Oskari Tauru

INTRANETIN TOTEUTUS WORDPRESS-SISÄLLÖNHALLINTA-JÄRJESTELMÄLLÄ

Opinnäytetyö

Liiketalouden ammattikorkeakoulututkinto

Tietojenkäsittelyn koulutus

2023

Kaakkois-Suomen ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tutkintonimike	Tradenomi (AMK)
Tekijä/Tekijät	Oskari Tauru
Työn nimi	Intranetin toteutus WordPress-sisällönhallintajärjestelmällä
Toimeksiantaja	Mainostoimisto Groteski Oy
Vuosi	2023
Sivut	37 sivua, liitteitä 2 sivua
Työn ohjaaja(t)	Miia Liukkonen

TIIVISTELMÄ

Opinnäytetyö käsittelee WordPress-sisällönhallintajärjestelmällä rakennettua intranetiä. Tavoitteena oli toteuttaa toimeksiantajan tarpeisiin sopiva alusta sisäisen viestinnän käyttöön yrityksessä. Intranetin pohjana käytettiin toimeksiantajan omaa räätälöityä WordPress-teemaa, joka sisälsi valmiiksi luotuja lohkoja sekä toiminnallisuuksia. Teeman lohkoja kehitettiin intranetin tarpeisiin sopiviksi sekä luotiin omia kustomoituja lohkoja. Toimeksiantajana toimii Mikkeliläinen Mainostoimisto Groteski Oy, joka erikoistuu räätälöityihin WordPress-verkkopalveluihin, WooCommerce-verkkokauppoihin, integraatioihin sekä markkinointiin digitaalisessa maailmassa perinteisten mainostoimistopalveluiden lisäksi.

Ennen varsinaista teknistä toteutusta, työssä käsitellään intranetin rakentamista varten käytetyt web-tekniikat. Näitä ovat mm. PHP, SCSS, Figma ja Vue.js. Työn teoriaosuus käsittelee web-tekniikoiden lisäksi yleisesti sisällönhallintajärjestelmien käyttötarkoituksia sekä ominaisuuksia. Työssä perehdytään syvemmin työn toteuttamiseen käytettyyn WordPress-sisällönhallintajärjestelmään ja sen lisäosiin sekä teemoihin.

Työn teknisessä toteutuksessa ensimmäiseksi suunniteltiin rautalankamalli Figmalla, minkä jälkeen rautalankamallia kehitettiin toimeksiantajan kanssa. Rautalankamallin pohjalta suunniteltiin ulkoasumalli käyttäen toimeksiantajan väriteemaa. Suunnitteluvaiheen jälkeen aloitettiin muokkaamaan toimeksiantajan WordPress-teemaa. Työn tuloksena saatiin toimiva ensimmäinen versio intranetistä, joka toimii toimeksiantajan tarpeisiin sopivana sisäisen viestinnän alustana. Intranet luotiin helposti muokattavaksi, joten toimeksiantajan on myös mahdollista kehittää sitä jatkossa omien tarpeidensa mukaisesti. Toimeksiantaja oli tyytyväinen intranetin ensimmäiseen versioon, joten se otettiin käyttöön yrityksessä.

Asiasanat: web-ohjelmointi, WordPress, tietojenkäsittely, sisällönhallintajärjestelmä

South-Eastern Finland University of Applied Sciences

Degree title	Bachelor of Business Administration
Author (authors)	Oskari Tauru
Thesis title	Intranet implementation with the WordPress content management system
Commissioned by	Mainostoimisto Groteski Oy
Time	2023
Pages	37 pages, 2 pages of appendices
Supervisor	Miia Liukkonen

ABSTRACT

The purpose of this thesis was to develop an intranet using the content management system WordPress. The main objective was to implement a platform suitable for the commissioner's needs for internal communication in the company. The intranet was implemented with the commissioner's own customized WordPress theme containing pre-created blocks and functionalities. This involved developing customized blocks for the intranet's needs and modifying the commissioner's theme blocks to fit the intranet.

Before the actual technical implementation, the theoretical part introduced web technologies that were used to develop the intranet. These web technologies were e.g., PHP, SCSS, Figma and Vue.js. In addition to web technologies, the theoretical part of the thesis dealt with the purposes and features of content management systems in general. As WordPress had a big part in the thesis, this content management system was introduced in more detail.

In the technical implementation, a wireframe was first designed with Figma. Based on the wireframe, an appearance model was designed using the commissioner's colour theme. After the planning phase, the commissioner's WordPress theme was introduced, and its modifying began. The main result of the thesis was a functional first version of the intranet, to serve as an internal communication platform suitable for the commissioner's needs. The intranet was created to be easily modified and possibly further developed. The commissioner was satisfied with the first version of the intranet, and it was deployed in the company.

Keywords: web programming, WordPress, data processing, content management system

SISÄLLYS

1 JOHDANTO

Opinnäytetyöni tavoitteena on luoda Mainostoimisto Groteski Oy:lle intranet WordPress-sisällönhallintajärjestelmällä yrityksen tarpeiden mukaan. Olin työharjoittelussa Groteskilla, jossa sain vahvan pohjan tekniikoihin, joita tarvitsen intranetin rakentamisessa.

Groteski on kasvava yritys mainostoimistona sekä digipalveluiden tuottajana. Heidän kasvunsa myötä kasvoi myös tarve sisäisen viestinnän alustalle, joka olisi helposti käytettävä sekä vaivattomasti yrityksen työntekijöiden saatavilla. Kehitin intranetin ominaisuudet toimeksiantajan kanssa heidän tarpeisiinsa sopiviksi.

Opinnäytetyössä keskityn verkkokehitykseen sopiviin tekniikoihin sekä menetelmiin. Työssä kerrotaan WordPress-sisällönhallintajärjestelmän käyttötarkoituksesta sekä lisäosien- ja teemojen hyödyntämisestä. Intranetin suunnitteluvaihe toteutetaan Figmalla, joka on käyttöliittymäsuunnittelun verkkosovellus. Intranetin pohjana käytetään Groteskin omaa räätälöityä WordPress-teemaa, joka sisältää valmiita asiakasprojekteihin luotuja lohkoja. Lohkoja luodaan sekä muokataan intranetin tarpeiden mukaan. Työssä kerrotaan, kuinka webtekniikoita on hyödynnetty intranetin luomisessa, joten työssä käsitellään vain työn kannalta tarpeelliset intranetin ominaisuudet. Intranetin ensimmäinen versio toteutetaan toimeksiantajalta saatujen ohjeiden mukaisesti.

Opinnäytetyöni toisessa luvussa esitellään työssä käytettyjen web-tekniikoiden perusteita sekä keskeisimpiä käsitteitä. Kolmas luku käsittelee sisällönhallintajärjestelmien käyttötarkoituksia ja ominaisuuksia niiden välillä. Luvussa pureudutaan syvemmin WordPress-sisällönhallintajärjestelmään. Neljännessä luvussa esitellään toimeksiantaja sekä käydään läpi intranetin tekninen toteutus vaihe vaiheelta. Luvussa viisi pohditaan kuinka intranetin ensimmäinen versio vastasi toimeksiantajan odotuksia sekä käydään läpi omia mietteitä työtä tehdessä.

2 WEB-TEKNIIKAT

Web-tekniikat ovat työkaluja ja tekniikoita, joita käytetään erilaisten laitteiden välisessä viestinnässä. Web-tekniikat voidaan luokitella seuraaviin osiin: World Wide Web (WWW), nettiselain, verkkopalvelin, verkkosivut ja verkkokehitys. (GeeksforGeeks 2022.)

Verkkokehityksellä tarkoitetaan verkkosivujen rakentamista, luomista ja ylläpitoa. Esimerkkejä verkkokehityksestä ovat web-suunnittelu, web-ohjelmointi ja tietokantojen hallinta. Verkkokehitys jaetaan kahteen eri osa-alueeseen, jotka ovat selainpuolen kehittäminen (engl. *Frontend Development*) ja palvelinpuolen kehittäminen (engl. *Backend Development*). (GeeksforGeeks 2022.)

Selainpuolen kehittämisellä tarkoitetaan- sitä osaa verkkosivusta, joka on suoraan vuorovaikutuksessa käyttäjän kanssa, ja taas palvelinpuolen kehittäminen on verkkosivuston osa, johon käyttäjä ei ole yhteydessä, näitä ovat esimerkiksi tietojen tallentaminen ja järjestäminen. (GeeksforGeeks 2022.)

2.1 PHP

PHP (engl. *Pre-Processor Hypertext*) on ohjelmointikieli, jota käytetään websovellusten suunnitteluun. Se on saanut vuosien varrella paljon suosiota palvelinpuolen skriptauskielenä. Se on helppokäyttöinen, mutta tehokas ohjelmointikieli, joka toimii useissa eri käyttöjärjestelmissä sekä tukee useita eri palvelemia. (Carr. & Gray 2018, 1.)

```php
<?php
    echo "Hello World";
?>
```

Kuva 1. Esimerkki PHP:n aloitus- sekä lopetustunnisteista

PHP:tä voidaan kirjoittaa lähes mihin tahansa tiedostossa, sen toimimiseen tarvitaan PHP-tunnisteet (engl. *tags*) ennen koodin alkua ja loppua (kuva 1). PHP-koodia voidaan upottaa haluttaessa keskelle HTML-tiedostoa tunnisteiden avulla. (Carr. & Gray 2018, 1.)

2.2 SASS, SCSS

SASS (*Syntically Awesome Style Sheets*) on CSS-tyylikielen esiprosessori, joka on yhteensopiva kaikkien CSS-versioiden kanssa. SASS tukee erilaisia kielten laajennuksia, ja se sisältää CSS-syntaksista puuttuvia ominaisuuksia, kuten muuttujat ja sisäkkäiset säännöt. Se myös auttaa vähentämään koodin kokonaispituutta hylkäämällä toistuvan CSS-koodin. (GeeksforGeeks 2022.)

SASS:n muuttujien avulla voidaan tallentaa tietoa, kuten värikoodeja. SASS tekee koodin kirjoittamisesta yksinkertaisempaa ja nopeampaa kuin tavallinen CSS. SASS:lla voidaan luoda osittaisia SASS-tiedostoja, jotka sisältävät pieniä osuuksia CSS-koodin kokonaisuudesta. Osittaiset SASS-tiedostot mahdollistavat helpomman koodin luettavuuden ja ylläpitämisen. (GeeksforGeeks 2022.)

Kuva 2. Esimerkki SASS-koodin ja CSS-koodin syntakseista

Kuvassa 2 vasemmalla on esimerkki SASS-koodin syntaksista ja oikealla puolella esimerkki CSS-koodin syntaksista. SCSS on SASS:n pääsyntaksi, jonka vuoksi vasemman puoleisen tiedoston tiedostopääte on .scss, joka tarkoittaa SASS-koodia.

2.3 Vue.js

Vue on JavaScriptin kehikko (engl. *framework*) ja ekosysteemi käyttöliittymien rakentamiseen. Se perustuu HTML-, CSS- ja JavaScript-standardien päälle

sekä tarjoaa deklaratiivisen ja komponenttipohjaisen ohjelmointimallin. Se kattaa useimmat yhteiset ominaisuudet, joita tarvitaan selainpuolen kehityksessä. Vue on suunniteltu joustavaksi ja omaksuttavaksi käyttötavan mukaan. Erilaisia käyttötapoja ovat mm. verkkokomponenttien upottaminen mille tahansa sivulle ja yksisivuisten sovellusten tekeminen. (Winter 2022.)

Tietojen sidonta antaa Vuelle mahdollisuuden päivittää dynaamisesti HTML-elementtejä, jotka ovat sidonnaisia taustalla oleviin Vue-objekteihin. Tämä mahdollistaa käyttäjän selaimessa toimivien verkkosovellusten luomisen, sekä tarjoaa interaktiivisen kokemuksen, joka ei vaadi sivun päivittämistä. (Winter 2022.)

