Klaus Heino

Käyttäjäystävällisten nettisivujen toteutus WordPress-järjestelmällä

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Tietotekniikka
Insinöörityö
1.12.2017

Metropolia

Tiivistelmä

Tekijä(t) Otsikko Sivumäärä Aika	Klaus Heino Käyttäjäystävällisten nettisivujen toteutus WordPress-järjestelmällä 36 sivua 1.12.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Lehtori Ilpo Kuivanen Riitta Haarnoja

Tämän insinöörityön tarkoituksena oli tuoda esille, mitä standardeja nykyinen verkkosivukehitys pitää sisällään. Asiakkaalle tehtiin verkkosivu-uudistus, jossa panostettiin sivujen ulkonäköön ja samalla huomioitiin käyttäjäystävällisyyden tärkeys. Asiakas tahtoi myös sivujen sisällön olevan vaivattomasti muokattavissa. Tähän käytettiin sisällönhallintaohjelmistona WordPressiä.

Työssä eriteltiin kehityksen eri vaiheissa useita keinoja tehdä sivuista käyttäjäystävällisiä, niin sivulla vierailevan kuin myös sivun sisällönhallitsijan näkökulmasta. Käyttäjäystävällisen verkkosivunäkymän suunnittelua tuotiin esille lukuisin esimerkein kertomalla Bootstrap-teknologian tuomista mahdollisuuksista, kuten responsiivisuudesta. Työn myöhemmässä, teknisemmässä osiossa esiteltiin sisällönhallintajärjestelmänä toimivan WordPressin hyötyjä. Siinä esiteltiin myös, kuinka staattisesta sivustosta alettiin luoda dynaamista sekä myös sitä kuinka, sisällönhallinnasta voidaan tehdä WordPressissä entistäkin helpompaa käyttämällä sen lisäosia eli plugineita.

Kokonaisuutena työssä pyrittiin antamaan kokonaiskäsitys myös siitä, minkälaisin vaihein dynaamisen ja selkeän verkkosivun kehittäminen tapahtuu. Sen oli myös tarkoitus todistaa, että käyttäjäystävällisyys on mekittävä osa nykyajan verkkosivukehitystä.

Avainsanat	WordPress, käyttäjäystävällisyys, responsiivisuus, Bootstrap

Abstract

Author(s)	Klaus Heino
Title	Creating user-friendly web sites with WordPress system
Number of Pages	36 pages
Date	1.12.2017
Degree	Bachelor of engineering
Degree Programme	Information Technology
Specialisation option	Software engineering
Instructor(s)	Ilpo Kuivanen, Senior Lecturer Riitta Haarnoja

The purpose of this thesis was to present what kind of standards today's website development includes. The customer was provided with new web pages by replacing their old ones and the objective was to concentrate on the design of the user interface and how to make it user-friendly. Additionally, the customer wanted it to be possible to easily modify the content of their pages and for this the content management system WordPress was used.

The thesis also analysed how to make sites customer friendly in different parts of the development. This was done from both the client's and the content manager's point of view. The design of user-friendly web pages is presented in various ways, e.g. by describing Bootstrap technology and its potential to create responsive layouts. The study also introduces the advantages of content management system WordPress, how the static pages were turned into dynamic ones and how it is possible to make content management even easier by using its plugins.

All in all, the aim of this thesis was to provide a comprehensive description of the different stages of the development of dynamic web sites and also prove, that user-friendly web-design is something that people should be aware of.

Keywords	WordPress, user-friendly design, responsiveness, Bootstrap

Sisällys

1 Johdanto

Insinöörityönä kehitettiin asiakkaalle modernit ja käyttäjäystävälliset verkkosivut, joiden sisältö tuli olla asiakkaalle päivitettävissä mahdollisimman helposti. Mitään teknillistä pohjaa uusia sivuja varten ei ollut lukuun ottamatta asiakkaan vanhojen sivujen tekstimateriaalia.

Vanhat asiakkaan sivut olivat tyypilliset staattiset sivut, joihin uusien merkintöjen tekeminen oli työlästä, ja pienienkin muutosten teko vaati lähdekoodieditorin käyttämistä. Muokattavuus oli toteutettava mahdollisimman käyttäjäystävälliseksi, ja tämän ominaisuuden toteuttamiseksi valittiin suosittu sisällönhallintaohjelmisto.

Sisällönhallintaohjelmistojen avulla tietotekniikassa kokematonkin henkilö voi helposti ja turvallisesti käsitellä sivujensa sisältöä ilman, että turvautuu jokaisessa vaiheessa tietotekniikaa paremmin osaavaan henkilöön. Sisällönhallintajärjestelmiä on maailmalla useita, ja monet niistä perustuvat avoimeen lähdekoodiin. Niille tyypillistä on tarjota käyttäjilleen front-end-kohtainen käyttöliittymä, jonka avulla käyttäjä voi helposti navigoida itsensä halutun sivuston sisällön äärelle ja muuttaa sitä. Muokattava sisällön tyyppi voi vaihdella kuvista tekstiin ja äänistä videoihin.

Valittaessa sisällönhallintajärjestelmää useiden eri vaihtoehtojen väliltä, päädyttiin käyttämään WordPressiä, sillä tämä järjestelmä vastasi asiakkaan senhetkiseen tilanteeseen parhaiten. Toisin kuin esimerkiksi vastineet Drupal ja Joomla, on WordPress helppokäyttöisempi pohja liikkeen sivustolle, jonka tarkoitus on vain näyttää informaatiota. Mikäli kyseessä olisi ollut jotakin monimutkaisempaa, kuten verkkokaupan luonti, olisi tullut harkita Joomlan käyttöä [1]. Lisäksi kun aikaa oli rajoitetusti, niin oli WordPressin valinta vaihtoehdoista turvallisin myös työn nopean valmistumisen kannalta.

Insinöörityössä on tarkoitus käsitellä WordPressiä yleisesti, kertoa sen koodirakenteista sekä eritellä sen ominaisuuksia ja liitännäisiä. Sen lisäksi käsitellään monipuolisesti käyttäjäystävällisyyttä sekä mobiiliyhteensopivuutta. Tuloksena syntyi omalla teemalla tuettu WordPress-sivusto, jonka sisällön muokattavuus oli helposti opeteltavissa koodia taitamattomillekin käyttäjille.

2 WordPressin historiaa ja teoriaa

WordPress on todistetusti käytetyin sisällönhallintajärjestelmä maailmassa. Lähteiden mukaan vuonna 2016 sen osuus sisällönhallintajärjestelmää käyttävistä sivustoista oli lähes 60 prosenttia ja reilu neljännes kaikista maailman nettisivuista. Sen käyttöliittymä on käännetty 40 eri kielelle suomi mukaan lukien. [2.]

