jamk.fi

WordPress-järjestelmän sisältölohkojen kehittäminen

Olli-Ilari Kärkkäinen

Opinnäytetyö
Toukokuu 2019
Luonnontieteiden ala
Tradenomi (AMK), tietojenkäsittelyn tutkinto-ohjelma

Jyväskylän ammattikorkeakoulu
JAMK University of Applied Sciences

jamk.fi

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Kärkkäinen, Olli-Ilari	Opinnäytetyö, AMK	toukokuu 2019
	Sivumäärä 35	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: x

Työn nimi

WordPress-järjestelmän sisältölohkojen kehittäminen

Tutkinto-ohjelma

Tietojenkäsittelyn tutkinto-ohjelma

Työn ohjaaja(t)

Tuikka, Tommi

Toimeksiantaja(t)

Aava & Bang Oy

Tiivistelmä

Internetin käytetyimmän sisällönhallintajärjestelmän WordPressin kehityssuunta kohti JavaScript-kielellä kehitettyä verkkosovellusta on jakanut paljon mielipiteitä WordPress-kehittäjien keskuudessa. WordPress on kehitetty PHP-kielellä, eikä sen kehityksessä ole aiemmin vaadittu muiden kielien käyttöä. Aava & Bang on kasvava markkinointitoimisto Keski-Suomesta, jossa päätettiin selvittää, voisiko uuden WordPress-kehitysmallin käyttöönotolla mahdollistaa verkkosivujen julkaisuprosessin ja käyttäjäkokemuksen kehittyminen.

Modernien JavaScript-tekniikoiden käytön yleistyminen ja sen mukana tulevat uudet käytänteet vaativat kuitenkin paljon uuden tiedon omaksumista ja tutkimustyötä, joka loi tarpeen kehittämistutkimukselle. Tutkimuksen tavoitteena oli tutkia parhaita käytänteitä WordPressin uusien sisältölohkojen kehityksessä. Lisäksi tarkoituksena oli selvittää, mitä ketteriä kehitysmenetelmiä voidaan hyödyntää yhden henkilön sovelluskehitysprosessissa.

Aineistoa kerättiin pääasiassa WordPressin virallisesta dokumentaatiosta sekä sen avoimesta lähdekoodista ja muista web-kehitystä käsittelevistä alan julkaisuista. Teoriapohjan avulla kehitettiin toimeksiantajayrityksenä toimineen Aava & Bangin käyttöön lisäosa WordPress-järjestelmään, jolle asetettiin tiettyjä vaatimuksia. Toteutettu lisäosa korjasi uudessa WordPress-versiossa ilmenneitä puutteita ja antoi rohkaisevaa tietoa uuden kehitysmallin tulevaisuuden käyttöönottoa varten.

Avainsanat (asiasanat)

WordPress, Gutenberg, web-kehitys, sovelluskehitys

Muut tiedot (Salassa pidettävät liitteet)

jamk.fi

Description

Author(s)	Type of publication	Date
Kärkkäinen, Olli-Ilari	Bachelor's thesis	May 2019
		Language of publication: Finnish
	Number of pages: 35	Permission for web publication: x

Title of publication

Content block development for WordPress content management system

Degree programme

Business Information Systems

Supervisor(s)

Tuikka, Tommi

Assigned by

Aava & Bang Oy

Abstract

The most used content management system of the internet, WordPress is quickly developing towards a web application written in JavaScript. WordPress is developed with PHP and it was the only language a developer needed to master in the WordPress ecosystem. This development has divided opinions across the field of WordPress development. Aava & Bang is a growing marketing agency from Central Finland that decided to explore if the new development model could improve their website developing process and user experience.

The growing use of modern JavaScript technologies and new practices in development require acquiring a great amount of new knowledge and doing research. This created the demand to conduct action research, the goal of which was to study the best practices in developing new content blocks to the WordPress system. In addition, it was desired to find the most suitable agile methods for the one-person development process.

The material for the study was mainly gathered from the official WordPress documentation, source code and other articles about web development. The material was used to develop a new WordPress plugin for the use of Aava & Bang's web development team. The developed plugin fixed few shortcomings of the new WordPress version and provided encouraging information for the future use of the new development model.

Keywords/tags (subjects)

WordPress, Gutenberg, web development, software development

Miscellaneous (Confidential information)

Sisältö

Kuviot

Määritelmät ja sanasto

API	Application Programming Interface. Rajapinta, jonka kautta eri ohjelmat voivat vaihtaa tietoja keskenään.
Avoin lähdekoodi	Kaikille käyttäjille avointa ja vapaasti muokattavaa koodia, josta voi levittää alkuperäistä sekä muokattua versiota.
DOM	Document Object Model. Sovelluksen käyttöliittymä kuvattuna puurakenteena.
Git	Hajautettu versionhallintaohjelmisto
Gutenberg	WordPress-sisällönhallintajärjetelmän uusi työkalu, jolla sivujen sisältöä muokataan.
JavaScript	Web-kehityksessä käytettävä dynaaminen ohjelmointikieli.
JavaScript-kirjasto	Kirjasto ennalta kirjoitettua JavaScript-koodia kehityksen helpottamiseen.
Käyttötapaus	Kuvaus toimijan ja tietojärjestelmän välisestä vuorovaikutuksesta, jolla on jokin tietty päämäärä.
Käyttäjätarina	Kuvaus, miten käyttäjä tai asiakas käyttää tuotetta.

Npm	Tietokanta, jossa voi jakaa JavaScript-ohjelmistoja paketteina.
Lisäosa	Sovellus, jolla voi laajentaa WordPressin toiminnollisuuksia.
Lohko	Elementti tai ns. rakennuspalikka, joista WordPress-sivustot rakennetaan.
PHP	Ohjelmointikieli, jonka kääntämiseen tarvitaan php-tulkki, joka palauttaa koodista muodostetun verkkosivun.
React	Moderni JavaScript-kirjasto interaktiivisten käyttöliittymien rakentamiseen.
SASS	Syntactically Awesome Style Sheets. CSS-kielen laajennos, joka lisää useita toiminnollisuuksia.
Scrum	Viitekehys, jota käytetään ketterässä ohjelmistokehityksessä.
Teema	Kokoelma sivupohjista ja tyylitiedostoista, jotka määrittävät WordPress-sivun ulkoasun.

UML	Unified Modeling Language. Standardisoitu mallinnuskieli, jonka kaavioilla mallinnetaan kehitettävää järjestelmää tai ohjelmistoa.
WordPress	Avoimeen lähdekoodiin perustuva sisällönhallintajärjestelmä.

1 Johdanto

WordPress on maailman eniten käytetty, avoimeen lähdekoodiin perustuva sisällönhallintajärjestelmä. WordPress on käytössä yli 33 %:ssa internetin verkkosivuista. WordPressiä käytetään sisällön hallintaan ja julkaisuun kaiken kokoisilla verkkosivuilla. WordPressiä käyttämällä on muun muassa tehty blogeja, yksinkertaisia sekä monimutkaisempia verkkosivustoja ja myös verkkosovelluksia. WordPress on ladattavissa ilmaiseksi ja sen lähdekoodi on vapaasti kehittäjän muokattavissa. (Features n.d.)

WordPress-kehityksen voi nähdä täysin omana osaamisalueenaan sen useiden eri aihealueiden vuoksi. WordPress-ydin on kehitetty PHP-ohjelmointikielellä, joka on yksi yleisimmistä verkossa käytetyistä kielistä. WordPress-kehityksessä tarvitaan myös jonkin verran tietämystä HTML-, CSS- ja JavaScript-kielistä. Oleellista on myös tietämys järjestelmälle ominaisista toiminta- ja suodatinkoukuista sekä teemojen ja lisäosien kehityksestä. (Ewer 2018.)

Joulukuussa 2018 WordPress sai version 5.0 mukana yhden merkittävimmistä tähänastisista muutoksista. WordPressin editori, jolla sivuston sisältöä hallintaan, koki täydellisen muutoksen. Editori oli jo vuosikausia pysynyt samanlaisena eikä se sellaisenaan sopinut muuhun, kuin yksinkertaisten teksti- ja kuvasisältöjen luomiseen. WordPress-kehittäjäyhteisö halusi luoda uuden, kaikille yhteisen mallin WordPress-lisäosien kehitykseen ja samalla korvata vanhempia toiminnollisuuksia. (Mullenweg 2018.)

Näin syntyi avoimeen lähdekoodiin perustuva Gutenberg-lisäosa, joka loppuvuodesta 2018 korvasi vanhan editorin kokonaan. Merkittävimpinä muutoksina nähdään uusi kehitysmalli, jossa React -JavaScript-kirjaston avulla kehitetään uusia toiminnollisuuksia lohkojen muodossa. Lohkot voidaan ajatella eräänlaisina rakennuspalikkoina, joita lisäämällä ja yhdistelemällä sivujen sisältöä voidaan hallita. Esimerkiksi jokainen sivun elementti, kuten otsikko, tekstikappale tai vaikka kuva on

oma lohkonsa, joita voidaan uudelleen käyttää ja asetella sivulle missä tahansa järjestyksessä. (Mt.)

Uuden React -JavaScript-kirjaston hyödyntäminen vaatii WordPress-kehittäjiltä paljon uusien menetelmien omaksumista. Tässä opinnäytetyössä tutustutaan syvemmin WordPressin uuden Gutenberg-editorin lohkojen kehitysmalliin ja kehitetään uusi lohko käytettäväksi. Opinnäytetyön toimeksiantajana on Aava & Bang Oy, jonka palvelutarjoomaan kuuluvat kaiken kokoiset WordPress-toteutukset. Toimeksiantajayritys osallistui tämän opinnäytetyön vaatimusmäärittelyyn.

2 Tutkimusasetelma

2.1 Opinnäytetyön tavoitteet ja rajaukset

Opinnäytetyön tavoitteena on tutustua syvemmin WordPressin uuden Gutenberg-editorin lohkojen maailmaan sekä kehittää uusi lohko. Lohkon kehitysidea syntyi toimeksiantajayrityksen tarpeesta. Uusi lohkojen kehitystapa vaatii WordPress-kehittäjältä paljon uuden tiedon omaksumista, sillä lohkojen kehityksessä WordPress-kehittäjille jo tutun PHP-ohjelmointikielen lisäksi tarvitaan tietämystä React -JavaScript-kirjastosta. React-kehitys eroaa selvästi perinteisestä JavaScript-sovelluskehityksestä. Myös vanhan editorin tuki loppuu aikaisintaan vuonna 2022, joten vanhaa editoria hyödyntävät WordPress-toteutukset voi joutua päivittämään Gutenberg-editoriin hyvinkin pian (Mullenweg 2018).

