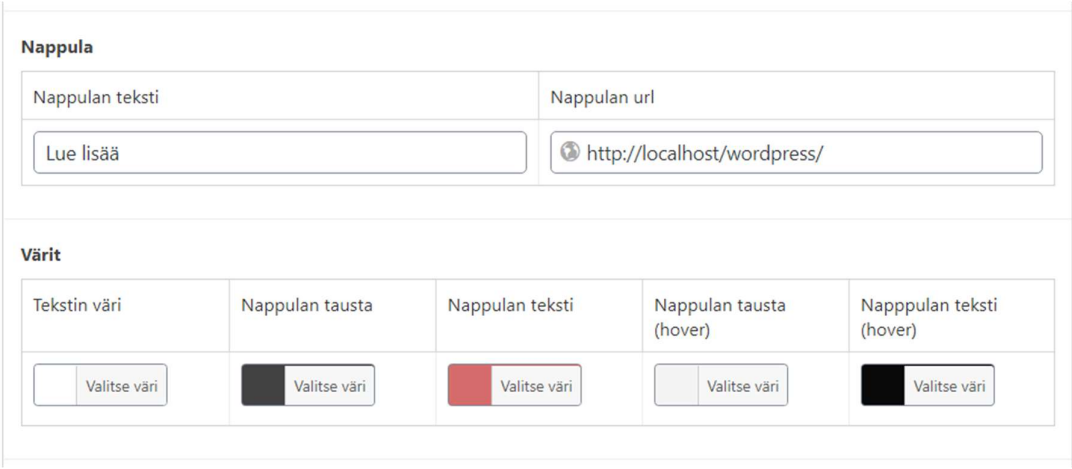
ACF-kentälle voidaan antaa arvoksi kentän pakollisuus, jolloin kenttä tulee täyttää, jotta komponentti voidaan lisätä sivulle. Banneri-asettelulla on ID-kenttä, jolla määritellään asettelun yksilöivä tunnus, kuten yllä olevassa kuvassa 32. ID-kenttä on pakollinen, jotta Bannerin yksilöivät CSS-määritykset vaikuttavat vain yhteen Banneriin. Kun sivu tallennetaan tai julkaistaan, niin testataan kaikki pakolliset kentät, että sisältävätkö ne arvon. Jos kenttä on tyhjä, niin kentän yläpuolelle ilmestyy varoitus. Sivua ei voida julkaista, jos pakollisia kenttiä on jätetty tyhjäksi.