Menestystarina sai alkunsa vuonna 2003 kun Matt Mullenweg ja Mike Little kopioivat b2-ohjelmiston vapaan lähdekoodin. b2 on sekin yhä käytetty sisällönhallintaohjelmisto, joka tunnetaan nykyisemmin nimellä b2evolution. Tästä huolimatta WordPressiä pidetään b2:n virallisena seuraajana. WordPressin koodi on kirjoitettu PHP:llä ja se käyttää MySQL- tai MariaDB tyypin tietokantaa. WordPress on saavuttanut erityistä suosiota etenkin blogia ylläpitävien käyttäjien keskuudessa, sillä WordPress tarjoaa helposti käyttöönotettavat ominaisuudet niitä varten. Mikään ei tietenkään estä luomasta perinteisempää sivustoa, jonka sisältöä ei tarvitse säännöllisesti päivittää. [2.]

Kun versio 1.0 julkaistiin tammikuussa 2004, niin jo se sisälsi suoraviivaisen asennuksen ja joukon muita ominaisuuksia, joita järjestelmän käyttäjät ylistävät vielä tänä päivänä. Se saavutti suurta suosisota jo ensihetkinään, mutta voimakkaaseen kasvuun suosio lähti 2005-2007 välisenä aikana, jolloin WordPress sai oman, hyvästä rahoituksesta nauttivan, kaupallisen yksikkönsä nimeltä Automattic. Tämän jälkeen WordPress on jatkanut voitollista kulkuaan aina näihin päiviin asti, eikä sen tulevaisuuteen liity suurempia uhkakuvia tai kilpailioita. [3.]

Vaikka WordPress onkin erinomainen järjestelmä, ei se ole suinkaan täydellinen. Erääksi sen merkittävistä heikkouksista on ironisesti sen suosio, joka on asettanut järjestelmän ja sen työkalut vuosien varrella hakkerien kovalle rasitteelle. Esimerkiksi vuonna 2011 suurta suosiota WordPress-sivustoissa saavuttanut työkalu ”timthumb” osottautui vaaralliseksi. Apuväline salli helpon kuvan koon muokkaamisen, mutta samalla avasi takaportin, jonka kautta hakkerit saattoivat ladata haitallisia tiedostoja sen kansiorakenteisiin. [4.] WordPressin kehitystiimi on kuitenkin ollut ahkera tietoturvaan liittyvissä asioissa, ja järjestelmää päivitetäänkin säännöllisesti. Vuonna 2013 julkaistu WordPressin versio 3.7 toi automaattiset päivitykset koskien tietoturvallisuutta ja stabiiliutta. [5.]

2.1 Teemat

WordPress on oikeastaan vain työkalu: sen on tarkoitus mahdollistaa datan muokattavuus ja tietokantaan varastointi hallintapaneelin kautta. Se miltä itse nettisivu näyttää vierailijoille selaimen näkymässä eli front-endissä ja millaisia eri muokkaamismahdollisuuksia se tarjoaa, riippuu vain ja ainoastaan valitusta teemasta ja mahdollisista lisäosista.

Teemoja on saatavilla asennettavaksi sekä maksullisena että ilmaiseksi. Maailmanlaajuinen yhteisö takaa sen, että visuaalisesti kauniin ja trendikkään sivuston saa perustettua helposti. Pelkästään maksuttomiakin teemoja löytyy tuhansittain.

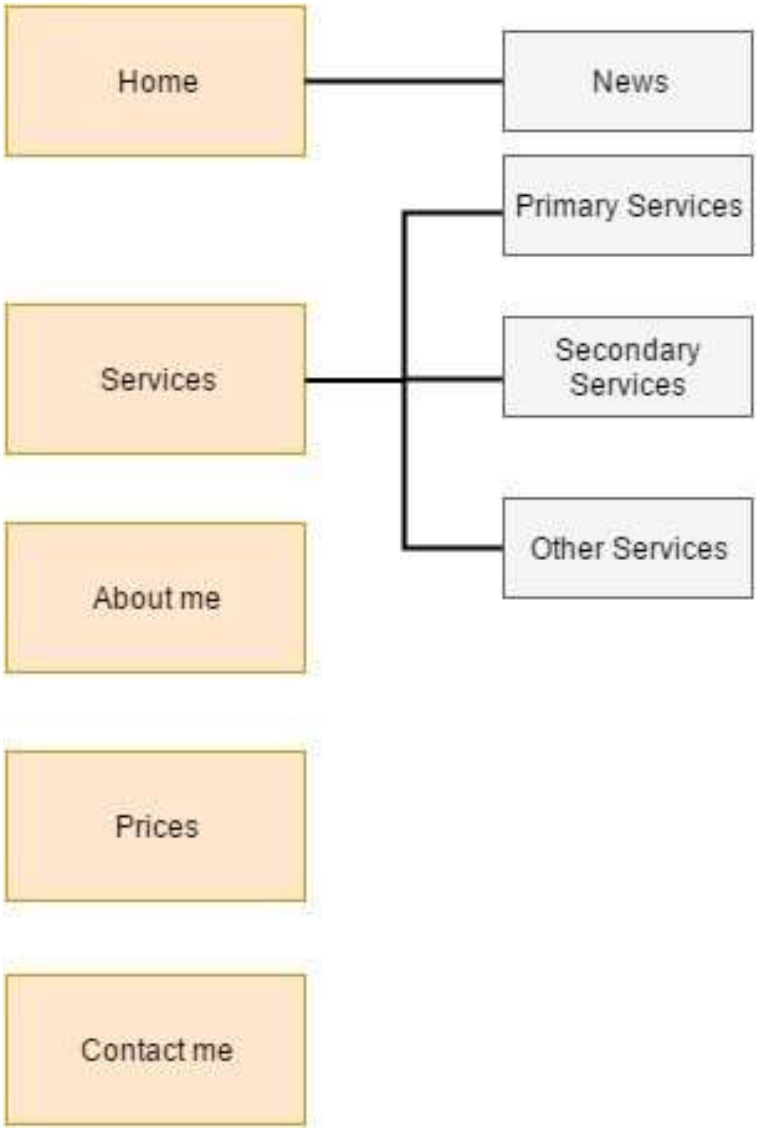
Insinöörityön tekeminen olisi helposti onnistunut valitsemalla jonkin valmiin teeman, jonka tarjoamat ominaisuudet ja visuaalisuus olisivat sopineet asiakkaalle. Tällaisen projektin täytäntöön laittaminen olisi vaatinut todella vähän tai ei välttämättä ollenkaan verkkosovellusten ja -sivujen ohjelmointiin liittyvää kokemusta, ja sivusto olisi voinut olla käyttövalmis muutamassa illassa.

Äsken mainitussa skenaariossa opeteltavat asiat olisivat rajoittuneet lähinnä WordPressin asennustyöhön sekä ko. ohjelmiston tiedostorakenteiden opettelemiseen. Kaikki se logiikka, millä tavoin teema ja sen koodi kommunikoivat WordPressin käyttöliittymän kanssa olisi sen sijaan jäänyt oppimatta. Mikäli haluaa olla varma, että käytetty teema on varmasti toimiva sekä hyvin optimoitu, kannattaa harkita oman teeman kirjoittamista. Samalla saa ymmärrystä WordPressin PHP-koodin yleisistä funktioista, jolloin opittuja asioita voi hyödyntää muidenkin tekemien teemojen kanssa. Pyörää ei välttämättä aina tarvitse keksiä uudellen, kun voi valita aluksi sopivan teeman ja räätälöidä siitä omanlaisensa.