Vaikka Gutenberg-editori on jo julkaistu osana WordPress-ydintä, on sen kehitys kirjoitushetkellä vielä hyvin varhaisessa vaiheessa. Editoriin valmiiksi kehitettyjen lohkojen asetukset ovat vielä puutteelliset. Puutteiden vuoksi editorilla ei pysty toteuttamaan elementtejä, jotka saattavat olla varsin yleisiä verkkosivustojen maailmassa. Editorin lohkojen avulla voidaan kyllä lisätä aiempaa monipuolisempaa sisältöä, mutta lohkojen ulkoasun muokkaamismahdollisuudet ovat hyvin rajalliset.

Opinnäytetyön teoriaosuudessa käsitellään WordPressin tärkeimpiä ominaisuuksia ja toimintoja sekä WordPress-kehityksessä tarvittavia yleisimpiä tekniikoita. Lisäksi selvitetään, mitä ketteriä kehitysmenetelmiä voidaan hyödyntää. Työn toiminnallisessa osuudessa kehitetään valituilla menetelmillä Gutenberg-lohko, jolle suoritetaan erikseen vaatimusmäärittely. Tutkimuksen ja lohkon kehitysprosessin ansiosta syntyvää tietoa on tarkoitus hyödyntää tulevaisuudessa uusien lohkojen kehityksessä toimeksiantajayrityksessä.

2.2 Tutkimuskysymykset

Uuden kehitysmallin omaksumisella toimeksiantajayritys tavoittelee verkkosivuston käyttäjäkokemuksen parantamista, joka samalla tulee näkymään asiakastyytyväisyydessä. Kyvyn kehittää itse kustomoitavia lohkoja uskotaan kehittävän verkkopalvelutiimin julkaisuprosessia ja avaavan uusia mahdollisuuksia yrityksen liiketoiminnassa. Tuki vanhalle kehitysmallille tulee jossain vaiheessa päättymään, joten tutkimusongelmaksi muodostuu puutteet uudemman kehitysmallin käytänteiden tuntemisessa. Tutkimusongelma päädyttiin jakamaan kolmeen seuraavaan tutkimuskysymykseen:

1. Mitkä ovat parhaita käytänteitä Gutenberg-lohkojen kehityksessä?
2. Mitä vaatimuksia kehitettävällä lohkolla on?
3. Mitä ketteriä kehitysmenetelmiä voi hyödyntää yhden henkilön sovelluskehitysprosessissa?

2.3 Tutkimus- ja kehittämismenetelmät

Tutkimusmenetelmäksi valikoitui kehittämistutkimus. Edelsonin (2002) mukaan kehittämistutkimuksessa yhdistyvät kehittäminen sekä tutkimus syklisessä muodossa. Puolestaan Barab ja Squire (2004) määrittelevät kehittämistutkimuksen lähtevän muutostarpeesta, jonka tuloksena syntyy tuotos. (Kananen 2012, 19.) Tämän opinnäytetyön tutkimuksen tarkoituksena on saada ajankohtaista tietoa yhdestä toimeksiantajayrityksen tuotteesta, eli WordPress-sivuston uudesta

kehitysmallista, ja luoda tämän uuden mallin mukainen lisäosa, joka poistaa sivuston sisällön hallinnoimisessa havaittuja puutteita. Työ aloitetaan hankkimalla aineistoa WordPressin sekä sen Gutenberg-editorin dokumentaatiosta ja WordPressin kehittäjän, Matt Mullenwegin blogista sekä muista web-kehitystä käsittelevistä alan julkaisuista. Toimeksiantajayrityksen kanssa suoritettava vaatimusmäärittely määrää reunaehdot opinnäytetyölle.

Kehitys Scrum-menetelmällä

Scrumilla tarkoitetaan viitekehystä, jonka avulla tuottavasti ja luovasti kehitetään, toimitetaan ja ylläpidetään monimutkaisia, mahdollisimman paljon lisäarvoa tuovia tuotteita. Scrumin taustalla vaikuttaa empiirinen prosessinhallintateoria. Teorian mukaan tiedon nähdään perustuvan kokemukseen ja päätösten tekoon jo tiedossa olevien tosiasioiden pohjalta. Scrum-menetelmä käyttää toistavaa ja lisäävää lähestymistapaa optimoidakseen ennustettavuutta ja riskinhallintaa. Scrumille ominaista on pienikokoinen tiimi, joka on sopeutumiskykyinen ja joustava. Scrumia voivat hyödyntää niin yksittäiset kuin useamman tiimin verkostot. Scrum-tiimin jäsenten tulee sitoutua tiimin tavoitteisiin sekä arvoihin. Rohkeus, sitoutuminen, kunnioitus ja avoimuus ovat tiimin keskeisiä arvoja, joiden omaksuminen on ratkaiseva tekijä menetelmän onnistumisessa. (Schwaber & Sutherland 2017, 4 – 6.)

Scrumin tapahtumat ja tuotokset

Tätä opinnäytetyötä työstettiin yhden henkilön voimin, joten usean henkilön tiimille tarkoitettu menetelmä koettiin liian raskaaksi ja aikaa vieväksi. Menetelmästä valikoitiin käytettäväksi sen tärkeimmät tapahtumat ja tuotokset. Sprintillä tarkoitetaan kehitysjaksoa, jolla on jokin päämäärä, kuten esimerkiksi lisätä toiminnollisuus tuotteeseen tai korjata virheitä ohjelmoinnissa. Sprintin tehtävien tulee edistää sen päämäärän saavuttamista. Sprintit ovat yleensä noin viikosta kuukauteen pitkiä. (Andrews 2017.) Tässä opinnäytetyössä käytettiin kahta sprinttiä. Ensimmäinen sprintti kesti kaksi viikkoa, jonka aikana pääosa kehitystyöstä tehtiin. Toiseen sprinttiin varattiin viikko aikaa, jolloin tuotosta testattiin ja tehtiin tarvittavat korjaukset.

Sprintti aloitetaan sprintin suunnittelusessiolla, jossa luodaan tuotteelle kehitysjono. Kehitysjonoon listataan kaikki ne tehtävät, jotka suorittamalla tuote täyttää ”valmiin määritelmän” sprintin lopussa. Tehtävät voidaan pisteyttää niiden vaativuuden ja ajankäytön mukaan. Sprintin lopussa tehtyjen tehtävien pisteet lasketaan yhteen ja nähdään, miten monta pistettä on saavutettu. Seuraavan sprintin suunnittelussa on huomioitava, että tulevien tehtävien pisteiden summa on noin sama kuin edellisen sprintin aikana saavutetut pisteet. (Andrews 2017.) Tämän opinnäytetyön molemmat sprintit aloitettiin lyhyellä suunnittelusessioilla, joissa suunniteltiin tulevan sprintin tehtävät.

Sprintin päivät alkavat muutaman minuutin pituisella daily scrumilla, jonka aikana on tarkoitus tehdä itsearviointia. Daily scrumin sisältönä voidaan tarkastella edellisenä päivänä tehtyjä tehtäviä ja arvioida kehityskohtia sekä miettiä tulevan päivän tehtäviä. Päivän tehtäviä varten voi perustaa erillisen tehtävätaulun, jonka avulla tehtävät voi jakaa tekemättömiin, kesken oleviin ja valmiisiin tehtäviin. (Mt.)

Scrumin yksi perusperiaatteista on saada tehty työ julkiseksi sopivin väliajoin. Tämä tapahtuu sprintin lopun julkaisuvaiheessa. Julkaistavan työn ei tarvitse olla valmis tuote, vaan oleellista on saada työ testattavaksi ja kerätä palautetta, jotka voidaan ottaa huomioon myöhemmin seuraavassa sprintissä. Sprintin loputtua oleellista on myös tarkastella syvemmin kuluneen sprintin työskentelyä. Tätä varten pidetään yleensä noin kahden tunnin mittainen retrospektiivi. (Mt.) Andrewsin (mt.) mukaan retrospektiivissä työn tuloksia voidaan reflektoida esimerkiksi seuraavin kysymyksin:

1. Mitä olet saavuttanut?
2. Saavutettiinko sprintin tavoitteet?
3. Mikä toimi erityisen hyvin kuluneessa sprintissä? Mikä olisi voinut toimia paremmin?
4. Onko olemassa tekijöitä, joka haittaavat työskentelyn tehokkuutta?
5. Aiheuttaako jokin asia stressiä? Mistä nautit työskentelyssä erityisen paljon?

Reflektoinnin perusteella valitaan kaksi yksinkertaista asiaa, joita voidaan parantaa seuraavassa sprintissä:

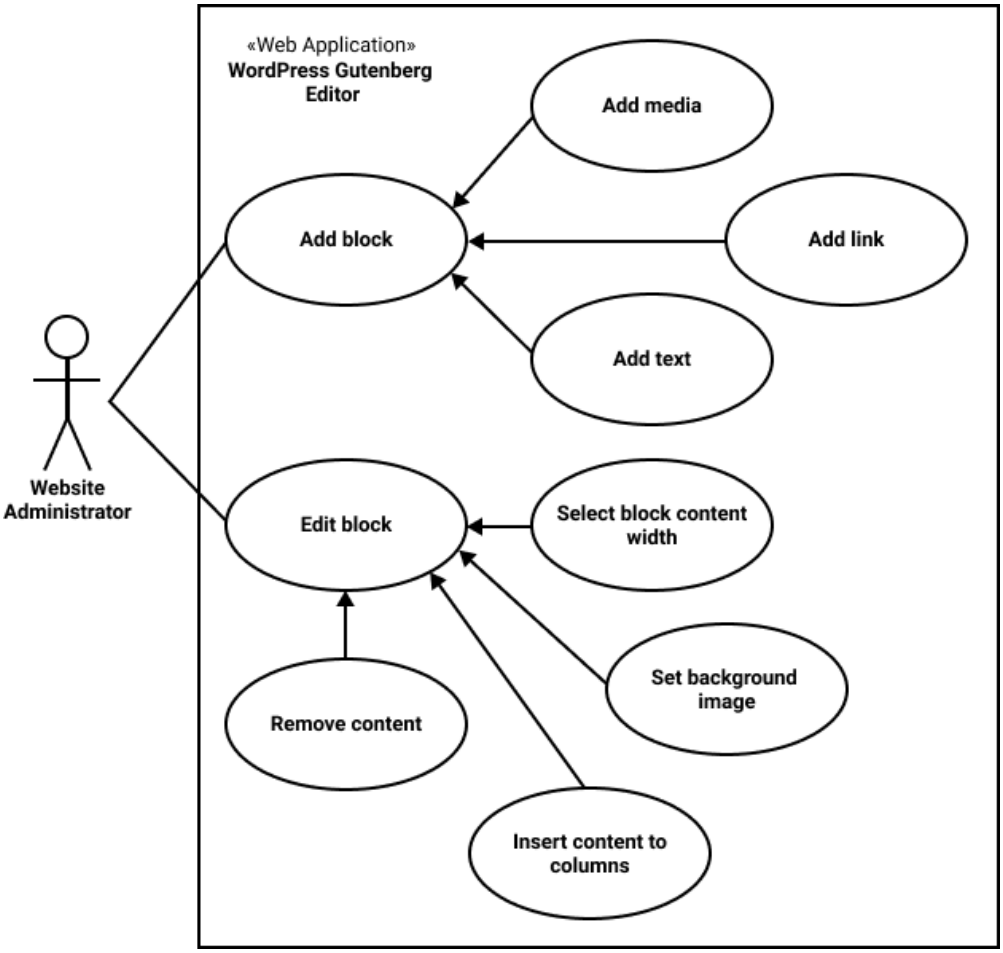
1. Yksi asia, joka tekee työstä tuottavamman.
2. Yksi asia, joka tekee työn teosta mielekkäämpää.