Vuea voidaan kirjoittaa myös HTML-koodin seassa *script*-tunnisteiden sisässä. Usein rakennus (engl. *build*) työkaluja tukevissa Vue-projekteissa voidaan luoda komponentteja käyttämällä HTML-tiedostomuotoa. Näitä tiedostomuotoja kutsutaan nimellä Single-File-Component (SFC), kuten nimi kertoo, tällainen tiedostomuoto kapseloi komponentin logiikan (JavaScript), mallin (HTML) ja tyylit (CSS) yhteen tiedostoon. (Vue.js 2023.)

2.4 Responsiivisuus

Jokaisen verkkosivun toimimisen ja helppokäyttöisyyden kannalta responsiivisuus tulisi olla kunnossa ja testattuna. Responsiivisuudella tarkoitetaan sitä, että verkkosivusto suunnitellaan erilaisten käyttäjien päälaitteiden mukaisesti. Yleisesti verkkosivut tehdään aina ensimmäiseksi työpöytäkäyttöön sopiviksi, mutta koodin avulla saman verkkosivun saa skaalautumaan- vaikka mobiililaitteille.

Sivuston responsiivisuus vaikuttaa voimakkaasti hakukonenäkyvyyteen. Useat hakukoneet huonontavat sivuston näkyvyyttä hakutuloksissa, jos sivusto ei ole responsiivinen. Hakukonenäkyvyyden lisäksi responsiivisuudella parannetaan myös käyttäjäkokemusta. (Suovesi 2022.)

Responsiivisuuden toteuttamiseen löytyy useita erilaisia ohjelmistokehyksiä (engl. *framework*), joista käytetyin on Bootstrap. Groteski käyttää suurimmaksi

osaksi verkkosivuissaan CSS Grid Layoutia, joka tarkoittaa CSS:n omaa kaksiulotteista ruudukkomaista rakennetta, joka kirjoitetaan suoraan CSS-koodiin. Opinnäytetyössäni käytetään myös CSS Grid Layoutia.

2.5 Figma

Figma on suunnitteluohjelma, jolla voidaan suunnitella verkkosivustoja sekä erilaisia käyttöliittymiä esimerkiksi mobiilisovelluksille. Figma on siitä loistava, että koodin kirjoittamista ei tarvitse, vaan voidaan keskittyä ainoastaan siihen, miltä sivun rakenne ja ulkonäkö tulee näyttämään. Figmalla voidaan luoda verkkosivustosta havainnollistava prototyyppi, joka voidaan esitellä asiakkaalle ilman, että verkkosivua on edes aloitettu koodaamaan. (Webguru 2021.)

Figman perusversio on täysin ilmainen ja se toimii suoraan selaimessa, halutessaan Figmasta löytyy myös työpöytäversio. Sovelluksessa piirtäminen tapahtuu näkymässä, johon voidaan raahata valmiita ui-komponentteja. Halutessaan komponentteja voidaan luoda itse. Figma-projekti voidaan jakaa tiimin kanssa, mikä mahdollistaa saman suunnitteluprojektin monipuoliset yhteistyötoiminnot. Suunnitteluprojekti voidaan jakaa myös asiakkaalle ja hän voi tarvittaessa lisätä kommentteja projektiin. Samankaltaisia piirto-ohjelmia ovat Adoben Photoshop, Illustrator ja XD. (Webguru 2021.)

3 SISÄLLÖNHALLINTAJÄRJESTELMÄT

Sisällönhallintajärjestelmä (engl. *content management system* CMS) on ohjelmistosovellus, jonka avulla käyttäjät voivat luoda, muokata, tallentaa sekä julkaista digitaalista sisältöä. Sisällönhallintajärjestelmässä on CMS-työkaluja, jotka ovat pääasiassa online-työkaluja, joiden avulla voidaan hallita, ylläpitää ja päivittää verkkosivuja. Sisällönhallintajärjestelmä tarjoaa graafisen käyttöliittymän työkaluilla verkkosisällön luomiseen, muokkaamiseen ja julkaisemiseen ilman, että tarvitsee kirjoittaa koodia tyhjästä. (Amsler & Churchville 2021.)

3.1 Sisällönhallintajärjestelmän ominaisuudet

Sisällönhallintajärjestelmien ydintoimintoja ovat intuitiivinen indeksointi (engl. *Intuitive indexing),* formaattien hallinta (engl. *Format management*), version

ominaisuudet (engl. *Revision features*) ja julkaiseminen (engl. *Publishing*). Nämä ydintoiminnot löytyvät kaikista sisällönhallintajärjestelmistä ja niiden lisäksi löytyy myös sisällönhallintajärjestelmä kohtaisia ominaisuuksia. Ydinominaisuuksilla voidaan helpottaa mm. hakutoimintojen käyttöä, muuttaa skannatut paperiasiakirjat ja vanhat sähköiset asiakirjat HTML- tai PDF-dokumenteiksi, muokata ja päivittää sisältöä ensimmäisen julkaisun jälkeen sekä seurata tiedostoihin tehtyjä muokkauksia. (Amsler & Churchville 2021.)

Sisällönhallintajärjestelmän käyttämisessä on useita erilaisia etuja, joita ovat helppokäyttöisyys, helppo tiedon etsiminen, saatavuus mistä tahansa, useiden käyttäjien salliminen, välittömät sisältöpäivitykset, helppo skaalautuvuus sekä helppo päivittäminen. (Amsler & Churchville 2021.)

3.2 Sisällönhallintajärjestelmän valinta

Sisällönhallintajärjestelmän valinnassa tulee ottaa monia asioita huomioon, esimerkiksi helppokäyttöinen editorin käyttöliittymä ja älykkäät hakutoiminnot. Kuitenkin joidenkin organisaatioiden kohdalla nämä perusominaisuudet eivät ole riittävät ja yleensä he panostavat enemmän tarkempiin ja räätälöityihin ominaisuuksiin ja vaatimuksiin. Sisällönhallintajärjestelmän valintaan vaikuttaa organisaation koko ja maantieteellinen hajautus. Sisällönhallintajärjestelmää valitessa tulee tietää, kuinka monta henkilöä tulee käyttämään sovellusta, tukeeko se monikielisyyttä ja minkä kokoista tukitiimiä tarvitaan sen ylläpitämiseen. (Amsler & Churchville 2021.)

Täytyy arvioida liiketoiminnan tarpeet, onko sisällönhallintajärjestelmässä organisaation nykyisiä tarpeita ja mahdollisia liiketoimintasuunnitelmia mahdollistavat ominaisuudet. Löytyykö sisällönhallintajärjestelmästä tulevaisuuteen suuntautuvia ominaisuuksia, kuten RESTful API. RESTful API tarkoittaa käyttöliittymää, jota kaksi tietokonejärjestelmää käyttää tiedon vaihtamiseen turvallisesti internetissä. Markkinoilta löytyy monia ilmaisia ja tilauspohjaisia sisällönhallintajärjestelmiä henkilökohtaiseen tai yrityskäyttöön. Niistä suosituimmat ovat mm. Joomla, WordPress, Drupal ja Wix, jotka kaikki ovat ilmaisia avoimeen lähdekoodiin perustuvia järjestelmiä. (Amsler & Churchville 2021.)

Groteskilla käytetään WordPressiä sisällönhallintajärjestelmänä, sillä se vastaa yrityksen ja liiketoimintasuunnitelman tarpeita. Opinnäytetyöni pohjaksi on valittu WordPress, sillä sain työharjoittelujaksoni aikana kokemusta sen käytöstä ja se tuli tietyissä määrin tutuksi.

3.3 WordPress-sisällönhallintajärjestelmänä

WordPress on avoimen lähdekoodin kaikille suunniteltu ilmainen ohjelmisto, jolla käyttäjien on helppo luoda verkkosivustoja erilaisiin käyttötarkoituksiin. Se sai alkunsa vuonna 2003, kun Mike Little ja Matt Mullenweg kehittivät hyvin rakennetun ja visuaalisesti tyylikkään henkilökohtaisen julkaisujärjestelmän. Nykyään WordPress on rakennettu PHP:lle, MySQL:lle ja lisensoitu GPLv2:lla ja sillä on tehty 43 % kaikista verkkosivuista verkossa. Avoimen lähdekoodin WordPress-projekti on kehittynyt ammattitaitoisten, innostuneiden kehittäjien, suunnittelijoiden, tutkijoiden, bloggaajien ja muiden tukemana. WordPress on täysin muokattavissa omien tarpeiden mukaan. (WordPress 2019.)

3.3.1 Versiot

WordPressistä löytyy kaksi erillistä versiota, toinen niistä on WordPress.com, joka on WordPressin omalla palvelimella käytettävä versio, ja toinen on WordPress.org, joka on omalle palvelimelle ladattava versio. (Hakukonemestarit 2020.)

WordPress.com on alkeellisien verkkosivujen tai blogien luontia varten. Se on täysin ilmainen, eikä päivityksiä tarvitse asentaa itse. Siinä on kuitenkin rajoitteita verrattuna WordPress.org versioon. Luotujen verkkosivujen osoite on aina muotoa omasivu.wordpress.com ja siihen liittyy tiettyjä ehtoja, kuten pakollinen rekisteröinti ja käyttöehtojen hyväksyminen. WordPress.comin perusversio on todella suppea, sillä siihen ei saa asennettua lisäosia ilman lisäkustannuksia. WordPress.com toki tarjoaa erikseen ostettavia paketteja, joilla saadaan itse ladata lisäosia ja teemoja, mutta ilman maksullista versiota, verkkosivustasi ei saa omalaatuista sillä, suurin osa ilmaisversion teemoista onkin loppuun kulutettuja ja paljon käytettyjä unohtamatta *Powered by WordPress* tekstiä. Vaikka käyttäjältä löytyisikin osaamista koodaamisesta, PHP-koodia ei pysty muokkaamaan ilmaisessa versiossa. Täytyy muistaa, jos verkkosivusi

tehdään WordPress.comilla, et omista blogiasi etkä sen sisältöjä. (Hakukonemestarit 2020.)

Suurin ero versioiden välillä löytyy WordPress.orgista, jonka sivuilta voidaan ladata uusin versio ohjelmistosta omalle palvelimelle. Näin omistat sivusi ja kaiken sen sisällön, lisäksi saat täyden vapauden tehdä sivullesi mitä haluat. WordPress.org on ilmainen eikä vaadi rekisteröintiä tai käyttöehtojen hyväksymistä. Tämä mahdollistaa täyden vapauden lisäosien asentamiseen ja koodin muokkaamiseen. Ainoa niin kutsuttu haittapuoli WordPress.orgissa on se, että ohjelmiston ja päivitysten asentaminen, varmuuskopiointi sekä palvelintilan ostaminen pitää tehdä itse. (Hakukonemestarit 2020.)

WordPress julkaistiin vuonna 2003 ja siitä lähtien se on saanut useita uusia versioita ja ominaisuuksia. Sitä on päivitetty lähes jokaisena vuotena julkaisunsa jälkeen. Käytän työssäni omalle palvelimelle ladattavaa WordPressin uusinta versiota, joka on versio 6.1.

3.3.2 Lisäosat ja teemat

WordPressin teemat ovat valmiiksi rakennettuja verkkosivupohjia, joita voidaan muokata vastaamaan käyttäjän omia toiveita. Teemoilla vaikutetaan verkkosivuston visuaaliseen ilmeeseen. WordPressiin saa hankittua useita ilmaisia teemoja, mutta parempia ominaisuuksia haluavalle on olemassa maksullisia teemoja. Lähes jokainen teema pitää sisällään valmiiksi suunniteltuja sivupohjia, joihin käyttäjän tarvitsee vain lisätä omat kuvat ja sisältötekstit. (WP-teemat 2023.)