Omien WordPress-teemojen luonti on paitsi taito myös uramahdollisuus. Jo pelkästään Suomessa on useita yrityksiä, jotka työstävät asiakkailleen verkkosivuja käyttäen WordPressiä julkaisualustanaan. Jotkut saattavat taas työllistyä freelancereina ja tehdä kehittäjinä töitä täysipäiväisesti tai oman työnsä ohessa.

3 Tuotannossa käytetyt menetelmät ja teknologiat

Kuten jo johdannossa mainittiin, niin projektia lähdettiin tekemään melkein vapain käsin. Ensimmäisessä palaverin jälkeen materiaalina oli selvillä ainoastaan osa sivujen informaation eli kuvien ja tekstien sisällöstä sekä kuva 1 informaatioarkkitehtuurimallista. Informaatioarkkitehtuurin tarkoitus on esitellä sivujen suhteet toisiinsa ja esitellä, mitä kautta pääsee mihinkin sivuun. Kuvassa on kaksi kerrosta, joista vasemmalta luettaen on ylin kerros eli sivut, joista pääsee siirtymään alemman kerroksen sivuille esimerkiksi linkkien kautta.



Kuva 1: Sivuston informaatioarkkitehtuuri

Projekti lähti käyntiin ensimmäiseltä asteelta eli vaatimusmäärittelystä. Piti osata ennalta miettiä ja määrittää, mitä toiminnallisia-, laadullisia- ja resurssivaatimuksia projekti tulisi sisältämään. Tuloksena haluttiin siis nettisivut, joiden informaatio olisi helposti muokattavissa asiakkaalle. Tätä varten teknologiaksi otettiin käyttöön WordPress, joka vastasi asiakkaan esittämiin toiveisiin sekä laadun- että myös resurssien näkökulmasta. Asiakkaalle oli tärkeää saada sivut luotua mahdollisimman edullisesti. WordPressin lisäksi projektiin sisältyi useita muita teknologioita, jotka liittyivät niin työympäristöön kuin käytettäviin kieliin.

3.1 Tuotantopalvelin

Vaikka WordPressin käyttöönotto itsessään onkin ilmaista, eivät mitkään verkkosivut maailmassa pyöri ilman toimivaa palvelinta. WP vaatii isäntäpalvelimeltaan lisäksi mahdollisimman tuoreen version PHP:stä ja MySQL-tietokannasta. Tätä vanhempiakin versioita tuetaan, mutta varoitetaan mahdollisista tietoturvauhista. Kirjoitushetkellä PHP:n version suositeltiin olevan 7 ja MySQL:n luku oli numeroltaan 5.6. Lisäksi palvelimen tulisi sisältää kovalevytilaa, jonka määrä riippuu luonnollisesti työstettävän sivuston koosta. Vaikka WP ei kovalevytilalle perso olekaan, niin monet sitä käyttävät piskuisemmatkin sivut käyttävät kymmeniä megatavuja kapasiteettia. [6.]

3.2 Kehityspalvelin

Jotta kehitettävän ohjelmiston testaaminen kehityksen aikana oli mahdollista, tarvitaan lokaali eli paikallinen palvelin. Tätä varten ladattiin WAMP-niminen ohjelmistopaketti, joka on suunniteltu Windows-käyttöjärjestelmälle. Tämän avulla luotiin kehitysympäristö, jossa oli mahdollista ajaa ja kehittää dynaamisia myös back-end-puolta tukevia nettisivuja.

3.3 HTML

HTML (HyperText Markup Language) on merkintäkieli, jota käytetään nettisivujen rakenteiden määrittämiseen. Sen tehtävänä on esittää, miten ja missä järjestyksessä nettisivun sisältö näytetään. Tämä sisältö koostuu elementeistä ja niillä voidaan määritellä tekstiä, kuvia tai hyperlinkkejä muihin sivuihin yms. [7.] HTML-elementeillä on usein myös attribuutteja, jotka ovat lisäarvoja. Näiden tarkoitus on vaikuttaa niihin liittyvien elementtien käyttäytymiseen monin eri tavoin.

Yhdessä elementit rakentavat DOM-puun: Rakenteen jossa elementin ”body” sisälle kirjoitetaan se, mitä sivulla halutaan näytettävän. Body:n ulkopuolella taas määritetään erilaiset riippuvuussuhteet esim. merkistöstandardeihin sekä seuraavaksi käsiteltäviin tyylitiedostoihin. Se miltä HTML:n tuottama tulos itsessään on todella yksinkertaista ja jopa tylsää, mutta tämän korjaa seuraavaksi esiteltävä CSS.

```
<html>
   <head>
      <link rel="stylesheet" type="text/css" href="css/style.css">
      <meta charset="UTF-8">
       <title>HTML -example</title>
   </head>
   <body>
      <h1>Otsikko</h1>
      <p>Paragrafi</p>
      <p>Toinen paragrafi</p>
   </body>
</html>
```

Koodiesimerkki 1: HTML -koodi esimerkki

3.4 CSS

CSS (Cascading Style Sheets) on esityskieli, ja sen tehtävänä on määrittää, millaisia visuaalisia ominaisuuksia HTML:n tuottamat elementit saavat. CSS pystyy vaikuttamaan mm. väreihin, kokoon, asetteluun ja fonttien määritykseen. [8.] Kun CSS ja HTML toimivat yhdessä, voidaan muodostaa kauniita ja nykyajan visuaaliset standardit täyttäviä sivuja.

3.5 PHP

PHP (PHP: Hypertext Preprocessor) on ohjelmointikieli ja sen avulla kehitettävälle sivustolle voidaan lisätä älyä ja dynaamisuutta. PHP:n ansiosta saadaan tehtyä ominaisuuksia ja tulostettua sisältöä, joka voidaan hakea automaattisesti esim. sivuston käyttämästä tietokannasta tai tiedostosta. Se poikkeaa myös edellä mainituista kielistä suoritustavaltaan: Käyttäjän siirtyessä

sivustolle ajautuvat HTML ja CSS automaattisesti selaimessa. PHP suoritetaan nettisivun isäntäpalvelimella, minkä jälkeen vasta saadut tulokset lähetetään vierailijan selaimelle.