Vaatimusmäärittely

Garbarin (2016) mukaan ISO/IEC/IEEE 29148:2011-standardi kuvaa vaatimusmäärittelyn eduiksi sen tarjoaman realistisen perustan tuotteiden kulujen, riskien ja aikataulujen arvioimiseksi. Se tarjoaa perustan tuotteen parantamiselle ja käyttöönotolle uusille käyttäjille tai uusille käyttöympäristöille. Vaatimusmäärittely pakottaa arvioimaan tiukkoja vaatimuksia ennen suunnittelun aloittamista ja minimoi näin myöhemmän uudelleensuunnittelun. Siinä luodaan perusta hankkijoiden ja tavarantoimittajien väliselle sopimukselle siitä, miten tuote on tarkoitus tehdä. (Garbar 2016.)

Käyttötapaukset

Tässä opinnäytetyössä vaatimusmäärittelyssä ei kuitenkaan luotu erillistä dokumenttia määrittelyistä, vaan hyödynnettiin käyttötapauksien ja käyttäjätarinoiden kuvaamista. Käyttötapaus on menetelmä, jonka avulla voidaan tunnistaa, selventää ja järjestää järjestelmävaatimukset. Käyttötapaus muodostuu järjestelmien ja käyttäjien välisestä vuorovaikutuksesta tietyssä ympäristössä, ja se liittyy tiettyyn tavoitteeseen. Käyttötapauksessa tulisi olla kaikki järjestelmän toiminnot, joilla on merkitystä käyttäjille. Käyttötapaus voidaan ajatella kokoelmana mahdollisista skenaarioista, jotka liittyvät tiettyyn tavoitteeseen, tosin käyttötapaus ja tavoite katsotaan joskus toistensa synonyymeiksi. (Rouse 2007.) Käyttötapauksien mallintamiseen voidaan hyödyntää käyttötapauskaaviota (ks. kuvio 1). Käyttötapauskaavio on UML-mallinnuskielen mukainen kaavio, jolla kuvataan aktorin, eli toimijan vuorovaikutusta tietojärjestelmässä. (Fakhroutdinov n.d.)



Kuvio 1. Kehitettävän lohkon käyttötapaukset

Käyttäjätarinat

Käyttäjätarinat ovat yksi Scrum-menetelmän pääartefakti. Näiden tarinoiden avulla kehittäjien on helppo luoda kohtuullinen arvio toiminnollisuuksien toteuttamisen työmäärästä. Tarinoita voidaan käyttää kuvaamaan hyvin monenlaisia vaatimustyyppejä. Osa tarinoista voi olla hyvin samankaltaisia käyttötapauksien kanssa, kun osa taas voi muistuttaa hyvin paljon teknisiä vaatimuksia. (Ambler n.d.)

1. Käyttäjä haluaa lisätä tekstiä lohkoon.
2. Käyttäjä haluaa muotoilla tekstiä.
3. Käyttäjä haluaa asetella tekstiä useampaan kuin yhteen sarakkeeseen.
4. Käyttäjä haluaa tekstille värin valintamahdollisuuden.
5. Käyttäjä haluaa valita lohkolle taustakuvan.

6. Käyttäjä haluaa valita sisällön leveyden.

3 Kehitystyössä käytettäviä sovelluksia, tekniikoita ja työkaluja

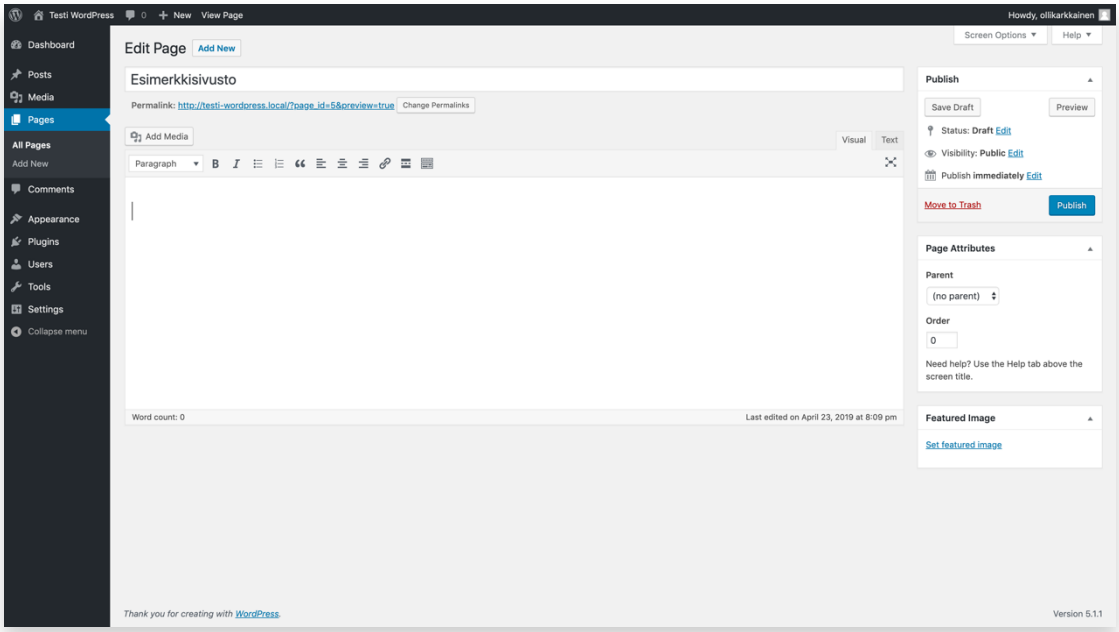
3.1 WordPress

WordPress on maailman suosituin sisällönhallintajärjestelmä. Sen markkinaosuus sisällönhallintajärjestelmistä on hieman yli 60 %. (Usage of content management systems n.d.) WordPressin ensimmäinen versio julkaistiin vuonna 2003 Matt Mullenwegin ja Mike Littlen kehittämänä. WordPress kehitettiin alun perin blogien julkaisua varten, mutta nykyään sähköisten artikkeleiden lisäksi sillä voidaan julkaista myös verkkosivuja. WordPress perustuu avoimeen lähdekoodiin ja on ladattavissa täysin ilmaiseksi. Se asennetaan käyttäjän omalle palvelimelle, jossa käyttäjä itse ylläpitää sovellusta. WordPress suosittelee palvelimen tukevan uusimpia versioita PHP:stä, MySQL:stä tai MariaDB:stä sekä tukevan myös salattua SSL-yhteyttä. Palvelinohjelmana suositellaan käytettävän joko Apachea tai Nginx:ää. (Requirements n.d.)

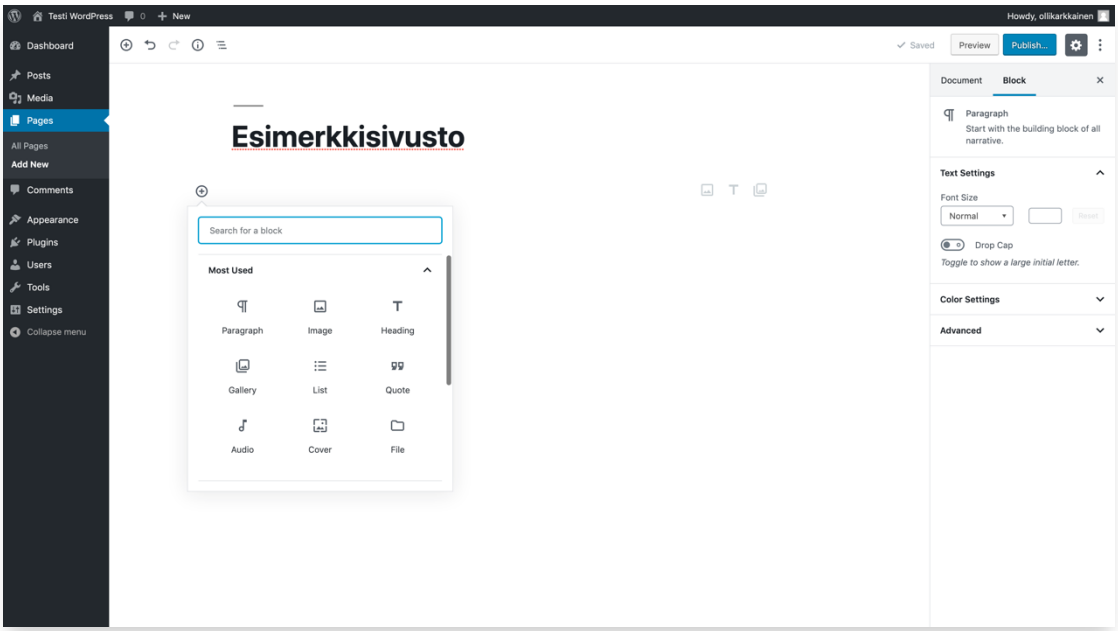
WordPress on kehitetty lähes kokonaan PHP-ohjelmointikielellä lukuun ottamatta sen uuden Gutenberg-editorin lohkoja. WordPressin voidaan katsoa koostuvan kolmesta tärkeästä komponentista (Introduction to Plugin Development n.d.). WordPressin tärkein komponentti, ydin, sisältää kaikki sen oletustoiminnot (Hughes 2017). Toinen tärkeä komponentti on lisäosat, joiden avulla voidaan kehittää uusia toiminnollisuuksia WordPress-ytimen rinnalle ohjelmointirajapintaa hyödyntäen. Kolmas komponentti on puolestaan teemat, joilla määritellään sivuston ulkoasu ja asettelu. Teemat hakevat sivuston sisällön WordPressin tietokannasta ja näyttävät sen selaimelle. (Introduction to Plugin Development n.d.; What is a Theme n.d.) Lisäosia ja teemoja voivat kehittää kuka tahansa ja yleensä puhuttaessa WordPress-kehityksestä tarkoitetaan nimenomaan lisäosien ja teemojen kehittämistä. Kehitetyt

teemat ja lisäosat voidaan ladata WordPressin hakemistoon kaikkien ladattaviksi ilmaiseksi tai maksulliseksi. (Theme Directory n.d.)

WordPressin vanhempi editori on rakennettu pääosin tekstin kirjoittamista varten (ks. kuvio 1). Nykypäivänä kuitenkin sivuille ja artikkeleille pitää voida lisätä hyvin monimuotoista mediaa, kuten mediaupotuksia, karttoja, yhteydenottolomakkeita, kuvia ja videoita. Uudessa editorissa sisältöä voidaan syöttää lohkojen muodossa (ks. kuvio 3). Gutenberg-lohkot ovat sisältöelementtejä, joita ylläpitäjä voi yhdistää ja asetella haluamaansa järjestykseen luodakseen ainutlaatuisia asetteluja. (Mullenweg 2018.)