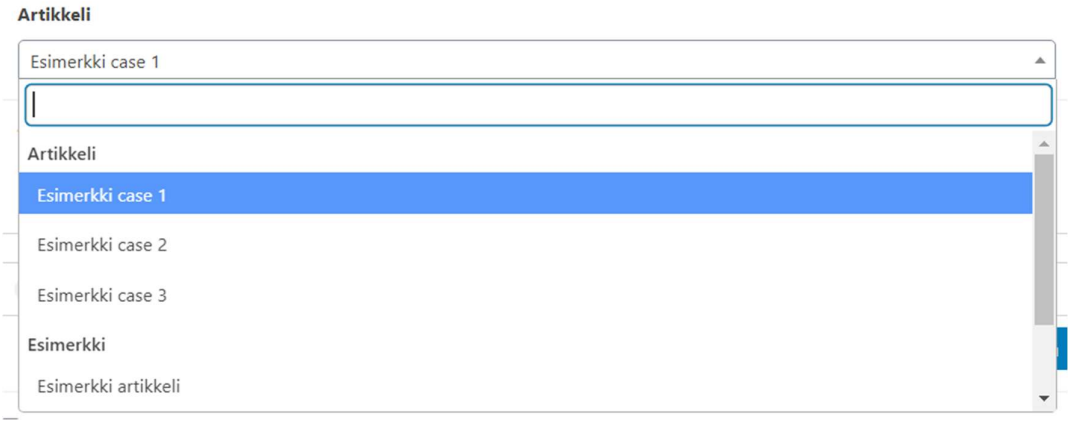
Kuva 30. Maksimimäärän testaus

Kentille, jotka ovat tyypiltään asettelu, voidaan asettaa rivien minimi- ja maksimiarvot. Esimerkiksi ”joustava sisältö”- ja ”toista rivejä” -kentille voidaan asettaa rivien määrät. Yllä olevassa kuvassa 33. on Banneri-asettelu, jossa on asetettu jo yhden Bannerin määritykset. Bannerin kentälle ”joustava sisältö” on asetettu maksimimääräksi yksi Banneri, jolloin yhteen komponenttiin saadaan asettaa vain yksi Banneri. Kenttien minimi- ja maksimiarvot voidaan myös asettaa samaksi arvoksi, jolloin kenttiä on sivunmuokkauksessa aina sen arvon verran. Alla olevassa kuvassa 34. on Bannerin Värit- ja Nappula-kentät, joille

on asetettu minimiksi ja maksimiksi yksi. Kenttiä on sivunmuokkauksessa vain yhdet, vaikka kentät ovat tyypiltään ”toista rivejä”.



Kuva 31. Kenttien minimi- ja maksimiarvo



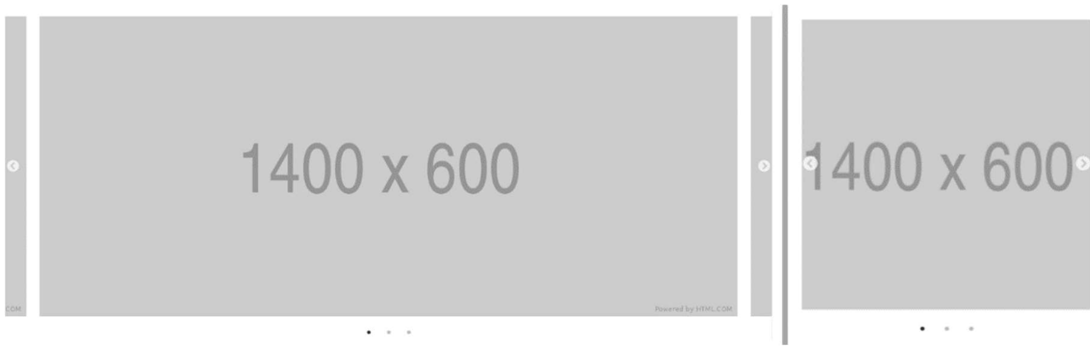
Kuva 32. Artikkeliolion valinta

Artikkeliolio-kentälle voidaan asettaa artikkelien tyyppi, jolla rajataan, minkälaisia artikkeleita voidaan valita. Yllä olevassa kuvassa 35. on Artikkelinosto-komponentin artikkelikenttä, jolle annetaan arvoksi artikkeliolio. Artikkelin tyypiksi on rajattu artikkelit ja esimerkki, joka on kustomoitu artikkelityyppi. Artikkelityyppien lisäksi voidaan rajata artikkeleita niiden kategorioiden mukaan ja voidaan valita myös sivuja.