PHP:tä käytetään yleisesti upotettuna sivuston HTML rakenteisiin. Ilman PHP:n ominaisuuksia HTML:n tuottamat elementit ovat paitsi staattisia myös niiden tulostaminen sivustoon pitää tehdä lisäämällä aina yksi elementin tekevä rivi HTML-skriptiin. PHP:n avulla voidaan luoda funktioita ja erinäisiä yleisiä rakenteita kuten if- ja for-lausekkeita.

```
<html>
    <head>
        <link rel="stylesheet" type="text/css" href="css/style.css">
        <meta charset="UTF-8">
        <title>PHP -example</title>
    </head>
    <body>
        <h1>Tama on otsikko</h1>
        <?php
            $index = 1; $name =
            'paragrafi';

            for ($index; $index <= 5; $index++) {

                if ($index === 5) {
                    echo 'Seuraavaksi tulostetaan viimeinen' . $name;
                }
                <p>Tama on paragrafi</p>
        ?>
        <?php
            }
        ?>
    </body>
</html>
```

Koodiesimerkki 2: PHP -koodiesimerkki

PHP-koodiesimerkistä voidaan havaita, että se sisältää paljon yhteneväisyyksiä aiemmin esitellyn HTML-koodiesimerkin kanssa. Siihen on kuitenkin myös lisätty rivejä, joiden koodi saattaa

näyttää tutulta eri ohjelmointikieliä opiskelleelle. PHP:n avulla voidaan HTML:n elementtejä iteroida, asettaa ehtoja sekä määrittää muuttujia. Toisin kuin esimerkiksi Javassa tai C++:ssa, muuttujille ei tarvitse määritellä tyyppiä, vaan kunkin muuttujan eteen lisätään $-merkki. PHP pystyy tunnistamaan tyypin jo muuttujaan asetettavan arvon perusteella. [9.]

Esimerkki on pelkkä raapaisu kielen tarjoamista mahdollisuuksista. PHP:ta ja sen liittymistä WordPressiin sekä tietokantoihin tullaan käsittelemään myöhemmin tässä dokumentissa.

3.6 Bootstrap

Bootstap on suosittu, ilmainen rajapinta verkkosovelluskehityksessä. Tämä alun perin Twitteriä varten luotu projekti tarjoaa kehittäjälle runsaasti ominaisuuksia ja komponentteja, joiden ansiosta modernien nettisivujen tekeminen onnistuu vaivattomammin.

Bootstrap on tiedostorakenteeltaan laaja hakemisto, johon kuuluu pääasiassa css- ja jstiedostoja. Jotta esim. tyylitiedostojen tarjoamat ominaisuudet saa käyttöönsä, on niihin tehtävä niihin linkitys, joka tapahtuu helposti samaan tapaan kuin mihin tahansa muihin tyylitiedostoihin.

3.6.1 Grid-systeemi

Yksi Bootstrapin suosituimmista sekä merkittävimmistä ominaisuuksista on varmasti Grid-systeemi, joka tekee sivuston responsiivisuuden toteuttamisesta helppoa. Responsiivisuus on iso trendi tämän hetken verkkosovelluskehityksessä, ja sillä tarkoitetaan mukautuvuutta. Responsiivisesti toimiva sivusto suoriutuu käyttäjäystävällisesti laitteesta riippumatta mukauttaen sisällön kehittäjän haluamalla tavalla. Tällainen sivusto oikein toteutettuna on käyttäjäystävällinen olipa sitten vierailija verkossa älypuhelimella, tabletilla tai tietokoneella.

Bootstrapin Grid-systeemi perustuu kolumni- eli sarakekohtaiseen leikkaukseen, joka jakaa sivun kahteentoista osaan leveyssuunnassa. Sillä on valmiiksi viisi erilaista kerrosta, joista kukin liittyy tiettyyn tuettuun laitteeseen. Se miten eri elementeille määritellään tapa, miten ne näytetään eri tasoilla, tapahtuu attribuuteilla. Lisäksi näiden attribuuttien sisältävien elementtien parentteina käytetään tiettyjä Bootstrap-elementtejä.

```
<body>
  <div class="container">
      <div class="row">
         <div class="col-md-12">col-md-12</div>
      </div>
      <div class="row">
            <div   class="col-md-3">col-md-3</div>
            <div   class="col-md-3">col-md-3</div>
            <div   class="col-md-3">col-md-3</div>
            <div   class="col-md-3">col-md-3</div>
       </div>
    </div>
</body>
```

Koodiesimerkki 3: Bootstrap Grid -esimerkki

Elementti ”container” toimii kaikkien sarakkeiden tuoman mukautuvuuden ”kääriänä”. Sen lapsena toimiva ”row” puolestaan, huolehtii elementtien rivittämisestä, minkä avulla sarakkeita voidaan ryhmitellä lukumäärällisesti erikokoisiin joukkoihin. ”row”-elementtien lapsina toimivat taas ”div”-elementit, joiden attribuutiksi asetetaan col-**-**. Tämän attribuutin arvon nimi vaihtelee haluttujen ominaisuuksien mukaan. Sarakkeiden sisään kirjoitetaan lopulta itse sisältö, mikä halutaan esittää sarakkeen määrittämällä alueella.

Sarakeattribuuttiin liittyy numero, jonka arvo vaihtelee yhdestä kahteentoista, ja se luonnollisesti merkitsee, kuinka paljon tilaa sarakkeen sisällölle tahdotaan varata. Kuvan 2 esimerkki on luonut selaimen näyttämään kuudesta sarakkeesta: Yhdelle riville voidaan mahduttaa sarakkeita niin kauan, kunnes niiden vaatiman tilan summa ylittää 12. Riviä ei ole kuitenkaan pakko käyttää koko leveydeltään, kunhan seuraaville sarakkeille määrittelee vanhemmaksi uuden ”row”-elementin.

Kuva 2: Grid-Systeemi

Numeron lisäksi attribuutin nimessä tulee olla kaksikirjaiminen merkkijono, joka viittaa, mihin tasoon attribuutin tuomalla säännöllä halutaan viitata. Merkkijono voi olla xs, sm, md tai lg. Näillä halutaan viitata tiettyihin ennalta määrättyihin pikseleillä laskettuihin leveyksiin, jotka toimivat pysäytyspisteinä. Esimerkiksi elementti, jonka attribuuttina toimii col-lg-4, vie näkymästä kolmanneksen tilaa ja vain jos selaimen senhetkinen leveys on yli 1200 pikseliä. ”col-md-4”-attribuutin pysäytyspiste on vastoin 960px. Mikäli selain vie vähemmän tilaa esim. mobiilinäkymässä, niin ottaa ko. sarake automaattisesti koko rivin verran tilaa sisällölleen. Jos Grid-systeemi toimisi näin, olisi elementeille vain kaksi eri näkymää: koko ruudun leveys tai sarakkeen varaama leveys. Sarakeattribuutit kuitenkin toimivat siinä missä muutkin attribuutit, joten niitä voidaan asettaa elementille useita. Tämän ansiosta kehittäjä voi antaa kuhunkin tasoon omanlaisensa säännön.

3.6.2 Komponentit

Komponentit ovat Bootstrapin tarjoamia, valmiita elementtejä, joihin on tehty valmiiksi viimeistellyt tyylimäärittelyt. Samoin kuten Grid-systeemin sarakkeiden kanssa, komponentit voidaan määrittää antamalla elementille haluttu attribuutti. Vaikka komponentit ovatkin hyödyllisiä, niiden nopean käyttöönoton ja uudelleenkäytettävyytensä ansiosta, eivät niiden valmiiksi generoidut tyylit aina välttämättä miellytä. Onneksi näitä tarpeen vaatiessa voi ylikirjoittaa esim. halutessa vaihtaa taustaväriä. Tosin joskus oman komponentin luominen voi onnistua jopa nopeammin varsinkin, jos ristiriitoja on kovin paljon. Komponentteja saadaan rakennettua viittaamalla Bootstrapin CSS-tiedostossa oleviin tyyleihin.