Teeman tärkeimmät ominaisuudet ovat helppokäyttöisyys, monipuolisuus, luotettavuus ja nopeus. WP-teemat (2023) on listannut muutamia kysymyksiä, joita kannattaa harkita teemaa valittaessa.

- Minkälaista sivustoa suunnitellaan?
- Onko teema responsiivinen?
- Mitä selaimia teeman tulee tukea?
- Mitä toiminnallisuuksia teema sisältää?
- Tukeeko teema monikielistä sivustoa?
- Tuleeko teeman mukana Page Builder?
- Tarvitaanko apua tai ylläpitotukea teeman kanssa?

- Onko teeman suunnittelussa huomioitu hakukoneoptimointi?
- Minkälaisia arvosteluja teema on saanut?

Teemoja voidaan ladata ilmaiseksi tai ostaa suoraan netistä. Ne on helppo asentaa ja ottaa käyttöön. Hankittuasi uuden teeman, saat pakatun .zip-tiedoston, joka ladataan WordPressin hallintapaneelista kohdasta *Ulkoasu* > *Teemat* > *Lisää uusi teema*. (WP-teemat 2023.)

WordPress-lisäosat ovat suunniteltu helpottamaan sivustojen erilaisien toiminnallisuuksien toteuttamista ja mahdollistamaan monenlaisien ominaisuuksien lisäämisen ilman koodaustaitoja. Markkinoilta löytyy lähes 60 000 erilaista ilmaista lisäosaa, joiden lisäksi on olemassa maksullisia lisäosia, joten tarvittavien ja parhaiden lisäosien löytäminen voi olla haastavaa. (Zoner 2021.)

Lisäosien avulla voidaan toteuttaa toimintoja, joita WordPressistä ei löydy sisäänrakennettuna, kuten kuvien optimointi, sivuston nopeuden parantaminen ja monikielisen version luominen. Ennen lisäosan asentamista tulee huomioida, onko lisäosan asentaminen luotettavaa ja onko se yhteensopiva oman verkkosivuston kanssa. Lisäosia voidaan asentaa, päivittää ja käyttöönottaa WordPressin hallintapaneelin kohdasta *Lisäosat* > *Lisää uusi*. (Zoner 2021.)

Usein lisäosista on saatavilla ilmainen sekä maksullinen versio. Maksullinen versio sisältää laajemmat toiminnallisuudet, mutta se ei aina välttämättä ole korkean hinnan arvoinen. Usein kannattaa käyttää tarpeeksi aikaa lisäosien tutkimiseen ja siihen, tarvitaanko välttämättä maksullista versiota. Lisäosien valinnassa kannattaa olla tarkkana, sillä ne saattavat vaikuttaa negatiivisesti koko sivuston toimintaan ja nopeuteen. Lähtökohtaisesti olisi hyvä asentaa vain yksi lisäosa yhtä tarkoitusta varten, näin lisäosia ei kerry sivustolle liikaa. Zoner (2021) on listannut muutamia tärkeitä huomioita, joita kannattaa tarkistaa ennen lisäosan asentamista.

- lisäosan säännöllisten päivitysten saatavuus
- yhteensopivuus oman WordPress-version ja muiden lisäosien kanssa
- lisäosan aktiivisten latauksien määrä
- lisäosan arvostelut.

Kaikki asennetut WordPress-lisäosat tulisi olla käytössä ja päivitetty viimeisimpään versioon, jos lisäosaa ei enää tarvitse, se kannattaa poistaa kokonaan. Näin saadaan sivusto toimimaan mahdollisimman nopeasti, koska turhat käyttämättömät lisäosat saattavat pahimmassa tapauksessa hidastaa sivuston toimintaa. Päivittämättömät lisäosat ovat aina riski sivuston tietoturvan kannalta. (Zoner 2021.)

Työssä käytän toimeksiantajan omaa kustomoitua WordPress-teemaa, joka sisältää muutamia toimeksiantajan määrittelemiä lisäosia teeman kanssa käytettäväksi. Teeman rakenteesta ja käyttämistäni lisäosista kerrotaan lisää luvussa 4.2 *Intranetin rakenne ja lohkon luominen.*

4 INTRANETIN TOTEUTUS

Toteutan opinnäytetyöni Mainostoimisto Groteski Oy:lle. Groteski on vuonna 2012 perustettu mikkeliläinen mainostoimisto ja se työllistää 13 alan ammattilaista. Groteski on jo 10 vuoden ajan toteuttanut asiakkaille graafista suunnittelua ja brändejä sekä räätälöityjä asiakaslähtöisiä verkkopalveluratkaisuja. Yritys on jaettu kahteen tiimiin: visuaalisen suunnittelun sekä digitaalisten ratkaisujen tiimiin.

Groteski erikoistuu räätälöityihin WordPress-verkkopalveluihin, WooCommerce-verkkokauppoihin, integraatioihin sekä markkinointiin digitaalisessa maailmassa. Groteskin liikevaihto on kasvanut viimeisten vuosien aikana reilusti, noin 30 prosenttia vuodessa ja digitaalisten ratkaisujen liiketoiminta noin 50–70 prosenttia vuodessa. Groteski käyttää WordPressiä toteutukseen, Microsoft Visual Studiota koodiin editoimiseen, WinSCP:tä SFTP-sovelluksena ja Figmaa verkkosivustojen visuaaliseen suunnitteluun.

Sain toimeksiantajalta työksi toteuttaa yritykselle intranetin sisäisen viestinnän käyttöön. Kävin ennen työn aloittamista pienimuotoisen kyselyn Groteskin henkilöstön kanssa siitä, minkälaisille ominaisuuksille intranetissä on tarvetta. Kysely tuotti tulosta ja sain todella selvän kuvan, mitkä ovat ne tärkeimmät ensimmäisen version ominaisuudet. Intranet oli tarkoitus tehdä helposti jatkokehitettäväksi ja tulevaisuudessa ehkä valmiiksi pohjaksi, jota voidaan myydä

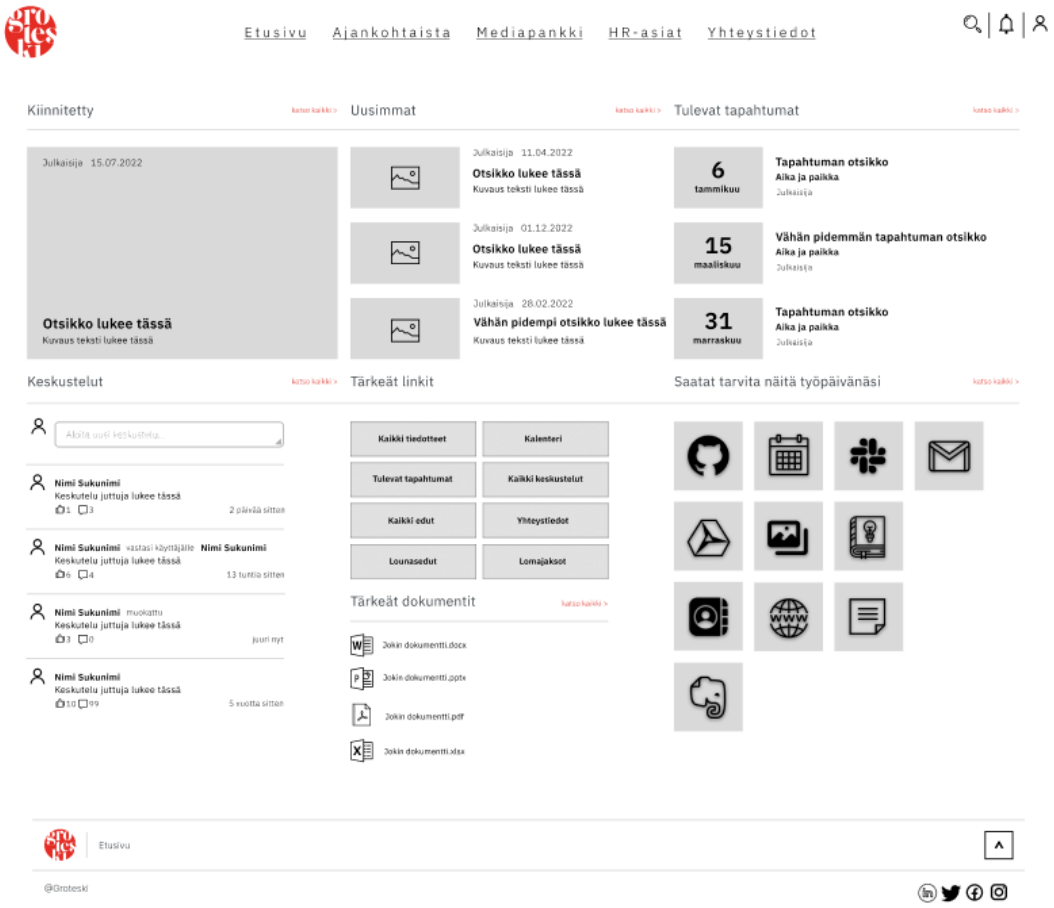
asiakkaille. Työssä sivuutan intranetin jatkokehityksen, koska opinnäytetyöni käsittelee vain intranetin ensimmäisen version ulossaantia.

4.1 Rautalanka- ja ulkoasumalli

Suunnitteluvaiheen tarkoituksena on tehdä rautalankamalli verkkosivustosta haluttujen ominaisuuksien osalta. Harjoittelussa pääsin käyttämään Figmaa ja koin sen todella käteväksi käyttöliittymäsuunnittelun sovellukseksi. Intranetin rautalanka- ja ulkoasumalli on suunniteltu Figmalla. Groteskilla ennen käytettiin käyttöliittymäsuunnitteluun Adobe XD:tä, mutta Figman nopea kasvu ja laajempien ominaisuuksien takia Groteski on vaihtanut käytettäväksi Figmaa yrityksessä.

Ennen ensimmäisen rautalankamallin suunnittelua- etsin netistä useiden muiden yritysten intranetin ulkoasuja ja keräsin tietoa intranetiin tarvittavista ominaisuuksista. Sovimme toimeksiantajan kanssa, että suunnittelen ensimmäisen version intranetin etusivusta ilman, että Groteski ilmaisee oman näkemyksensä intranetin rakenteesta ja ominaisuuksista. Palaverissa näytin rautalankamallin toimeksiantajalle ja sain palautetta sekä kehitysehdotuksia intranetiä varten. Ensimmäinen versio on tärkeä, että saadaan molempien osapuolien ideat esille ja voidaan yhdessä kehittää verkkosivuston rautalankamallia.

Tein ensimmäisen version jälkeen vielä muutaman erilaisen version etusivun rautalankamallista, jotka esittelin toimeksiantajalle. Näissä otin huomioon toimeksiantajan palautteen sekä esille tulleet kehitysehdotukset. Kuvassa 3 on esitelty toimeksiantajan valitsema etusivun rautalankamalli. Rautalankamallista löytyy ajankohtaiset ja uusimmat tiedotteet sekä tulevat tapahtumat, keskustelualue, tärkeät linkit, ladattavat dokumentit ja itselle kustomoitava sovellukset työpäivää varten osio. Sivulla on myös ylä- ja alatunnisteet eli navigaatio sekä footer.



Kuva 3. Toimeksiantajan valitsema intranetin rautalankamalli

Kun toimeksiantaja hyväksyi rautalankamallin, aloin suunnittelemaan rautalankamallin pohjalta ulkoasumallia. Tämä tarkoittaa sitä, että rautalankamallissa olevat elementit muutetaan tyyliteltyyn versioon. Yleensä organisaatiolta löytyy omat teemavärit, joiden pohjalta ulkoasumalli suunnitellaan. Jos organisaatiolla ei ole teemaa, voidaan valita sopiva väripaletti ulkoasumallia varten. Tässä tapauksessa, kun intranet suunniteltiin Groteskille, käytin heidän omaa väriteemaansa.