Kuvio 2. Kuvakaappaus vanhasta editorista WordPress-hallintapaneelissa



Kuvio 3. Kuvakaappaus uudesta Gutenberg-editorista

Editorin lisäksi WordPressin asetuksia muokataan muun muassa customizer-osiolla, vimpainosiolla ja lyhytkoodeilla. Customizer-osiolla voidaan muokata teeman asetuksia. Vimpainosiolla lisätään sisältöä teeman sivupalkkiin, mikäli sellainen teemasta löytyy. Lyhytkoodilla tarkoitetaan lyhyttä koodinpätkää, jolla vanhan editorin tekstikenttään upotetaan jonkin lisäosan toiminnollisuus. (Theme Options – The Customize API n.d.; WordPress Widgets n.d.; Shortcodes n.d.)

Mullenwegin (2018) mukaan WordPressin tulevissa kehitysvaiheissa lohkot korvaavat edellä mainitut tekniikat. Mullenweg (mt.) myös mainitsee artikkelissaan mahdollisuuden rakentaa koko WordPress toimimaan tulevaisuudessa JavaScriptillä. Tämän vuoksi WordPress-kehittäjien on erityisen ajankohtaista tutkia lohkojen kehityksen käytänteitä ja samalla hankkia tietoa modernista JavaScript-kehityksestä.

3.2 PHP

PHP on ohjelmointikieli, jota käytetään erityisesti dynaamisten verkkosivujen kehityksessä. PHP-koodi voidaan upottaa osaksi HTML-kieltä (ks. esimerkkikoodi 1). PHP-koodille on määritelty aloitus- ja lopetusmerkit, `<?php` ja `?>` , joiden sisään

koodi kirjoitetaan. PHP:tä ei käännetä konekieliseksi, vaan palvelimen PHP-tulkki palauttaa koodista muodostetun verkkosivun, eli loppukäyttäjän on mahdoton päästä näkemään itse PHP-koodia. (What is PHP? N.d.) PHP:tä käytetään yli kahdessakymmenessä miljoonassa verkkosivussa tai sovelluksessa. PHP:n suuren käyttäjämäärän nähdään johtuvan sen avoimesta lähdekoodista, suhteellisen helposta opettelusta ja säännöllisistä päivityksistä. PHP:n suuri etu on myös sen kyky käsitellä MySQL-tietokantaa. (What is PHP? Write your first PHP program. n.d.)

```
<!DOCTYPE HTML>
<html>
    <head>
        <title>Example</title>
    </head>
    <body>
         <h1>
             <?php echo 'Hello, World'; ?>
         </h1>
    </body>
</html>
```

Esimerkkikoodi 1. PHP-koodin upotus HTML-koodiin

3.3 React

React tai React.js on Facebookin kehittämä JavaScript-kirjasto. React kehitettiin ratkaisuksi haasteisiin monimutkaisten käyttöliittymien kehityksessä, joissa tietokokonaisuudet muuttuvat ajan mittaan. Reactin idea syntyi Facebookin insinöörien tarpeista kehittää skaalautuvia ja ylläpidettäviä palveluita, joissa uusien toimintojen lisääminen ei rikkoisi tai pakottaisi uudelleen rakentamaan sovellusta. Facebookin insinöörit halusivat myös sovellusten tietojen muuttuvan käyttöliittymissä dynaamisesti. Kun vanhemmilla tekniikoilla toteutetun sovelluksen tietokokonaisuudet muuttuivat, sovellus ajoi koko muuttuneen koodin uudestaan, joka johti ruudun välähdyksiin ja suorituskyvyn hidastumiseen. (Gackenheimer 2015, 1 – 3.)

Virtual DOM

DOM, eli Document Object Model tarkoittaa puurakennetta sovelluksen käyttöliittymästä. Aina kun sovelluksen käyttöliittymän tila muuttuu, DOM päivitetään vastaamaan kyseistä muutosta. DOMin päivittäminen usein on kuitenkin hyvin raskasta suorituskyvyn kannalta ja näin sovellus hidastuu. React käyttää virtuaalista DOMia käyttöliittymän päivittämisessä, joka parantaa sovelluksen suorituskykyä huomattavasti. (Hamedani 2018.)

Virtuaalisella DOMilla tarkoitetaan virtuaalista kuvausta DOM-rakenteesta. Joka kerta kun sovelluksen tila muuttuu, React päivittää virtuaalisen DOMin todellisen DOMin sijaan. Kun käyttöliittymään lisätään uusia elementtejä, luodaan aina uusi virtuaalinen DOM, joka esitetään myös puurakenteena. Jokainen elementti on niin kutsuttu solmu (engl. node) tässä puussa. Mikäli näiden elementtien tila muuttuu, luodaan aina uusi virtuaalinen DOM-puu, jota verrataan edellisen puun rakenteeseen. Tämän jälkeen virtuaalinen DOM laskee parhaan mahdollisen tavan päivittää todellista DOM-puuta. Näin Reactin avulla varmistetaan, että todellinen DOM kokee aina mahdollisimman vähän muutoksia. (Mt.)

JSX

React-dokumentaatiossa törmää usein käsitteisiin nimeltä ES5, ES6 ja JSX. Näistä ES5 ja ES6 ovat Ecma-organisaation kehittämiä JavaScript-kielen standardisoituja määritelmiä. ES5:llä tarkoitetaan usein ”tavallista” JavaScriptiä. Se julkaistiin vuonna 2009 ja sitä tukevat kaikki käytetyimmät verkkoselaimet. ES6 on vuonna 2015 julkaistu uudempi versio, joka lisäsi JavaScript-kieleen muutamia toimintoja ja parannuksia. ES6 ei ole vielä täysin tuettu kaikkien selaimien toimesta, joten ES6-koodi joudutaan kääntämään taaksepäin yhteensopivaan muotoon. (Lerner n.d.)

JSX:llä tarkoitetaan syntaksilaajennosta JavaScriptiin. Suurin osa React-kehittäjistä käyttää JSX-syntaksia. (Tutorial: Intro to React n.d.) JSX on JavaScript-kielen laajennus, joka muistuttaa hieman HTML-kieltä. Ideana on yhdistää samoihin komponentteihin HTML-kielellä rakennetut käyttöliittymät ja sen taustalla tapahtuva logiikka, jotka on perinteisesti pitänyt rakentaa erillisiksi osiksi. React ei pakota käyttämään JSX-syntaksia, mutta sitä suositellaan käytettävän käyttöliittymän

rakenteen hahmottamiseksi JavaScript-koodin sisällä. JSX-syntaksi kootaan aina lopuksi tavalliseksi JavaScript-koodiksi. (Introducing JSX n.d.)

Komponentit ja funktiot

Kaikkien React-sovellusten ytimessä ovat komponentit. Komponentti on itsenäinen moduuli, josta käyttöliittymät rakennetaan. React-komponentit ovat yhdistettävissä, eli komponentti voi sisältää myös useita muita komponentteja. Useat pienemmät komponentit muodostavat sovelluksen lopullisen käyttöliittymän rakenteen. (Lerner n.d.) Kun sovelluksen tietokokonaisuudet päivittyvät, React päivittää vain tarvittavat komponentit (Tutorial: Intro to React n.d.). Komponentti voidaan luoda Käyttämällä funktioita tai luokkia (ks. esimerkkikoodit 2 ja 3), jotka hyväksyvät parametreja nimeltä `props`, eli properties, ja palauttavat lopuksi React-elementtejä. (Components and Props n.d.)

```
function Welcome(props) {
    return <h1>Hello, {props.name}</h1>;
}
```

Esimerkkikoodi 2. Komponentin muodostus funktiona (Components and Props n.d., muokattu)

```
class Welcome extends React.Component {
  render() {
    return <h1>Hello, {this.props.name}</h1>;
  }
}
```

Esimerkkikoodi 3. Komponentin muodostus luokkana (Components and Props n.d., muokattu)

```
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}

const element = <Welcome name="World" />;
ReactDOM.render(
  element,
  document.getElementById('root')
);
```

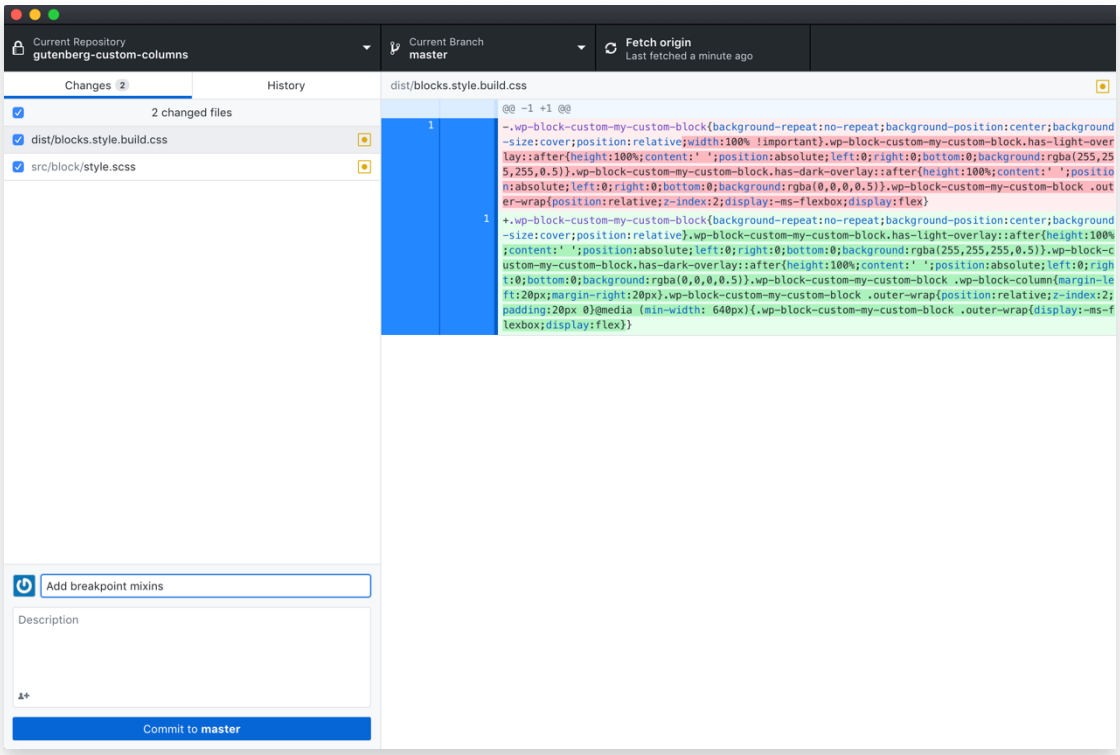
Esimerkkikoodi 4. React kohtelee pienillä kirjaimilla alkavia komponentteja DOM-tageina. Esimerkiksi `<h1 />` esittää HTML-kielen tagia, kun taas `<Welcome />` esittää komponenttia. Komponentti renderöidään näyttöruudulle kutsumalla `ReactDOM.render()` -metodia `<Welcome name="World" />` -elementillä. Tämän jälkeen React kutsuu `Welcome`–komponenttia, `{name: 'World'}` `props`-parametrina. `Welcome`-komponentti palauttaa `<h1>Hello, World</h1>` -elementin lopputuloksena (Components and Props n.d., muokattu)

3.4 Versionhallinta

Tämän opinnäytetyön versionhallinnassa hyödynnetään Git-työkalua. Git on Linus Torvaldsin kehittämä versionhallintaohjelmisto, joka syntyi tarpeesta kehittää avoimen lähdekoodin ohjelmisto, joka toimisi nopeasti suurissa projekteissa, joissa saattoi olla satoja tai jopa tuhansia kehittäjiä. (Brown 2018.) Gitissä projektit tallennetaan ohjelmavarastoihin (engl. repositories), jotka voivat sisältää useita brancheja eli versioita projektista. Versioiden muutoksia voidaan verrata toisiinsa ja tarpeen tullen versiot voidaan yhdistää (engl. merge) helposti. (Branching and Merging n.d.) Git on komentorivisovellus, joten sitä käytetään tietyillä komennoilla tietokoneen ohjaukseen tarkoitetulla komentotulkilla (Dudler n.d.).