3.7 Staattisen pohjan suunnittelu ja luonti

Kun verkkosivuja tai mitä tahansa muuta ohjelmistoprojektia lähdetään työstämään kooditasolla ja valmista materiaalia on rajallisesti, tulee edetä pienin askelin. Dynaamisten ominaisuuksien kuten sisällön muokattavuuden miettiminen on epäedullista, jos ei tiedetä edes millaisessa visuaalisessa muodossa sivusto tahdottaisiin näytettävän. Tämän vuoksi lähdettiin aluksi luomaan sivustoa staattisella pohjalla, johon liittyi HTML:n, CSS:n ja Bootstrapin käyttöä. Tämän luvun tarkoitus onkin kertoa, mitä tapahtui ennen kuin WordPress astui osaksi projektia. Kutakin sivua varten laadittiin paperiset prototyypit, joiden pohjalta sivuja aloitettiin rakentamaan.

3.7.1 Sivuston värimaailman pohdinta

Nettisivujen värisuunnittelu on tärkeää ja olennaista. Värit ovat erittäin voimakas suunnitteluväline. Niitä tulee käyttää mahdollisimman tehokkaasti, sillä eri värit vetävät katsojan huomion puoleensa. Väreihin liittyy sekä emotionaalisia että sosiaalisia merkityksiä. [10.]

Yksi väri oikein suunniteltuna voi lisätä työskentelyn tehokkuutta, nopeutta ja tarkkuutta, mutta tosin yksi väri liikaa voi romuttaa koko sommittelun. Värien maksimiksi sanotaan 5 +- 2 väriä, jos tahdotaan käyttäjän muistavan värien merkitys. [10] Ihmisen lyhyt muisti pystyy muistamaan korkeintaan seitsemän asiaa ja niiden merkityksen kerrallaan.

Väri vaikuttaa voimakkaasti myös tunteisiin, mitä voi suunnittelussa käyttää hyväksi. Siksi tuleekin valita värit, jotka vastaavat haluttuihin käyttäjän tunteisiin. Esim. mikäli työstettäisiin seurustelupalstaa tulisi valita värejä, jotka herättäisivät käyttäjissään intohimon tunteita. Eli punaista, pinkkiä tai oranssia. Nämä värit vastaavat intohimon, ystävällisyyden, lämmön ja romantiikan tunnetiloja. Värejä voidaan käyttää tarvittaessa ilmaisemaan myös negatiivisia asioita. Esim. punaista käytetään globaalisti ilmaisemaan vaaroja tai virhetiloja. [11.]

Asiakkaan toiveiden mukaisesti lähdettiin etsimään luonnonläheisiä sävyjä: värejä, jotka viestittäisivät terveyttä, elämää, luontoa, rauhaa ja henkisyyttä. Aiemmin mainittiin, että sivustolle tulisi valita mielellään kolmesta seitsemään väriä ja päädyttiin valitsemaan väripaletti, joka valkoisen ja mustan lisäksi koostuisi kuvan 3 sävyistä.

Kuva 3: Sivuston väripaletti

Sivuston värimaailma tuli siis koostumaan seitsemästä väristä. Niiden käyttö ei ollut yhtä runsasta toisiinsa nähden, mutta jokaisella niistä oli tietty rooli ja elementti, jonka kanssa ne esiintyivät. Erilaiset otsikointiin kuuluvat elementit päätettiin värjätä kuvan 3 väripaletin oikeanpuolinamaisella sinisellä värillä, kun taas linkkien värittämiseen käytettiin sinisestä vasemmalla olevaa turkoosia sävyä.

3.7.2 Navigaation sovittaminen

Nimensä mukaisesti navigaatiopalkin on tarkoitus valaista käyttäjää, millaista sisältöä sivusto tarjoaa. Sen tyylikkään muotoilemisen lisäksi on huolehdittava, että linkit eri sivuille on oikein aseteltu. Usein nykyään suositaan ratkaisuja, joissa navigaatioon ei kuulu linkkiä ”Kotisivu” tai ”Home”, vaan kotisivun linkki voi olla joko firman logo tai firman nimi. Kannattaa myös suosia standardia, jonka mukaisesti linkit esitetään vaakatasossa 4, vaikka joissakin designeissa allekkain voi myös olla toimiva, ellei parempi vaihtoehto.

Kuva 4: Navigaatio

Koska informaatioarkkitehtuurista saatiin suhteessa hyvinkin yksinkertainen, mahtuivat navigaatiossa olevat linkit yhdelle riville. Kuten jo luvussa ”Sivuston värimaailman pohdinta” mainittiin, niin ihmisen lyhyt muisti pystyy käsittelemään vain seitsemää asiaa kerrallaan. Mikäli navigaatiossa olevien linkkien määrä ylittää ko. luvun, on syytä pohtia sisällön jäsentelyä uudelleen. Samalla vältyttiin tekemästä erillisiä navigaatioita alemmille tasoille tai käytettävyydeltään heikkoja pudotusvalikoita. [12.]

Navigaation linkit mahtuivat vaakatasossa normaaleilla pöytäkoneilla, mutta älypuhelimen näkymässä näin ei ole. Täysmittaisen sivusto navigaation esittäminen on mobiilissa usein jo tilanpuutteen vuoksi sula mahdottomuus. Responsiivisuuden nimissä puhelimille oli luotava omanlaisensa näkymä, jonka tuli olla erilainen, mutta silti yhtä tehokasta käyttää. Tätä varten käytettiin yleisesti tunnettua "hampurilaisvalikkoa": Kuvan 5 esittämässä näkymässä sivun ylälaidassa näytetään ainoastaan kotisivun linkin lisäksi nappula, joka avaa listauksen linkeistä muihin sivuihin. Täten listaus näytetään aina kun tarpeellista.

Kuva 5: Navigaatio puhelinnäkymässä

Mobiilinäkymän sai kehitettyä helposti Bootstrapin tarjoamalla ”navbar”-komponentilla. Kuvassa 6 rivillä 34 määritetään ”button”-elementti ja sille määritellään joukko erilaisia attribuutteja. ”navbar-toggle” määrittää näkymän leveyden murtorajan, jolloin nappula näytetään, mikä pätee jo tablettinäkymissä. Samalla rivin attribuutti ”collapse” mahdollistaa navigaation ”romahduttamisen” alas nappulaa painattaessa. Viimeisenä rivillä 44 oleva ”navbar-collapse” taas toimii viittauksena siihen sisältöön, mikä halutaan näyttää nappulaa painetaessa, ja tämä löytyy riveiltä 45-48.

```
31      <div class="navbar navbar-inverse navbar-fixed-top" role="navigation">
32          <div class="container">
33              <div class="navbar-header">
34                  <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-collapse">
35                      <span class="sr-only">Toggle navigation</span>
36                      <span class="icon-bar"></span>
37                      <span class="icon-bar"></span>
38                      <span class="icon-bar"></span>
39                  </button>
40                  <div id="title">
41                      <a id="title-link" href="index.html"><h1 id="title-text">Aina Ilona</h1></a>
42                  </div>
43              </div>
44              <div class="navbar-collapse collapse">
45                  <ul class="nav navbar-nav navbar-right" id="nav-head">
46                      <li><a href="services.html">Palvelut</a></li>
47                      <li><a href="about-me.html">Riitta</a></li>
48                      <li><a href="prices.html">Hinnasto</a></li>
49                  </ul>
50              </div>
51          </div>
52      </div>
```