Kuten rautalankamallissa, suunnittelin muutamia erilaisia versioita intranetin etusivun ulkoasumallista. Tarkoituksena oli suunnitella hillitty, mutta toimeksiantajan väriteemaa omaksuva intranet. Groteskin teeman punaista ja tummansinistä on käytetty korostamaan elementtejä etusivulla. Etusivulta löytyy myös leijumis- (engl. *hover*) efektejä, kun käyttäjä vie hiiren kohdistimen nappien ja linkkien päälle. Etusivun pohjavärinä on käytetty vaalean sinistä, jotta etusivulla voidaan korostaa elementtien taustaväriä valkeaa väriä hyödyntäen. Liitteessä 1 on kuvattu toimeksiantajan valitsema etusivun ulkoasumalli.

Suunnittelin myös rautalanka- ja ulkoasumalleja intranetin muista sivuista. Muita sivuja ovat ajankohtaistasivu, mediapankki, hr-asiat ja yhteystiedot. Työn kannalta ei ole tarpeellista käydä läpi jokaisen sivun rautalanka- sekä ulkoasumalleja. Seuraavissa luvuissa käymme läpi teknisen toteutuksen vaiheita sekä muutamme suunnitellut rautalanka- ja ulkoasumallit toimivaksi intranetiksi.

4.2 Intranetin rakenne ja lohkon luominen

Työn toteuttamista varten asennetaan WordPress paikallisesti omalle koneelle. Käytin tässä vaiheessa *Local*-nimistä työkalua, joka on ilmaiseksi ladattavissa Local WP:n omilta verkkosivuilta. Työkalun avulla voidaan perustaa useita lokaaleja WordPress-sivustoja omalle koneelle. Oman lokaalin WordPress-sivuston käyttöönottamiseen löytyy ohjeet *Local*-työkalun omilta sivuilta.

Projektia varten luotiin *Github*-palveluun arkisto (engl. *repository*). Github on versionhallintatyökalu, jonka avulla on helppo palata projektin aiempaan versioon, jos nykyisessä versiossa on ilmennyt ongelmia. Github on myös loistava paikka koodin tallennuspaikkana ja tiimissä työskentelyä varten. Tein *Github*-arkiston sitä varten, koska teen vain ensimmäisen version intranetistä ja sen jatkokehittäminen on silloin helppoa, kun toimeksiantajalla on nykyiset koodit tallessa Githubissa. Intranetin kanssa työskenteli myös toimeksiantajan yksi web-kehittäjä, joka auttoi minua pitämään koodin toimeksiantajan antamien ohjeiden mukaisessa muodossa.

Toimeksiantajalla on oma WordPress-teema, jonka pohjalta tehdään verkkosivuja yrityksessä. Teema pitää sisällään valmiiksi koodattuja lohkoja, jotka mahdollistavat nopean asiakasprojektien aloittamisen. Käytän itse Groteskin omaa teemaa intranetin pohjana. Teeman käyttöönoton jälkeen on hyvä ladata tarvittavat lisäosat projektiin, jotkin näistä tulevat jo valmiiksi teeman kanssa. Alla on lueteltuna lista käyttämistäni lisäosista opinnäytetyöni toteuttamisessa.

- Advanced Custom Fields PRO
- All-in-One WP Migration
- Broken Link Checker

- Complianz | GDPR/CCPA Cookie Consent
- Custom Post Type UI
- Easy WP SMTP
- FileBird Lite
- Gravity Forms
- Gravity Forms Polls
- Imagify
- ManageWP – Worker
- miniOrange SSO using SAML 2.0
- Post Types Order
- Simple History
- Simple Local Avatars
- Sticky Posts – Switch
- UpdraftPlus - Backup/Restore
- W3 Total Cache
- Yoast Duplicate Post
- Yoast SEO
- Zapier for WordPress

Groteskin teema pitää sisällään useita erilaisia tiedostoja, mutta en näe olennaiseksi käydä läpi jokaista teeman tiedostoa, vaan keskityn itse luomiini tiedostoihin intranetiä rakentaessa. Loin teemaan tiedoston nimeltä front-page.php, joka toimii intranetin etusivun muodostuksessa. Tiedostoon haetaan WordPressin omalla *get*-funktiolla header.php- ja footer.php-tiedostot. Nämä ovat nimensä mukaisesti sivuston ylä- ja alatunnisteet. Header.php sisältää sivuston html, head ja meta-tag:it sisältöineen. Footer.php sisältää sivun alaosaan tarvittavat elementit, esimerkiksi sosiaalisen median napit ja copyright-logon. Näiden lisäksi teemaan haetaan etusivulla tarvittavat lohkot, joihin palataan myöhemmin tässä luvussa.

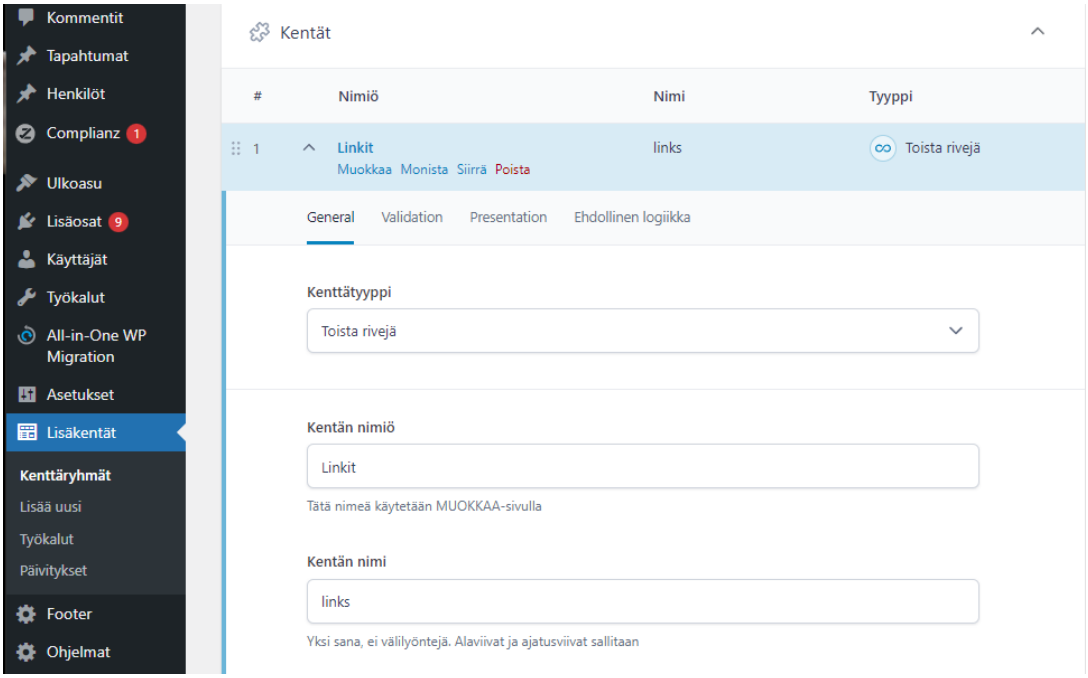
Teeman kansiorakenne on ositettuna omiin kansioihinsa, mikä tarkoittaa, että PHP-, SCSS- ja JavaScript-tiedostot ovat omissa kansioissaan. Groteskin teemassa on esimerkiksi oma kansio funktioille, lohkoille, lohkojen tyylitiedostoille ja tarvittaville muuttujilla, joita ovat mm. värit sekä fontit. Groteski käyttää *Gulp*-nimistä työkalua, joka kokoaa ositetut SCSS- ja JavaScript-tiedostot minimoituun muotoon. Minimoitujen tiedostojen nimet ovat työssä site.min.css ja main.min.js. Gulp poistaa koodista ylimääräiset merkit mm. välilyönnit sekä rivinvaihdot ja pakkaa koodin yhdeksi globaaliksi tiedostoksi. Tämä mahdollistaa pienemmän tiedostokoon, joka nopeuttaa verkkosivuston toimintaa.

Intranet sisältää viisi pääsivua, näiden lisäksi hr-asiat pääsivun alla on muutamia alasivuja. Tämän sivun nimi vaihdettiin myöhemmin työntekijälle-sivuksi, jotta se olisi selkeämpi. Sivut luodaan WordPressin hallintapaneelista kohdasta *sivut* > *lisää uusi*. Kun uusi sivu on luotu, se ei sisällä vielä minkäänlaista sisältöä, tähän tarkoitukseen tarvitaan lohkoja.

Kuten mainitsin, Groteskin teemasta löytyy valmiiksi luotuja lohkoja. Intranetti kuitenkin sisältää ominaisuuksia, joita ei voida toteuttaa valmiiksi olevilla lohkoilla. Tätä varten luodaan omia kustomoituja lohkoja. Kaikki työtä varten luotujen lohkojen ja Groteskin teeman valmiiden lohkojen PHP-tiedostot löytyvät teeman hakemiston *blocks*-nimisestä kansiosta.

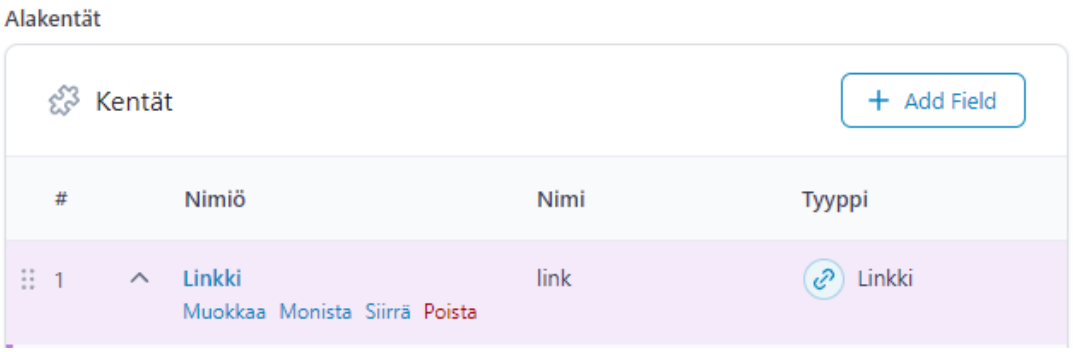
Kun teemaan luodaan uusia lohkoja, niille tehdään syöttötietojen kentät WordPressin hallintapaneelissa. Tämä tapahtuu WordPressin hallintapaneelista kohdassa *lisäkentät* > *kenttäryhmät* > *lisää uusi*. Kun kenttäryhmä on luotu, sille annettuja arvoja voidaan poimia koodissa ja näin voidaan tulostaa syöttötiedot HTML-rakenteeseen WordPress-verkkosivulle. Kenttäryhmien selkeästi nimeäminen on todella tärkeää, jotta haluttua kenttäryhmää on helppo kutsua koodieditorissa.

Seuraavaksi käydään läpi, kuinka oman kustomoidun lohkon voi lisätä teemaan. Käytän esimerkissä lohkoa nimeltä *Linkit*, jota käytetään intranetin etusivulla. Koska kyseessä on lohko, joka sisältää linkkejä, sen kenttäryhmän otsikoksi annetaan sitä kuvaava nimi, joka on tässä tapauksessa ”Linkit”. Kenttäryhmälle luodaan uusi kenttä, jonka nimiöksi annetaan nimi ”Linkit”, tämä nimi tulee näkyville WordPressin hallintapaneeliin. Kentälle annetaan myös toinen nimi, jonka avulla kyseinen kenttä voidaan poimia koodieditorissa. Koodissa poimittava nimi ei saa sisältää välilyöntejä, koska koodieditori ei välttämättä tunnista sitä muuttujaksi. Kuvassa 4 *Kentän nimi* kohdalle annettu arvo on koodissa poimittava arvo, joka on ”links”.