Jotta projektin versiot olisivat tallessa myös muualla kuin kehittäjän tietokoneella, voidaan projekti tallentaa GitHub-sivustolle. GitHub on pilvipalvelu, jonne voidaan tallentaa Git-versionhallintaa hyödyntävät projektit. GitHub mahdollistaa usean kehittäjän samanaikaisen työskentelyn projektin parissa. (Brown 2017.) GitHubin ohjelmavarastot ovat nykyisin ilmaisia rajoitetuin ominaisuuksin. GitHub tarjoaa Git-

työkaluun graafisen käyttöliittymän. Gitin käyttöä paikallisesti omalla tietokoneella voidaan helpottaa GitHub Desktop -sovelluksella (ks. kuvio 4), jolla versionhallintaa voidaan suorittaa lähes kokonaan ilman komentotulkin ja useiden, melko vaikeasti muistettavien komentojen käyttöä. (What is GitHub Desktop and who is it for? 2018.)



Kuvio 4. Kuvakaappaus GitHub Desktop -sovelluksesta

3.5 Kehitysympäristö

Lohkon kehitys aloitetaan asentamalla kehitysympäristö. Käyttöjärjestelmänä käytetään Mac OS 10.14.3. Tekstieditoriksi ohjelmointityöhön valikoitui GitHubin kehittämä Atom, joka sisältää oletuksena tuen PHP- ja JSX-syntakseille. Atom myös sisältää helppokäyttöisen paketinhallintajärjestelmän, jolla siihen voidaan ladata lisäosia. Lisäosien avulla Atomiin pystytään lisäämään esimerkiksi tukia muille ohjelmointikielille. (Atom Basics n.d.; Why Atom? N.d.)

WordPress-kehitystä varten on hyvä pystyttää paikallinen WordPress-asennus. Näin tehtyjä muutoksia pääsee näkemään reaaliajassa ja testaamaan kehittäjän omalla

tietokoneella eikä muutoksia tarvitse ensin viedä tuotantopalvelimelle. WordPressin paikalliseen käyttöön on tarjolla useita vaihtoehtoja. Perinteisesti tähän tarkoitukseen on asennettu tietokoneelle paikallinen palvelin, johon WordPress asennetaan. Tämä vaihtoehto sisältää kuitenkin monia vaiheita ja paljon manuaalista työtä.

Tässä opinnäytetyössä tullaan kokeilemaan tuoreempaa ratkaisua, eli Flywheelin tarjoamaa Local-sovellusta, joka sen kehittäjien mukaan automatisoi lähes kaikki vaiheet WordPressin asennuksessa. Local-sovellus on ollut markkinoilla vuodesta 2016 ja yksittäiselle käyttäjälle se on täysin ilmainen. (About n.d.) Localin asennuksen yhteydessä asennetaan automaattisesti sen käyttämät työkalut. Local käyttää VirtualBox- ja Docker-työkaluja virtuaalikoneen luomiseen, jonka päälle se asentaa WordPressin automaattisesti. (Features n.d.)

4 Lohkon kehitys

4.1 Create Guten Block -työkalun käyttö

Gutenberg-lohkon kehitys aloitetaan luomalla kansiorakenne projektille. Kuten muutkin kustomoidut WordPress-toiminnollisuudet, myös lohkot kehitetään lisäosien avulla. Jotta WordPress tunnistaisi projektin lisäosaksi, tulee sille luoda lisäosalle ominainen PHP-päätiedosto (ks. esimerkkikoodi 5). Tämän lisäksi lohkon React-kehitysmalli vaatii toimiakseen useiden konfiguraatioiden ja riippuvuuksien asennusta. Tämän työvaiheen helpottamiseksi on kuitenkin kehitetty työkalu. Create Guten Block on Ahmad Awaisin kehittämä avoimen lähdekoodin työkalu, joka luo automaattisesti lohkon vaatimat kansiorakenteet, tiedostot ja konfiguraatiot sekä lataa tarvittavat JavaScript-paketit. (Klein 2018.)

```
<?php
/**
 * Plugin Name: My Custom Block
 * Description: My Custom Block – is a Gutenberg plugin
created via create-guten-block.
 * Author: Olli-Ilari Kärkkäinen
 * Author URI: https://github.com/ollikoo
 * Version: 1.0.0
 * License: GPL2+
 * License URI: https://www.gnu.org/licenses/gpl-2.0.txt
 *
 * @package MCB
 */

// Exit if accessed directly.
if ( ! defined( 'ABSPATH' ) ) {
    exit;
}

/**
 * Block Initializer.
 */
require_once plugin_dir_path( __FILE__ ) .
'src/init.php';
```

Esimerkkikoodi 5. Lisäosan päätiedoston header-osio, jossa tulee mainita ainakin lisäosan nimi, tekijä ja versio

Työkalua käytetään komentotulkilla ja se vaatii toimiakseen Node.js- ja npm-työkalujen asennuksen (mt.). Tämän jälkeen Projektikansio luodaan navigoimalla komentotulkilla paikallisen WordPress-asennuksen `plugins`-kansioon. `Plugins`-kansioon päästään Local-sovellusta käytettäessä oletuksena `cd /Users/<Käyttäjä>/Local\ Sites/<WordPress-asennuksen nimi>/app/public/wp-content/plugins/` -komennolla. Seuraavaksi ajetaan komento `npx create-guten-block <Projektin nimi>`, joka luo projektikansion annetulla nimellä. Kehitys alotetaan siirtymällä lisäosakansion juureen ja ajamalla `npm start`-komento, joka käynnistää kehitystilan (mt.).

Kehitystilassa Create Guten Blockin mukana asennetut riippuvuudet kääntävät kirjotetut JavaScript- ja CSS/SASS-koodit yhteensopivampaan muotoon. Kehitystilan ollessa päällä komentotulkki ilmoittaa myös mahdollisista virheistä koodissa. Kun kehitystyö on valmis, ajetaan `npm run build`-komento, joka kokoaa kirjoitetuista koodeista optimoidut versiot `dist`-kansioon. (Mt.)

4.2 Gutenberg-arkkitehtuuri

Gutenbergin arkkitehtuuri koostuu erilaisista JavaScript-kirjastoista. Kirjastojen ideana on mahdollistaa jo valmiiksi kehitettyjen komponenttien, metodien ja funktioiden hyödyntäminen lohkojen kehityksessä. Tämä luku käsittelee vain tässä opinnäytetyössä hyödynnettyjä tärkeimpiä kirjastoja (ks. esimerkkikoodi 6).

```
const { __ } = wp.i18n;
const {
  registerBlockType,
  synchronizeBlocksWithTemplate,
} = wp.blocks;
const {
  BlockControls,
  BlockAlignmentToolbar,
  InspectorControls,
  InnerBlocks,
  Editable,
  MediaUpload,
  PanelColorSettings,
  getColorClassName,
  getColorObjectByColorValue,
} = wp.editor;
const {
    Toolbar,
    Button,
    Tooltip,
    PanelBody,
    PanelRow,
    SelectControl,
} = wp.components;
```

Esimerkkikoodi 6. Komponenttien tuonti Gutenberg-kirjastoista

Elements-kirjasto

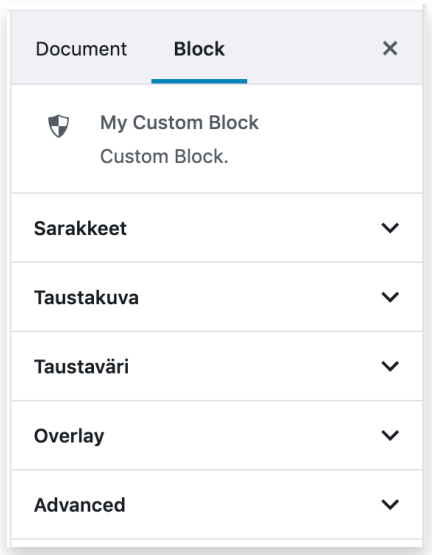
Näistä kirjastoista tärkein on Element-kirjasto, joka on Reactin päälle rakennettu abstraktiokerros. Abstraktiokerroksen ajatuksena on tarjota rajapinta kolmannen osapuolen koodiin, kuten tässä tapauksessa Reactiin. Tämän rajapinnan avulla riippuvan koodin liitäntä pysyy edelleen samana, vaikka taustalla olevaan kolmannen osapuolen koodiin tulisi muutoksia. Element-kirjastoa käytetään viittaamalla `wp.element`-olioon, joka tarjoaa käytettäväksi `Reactista` tutut `createElement`- ja `render`-metodit. (@wordpress/element n.d.)

Block Library -kirjasto

Varsinkin Gutenberg-kehitystä aloitteleville erityisen hyödyllinen kirjasto on Block Library. Nimensä mukaisesti se on kirjasto, joka sisältää editoriin valmiiksi kehitettyjen lohkojen lähdekoodit. Kehitettäessä täysin uutta lohkoa, voi Gutenberg-dokumentaation sijaan katsoa mallia juurikin Block Libraryn lohkoista. Esimerkiksi tämän opinnäytetyön yksi vaatimuksista oli, että kehitettävän lohkon sisältöä voidaan jaotella useisiin sarakkeisiin. Tämän toiminnon kehityksessä pystyttiin hyödyntämään osittain Sarakkeet-lohkon (engl. Columns) logiikkaa, joka nopeutti huomattavasti kehitystyötä.