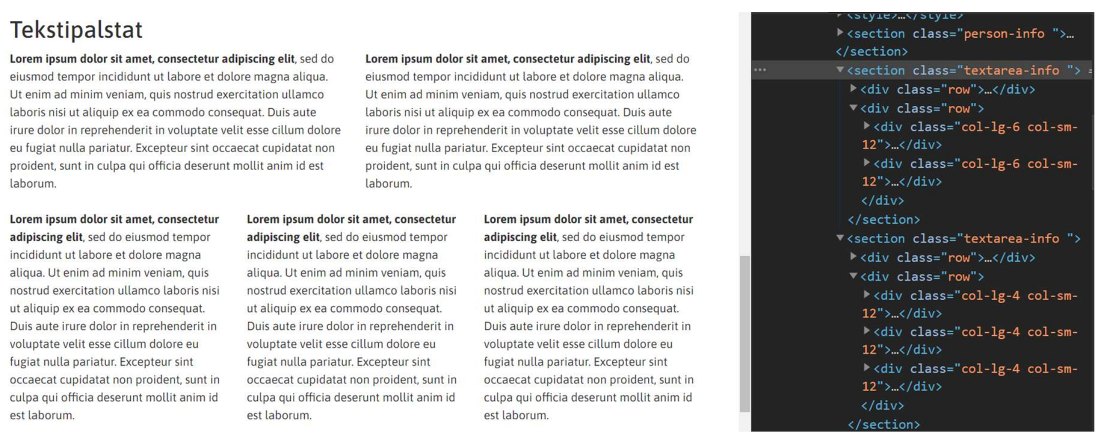
3.2.2 Testaus asiakkaan näkökulmasta

Komponenttien ulkonäköön voidaan vaikuttaa siten, että se eroaa tietokone- ja mobiilikoossa. Alla olevassa kuvassa 36. on komponentin Kuvakaruselli ulkonäkö. Vasemmalla kuvassa on karuselli, silloin kun on käytössä tarpeeksi leveä ruutu, ja oikealla on karuselli

silloin, kun laite on mobiililaite. Kuvakarusellilla on kaksi eri kuvakokoa, slider ja square, joita käytetään ruutukoon mukaan. Käyttäjän laitetta testataan käyttäen funktiota wp_is_mobile, joka palauttaa totuusarvon. Kuvan koko vaihdetaan mobiilissa vaakakuvasta neliöksi, jotta kuva ei ole liian matala pienillä laitteilla. Kuvakarusellin lisäksi Hero-kuva-komponentti toimii samalla tavalla kuvan koon mukaan.