Kuva 4. Kenttäryhmän esimerkki

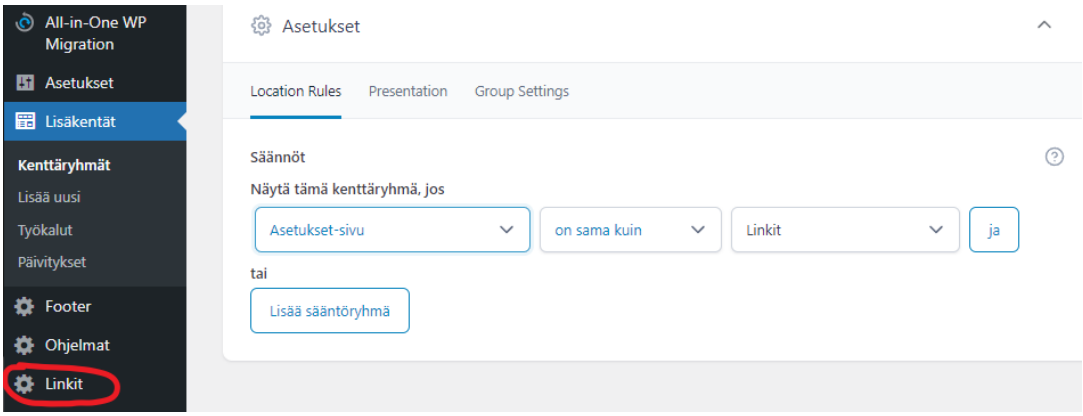
Kentälle annetaan kenttätyyppi, joka tarkoittaa kentän toiminnallisuutta. Kun kenttäryhmän tyypiksi on valittu ”toista rivejä”, sille täytyy luoda alakenttä, joka tarkoittaa yksittäistä elementtiä kenttäryhmässä. Alakenttiä voidaan luoda niin monta kuin on tarvetta, mutta tässä tapauksessa tarvitaan vain yksi alakenttä nimeltä ”Linkki” (kuva 5).



Kuva 5. Esimerkki alankentistä

Alakentät nimetään aina yksikössä, koska ne ovat kenttäryhmän yksittäinen elementti, joten koodieditorissa käytettävä nimi on ”link”. Alakentille annetaan kenttätyyppi tarvittavan ominaisuuden mukaan. Kuten aikaisemmin mainitsin, esimerkissä teemme lohkoa, joka sisältää linkkejä, annamme sen tyypiksi ”Linkki”. Tyyppejä löytyy moneen tarkoitukseen, jos halutaan luoda lohko, joka sisältää dokumentteja, sen tyypiksi annetaan ”Tiedosto”.

Viimeinen asia kenttäryhmän luomisessa on sen asetukset. Asetuksilla luodaan kenttäryhmälle sääntö, jolla määritetään, missä näkymässä kenttäryhmä näytetään hallintapaneelissa. Groteskin teeman oletuslohkot käyttävät sääntöä ”Lohko”, mutta tässä tapauksessa sääntö on ”Asetukset-sivu”, jolloin luotu kenttäryhmä on näkyvillä suoraan hallintapaneelin valikossa (kuva 6).



Kuva 6. Esimerkki lohkon säännöstä

Groteskin teeman oletuslohkoja kutsutaan functions.php-tiedostossa acf_register_block-funktiolla, jonka avulla määritetään lohkon nimi ja sivupohja, jossa lohkoa halutaan käyttää. Edellä mainitulla rekisteröintitavalla saadaan lohko näkyville WordPressin muokkausnäkymään, jossa sisällönsyöttäminen tapahtuu. Tätä tapaa käytetään toistuvissa lohkoissa, joita näytetään usealla sivulla ja niiden sisältö muuttuu sivukohtaisesti. Kun luodaan lohko, jota käytetään vain yhdellä sivulla, kuten esimerkkinä oleva lohko nimeltä *Linkit*, käytetään samassa tiedostossa olevaa acf_add_options_page-funktiota.

```php
$block_name  = 'Kaksi saraketta';
$block_slug  = 'two-column-block';
$description = 'Lohko kahdella sarakkeella';

acf_register_block_type(
    array(
        'name'            => $block_name,
        'title'           => $block_name,
        'description'     => $description,
        'render_template' => "blocks/$block_slug.php",
        'keywords'        => array( $block_name ),
    )
);
```

```php
acf_add_options_page(
    array(
        'page_title' => 'Linkit',
        'menu_title' => 'Linkit',
        'menu_slug'  => 'links-settings',
        'redirect'   => false,
    )
);
```

Kuva 7. Acf_register_block- ja acf_add_options_page-funktioiden rakenne

Tämä funktio eroaa normaalisti käytettävästä acf_register_block-funktiosta siten, että luotu lohko saadaan näkyville WordPressin hallintapaneelin valikkoon

ja näin sisällön syöttäminen tapahtuu suoraan valikon ”Linkit” asetuksen kohdalta. Kuvassa 7 vasemmalla puolella on Groteskin teeman oletuslohko nimeltä *Kaksi saraketta,* joka käyttää perinteistä acf_register_block-funktiota. Oikealla on lohkossa nimeltä *Linkit* käytettävä acf_add_options_page-funktio.

Jokaisella lohkolla on oma vastaavalla tavalla nimetty PHP-tiedostonsa, joka löytyy aiemmin esitellystä *blocks*-nimisestä kansiosta. PHP-tiedostossa otetaan ACF-kentistä, eli WordPressin hallintapaneelissa olevista kenttäryhmistä syöttötiedot muuttujille, minkä jälkeen luodaan tarvittava HTML-rakenne, johon muuttujien sisällöt tulostetaan. Kun kenttäryhmässä on käytetty tyyppiä ”toista rivejä”, käydään koodissa kenttäryhmä ”Linkit” läpi *foreach*-silmukalla, jolla saadaan tulostettua yksittäiselle elementille muuttujien sisällöt. Kuvassa 8 on *Linkki*-lohkon PHP-tiedosto nimeltä block-links.php, joka sisältää lohkon HTML-rakenteen sekä PHP-koodin.

```php
<?php $links = get_field( 'links', 'option' ); ?>

<section class="block-links">
    <header class="block-header">
        <h2 class="h3 section-title">
            Tärkeät linkit
        </h2>
    </header>

    <div class="block-content">
        <?php if ( $links ) : ?>
            <div class="links">
                <?php foreach ( $links as $link ) : ?>
                    <?php
                    $href = esc_url( $link['link']['url'] );
                    $title = esc_attr( $link['link']['title'] );
                    $target = esc_attr( $link['link']['target'] );
                    ?>
                    <a class="link" href="<?php echo $href; ?>" title="<?php echo $title; ?>" target="<?php echo $target; ?>">
                        <?php echo $title; ?>
                    </a>
                <?php endforeach; ?>
            </div>
        <?php endif; ?>
    </div>
</section>
```

Kuva 8. Block-links.php-tiedoston rakenne

Block-links.php-tiedostolle luodaan samanlailla nimetty tyylitiedosto nimeltä block-links.scss, joka sisältää ainoastaan *Linkki*-lohkon ulkoasumäärittelyt (kuva 9). Koska intranet on tulossa toimeksiantajan käyttöön, otin mallia aiemmin suunnitellusta ja toimeksiantajan hyväksymästä ulkoasumallista tyylien koodaamisessa. Lohko jaetaan kahteen täysin samankokoiseen kolumniin antamalla *links*-nimiselle luokalle *grid-template-columns*-ominaisuus, johon määritetään arvoksi kolumnien määrä ja koko. Tällä tarkoitetaan sitä, että lohkoon lisätyt linkit ovat vierekkäin kahdessa kolumnissa.

```scss
.links {
    display: grid;
    grid-template-columns: minmax(0, 1fr) minmax(0, 1fr);
    gap: 15px;

    .link {
        background-color: $darkblue;
        padding: 25px 15px;
        text-decoration: none;
        color: $white;
        text-align: left;
        font-weight: 700;
        text-transform: uppercase;
        letter-spacing: -0.5px;
        border-right: 10px solid $red;
        transition: all 0.25s;

        &:hover {
            border-right: 20px solid $darkblue;
            background-color: $red;
        }
```

Kuva 9. Block-links.scss-tiedoston rakenne

Loin kaikki intranetissä käytettävät värit globaaleiksi muuttujiksi teeman hakemistossa olevaan *variables*-kansion *colors.scss*-tiedostoon, joten jatkossa voidaan kirjoittaa värille annettu muuttuja, kun väriä halutaan käyttää tyylitiedostossa. Kuvassa 9 yksittäiselle linkille, eli *link*-nimiselle luokalle on annettu taustaväriksi globaali muuttuja ”$darkblue”, joka vastaa värikoodia ”#0f1127”. Yksittäiselle linkille annetaan leijumisefekti, jolla vaihdetaan linkin taustaväri muuttujaan ”$red”, kun käyttäjä vie hiiren kohdistimen linkin päälle. Lohkolle tehdään muut olennaiset ulkoasumäärittelyt, jotta se vastaa ulkoasumalliin suunniteltua lohkoa. Näitä ulkoasumäärittelyitä ovat mm. fontin paksuus (engl. *font-weight*), täyte (engl. *padding*) ja väri (engl. *color*). Kuvassa 10 on valmis tyylitelty yksittäinen linkki *Linkki*-lohkossa ilman leijumisefektiä ja sen kanssa.

Kuva 10. Tyylitelty linkki *Linkki*-lohkossa

Kaikkien intranetissä käyttämieni kustomoitujen lohkojen luominen ja rakentaminen tapahtuu käytännössä esimerkkinä toimineen *Linkki*-lohkon tavoin. Lohkoilta löytyy aina oma kenttäryhmä hallintapaneelissa, PHP-tiedosto ja SCSS-tiedosto. Jokaiselle sivulle on luotu omat kustomoidut lohkot, joten voimme kutsua valmista tekemäämme lohkoa koodissa get_template_part-funktiolla suoraan halutulle sivupohjalle. Kuvassa 11 *Linkki*-lohkoa kutsutaan front-page.php-tiedostossa muiden etusivun lohkojen kanssa.

```php
get_header();
?>

    <main id="primary" class="site-main">
        <div id="vue-app">
            <div class="grid-full main-grid">
                <?php get_template_part( 'blocks/block', 'stickied' ); ?>
                <?php get_template_part( 'blocks/block', 'news' ); ?>
                <?php get_template_part( 'blocks/block', 'upcoming-events' ); ?>
                <?php get_template_part( 'blocks/block', 'comments' ); ?>

                <div class="combined">
                    <?php get_template_part( 'blocks/block', 'links' ); ?>
                    <?php get_template_part( 'blocks/block', 'documents' ); ?>
                </div>

                <?php get_template_part( 'blocks/block', 'programs' ); ?>
            </div>
        </div>
    </main><!-- #main -->

<?php
get_footer();
```

Kuva 11. Front-page.php tiedoston rakenne

4.3 Teeman valmiin lohkon hyödyntäminen

Seuraavaksi hyödynnetään Groteskin teemassa olevia valmiita oletuslohkoja. Aiemmin mainitsin, että teemasta löytyvät lohkot eivät sellaisenaan vastaa intranetin tarpeita. Lohkot on koodattu PHP-koodilla, joten teemassa olevia lohkoja voidaan muokata intranetiin sopiviksi.

Loin intranetiin sivun nimeltä *Mediapankki*, tämän sivun tarkoituksena on kasata Groteskin graafisen ohjeiston lisäksi logoversiot, tietoa värien käytöstä, typografiasta ja muista olennaisista visuaalisista elementeistä. Koska sivu on itsessään todella yksinkertainen, käytetään sivun luomisessa teeman valmista lohkoa nimeltä *Teksti ja kuva*. Kaikki teemassa olevat lohkot ovat luotu esimerkkinä toimineen *Linkki*-lohkon tavoin, joten en kerro toistamiseen lohkojen luomisesta vaan keskityn lohkon hyödyntämistä varten tarvittaviin muokkauksiin tässä luvussa.