Components-kirjasto

Components-kirjasto sisältää komponentteja, joita käytetään käyttöliittymien luomiseen WordPress-hallintapaneelissa (@wordpress/components n.d.). Tässä työssä lohkon asetukset aseteltiin PanelBody- ja PanelRow-komponenttien avulla haitarivalikkoon editorin oikeanpuoleiseen inspector-palkkiin (ks. kuvio 5).



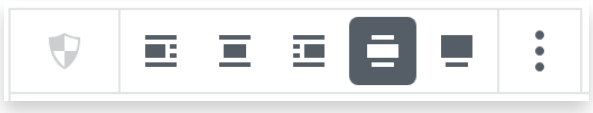
Kuvio 5. Kuvakaappaus luodusta lohkon asetusten haitarivalikosta

```
<InspectorControls>
    <PanelBody
        title={ __( 'Sarakkeet', 'custom' ) }
    >
        <PanelRow>
          <SelectControl
            label={__('Valitse lukumäärä', 'custom')}
            value={templateControl}
            options={[
              { value: "one", label: __("Yksi", "custom")
},
              { value: "two", label: __("Kaksi",
"custom") },
              { value: "three", label: __("Kolme",
"custom") }
            ]}
            onChange={templateControl => setAttributes({
templateControl })}
          />
        </PanelRow>
    </PanelBody>
<InspectorControls>
```

Esimerkkikoodi 7. InspectorControls-komponentin sisään luotu PanelBody- ja PanelRow-komponentit luovat automaattisesti haitarivalikon editorin inspector-palkkiin

Editor-kirjasto

Itse Gutenberg-editorin koodi on sisällytetty Editor-kirjastoon. Kyseinen kirjasto sisältää myös kehittäjälle tarpeellisia uudelleenkäytettäviä komponentteja. (@wordpress/editor n.d.) Esimerkiksi BlockControls-komponenttilla voidaan luoda ryhmä painikkeita lohkon asetusten muokkaamiseen, joka tulee näkyviin lohkon yläreunaan, kun se on valittuna (ks. kuvio 6).



Kuvio 6. Kuvakaappaus lohkon asettelua säätävästä BlockControls-komponentista

Internationalization-kirjasto

Lohkojen käyttöliittymää luodessa on oleellista ottaa huomioon myös niiden käännettävyys muille kielille. Gutenbergiin on luotu i18n-kirjasto, jonka avulla kaikista lohkon string-tietotyypin merkkijonoista voidaan tehdä käännettävät. String-merkkijonot käännetään `__` -funktiolla, joka tuodaan JavaScript-koodiin luomalla muuttuja `const { __ } = wp.i18n;`. Funktiota käytetään kirjaamalla string-merkkijonot muotoon `__( 'Hello World!', 'nimiavaruus' );`. Mikäli valmis lisäosa ladataan jaettavaksi wordpress.org-palveluun, sen käännettävät merkkijonot menevät automaattisesti WordPress-yhteisön käännettäväksi. Lisäosaan voidaan liittää myös itse tehty käännöstiedosto. (Internationalization n.d.)

4.3 Lohkon rekisteröinti

Lohko rekisteröidään `registerBlockType()`-funktiolla, johon syötetään ensimmäisenä argumenttina lohkon nimi muodossa `nimiavaruus/lohkon-nimi`, jossa nimiavaruus on joko lisäosan tai teeman nimi. Toisena argumenttina syötetään konfiguraatio-olio, joka sisältää lohkon ominaisuudet. Ominaisuuksista neljä ensimmäistä ovat pakolliset string-tietotyyppinä syötettävät `title`, `icon`, `category` ja `keywords`. `Title` on lohkon nimi, joka näytetään

hallintapaneelissa. `Icon` puolestaan on myös hallintapaneelissa näytettävä lohkon ikoni, jona voidaan käyttää WordPressin Dashicons-hakemistosta löytyviä ikoneita tai vaihtoehtoisesti tuoda omavalintainen SVG-ikoni. `Category` on lohkon kategoria, jonka avulla lohko voidaan lisätä johonkin ennalta määritetyistä kategorioista. Lohkolle on myös mahdollista luoda kokonaan oma kategoriansa. `Keywords` taas on taulukko, johon syötetään avainsanoja, joilla lohkon löytää helpommin hallintapaneelista. (Block Registration n.d.)

Edellä mainittujen lisäksi oleellisia ominaisuuksia ovat `attributes`, `edit` ja `save`. `Attributes` tarkoittaa lohkossa käytettäviä attribuutteja. Attribuuteille annetaan nimen lisäksi ominaisuuksia, kuten esimerkiksi tietotyyppi ja oletusarvo. (Block Registration n.d.)

```
attributes: {
    blockAlignment: {
        type: 'string',
        default: 'wide',
    },
    templateControl: {
        type: 'string',
        default: 'one',
    },
    imgURL: {
        type: 'string',
    },
    imgID: {
        type: 'number',
    },
    imgAlt: {
        type: 'string',
    },
    backgroundColor: {
        type: "string",
    },
    overlayControl: {
        type: "string",
        default: "none",
    }
},
```

Esimerkkikoodi 8. Kehitystyössä käytettyjä lohkon attribuutteja

`Edit` ja `save` puolestaan ovat funktioita. `Edit`-funktio kuvaa lohkon rakennetta editorin yhteydessä. Se myös määrittelee mitä editori tekee, kun lohkoa käytetään. `Edit`-funktiolle voi syöttää aiemmin mainitut attribuutit argumenttioliona. Samaan argumenttiolioon voi sisällyttää myös monia valmiiksi kehitettyjä ominaisuuksia. Näistä tärkeimmät ovat `className`, joka muodostaa lohkon nimestä automaattisesti luokan sekä `setAttributes`, joka mahdollistaa lohkon attribuuttien päivityksen käyttäjän toimesta. Lopuksi `save`-funktiolla tallennetaan lohkon attribuuttien arvot ja lopulliset merkkaukset WordPressin tietokantaan. Useimmissa lohkoissa `save`-funktio palauttaa elementin, joka määrittää lohkon ulkonäön sivuston vierailijoille näkyvällä front end-puolella. (Edit and Save n.d.)

Sisäisten lohkojen hyödyntäminen

Mikäli lohkon sisälle halutaan liittää muita lohkoja, voidaan `edit`- ja `save`-funktioissa käyttää `InnerBlocks`-komponenttia (ks. esimerkkikoodi 8). Tämän työn yksi tärkeä vaatimus oli, että lohkoon tulee voida syöttää monimuotoista sisältöä yhteen tai useampaan sarakkeeseen. Gutenberg-editori sisältää jo valmiiksi lohkoja, joilla voi syöttää tarvittavia sisältöjä, joten `InnerBlocks`-komponenttia päätettiin käyttää näiden lohkojen lisäämiseen. Koska `edit`- ja `save`-funktiot voivat kumpikin renderöidä vain yhden `InnerBlocks`-komponentin, ei useamman sarakkeen asettelua voitu luoda vain lisäämällä useampi komponentti. `InnerBlocks`-komponentin sisälle voidaan kuitenkin määritellä valmiita asetteluita (engl. templates) muista lohkoista. Sarakkeiden luomiseen käytettiin kolmea asettelua, joissa toistetaan Gutenbergin sarakelohkoa halutun sarakkeiden lukumäärän verran. Gutenbergin sarakelohko sisältää oman `InnerBlocks`-komponentin, joten näin useamman komponentin rajoitus renderöinnissä voitiin kiertää.

```
<div className={ classes } style={ divStyle }>
    <div className={'outer-wrap'}>
        <InnerBlocks
            template={
TemplateSelector({templateControl}) }
            allowedBlocks= { ALLOWED_BLOCKS }
            templateLock= "all"
        />
    </div>
</div>
```

Esimerkkikoodi 8. `Edit`-funktiossa palautettu editorissa näkyvä elementti, joka sisältää `InnerBlocks`-komponentin, jonka asettelu määritetään erillisessä funktiossa

```
const ALLOWED_BLOCKS = [ 'core/column' ];

const TEMPLATE_ONE = [
    ['core/column', {} ]
];
const TEMPLATE_TWO = [
    ['core/column', {} ],
    ['core/column', {} ]
];
const TEMPLATE_THREE = [
    ['core/column', {} ],
    ['core/column', {} ],
    ['core/column', {} ]
];

function TemplateSelector(columns) {
    if (columns.templateControl == "one") {
      return TEMPLATE_ONE;
    } else if(columns.templateControl == "two") {
      return TEMPLATE_TWO;
    } else if(columns.templateControl == "three") {
      return TEMPLATE_THREE;
    }
    ;
};
```

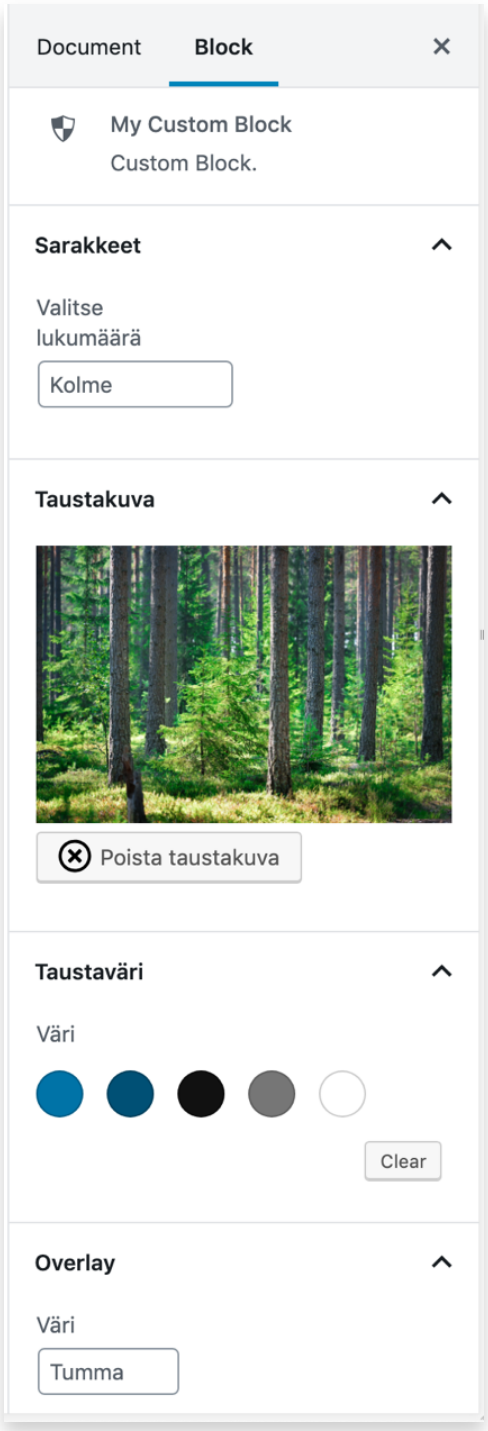
Esimerkkikoodi 10. Kolme eri asettelua, joissa toistetaan Gutenbergin sarakelohkoa sekä asettelun määrittelevä funktio