Kuva 6: Navigaation luova HTML

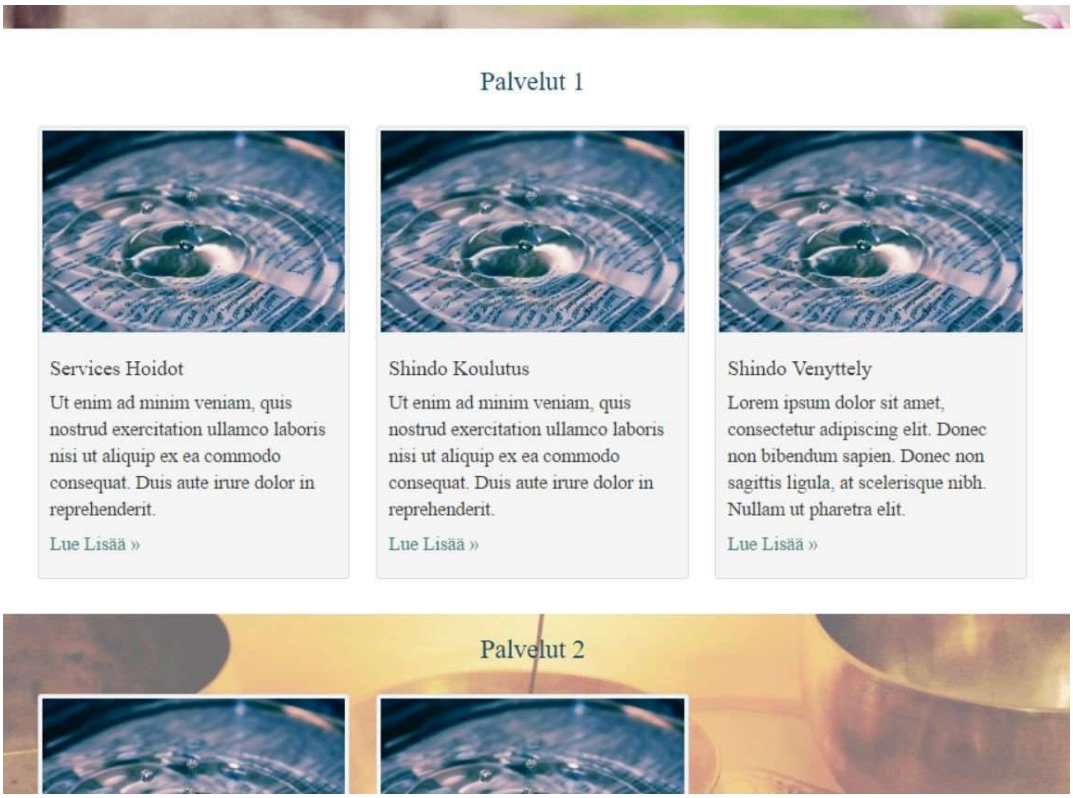
3.7.3 Informaatioarkkitehtuurin jäsentely järkeväksi

Prototyyppien piirtämisen ja miettimisen aikana tehtiin asiakkaan kanssa yhteisymmärryksellä muutoksia alun perin laadittuun 1 IA-arkkitehtuuri malliin. Kuvasta 1 näkyy, kuinka sivulta ”Services” tulisi päästä kolmelle eri alasivulle, mikä tapahtuu tyypillisesti linkkien avulla. Vaikka näiden alasivujen graafinen suunnittelu olisikin ollut toisistaan poikkeamatonta, niin viimeistään kunkin sivun saattaminen muokattavaksi olisi vaatinut lisää aikaa. Lisäksi verkkosivujen ja niiden esittämän sisällön suunnittelussa, on pyrittävä mahdollisimman yksinkertaiseen ja käyttäjäystävälliseen jäsentelyyn. Mitä suurempi sivusto on, sitä enemmän on kiinnitettävä huomiota siihen, ettei käyttäjä eksy etsiessään haluamaansa tietoa: Valtava tiedon määrä sekä käyttäjän huomiosta kilpailevien elementtien välinen kilpailu saattaa johtaa turhautumiseen. [13.]

Myönnettäköön, että alkuperäinen yhdeksän sivun malli ei olisi kärsinyt vielä huonosta käytettävyydestä, sillä käyttäjä olisi pystynyt navigoimaan sivustolla vielä vaivatta, jos yläsivuja varten luotaisiin paluulinkit esim. ”murupolkujen” avulla. Mutta jos yleisesti halutaan luoda sivu eikä sen sisältämää informaatiota ole paljon, on hyvä miettiä muita mahdollisuuksia esitystavalle.

Kun selvisi, ettei sivulle ”Palvelut” ollut alun perinkään aikomus laittaa linkkien lisäksi paljoa informaatiota, niin päädyttiin kaikkien kolmen alasivun sisältö mahduttaa ”Palvelut”-sivulle. Käytettävyyden kannalta onkin tällöin pohdittava, onko sivulle saapunut käyttäjä kiinnostunut kaikesta sisällöstä vai pelkästään sen osasta: tieto on esitettävä siten, että käyttäjä löytää tehokkaasti haluamansa.

Hyvää käytettävyyttä lähdettiinkin luomaan hahmolakeja käyttäen. Nämä ovat sääntöjä tai heuristiikkoja, joilla kuvataan, miten ihminen yhdistelee havaintojensa yksityiskohtia kokonaisuuksiksi.



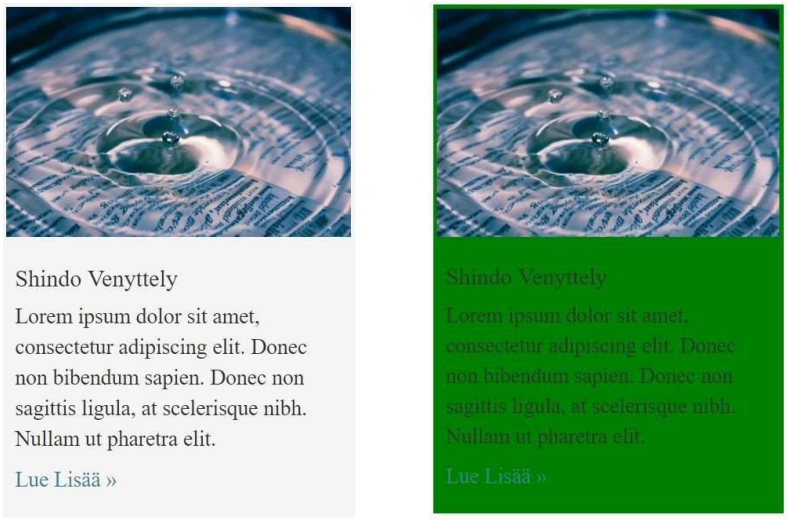
Kuva 7: Sivun sisällön jakaminen alueisiin

”Palvelut”-sivulla tuli kolme eri palvelukategoriaa, jotka edelleen sisälsivät omia alakategorioitaan. Sivun sisältö jaettiin kolmeen osaan ja kukin näistä edelleen osiin, joiden määrä vaihteli. Kuvassa 7 ”Alue”-hahmolakiin nojaten aluksi sivun kolme palvelumuotoa ladottiin allekkain. Keskimmäiselle alueelle määriteltiin kuva, jonka merkitys oli piirtää visuaalisesti selkeät rajat kullekin palvelukategorialle. Kategoriat saatiin täten jäsenneltyä niin, että käyttäjä löytää haluamansa kohteen tehokkaammin.