Teeman valmiilla lohkoilla on omat kenttäryhmänsä, *Teksti ja kuva*-lohkon kenttäryhmä sisältää syöttötiedot tekstille ja kuvalle. *Mediapankki*-sivun tarkoitusta varten lisäsin lohkolle kolmannen syöttötiedon, joka on linkeille. Kenttäryhmä on muuten samankaltainen, kuin esimerkkinä käytetty *Linkki*-lohko,

mutta säännöksi on annettu ”Lohko”, joka tarkoittaa, että kenttäryhmä näytetään lohkona WordPressin muokkausnäkymässä. Kun säännöksi on annettu ”Lohko” käytetään edellisessä luvussa esiteltyä functions.php-tiedostossa olevaa acf_register_block-funktiota lohkon rekisteröintiin.

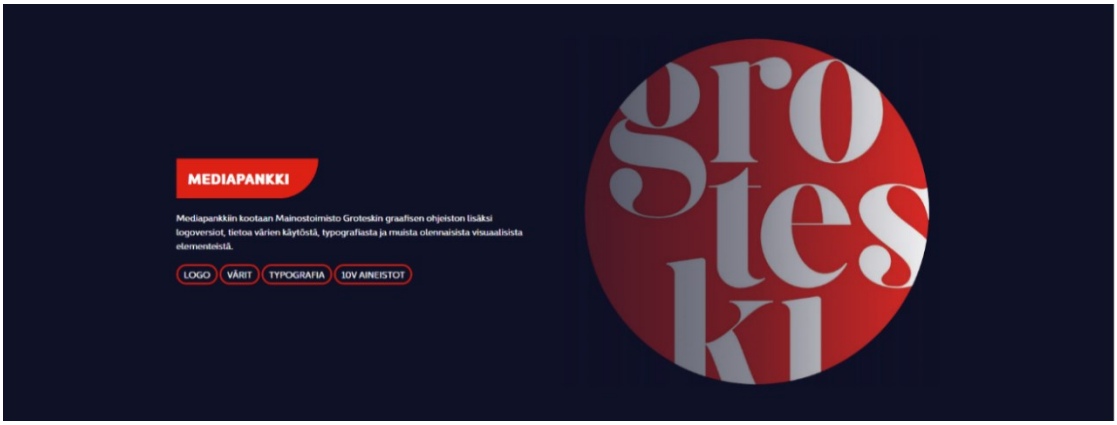
Lohkolla on oma PHP-tiedostonsa nimeltä text-and-image-block.php, joka sisältää lohkon HTML-rakenteen ja tulostamisen tekstille sekä kuvalle. Lisäsin PHP-tiedostoon tulostamisen linkeille. Kuvassa 12 on text-and-image-block.php-tiedoston rakenne, jossa ensimmäisenä on tekstin syöttötietojen tulostaminen HTML-rakenteeseen, jonka jälkeen linkit tulostetaan suoraan tekstin alle *Linkki*-lohkon tavoin. Viimeisenä tiedostossa on kuvan syöttötietojen tulostaminen HTML-rakenteeseen.

```
<section <?php groteski_block_id(); ?> class="text-and-image-block">
    <div class="grid">
        <div class="text-block">
            <div class="block-header">
            <?php groteski_block_title(); ?>
            </div>
            <?php if ( get_field( 'text' ) ) : ?>
                <?php the_field( 'text' ); ?>
            <?php endif; ?>

            <?php if ( have_rows( 'links' ) ) : ?>
                <div class="links">
                    <?php while ( have_rows( 'links' ) ) : the_row(); ?>
                        <?php $link = get_sub_field( 'link' ); ?>

                        <?php if ( $link ) : ?>
                            <?php
                            $href = esc_url( $link['url'] );
                            $title = esc_attr( $link['title'] );
                            $target = esc_attr( $link['target'] );
                            ?>

                            <a class="link" title="<?php echo $title ?>" href="<?php echo $href ?>" target="<?php echo $target; ?>">
                                <?php echo $title; ?>
                            </a>
                        <?php endif; ?>
                    <?php endwhile; ?>
                </div>
            <?php endif; ?>
        </div>

        <div class="image-block">
            <?php $img = get_field( 'image' ); ?>
            <?php $alt = $img['alt']; ?>
            <?php $source = $img['sizes']['large']; ?>

            <img class="media-image" alt="<?php echo $alt ?>" src="<?php echo $source ?>"/>
        </div>
    </div>
</section>
```

Kuva 12. Muokatun *Teksti ja kuva*-lohkon rakenne

Lohkolla on oma tyylitiedosto nimeltä text-and-image-block.scss, josta löytyy ulkoasumäärittelyt kyseiselle lohkolle. Lisäsin lohkolle ulkoasumäärittelyt linkeille. Tässä käytin sisällyttämis- (engl. *include*) toimintoa, joka mahdollistaa globaalin tyylitiedoston kutsumista toisessa tiedostossa. Globaalit tyylitiedostot pitävät sisällään yleensä sivustolla toistuvien elementtien tyylit. Globaalit tyylitiedostot kutsutaan lisäämällä tiedoston alkuun ”@use ”mixins/all” as *;”, joka

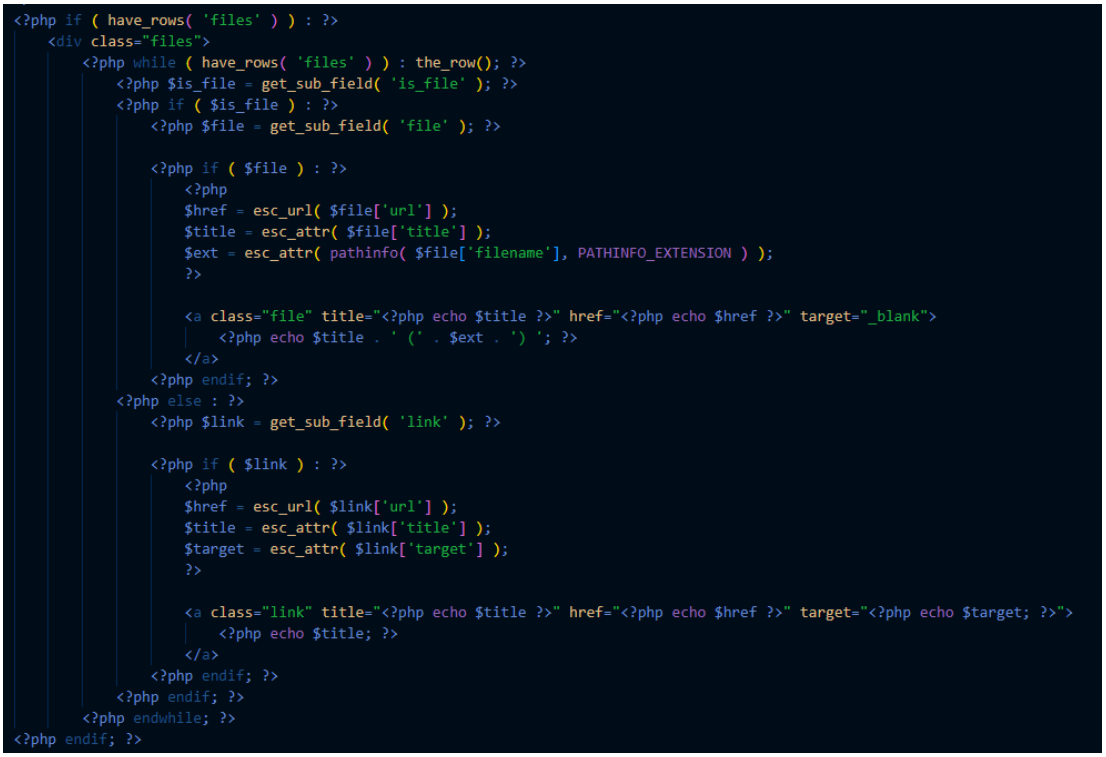
kertoo tiedostolle, että käytetään mainitussa tiedostopolussa olevia globaaleja tiedostoja. Tämän jälkeen kutsutaan globaalissa tiedostossa nimeltä links.scss olevaa joukkoa (engl. *mixin*) *links* komennolla ”@include links;”. Näin lohkoon lisätyt linkit saavat itselleen globaaliin tyylitiedostoon määritellyt ominaisuudet. Kuvassa 13 on *Mediapankki*-sivun muokattu *Teksti ja kuva*-lohko tyylien kanssa.



Kuva 13. Muokattu *Teksti ja kuva*-lohko tyylien kanssa

Aiemmin tehdyssä etusivun *Linkki*-lohkossa on myös linkkejä, mutta näiden ulkoasumäärittelyssä ei käytetty globaalia tyylitiedostoa, koska halusin etusivulla olevien linkkien olevan erinäköisiä intranetin muilla sivuilla olevista linkeistä. Muut intranetissä käytettävät linkit käyttävät globaalia tyylitiedoston ulkoasumäärittelyä.

Mediapankki-sivulla on myös useita *Media*-lohkoja, joiden sisältönä on aiemmin luotujen linkkien sisällöt. Linkkiä painamalla sivu raahautuu linkin nimeä vastaavalle *Media*-lohkolle. *Media*-lohko ei ole Groteskin teeman oletuslohko, joten loin sen samanlailla WordPressin hallintapaneelista kuin aiemmin luomani lohkot. Tässä lohkossa tarvitaan kenttäryhmän alakentän kenttätyypiksi ”Tosi / Epätosi -valinta”, jolla tarkistetaan, onko annettu syöttötieto tyyppiä tiedosto vai linkki, koska *Media*-lohkon sisältö voi olla joko ladattava tiedosto tai avattava linkki. *Media*-lohkolle luodaan media-block.php-niminen tiedosto, jossa tulostetaan syöttötietojen sisällöt HTML-rakenteeseen (kuva 14).



```php
<?php if ( have_rows( 'files' ) ) : ?>
    <div class="files">
        <?php while ( have_rows( 'files' ) ) : the_row(); ?>
            <?php $is_file = get_sub_field( 'is_file' ); ?>
            <?php if ( $is_file ) : ?>
                <?php $file = get_sub_field( 'file' ); ?>

                <?php if ( $file ) : ?>
                    <?php
                    $href = esc_url( $file['url'] );
                    $title = esc_attr( $file['title'] );
                    $ext = esc_attr( pathinfo( $file['filename'], PATHINFO_EXTENSION ) );
                    ?>

                    <a class="file" title="<?php echo $title ?>" href="<?php echo $href ?>" target="_blank">
                        <?php echo $title . ' (' . $ext . ') '; ?>
                    </a>
                <?php endif; ?>
            <?php else : ?>
                <?php $link = get_sub_field( 'link' ); ?>

                <?php if ( $link ) : ?>
                    <?php
                    $href = esc_url( $link['url'] );
                    $title = esc_attr( $link['title'] );
                    $target = esc_attr( $link['target'] );
                    ?>

                    <a class="link" title="<?php echo $title ?>" href="<?php echo $href ?>" target="<?php echo $target; ?>">
                        <?php echo $title; ?>
                    </a>
                <?php endif; ?>
            <?php endif; ?>
        <?php endwhile; ?>
<?php endif; ?>
```

Kuva 14. Media-block.php-tiedoston rakenne

Koska kenttäryhmälle on annettu tyypiksi ”Tosi / Epätosi -valinta”, kuvassa 14 tehdään *if-else* lauseke, jolla tarkistetaan, toteutuuko syöttötietojen ehdot. Ensiksi tarkistetaan, onko annettu syöttötieto tyypiltään tiedosto, jos ehto on tosi, tulostetaan HTML-rakenteeseen tiedosto. Jos ensimmäinen ehto ei täyty, tulostetaan HTML-rakenteeseen linkki. Näin *Media*-lohkon muokkausnäkymässä kysytään käyttäjältä rastiruutuun tavalla, onko annettu media muotoa tiedosto vai linkki. Jos käyttäjä laittaa rastin ruutuun, syöttöalue hyväksyy vain tiedostoja, jos käyttäjä ei laita rastia ruutuun, syöttöalue hyväksyy vain linkkejä.