5 Pohdinta

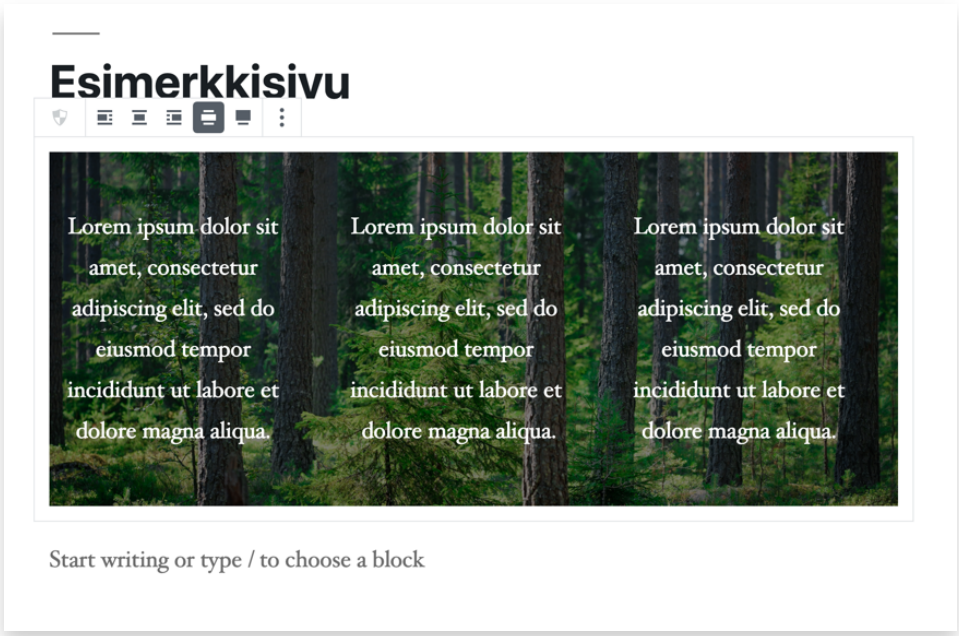
5.1 Kehitystyön tulokset

Opinnäytetyön tutkimusongelman muodostivat toimeksiantajayrityksen puutteet kokonaan uuden WordPress-kehitysmallin tuntemisessa. Uuden Gutenberg-editorin sekä sen kehitysmallin käyttöönotto ei ole kuitenkaan vielä täysin pakollista, sillä vanhempaa editoria sekä kehitysmallia tullaan tukemaan ainakin vuoteen 2022. WordPress-kehittäjäyhteisön tavoitteena oli uuden päivityksen myötä helpottaa järjestelmän kehitystä ja parantaa samalla käyttäjäkokemusta. Näiden tekijöiden kehittämisellä toimeksiantajayritys voisi teoriassa saavuttaa kilpailuedun, mutta toisaalta näin tuoreen teknologian käyttöönotossa on myös omat riskinsä, minkä vuoksi selvitys aiheesta nähtiin tarpeellisena.

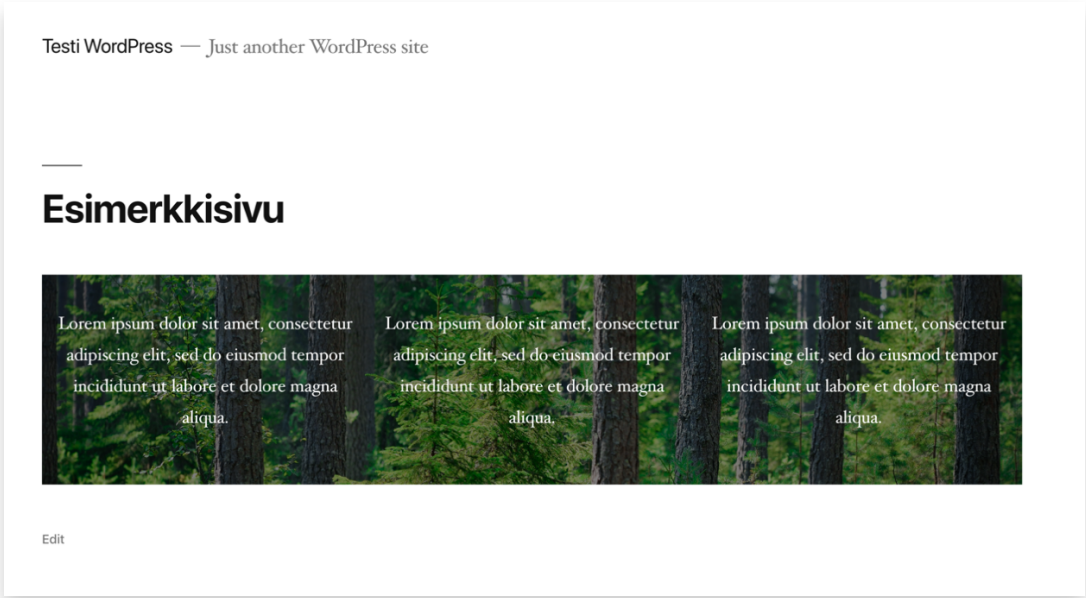
Tämän opinnäytetyön tuloksena kehitettiin sisältölohko toimeksiantajayrityksen käyttämään WordPress-sisällönhallintajärjestelmään. Vaatimusmäärittelyssä kuvattuja reunaehtoja lohkolle olivat monimuotoisen sisällön luominen, taustakuvan tai -värin valitseminen ja lohkon sisällön asettelu useampaan sarakkeeseen sekä sen leveyden määrittäminen. Vaaditut toiminnollisuudet ovat web-kehityksen maailmassa hyvinkin tavallisia, mutta valmiiksi kehitetyillä lohkoilla ei kaikkia vaadittuja ominaisuuksia ollut. Valmiiksi kehitetyillä lohkoilla pystyy kuitenkin lisäämään monia eri sisältömuotoja, kuten mediaupotuksia, kuvia, karttoja ja tekstiä, minkä takia kehitettävään lohkoon luotiin mahdollisuus liittää näitä lohkoja. Myös muiden vaatimusten täyttämiseen pystyttiin hyödyntämään Gutenbergin kirjastojen komponentteja sekä valmiiden lohkojen avointa lähdekoodia. Lopputuloksen voi nähdä käyttäjäystävällisenä, sillä lohko sisältää selkeät ja yksinkertaiset asetukset (ks. kuvio 7). Lohko myös näyttää lähes samalta sekä editor- että front end-puolella, joka helpottaa sivuston sisällön muokkausta (ks. kuviot 8 ja 9).



Kuvio 7. Kuvakaappaus lohkon asetuksista inspector-paneelista



Kuvio 8. Kuvakaappaus lohkosta editorin puolella



Kuvio 9. Kuvakaappaus lohkosta sivun front end-puolelta

5.2 Gutenberg-kehityksen parhaat käytänteet

Yksi tämän opinnäytetyön tavoitteista oli tutkia ja soveltaa Gutenberg-lohkon kehityksen parhaita käytänteitä. Työhön onnistuttiin valikoimaan juuri oikeat

käytänteet, sillä koko prosessiin kului suunnitelman mukaisesti noin kaksi viikkoa osa-aikaisesti työskennellen. Smithin (2009) mukaan Hunt ja Thomas (1999) kuvaavat yhdeksi ohjelmistokehityksen pääperiaatteeksi itsensä toistamisen välttämisen. Tässä työssä esitellyillä käytänteillä onnistuttiin erityisesti minimoimaan usein toistettavia työvaiheita ja näin nopeuttamaan sekä helpottamaan kehitystyötä. Ohjelmointityössä ei kehittäjä voi ottaa kovinkaan suuria vapauksia, sillä lohko vaatii toimiakseen hyvin pitkälle ennalta määritetyn rakenteen, mikä toisaalta yksinkertaistaa ohjelmointityötä ja tekee koodista luettavamman sekä vähentää yleisten virheiden määrää.

WordPress tarjoaa Gutenberg-lohkojen kehitykseen omat tutoriaalinsa, jotka sisältävät kuitenkin turhan monta vaihetta. Lohkon kehitysvaiheeseen päästäkseen, joutuu seuraamaan ohjeita useamman sivun ja eri vaiheen verran. Kehitystä nopeuttamaan löydettiin Create Guten Block -niminen työkalu, jonka avulla automatisoidaan kehitystyön kannalta kriittisen projektihakemiston ja tärkeiden tiedostojen luonti ja kehitystyössä tarvittavien JavaScript-pakettien lataus. Työkalu perustuu avoimeen lähdekoodiin ja on ladattavissa GitHub-palvelusta, jossa sen käyttöön on tarjottu myös selkeät ohjeet.

Gutenberg-kehityksen parhaisiin käytänteisiin kuuluu myös paikallisen WordPress-asennuksen pystyttäminen. Näin tehtyjä muutoksia pääsee näkemään reaaliajassa, eikä niitä tarvitse viedä tuotantopalvelimelle kaikkien sivuston vierailijoiden nähtäväksi. WordPress tarjoaa omassa dokumentaatiossaan vaihtoehdoksi paikallisen palvelimen asennusta tietokoneelle ja uusimman WordPress-version lataamista internetistä ja asentamista palvelimelle. Tämän vaihtoehdon voi kuitenkin nähdä vanhentuneena sen monivaiheisuuden ja turhan toistamisen vuoksi. Paikallista asennusta helpottamaan löydettiin modernimpi ratkaisu, Flywheelin tarjoama Local-sovellus, joka automatisoi prosessin lähes kokonaan.

Versionhallinnassa hyödynnettiin GitHub-palvelun tarjoamia ohjelmavarastoja ja GitHub Desktop -sovellusta. Git-sovelluskehitysprojekteissa versionhallintaa käytetään usein toistettavilla ja melko vaikeasti muistettavilla komennoilla komentotulkin kautta. Desktop-sovellus helpotti versionhallintaa huomattavasti, sillä

tehdyt muutokset voitiin viedä ohjelmavarastoon graafista käyttöliittymää käyttäen ilman edellä mainittuja komentoja ja komentotulkkia.

Vaikka kehitystyö suunniteltiin toteuttavan melko lyhyessä ajassa, tavoitteena oli kuitenkin tutkia, mitä projektinhallinnan menetelmiä voitaisiin hyödyntää. Menetelmistä käytettäväksi valikoitui ohjelmistokehityksessä varsin yleinen Scrum-menetelmä. Teoriataustasta kävi ilmi menetelmän pääkäyttötarkoitus muutaman henkilön kokoisten tiimien työskentelyssä. Oleellista tämän työn kannalta oli selvittää, miten useamman henkilön tiimille tarkoitettu menetelmä toimisi yhden henkilön kehitystyössä. Kerätyn aineiston perusteella Scrum-menetelmästä päädyttiin ottamaan käyttöön vain sen tärkeimmät tapahtumat ja tuotokset, kuten sprintti ja sen suunnittelusessio, daily scrum, retrospektiivi, kehitysjono ja tehtävätaulu. Näitä hyödyntäen kehitystehtävät voitiin suorittaa loogisessa järjestyksessä tehtävä kerrallaan. Tämän ansiosta aikaa jäi myös reflektointiin sekä toiminnollisuuksien testaamiseen. Scrumin periaatteiden mukaan kehitettävää lohkoa jaettiin sprinttien aikana testattavaksi mahdollisimman usein.