Kolmen alueen päälle saatiin edelleen ladottua pienempiä alueita. Nämä pienet alueet esittivät itse palveluita eli kohteita, mihin katsojan huomio tahdottiin suunnata. Ihmisen näköaisti luonnostaan poimii näkökentästä niitä partikkeleita, jotka näyttävät olevan silmää lähempänä. [10] Täten on tärkeää luoda illuusio, jolla halutut kohteet saadaan ikään kuin nousemaan taustastaan.

Valkoinen väri sopeutuu tyypillisesti hyvin taustaväriksi. Se ei kuulu tummiin, lämpimiin tai puhtaisiin väreihin, jotka aiheuttavat väistämättä syvyysvaikutuksen ja näyttävät olevan

lähempänä katsojaa. Eri palveluita esittävien alueiden taustaväriksi oli valittava tummempi väri ja tätä valitessa tuli olla tarkkana.



(a) Vaihtoehto 1 (b) Vaihtoehto 2

Kuva 8: Kaksi ei sopivaa väriasetelmaa

Jos väri oli vain asteen verran tummempi, niin tulos ei ollut kaunis, ja kohteen tarkastelu oli epämiellyttävää. Katsojan on vaikeaa keskittyä kohteeseen, jonka taustaväri tuskin erottuu ympäröivästä taustasta. Toisaalta, jos väri oli paljon tummempi ja lämpimämpi kuin tausta, olisi se hankaloittanut alueen sisällön lukemista. Myös kuvassa 8b on otettava huomioon, että liian räikeät värit voivat tehdä sivusta todella epämiellyttävän katseltavan. Liian usean elementin välinen kamppailu estää katselijaa keskittymästä mihinkään.

Kuvan 8 esimerkeistä valittiin vasen mutta tosin pienellä ja merkittävällä yksityiskohdalla: Kohteen merkkaavalle elementille asetettiin ”border”-niminen tyyli ja sen arvoksi 1 pikseli. Jo näin yksinkertaisella menetelmällä saadaan tulos, joka auttaa katsojaa keskittymään kohteen sisältöön. Tumman reunuksen luoma illuusio myös nostaa kohdetta hieman taustasta, mikä helpottaa myös katselua. Itseasiassa tumman reunuksen kirjoittamista ei tarvinnut erikseen tehdä, sillä kohteet päätettiin merkitä Bootstrapin valmiilla ”thumbnail”-komponenteilla. Elementin sisään pystytään helposti määrittelemään kuvia, otsikoita kuin tekstiäkin.

3.7.4 Modaalit

Kun sivun tahdotaan sisältävän paljon informaatiota kuten ”Services”-sivulla kävi, niin eräs suosittu tapa vähentää näkyvän informaation määrä ovat modaalit. Näihin laatikkomaisiin elementteihin törmää netissä ja älypuhelinsovelluksissa usein ja niihin liittyy aina tapahtuma, joka nostaa ne nettisivun päälle.

Kuva 9: Valittu väriasetelma

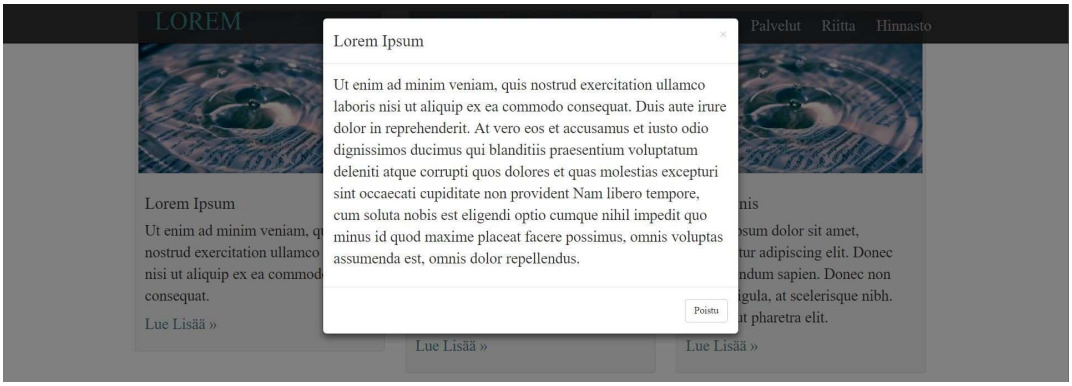
Tämän ansiosta verkkosivustoa ei välttämättä tarvitse suunnitella niin, että jokaista pitempää sisältöä kantaisi erillinen sivu. Modaaleilla voidaan nopeasti näyttää informaatiota käyttäjille ilman, että heidän tarvitsee hyppiä eri sivujen välillä. Tämä parantaa sivuston käytettävyyttä ja samalla myös kuormittaa sitä vähemmän, sillä turhilta sivun uudelleenlatauksilta vältytään. Modaaleilla olikin työssä suuri vaikutus sivun informaatioarkkitehtuurin rakenteeseen.

```
85      <div class="col-lg-12 col-sm-6">
86          <div class="caption">
87              <h3>Services Hoidot</h3>
88              <p> Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.</p>
89
90              <a class="read-more" data-toggle="modal" data-target="#exampleModal">Lue Lisää &raquo;</a>
91
92          </div>
93      </div>
94  </div>
95  <div class="modal fade" id="exampleModal" role="dialog">
96      <div class="modal-dialog">
97          <div class="modal-content">
98              <div class="modal-header">
99                  <button type="button" class="close" data-dismiss="modal">&times;</button>
100                 <h3 class="modal-title">Shindo Hoidot</h3>
101             </div>
102             <div class="modal-body">
103                 <p>Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.
104                     Duis aute irure dolor in reprehenderit.At vero eos et accusamus et iusto odio dignissimos ducimus qui
105                     blanditiis praesentium voluptatum deleniti atque corrupti quos dolores et quas molestias excepturi sint
106                     occaecati cupiditate non provident
107                 </p>
```

Kuva 10: Modaaliin liittyvä koodi

Edellisessä luvussa kerrottiin, kuinka eri palvelut esiteltiin Bootstrapin ”Thumbnail”-komponenteilla. Näiden sisään asetettiin kuvan, otsikon ja tekstisisällön lisäksi linkki ”Lue Lisää”, joka määriteltiin ohjaamaan käyttäjä tahdottuun modaaliin. Toimivien modaalien Kuvan 10 rivillä 90 luotava linkki sisältää attribuutit ”data-toggle” ja ”data-target”, jonka arvoksi asetetaan rivllä 95 määriteltävän modaalin attribuutti ”id”-vastine. Ei sovi unohtaa, että ”data-target”-arvon eteen tulee asettaa ”#”.