Media-lohkolla on oma tyylitiedosto nimeltä media-block.scss, josta löytyy *Media*-lohkon ulkoasumäärittelyt. Ulkoasun lisäksi tiedostossa on kuvat valitun syöttötyypin mukaan (kuva 15). Jos tyypiksi on valittu tiedosto, tulostetaan tiedoston perään latauskuvake ja jos tyyppi on linkki, tulostetaan linkin perään avauskuvake.

```
&.file::before {
    background-image: url("#{$icons}/downloadwhite.png");
    background-size: cover;
}

&.link::before {
    background-image: url("#{$icons}/openlinkwhite.png");
    background-size: cover;
}
```

Kuva 15. Media-block.scss-tiedoston kuvatyypin tulostaminen

Kaikki sivustolla käytetyt kuvat ladataan *assets*-kansion sisällä olevaan *icons*-kansioon. Näin kuvia voidaan käyttää missä tiedostossa halutaan. Kuvaa kutsutaan URL-osoitteen eli polun kautta ja perään kirjoitetaan kuvan nimi, tämä näkyy esimerkkinä kuvassa 15. Kaikki edellä mainittuun kansioon ladatut kuvat pakkaantuvat *webp*-muotoon, joka vie vähemmän tilaa ja se mahdollistaa verkkosivuston nopeamman latautumisen.

4.4 Vuen käyttäminen intranetissä

Intranetissä on käytetty muutamilla sivuilla Vue.js-kehikkoa. Groteskin web-kehittäjä on vastannut suurimmaksi osaksi intranetin Vue.js puolesta. Tämä on sen takia, koska minulla ei ollut aikaa opetella syvemmin Vuen käyttöä. Kerron ainoastaan Vue-tiedostoista, joiden rakentamisessa olin itse mukana.

Etusivulta löytyy *Ohjelmat*-lohko, joka käyttää Vuea. Tämä lohko on intranetiä varten luotu kustomoitu lohko, jonka rakentaminen tapahtui käytännössä *Linkki*-lohkon tavoin. *Ohjelmat*-lohko sisältää kaikki tärkeimmät ohjelmat, joita käyttäjä tarvitsee työpäivän aikana. Suunnitteluvaiheessa keksin, että käyttäjä voisi muokata ohjelmien järjestystä omien tarpeiden mukaan, joten lohkoon otettiin käyttöön Vue.

Lohkolla on aiempien lohkojen tavoin oma PHP-tiedosto, jonka nimi on block-programs.php. Teeman hakemistossa on *js*-niminen kansio, joka sisältää sivulla tarvittavat JavaScript-koodit. Kansion sisällä on *vue*-niminen kansio, joka nimensä mukaisesti sisältää Vue-tiedostot. *Ohjelmat*-lohkoa varten tarvitaan kaksi Vue-tiedostoa. Ensimmäinen tiedosto on GroteskiProgram.vue, joka ka-

saa yksittäisen elementin ominaisuudet sekä HTML-rakenteen. Tämän tiedoston sisältö viedään (engl. *export*) toiseen Vue-tiedostoon nimeltä GroteskiPrograms.vue.

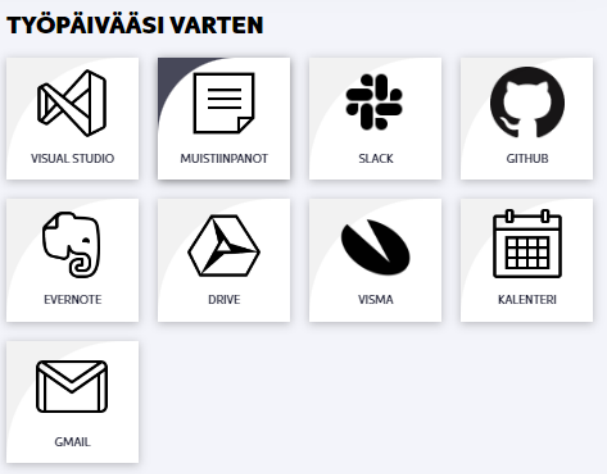
GroteskiPrograms.vue-tiedostossa tehdään liikkumislogiikka käyttäen Vuen omaa *vuedraggable*-lisäosaa, joka mahdollistaa elementtien paikan vaihtamisen. Kuvassa 16 nähdään osa logiikasta, jolla voidaan tallentaa elementtien uusi järjestys. Kun elementin raahaaminen aloitetaan *this.drag* arvo on totta (engl. *true*), kun raahaaminen lopetetaan arvo muuttuu epätodeksi (engl. *false*). *savePrograms*-funktio tallentaa elementin uuden sijainnin.

```
methods: {
  startDrag() {
    this.drag = true;

    if (this.timeout) {
      clearTimeout(this.timeout);
    }
  },
  endDrag() {
    this.drag = false;
    this.timeout = setTimeout(this.savePrograms, 1000);
  },
  savePrograms() {
    fetch(`${REST.url}/programs/save`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json; charset=utf-8',
        'X-WP-Nonce': REST.nonce,
      },
      body: JSON.stringify({
        programs: this.programs,
      }),
    })
      .then((res) => res.json())
      .then((response) => {
        console.log(response);
      });
  },
},
```

Kuva 16. GroteskiPrograms.php-tiedoston logiikkaa

Logiikan jälkeen tiedostoon luodaan ns. mallinne (engl. *template*), joka pitää sisällään *Ohjelmat*-lohkon päällä näkyvän latauskuvakkeen, jos elementtien latauksessa kestää odotettua pidempään. Kun elementit ovat latautuneet, niitä voidaan siirrellä haluamaansa järjestykseen. Lopuksi kutsutaan mallinetta block-programs.php-tiedostossa, jolloin lohko tulostetaan määriteltyyn HTML-rakenteeseen. Lopuksi Vue-tiedostot määritetään main.js-nimiseen tiedostoon, jossa luodaan Vue-sovellus, joka kuuntelee Vue-tiedostoja sivun latautuessa.

Lohkolle luodaan oma tyylitiedosto, joka sisältää lohkolle tarvittavat ulkoasumäärittelyt. Yksittäisen elementin vasemmassa yläkulmassa on vaaleanharmaa lovi, joka vaihtaa väriä tummansiniseksi, kun hiiren kohdistin on lohkon päällä (kuva 17). Tätä varten käytin *::before*- ja ::*after*-pseudoelementtejä, jotka mahdollistavat sisällön lisäämisen sivulle ilman, että ne ovat HTML-rakenteessa. Tämä tarkoittaa sitä, että vaaleanharmaa lovi ei ole luotu HTML-rakenteeseen, vaan se on luotu itse tyylitiedostossa. Kuvassa 17 on valmis *Ohjelmat*-lohko, jossa leijumisefekti on annettuna pseudoelementille.



Kuva 17. Tyylitelty *Ohjelmat*-lohko

Intranetissä on käytetty normaalia JavaScriptiä Vuen lisäksi esimerkiksi hakuja ilmoitusnapeissa. JavaScriptillä on luotu todella yksinkertaisia toimintoja, jotka tapahtuvat joko nappia painamalla tai hiiren kohdistimen viemisellä jonkin elementin päälle. Sivun alaosasta löytyy nappi, jolla voidaan palata takaisin sivun yläosaan. Nappi on luotu footer.php-tiedostoon perinteisillä *button*-tunnisteilla. Napin yksilöllinen tunniste (engl. *id*) on *scroll-up*, joka voidaan poimia JavaScript-koodissa main.js-tiedostossa (kuva 18). Napille annetaan klikkaus tapahtuma (engl. *onclick event*), jonka arvoksi määritetään ”window.scrollTo(0, 0)”, mikä tarkoittaa, että nappia painamalla sivu rullaa annettuihin koordinaatteihin.

```
$('#scroll-up').on('click', () => {
  window.scrollTo(0, 0);
});
```

Kuva 18. *scroll-up*-napin JavaScript logiikka

4.5 Responsiivisuuden testaaminen

Intranetin käyttömukavuuden kannalta tärkeimpiä ominaisuuksia on responsiivisuus. Sen avulla voidaan määrittää, miltä intranet näyttää eri päälaitteiden näyttökoissa. Responsiivisuutta tehdessä olen hyödyntänyt *breakpointteja*, mikä tarkoittaa, että verkkosivusto reagoi laitteen näyttökoon mukaan. Toisin sanoen *breakpointit* ovat näyttökoko pikseleissä, jolloin käytetään tiettyjä ulkoasumäärittelyitä. Kaikissa opinnäytetyössä käytetyissä lohkoissa on *breakpointit* käytössä.

Teemassa on valmiina sivuston oma globaali tyylitiedosto nimeltä layout.scss, joka sisältää sivuston sivupohjan responsiivisuuden. Kun lohkojen responsiivisuutta muokataan, se tehdään lohkon omaan tyylitiedostoon. Jokaiselle lohkolle on tehty omat responsiivisuudet mukaan lukien sivun ylä- ja alaosalle.

Opinnäytetyössä on käsitelty lohkojen luomista, PHP-koodin rakennetta, tyylien käyttöä ja Vuen toimimista. Koska jokaisen lohkon luominen tapahtuu käytännössä samalla tavalla, voidaan käyttää kustomoitua lohkoa nimeltä *Yhteystiedot* demonstroimaan intranetin responsiivisuutta. Tämä lohko sisältää organisaation henkilöstön yhteystietokortit.

Lohkon tyylitiedostoon nimeltä contacts-block.scss, luodaan lohkon ulkoasun lisäksi responsiivisuus. Lohkot tyylitellään Full HD eli 1920 x 1080 näyttökoon mukaan ensisijaisesti. Kun sivusto näyttää Full HD tilassa käyttäjäystävälliseltä, voidaan siirtyä *breakpointtejen* avulla alempiin näyttökokoihin. Näillä tarkoitetaan esimerkiksi tabletti ja mobiilinäkymiä. *Breakpointit* ovat määritetty *variables*-kansiossa olevaan breakpoints.scss-tiedostoon. *Breakpointeista* on tehty globaaleja muuttujia vastaamaan eri näyttökokoja. Jatkossa voidaan kutsua näyttökokoa vastaavaa muuttujaa, kun tehdään responsiivisuutta lohkolle.

Yhteystieto-lohkolle tehdään responsiivisuus kahdella erilaisella tavalla. Lohkossa käytetään useampaa tapaa, koska *Yhteystieto*-lohkon rakentamisessa on käytetty Vuea, joka aiheutti ongelmia *breakpointtien* käytössä. Yhteystietokortteja painamalla avautuu nippelitietokortti yhteystietokortin alle. Kun tein lohkon responsiivisuutta alempaan näyttökokoon, huomasin, että nippelitieto-

kortit aukesivat väärän yhteystiedon alle. *Breakpointit* sekoittivat yhteystietokorttien rivien järjestyksen, joka aiheutti edellä mainitun ongelman. Tämän takia lohkon responsiivisuus tehdään *Yhteystieto*-lohkon Vue-tiedostoon nimeltä GroteskiPerson.vue. Tämä tiedosto sisältää *Yhteystieto*-lohkon korttien aukeamislogiikan.