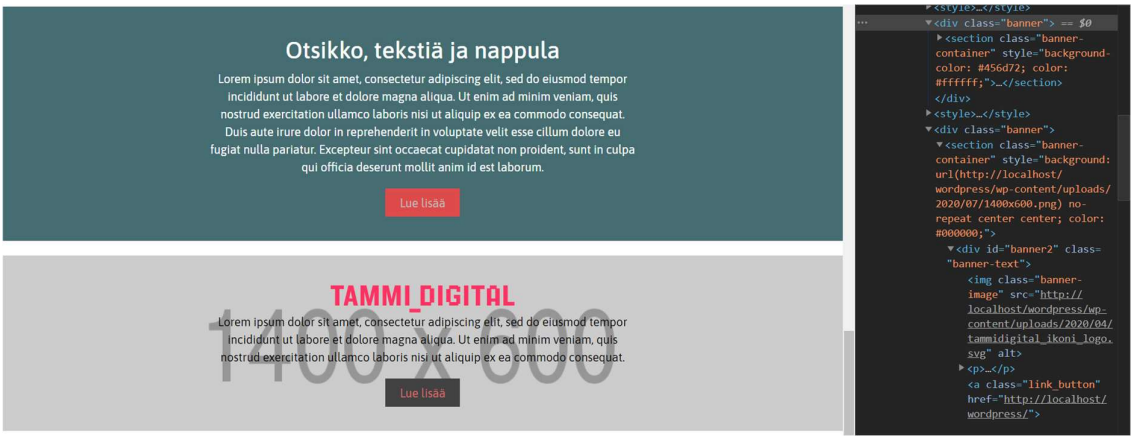


Kuva 36. Kuvakarusellin skaalaus



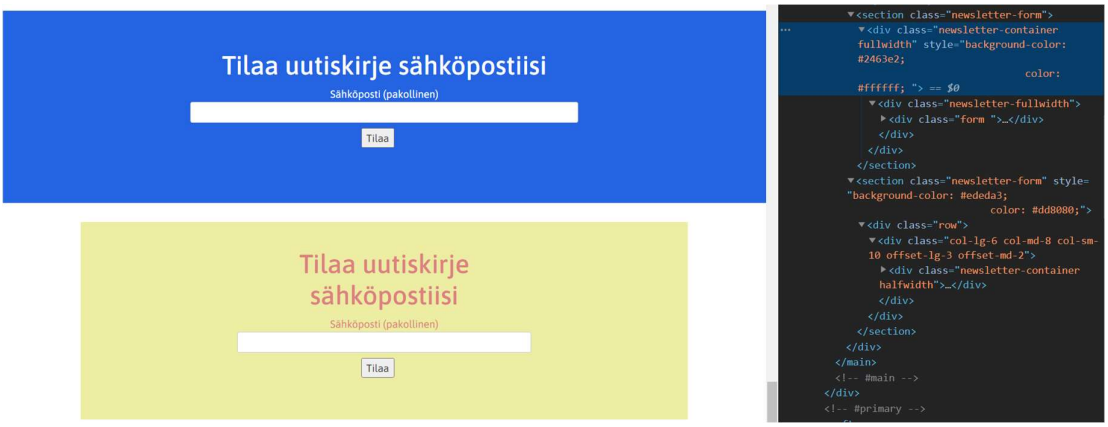
Kuva 33. Tekstipalstat-asettelu.

Komponenteissa voidaan kenttien määrän mukaan muokata myös Bootstrap-asettelua. Yllä olevassa kuvassa 37. on Yhteystiedot-komponentin asettelu Tekstipalstat, joita on sijoitettu sivulle käyttäen eri määrää palstoja. Palstojen määrä skaalautuu minimissään kahteen palstaan ja maksimissaan neljään palstaan. Palstojen määrän mukaan luodaan Bootstrap-asettelu, kuten ylemmässä Tekstipalstat-asettelussa sarakkeiden leveys on kuusi ja alemmassa leveys on neljä. Bootstrap-asetteluun on asetettu sarakkeiden iso- ja pienikokoinen arvo, jotka vaikuttavat työpöytä- ja mobiilikokoon, kuten mobiilikoossa tekstipalstat ovat täysleveitä.



Kuva 34. Bannerin asettelu.

Komponenteissa on myös erilaisia valintoja, joilla vaikutetaan komponentin sisältöön ja ulkonäköön. Esimerkiksi yllä olevassa kuvassa 38. on kaksi Banneri-asettelua, joille on valittu erilaiset taustat ja otsikot. Ylempään Banneriin on asetettu taustaksi väri, joka sijoitetaan ympäröivään section-elementtiin ja otsikoksi tekstiä, joka on asetettu h2-elementtiin. Alemmassa Bannerissa on taustaksi valittu kuva, joka on sijoitettu taustakuvaksi section-elementtiin ja otsikoksi myös kuva, joka asetetaan komponenttiin ylimmäksi elementiksi. Lisäksi alla olevassa kuvassa 39. vaikutetaan Uutiskirje asettelun ulkonäköön siten, että onko komponentin leveys koko ruudun levyinen vai Bootstrap container-elementin levyinen. Ylemmässä Uutiskirjeessä on asetettu leveydeksi täysileveä, jolloin sen HTML-rakenne on erilainen kuin alemmassa Uutiskirjeessä. Ylemmän uutiskirjeen rakennukseen ei ole käytetty Bootstrap-asettelua, vaan sen määrityksiin on käytetty prosentuaalisia arvoja. Vaihtoehtoisilla kentillä lisätään komponentin käytettävyyttä sillä, että ne ovat kustomoitavissa hyvin erinäköisiksi.



Kuva 35. Uutiskirje-asettelu.