5.3 Jatkotutkimusaiheita

Suurin kysymysmerkki WordPress-yhteisössä on ollut lohkojen virallisen kehitysmallin mukana tullut kehittäjille tuntemattomampi JavaScript-ohjelmointikieli. Kirjoitushetkellä tarjolle on jo tullut useita vaihtoehtoisia menetelmiä lohkojen kehitykseen. Näitä vaihtoehtoisia menetelmiä ovat muun muassa lisäosat, joiden avulla pystyy luomaan lohkoja WordPress-kehittäjille tutummalla PHP-kielellä. Useat vaihtoehtoiset menetelmät ovat kuitenkin vielä varhaisessa vaiheessa ja sisältävät omat rajoituksensa. Kiinnostavaa onkin nähdä, mikä menetelmä tulevaisuudessa saa suurimman käyttäjäkunnan taakseen. Vaihtoehtoiset menetelmät avaavat myös mahdollisuuden tutkia eroavaisuuksia niiden ja virallisemman menetelmän välillä. Kirjoitushetkellä on jo yleisesti tiedossa, että Gutenberg-editorista tuttuja lohkoja tullaan myöhemmissä kehitysvaiheissa hyödyntämään muiden asetuksien muokkaamisessa ja vanhempien ominaisuuksien korvaamisessa (Choyce 2018).

Lähteet

About. N.d. Tietoa Flywheel-yhtiöstä. Viitattu 23.4.2019. https://getflywheel.com/about/.

Ambler, S. N.d. Tietoa käyttäjätarinoista. Viitattu 14.4.2019. http://www.agilemodeling.com/artifacts/userStory.htm#Introduction.

Andrews, A. 2017. Scrum of One: How to Bring Scrum into your One-Person Operation. Artikkeli raywenderlich.com-sivustolla 2.6.2017. Viitattu 10.4.2019. https://www.raywenderlich.com/585-scrum-of-one-how-to-bring-scrum-into-your-one-person-operation.

Atom Basics. N.d. Atom-tekstieditorin dokumentaatio. Viitattu 23.4.2019. https://flight-manual.atom.io/getting-started/sections/atom-basics/.

Block Registration. N.d. Gutenberg-editorin käsikirja. Viitattu 26.4.2019. https://wordpress.org/gutenberg/handbook/designers-developers/developers/block-api/block-registration/.

Branching and Merging. N.d. Tietoa Git-versionhallinnasta. Viitattu 22.4.2019. https://git-scm.com/about/branching-and-merging.

Brown, K. 2017. What is GitHub, and What Is It Used For? Artikkeli How-To Geek www-sivustolla 6.9.2017. Viitattu 22.4.2019. https://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/.

Brown, Z. 2018. A Git Origin Story. Artikkeli Linux Journalin www-sivustolla 27.6.2018. Viitattu 22.4.2019. https://www.linuxjournal.com/content/git-origin-story.

Choyce, M. 2018. Gutenberg Phase 2. Artikkeli make.wordpress.org-sivustolla 9.12.2018. Viitattu 8.5.2019. https://make.wordpress.org/core/2018/12/08/gutenberg-phase-2/.

Components and Props. N.d. Dokumentaatio React-kehityksestä. Viitattu 19.4.2019. https://reactjs.org/docs/components-and-props.html.

Dudler, R. N.d. git – the simple guide. Opas Git-komennoista. Viitattu 22.4.2019. http://rogerdudler.github.io/git-guide/.

Edit and Save. N.d. Gutenberg-editorin käsikirja. Viitattu 2.5.2019. https://wordpress.org/gutenberg/handbook/designers-developers/developers/block-api/block-edit-save/.

Ewer, T. 2018. How to Get Started With WordPress Development. Artikkeli wpexplorer.com-sivustolla. Päivitetty 5.5.2018. Viitattu 8.5.2019. https://www.wpexplorer.com/getting-started-wordpress-development/.

Fakhroutdinov, K. N.d. Kuvauksia UML-diagrammeista. Viitattu 11.4.2019. https://www.uml-diagrams.org/use-case-diagrams.html.

Features. N.d. Local-sovelluksen ominaisuudet. Viitattu 23.4.2019. https://localbyflywheel.com/features/.

Features. N.d. WordPress-järjestelmän ominaisuudet. Viitattu 1.4.2019. https://wordpress.org/about/features/.

Gackenheimer, C. 2015. Introduction to React. Apress. E-kirja Googlen Books-palvelussa. Viitattu 19.4.2019. https://books.google.fi/books/about/Introduction_to_React.html?id=NZCKCgAAQBAJ&redir_esc=y.

Garbar, D. 2016. Software Requirements Specification Helps to Protect IT Projects From Failure. Artikkeli Belitsoftin www-sivustolla 8.8.2016. Viitattu 5.4.2019. https://belitsoft.com/php-development-services/software-requirements-specification-helps-protect-it-projects-failure.

Hamedani, M. 2018. React Virtual DOM Explained in Simple English. Artikkeli Mosh Hamedanin www-sivustolla 3.12.2018. Viitattu 20.4.2019. https://programmingwithmosh.com/react/react-virtual-dom-explained/.

Hughes, J. 2017. Themeisle www-sivuston artikkeli 19.1.2017. Viitattu 15.4.2019. https://themeisle.com/blog/wordpress-core-files/.

Internationalization. N.d. Gutenberg-editorin käsikirja. Viitattu 4.5.2019. https://wordpress.org/gutenberg/handbook/designers-developers/developers/internationalization/.

Introducing JSX. N.d. JSX-syntaksin esittely. Viitattu 18.4.2019. https://reactjs.org/docs/introducing-jsx.html.

Introduction to Plugin Development. N.d. WordPress-lisäosan kehittämisen käsikirja. Viitattu 15.4.2019. https://developer.wordpress.org/plugins/intro/.

Kananen, J. 2012. Kehittämistutkimus opinnäytetyönä. Jyväskylän ammattikorkeakoulu. Viitattu 9.4.2019.

Klein, F. 2018. The Beginner’s Guide to Create Guten Block. Create Guten Block -työkalun ohje. Päivitetty 16.4.2019. Viitattu 25.4.2019. https://wpdevelopment.courses/create-guten-block-beginners-guide/.

Lerner, A. N.d. What is JSX? Verkkokurssi React-kehityksestä. Osa 2. Viitattu 19.4.2019. https://www.fullstackreact.com/30-days-of-react/day-2/.

Lerner, A. N.d. What is React? Verkkokurssi React-kehityksestä. Osa 1. Viitattu 19.4.2019. https://www.fullstackreact.com/30-days-of-react/day-1/.

Mullenweg, M. 2018. WordPress 5.0: A Gutenberg FAG. Blogiartikkeli Matt Mullenwegin www-sivustolla 29.10.2018. Viitattu 16.4.2019. https://ma.tt/2018/11/a-gutenberg-faq/.

Requirements. N.d. Luettelo WordPress-sisällönhallintajärjestelmän vaatimuksista. Viitattu 11.4.2019. https://wordpress.org/about/requirements/.

Rouse, M. 2007. Use Case. Määrittely käyttötapaus-käsitteelle SearchSoftwareQualityn www-sivustolla. Viitattu 2.4.2019. https://searchsoftwarequality.techtarget.com/definition/use-case.

Schwaber, K. Sutherland, J. 2017. Scrum-opas. Viitattu 2.4.2019. https://scrumwell.files.wordpress.com/2018/03/2017-scrum-guide-fi-v1-02.pdf.

Shortcodes. N.d. WordPress-lisäosan kehittämisen käsikirja. Viitattu 11.4.2019. https://developer.wordpress.org/plugins/shortcodes/.
Smith, S. 2009. Don’t Repeat Yourself. Web-arkistosta haettu julkaisu oreilly.com-sivustolla. Päivitetty 24.11.2009. Viitattu 7.5.2019. https://web.archive.org/web/20131204221336/http://programmer.97things.oreilly.com:80/wiki/index.php/Don't_Repeat_Yourself.

Theme Directory. N.d. WordPress-teemojen tietokanta. Viitattu 14.5.2019. https://wordpress.org/themes/.

Theme Options – The Customize API. N.d. WordPress-teeman kehittämisen käsikirja. Viitattu 11.4.2019. https://developer.wordpress.org/themes/customize-api/.

Tutorial: Intro to React. N.d. React-JavaScript-kirjaston esittely. Viitattu 18.4.2019. https://reactjs.org/tutorial/tutorial.html.

Usage of content management systems. N.d. W3Techs-sivuston tilasto sisällönhallintajärjestelmien käyttäjämääristä. Viitattu 10.4.2019. https://w3techs.com/technologies/overview/content_management/all.

What is a Theme? N.d. WordPress teeman kehittämisen käsikirja. Viitattu 15.4.2019. https://developer.wordpress.org/themes/getting-started/what-is-a-theme/.

What is GitHub Desktop and who is it for? 2018. GitHub Desktop -sovelluksen dokumentaatio. Päivitetty 29.9.2018. Viitattu 22.4.2019. https://github.com/desktop/desktop/blob/development/docs/process/what-is-desktop.md.

What is PHP? N.d. Kuvaus PHP-ohjelmointikielestä. Viitattu 20.5.2019. https://secure.php.net/manual/en/intro-whatis.php.

What is PHP? Write your first PHP Program. N.d. Kuvaus PHP-ohjelmointikielestä. Viitattu 21.4.2019. https://www.guru99.com/what-is-php-first-php-program.html.

Why Atom? N.d. Atom-tekstieditorin dokumentaatio. Viitattu 23.4.2019. https://flight-manual.atom.io/getting-started/sections/why-atom/.

WordPress Coding Standards. N.d. WordPress-järjestelmän koodausohjeet. Viitattu 6.5.2019. https://make.wordpress.org/core/handbook/best-practices/coding-standards/.

WordPress Widgets. N.d. Dokumentaatio WordPress-lisäosien kehityksestä. Viitattu 11.4.2019. https://codex.wordpress.org/WordPress_Widgets
@wordpress/components. N.d. Gutenberg-editorin käsikirja. Viitattu 3.5.2019. https://wordpress.org/gutenberg/handbook/designers-developers/developers/packages/packages-components/.

@wordpress/element. N.d. Gutenberg-editorin käsikirja. Viitattu 3.5.2019. https://wordpress.org/gutenberg/handbook/designers-developers/developers/packages/packages-element/.

@wordpress/editor. N.d. Gutenberg-editorin käsikirja. Viitattu 3.5.2019. https://wordpress.org/gutenberg/handbook/designers-developers/developers/packages/packages-editor/.