Kuvassa 19 on funktio nimeltä *changeColumns,* jonka avulla määritetään, mikä *breakpointti* on käytössä ja kuinka monta yhteystietokorttia yhdellä rivillä on tällä näyttökoolla. Näin varmistetaan, että nippelitietokortti tulostuu aina oikean yhteystietokorttirivin jälkeen.

```
changeColumns() {
  const width = window.innerWidth;
  const element = document.querySelector(`#${this.blockId} .contacts`);

  if (width <= breakpoints.max_xxs) {
    element.style.setProperty('--columns', 1);
    this.columns = 1;
  } else if (width <= breakpoints.max_s) {
    element.style.setProperty('--columns', 2);
    this.columns = 2;
  } else if (width <= breakpoints.max_m) {
    element.style.setProperty('--columns', 3);
    this.columns = 3;
  } else {
    element.style.setProperty('--columns', 4);
    this.columns = 4;
  }
},
```

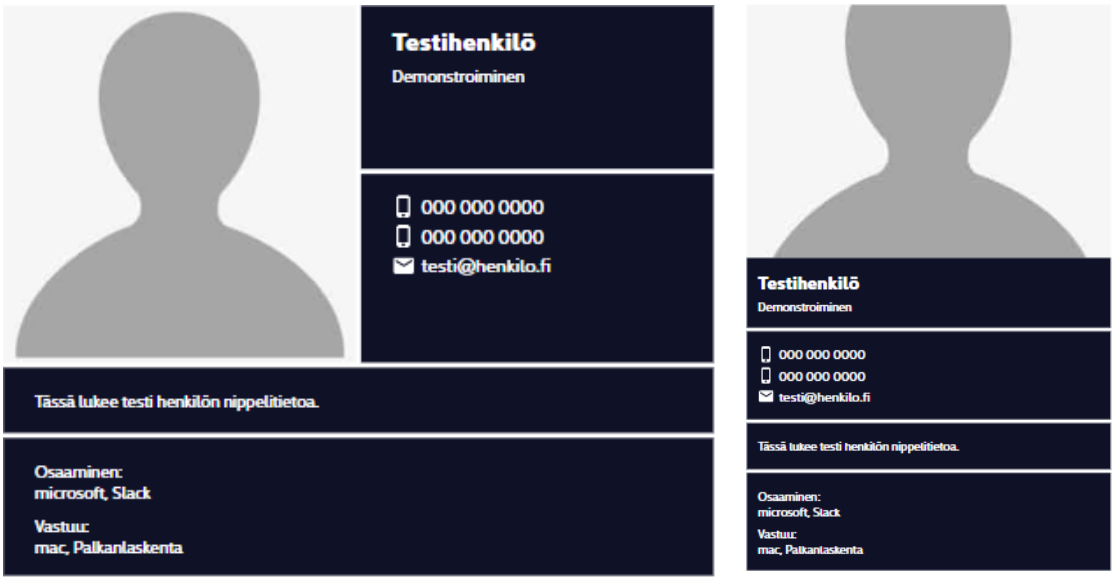
Kuva 19. GroteskiPersons.vue-tiedoston *changeColumns*-funktio

Nippelitietokortin responsiivisuus tehdään perinteisellä *breakpointtien* määrittelemisellä. Kuvassa 20 *single-contact*-luokka nimenomaan tarkoittaa avautuvaa nippelitietokorttia. Full HD näkymässä *grid-template-columns* jaetaan kolmeen yhtä suureen kolumniin. Kun näyttökoossa mennään alaspäin eli käytetään *breakpointtia* nimeltä $max-s, joka tarkoittaa maksimissaan 767 pikselin näyttökoon leveyttä, *grid-template-columns* jaetaan kahteen yhtä suureen kolumniin. Viimeinen lohkolle tarvittava *breakpoint* on $max-xs, joka tarkoittaa 575 pikselin näyttökokoon leveyttä, *grid-template-columns* muutetaan yhdeksi täysleveäksi kolumniksi. Kolumnijako vähenee aina, kun käytetään alempaa näyttökokoa.

```scss
.wrapper2 {
    position: relative;
    background-color: $white;
    grid-column: 1 / -1;

    .single-contact {
        grid-template-columns: minmax(0, 1fr) minmax(0, 1fr) minmax(0, 1fr);
        display: grid;
        text-align: left;
        gap: 5px;

        @media (max-width: $max-s) {
            grid-template-columns: minmax(0, 1fr) minmax(0, 1fr);
        }

        @media (max-width: $max-xs) {
            grid-template-columns: minmax(0, 1fr);
        }
```

Kuva 20. Contacts-block.scss-tiedoston responsiivisuuden rakenne

Responsiivisuutta voidaan helposti testata Google Chrome -selaimen kehittäjän työkaluja hyödyntäen. Kaikki intranetissä käytettyjen lohkojen responsiivisuudet ovat testattu kyseisellä tavalla. Loin esimerkkiä varten testihenkilön, ettei organisaation henkilöstön yhteystiedot ole näkyvillä opinnäytetyössäni. Normaalissa Full HD näkymässä *Yhteystieto*-lohkon nippelitietokortissa on kolme kolumnia vierekkäin. Kuvassa 21 vasemmalla on nippelitietokortti 767 pikselin näyttökoon leveydessä ja kolumnijako on kaksi. Oikealla puolella nippelitietokortti on 575 pikselin näyttökoon leveydessä ja kolumnijako on yksi, joka tarkoittaa, että kaikki elementit menevät allekkain.



Kuva 21. Nippelitietokortti 767px ja 575px näkymissä

5 YHTEENVETO JA POHDINTA

Työharjoitteluni aikana syksyn lopulla 2022 keskustelin toimeksiantajan kanssa, olisiko mahdollista toteuttaa opinnäytetyö harjoittelun jälkeen. Toimeksiantaja ehdotti minulle intranetiä opinnäytetyön aiheeksi, koska heille oli kertynyt tarve keskittää yrityksen sisäinen viestintä yhdelle selkeälle alustalle. Tämän alustan tulisi olla helposti löydettävissä ja käytettävissä yrityksen työntekijöille. Projekti kuulosti mainiolta ja todella kiinnostavalta opinnäytetyön aiheelta, koska harjoittelussa olin päässyt jo näkemään, kuinka verkkosivuja yrityksessä tehdään. Keskustelin toimeksiantajan kanssa, mitä tekniikoita intranetin rakentamista varten tarvitaan ja suurin osa tekniikoista olikin tullut harjoittelun aikana minulle tutuksi.

Intranetin rakentaminen alkoi suunnitteluvaiheesta, jossa suunnittelin erilaisia rautalankamalleja minulle uudella sovelluksella nimeltä Figma. Toimeksiantajan käyttöliittymäsuunnittelija opetti minulle Figman perusteita, joten sen haltuun ottaminen sujui ongelmitta. Käyttöliittymäsuunnittelu oli todella kiinnostava osa opinnäytetyötäni, vaikka en siitä paljoa kertonutkaan työn vaiheissa. Koin, että tekninen toteutus on projektin tärkein ja mielenkiintoisin osa-alue.

Suunnitteluvaiheen jälkeen intranetin rakentaminen jatkui toimeksiantajan omalla WordPress-teemalla, jota muokkasin luoden uusia lohkoja ja toiminnallisuuksia intranetiin sopivaksi. Opinnäytetyön aloittaminen tapahtui sulavasti, koska teema oli minulle tuttu ennestään. Emme sopineet intranetin valmistumiseen mitään aika rajaa, vaan pidimme palavereita viikoittain, joissa katsoimme, kuinka projekti on edennyt. Palavereissa sain uusia ehdotuksia ja palautetta, jotka auttoivat minua parhaaseen mahdolliseen lopputulokseen.

Työssä en kokenut tarpeelliseksi esitellä jokaisen intranetin sivun valmista lopputulosta. Minusta oli tärkeämpää kertoa työn etenemisestä vaiheittain, kuin näyttää suoraan valmis intranetin lopputulos ja kertoa siitä. Liitteessä 2 on nähtävissä valmis tuotos intranetin etusivusta, joka on toimeksiantajalla käytössä yrityksessä. Etusivun sisältöä on muokattu, ettei työssä näytetä mitään sinne kuulumatonta.

Opinnäytetyö oli loistava hetki oppia paljon uutta. Tuntui, että ennen harjoittelua minulla oli vain pintaraapaisu osaamista ja tietoa web-kehittämisestä. Harjoittelun aikana opin uusia hyödyllisiä taitoja, mutta tuntui, että opinnäytetyön myötä alkoi vasta syvempi oppiminen. Koen, että työn tavoiteltu lopputulos saavutettiin ja olen itse todella tyytyväinen siihen.

Kävin Groteskin toimistolla, kun sain intranetin ensimmäisen version valmiiksi. Sain yllätyksekseni kuulla, että intranet oli otettu käyttöön yrityksessä. Toimeksiantaja kertoi olevansa tyytyväinen uuteen alustaan, ja lopputuloksen olevan parempi kuin he osasivat edes kuvitellakaan. Intranet sisälsi kaikki yritykselle tarvittavat ominaisuudet ja se oli helppokäyttöinen. Intranetin jatkokehittäminen alkaa toimeksiantajan puolesta, jotta alustasta saadaan vielä enemmän käyttäjäystävällisempi ja yrityksen tarpeisiin sopiva.

LÄHTEET

Amsler, S. & Churchville, F. 2021. Content management system (CMS). WWW-dokumentti. Päivitetty 1.2.2021. Saatavissa: https://www.techtarget.com/searchcontentmanagement/definition/content-management-system-CMS [viitattu 15.2.2023].

Carr, D. & Gray, M. 2018. Beginning PHP: master the latest features of PHP 7 and fully embrace modern PHP development. E-kirja. Birmingham: Packt Publishing. Saatavissa: https://ebookcentral.proquest.com/lib/xamk-ebooks/reader.action?docID=5485034&query= [viitattu 1.2.2023].

GeeksforGeeks. 2022. SASS. WWW-dokumentti. Päivitetty 1.5.2022. Saatavissa: https://www.geeksforgeeks.org/sass/ [viitattu 1.2.2023].

GeeksforGeeks. 2023. Web Technology. WWW-dokumentti. Päivitetty 2.1.2023. Saatavissa: https://www.geeksforgeeks.org/web-technology/ [viitattu 1.2.2023].

Hakukonemestarit. 2020. Miten WordPress.com ja WordPress.org eroavat toisistaan. Blogi. Saatavissa: https://www.hakukonemestarit.fi/blogi/miten-eroaa-wordpress-com-ja-wordpress-org/ [viitattu 22.2.2023].

Lötjönen, J. 2021. Parhaat WordPress-lisäosat. Blogi. Julkaistu 26.6.2021. Saatavissa: https://www.zoner.fi/wordpress/parhaat-wordpress-lisaosat/ [viitattu 22.2.2023].

Suovesi, S. 2022. Mitä tarkoittaa responsiivisuus, ja miksi se on tärkeää. Blogi. Julkaistu 25.2.2022. Saatavissa: https://sininenharka.fi/mita-tarkoittaa-responsiivisuus-ja-miksi-se-on-tarkeaa/ [viitattu 7.3.2023].

Vue.js. 2023. Introduction. WWW-dokumentti. Saatavissa: https://vuejs.org/guide/introduction.html [viitattu 15.2.2023].

Webguru. 2021. Figma – Paras työkalu käyttöliittymien suunnitteluun. WWW-dokumentti. Julkaistu 14.10.2021. Saatavissa: https://webguru.fi/figma/ [viitattu 15.2.2023].

Winter, R. 2022. What is Vue.js. Blogi. Julkaistu 18.7.2022. Saatavissa: https://blog.hubspot.com/website/vue-js [viitattu 15.2.2023].

WordPress. 2019. About. WWW-dokumentti. Saatavissa: https://fi.wordpress.org/about/ [viitattu 22.2.2023].

WP-teemat.fi. 2023. Parhaat WordPress teemat. WWW-dokumentti. Saatavissa: https://wp-teemat.fi/ [viitattu 22.2.2023].

KUVALUETTELO

Kuvankaappaus toimeksiantajan hyväksymästä ulkoasumallista

Kuvankaappaus intranetin etusivusta yrityksen käytössä