3.3 Kirjaston levitystavan valinta ja suunnittelu

Komponenttikirjaston levityksessä on otettava huomioon sivulle lisäys sekä käytettävyys. Kirjaston levitystavaksi valittiin lisäosa, jotta se olisi lisättävissä myös olemassa oleville sivuille. Lisäosan kehityksessä tulee ottaa huomioon lisäosan päivitykset. Jos esimerkiksi huomataan komponentin tiedostoissa virheitä, niin komponentin tiedosto voitaisiin päivittää. Päivityksillä ei voida ottaa huomioon sitä, jos komponentin tiedostoihin on tehty muutoksia, jotka päivitysten takia tulevat ylikirjoitetuiksi. Kirjaston lisäosan käyttöliittymään voidaan lisätä tarvittavien kirjastojen lisäys lauseet, jotka käyttäjän on lisättävä functions.php-tiedostoon.

Lisäosa siirtää tarvittavat template-tiedostot teemaan kansioon ”Components” tai uudelleenohjaa komponenttien tiedostot lisäosan tiedostoihin esimerkiksi käyttämällä funktiota template_redirect. Uudelleenohjauksessa voidaan määrittää router.php-tiedostossa komponenttien tiedostojen sijainniksi lisäosan kansio. Tiedostojen lisäksi ACF-kenttäryhmä tulee tuoda sivustolle, jotta komponentit voivat toimia. Kenttäryhmän tuomiseksi voidaan käyttää WordPressin tuontityökalua, jolle annetaan JSON-tiedosto, joka sisältää kenttäryhmän konfiguraation, tai voidaan luoda lisäosalle tiedosto, joka tuo kenttäryhmät.

4 YHTEENVETO

Komponenttikirjaston valitut komponentit saatiin onnistuneesti suunniteltua ja kehitettyä. Komponenttien suunnittelussa otettiin huomioon mahdollisimman laaja kustomoitavuus niin komponenttien tyylimäärityksissä kuin komponentin asettelussa. Kehityksessä otettiin huomioon komponenttien rakenteen selkeys, jotta rakenteesta ei tulisi liian monimutkainen, ja yhtenevyys, jotta komponenttien rakenteet olisivat mahdollisimman samankaltaisia. Komponentit saatiin pilotoitua huomioiden kaksi eri näkökulmaa: kehittäjän ja asiakkaan näkökulma. Kehittäjän näkökulman pilotoinnissa otettiin huomioon komponenttien lisäys ja muokkaus sivunmuokkausnäkymässä. Asiakkaan näkökulman pilotoinnissa huomioitiin komponenttien ulkonäköä sekä skaalautuvuutta työpöytä- ja mobiilikokoon.

Komponenttikirjaston kehittämisen lisäksi tavoitteena oli suunnitella levitystapa. Levitystavaksi valittiin lisäosa, joka joko siirtää tarvittavat tiedostot teeman kansioon tai ohjaa teeman käyttämään template-tiedostoja lisäosan kansiosta. Lisäosan suunnittelu jäi keskeneräiseksi, sillä projektin aika loppui kesken.

Komponenttikirjaston kehitystä voitaisiin jatkaa lisäämällä uusia komponentteja, joilla voitaisiin laajentaa kirjaston kattavuutta. Uusien komponenttien lisäksi lisäosan suunnittelu ja kehitys tulisivat olemaan seuraavan version kehityksen kohteita. Olemassa olevien komponenttien kanssa olisi voitu jakaa ominaisuuksia erillisiin komponentteihin komponenttien asetteluiden sijasta, jotta ne eivät olisi niin monimutkaisia. Esimerkiksi Otsikko ja sisältö -komponentin Banneri-asettelu olisi voinut olla oma komponenttinsa asettelun sijasta.

Komponentteja voidaan käyttää sivupohjien sijasta luomaan www-sivuston sisältö. Sisältö on tällöin muokattavissa ja uudelleen järjestettävissä. Sisältö voidaan luoda suoraan haluttuun sivuun sivunmuokkausnäkymässä, jolloin sivun sisältö ei ole sidottu sivupohjan asetteluun. Sisältöön voidaan myös upottaa monimutkaisempia ominaisuuksia, kuten Kuvakarusellin, joita ei välttämättä perinteiselle sivulle voida lisätä ilman lisäosaa.

LÄHTEET

Advanced Custom Fields. 2020a. Edit content with Advanced Custom Fields for WordPress Developers. [viitattu 22.9.2020] Saatavissa: https://www.advancedcustomfields.com/

Advanced Custom Fields. 2020b. Documentation. [viitattu 23.9.2020] Saatavissa: https://www.advancedcustomfields.com/resources/

Azhar. 2020. What are WordPress hooks? How do actions and filters help to extend the functionality? learnwoo. [viitattu 22.10.2020] Saatavissa: https://learnwoo.com/what-are-wordpress-hooks/

Azmi, A. 2020. How Much of The Internet is WordPress. [viitattu 1.11.2020] Saatavissa: https://www.webhostingsecretrevealed.net/blog/wordpress-blog/wordpress-stats/

Hayes, D. 2017. The WordPress Template Hierarchy: What, Why, and How. WP Hierarchy. [viitattu 22.10.2020] Saatavissa: https://wphierarchy.com/

HostGator, 2020. How to Install WordPress Manually. [viitattu 7.9.2020] Saatavissa: https://www.hostgator.com/help/article/how-to-install-wordpress-manually

Hughes, J. 2017. An Introduction to WordPress Core Files. [viitattu 12.10.2020] Themeisle. Saatavissa: https://themeisle.com/blog/wordpress-core-files/

McCollin, R. 2014. Understanding and Working with Relationships Between Data in WordPress. [viitattu 13.10.2020] Saatavissa: https://code.tutsplus.com/tutorials/understanding-and-working-with-relationships-between-data-in-wordpress--cms-20632

Mullenweg, M. 2016. Moving Toward SSL. WordPress.org. [viitattu 7.9.2020] Saatavissa: https://wordpress.org/news/2016/12/moving-toward-ssl/

WordPress.org a. Democratize Publishing. [viitattu 4.9.2020]. Saatavissa: https://wordpress.org/about/

WordPress.org b. Make WordPress. [viitattu 7.9.2020]. Saatavissa: https://make.wordpress.org/

WordPress.org c. Features. [viitattu 7.9.2020] Saatavissa: https://wordpress.org/about/features/

WordPress.org d. Requirements. [viitattu 7.9.2020]. Saatavissa: https://wordpress.org/about/requirements/

WordPress.org e. Database Description. [viitattu 12.10.2020] Saatavissa: https://codex.wordpress.org/Database_Description

WordPress.org f. Administration Screens. [viitattu 10.9.2020] Saatavissa: https://wordpress.org/support/article/administration-screens/

WordPress.org g. Managing Plugins. [viitattu 11.9.2020] Saatavissa: https://wordpress.org/support/article/managing-plugins/

WordPress.org h. Plugin Basics. [viitattu 10.9.2020] Saatavissa: https://developer.wordpress.org/plugins/plugin-basics/

WordPress.org i. Hooks. [viitattu 18.9.2020] Saatavissa: https://developer.wordpress.org/plugins/hooks/

WordPress.org j. Plugin API. [viitattu 22.9.2020] Saatavissa: https://codex.wordpress.org/Plugin_API

WordPress.org k. Theme Handbook. [viitattu 8.9.2020] Saatavissa: https://developer.wordpress.org/themes/getting-started/what-is-a-theme/

WordPress.org l. Theme Development. [viitattu 8.9.2020] Saatavissa: https://codex.wordpress.org/Theme_Development

WordPress.org m. Stepping into Templates. [viitattu 8.9.2020] Saatavissa: https://codex.wordpress.org/Stepping_Into_Templates

WordPress.org n. Functions File Explained. [viitattu 8.9.2020] Saatavissa: https://codex.wordpress.org/Functions_File_Explained

WordPress.org o. Template Hierarchy. [viitattu 14.9.2020] Saatavissa: https://developer.wordpress.org/themes/basics/template-hierarchy/