Modaalien avulla oli mahdollista asettaa tekstisisältö niin, että kustakin palvelun tekstisisällöstä näytettäisiin sivulla vain pari ensimmäistä lausetta, ja koko sisältö tulisi esiin omassa modaalissaan linkin kautta. Kuvassa 11 esiintyvään modaaliin lisättiin myös tarvittavat elementit niistä poistumiseen. Mikään ei estä modaalien sisällön määrittämistä halutessaan täysin uusiksi.



Kuva 11: Linkin kautta ilmestynyt modaali

3.7.5 Puhelin- ja sähköpostiinkit käytettävyyden tukena

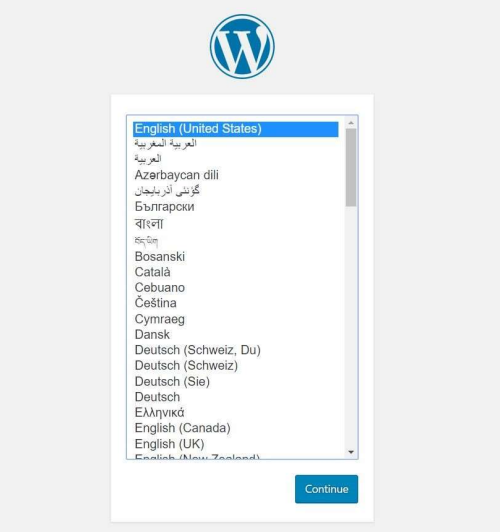
Yhteystiedot sivulle tuli luonnollisesti yrityksen osoitteen lisäksi sähköpostiosoite sekä puhelinnumero. Näitä varten on olemassa nykyisin kätevät linkkielementteihin liitettävät protokollat: puhelimelle ominainen on ”tel” ja sähköpostille taas ”mailto”.

Mikäli käytettävässä laitteessa on esimerkiksi puhelut mahdollistava ominaisuus, niin linkkiä painamalla käyttäjän on mahdollista valita haluamansa sovellus, jolla halutaan soittaa linkin sisältämään numeroon. Nämä ominaisuudet voivat tehostaa sivun käytettävyyttä jopa kaikilla laitteilla.

4 Staattisten sivujen kääntäminen dynaamisiksi WordPress-sivuiksi

Äskeisessä luvussa käsiteltiin erilaisia menetelmiä, joita modernin verkkosivun tekemisessä tulee ottaa huomioon. Oli mm. päätettävä sisällön asetelmista, värimaailmoista sekä responsiivisuuden toteuttamisesta. Kun sivusta oli staattisesti saatu valmiiksi, niin tuloksena oli visuaalisesti näyttävät sivut, joiden tekstit oltiin merkitty ”Lorem ipsum”-tekstein, jota yleisesti käytetään täytetekstinä suunniteltaessa sivuston ulkoasua. [14]

Kun sivusto oli saatu tähän pisteeseen, voitiin WordPressin liittäminen osaksi sivustoa aloittaa. Tämä luku käsittelee, kuinka WordPress asennetaan tietokantoineen ja miten voidaan helposti vähentää työn määrää valmiilla tiedostopaketeilla, jotka tarjoavat alustan kehitettävälle teemalle. Staattinen sivusto jätetään tässä vaiheessa hetkeksi hautumaan ja siirrytään käsittelemään WordPress-ohjelmiston asennusta omassa kehitysympäristössä.



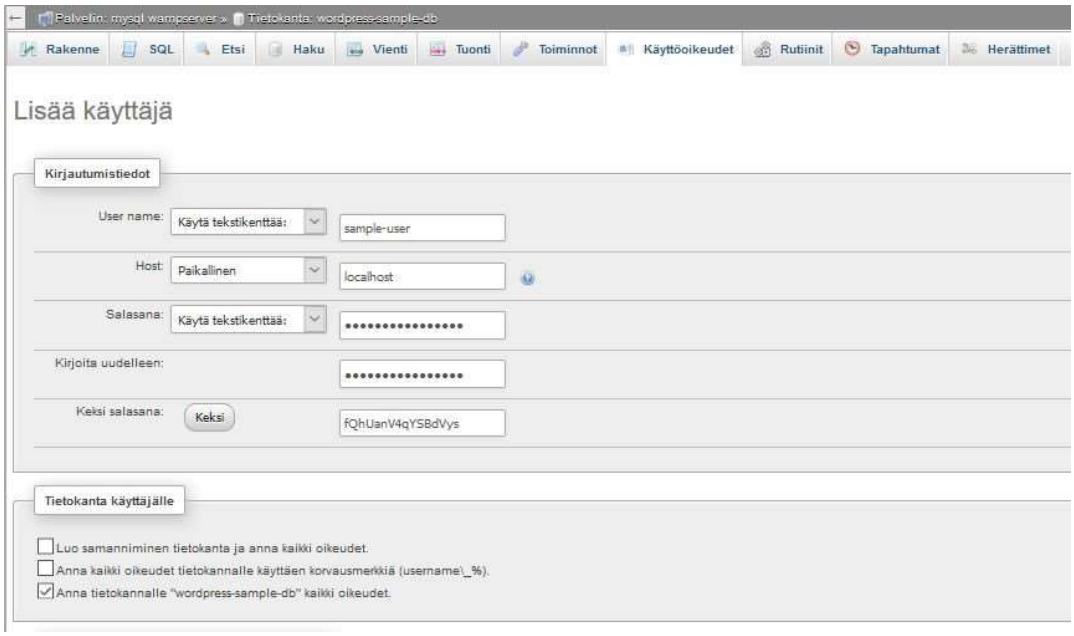
Kuva 12: Selaimeen ilmestynyt WordPress-asennusvalikko

4.1 WordPressin asentaminen

Aivan aluksi tulee ladata WordPressin asennuspaketti sen virallisilta kotisivuilta osoitteesta www.wordpress.org. Tämän dokumentin kirjoitushetkellä versio oli 4.8.1. Asennuspaketti sisältää joukon PHP -tiedostoja, mikä tarkoittaa, että tässä vaiheessa viimeistään tulee siirtyä asennetun WAMP:in käyttämiseen osana kehitystä. Tämä käy varsin helposti, sillä ladattu ja purettu tiedostopaketti tulee siirtää hakemistoon, jonka tulisi olla ../wamp/www, joka on juurihakemisto kaikille kehitettäville projekteille. Jos ladattu hakemisto nimetään ”WP-projekti”, niin tähän sisältöön pääsee käsiksi selaimen kautta kirjoittamalla hakukenttään ”localhost/WP-projekti”. Jos tämä ei onnistu, johtuu se usein siitä, ettei WAMP:ia ole käynnistetty. Tämän jälkeen, mentäessä WordPressin asennushakemistoon, tulisi selaimeen avautua kuvan 12 mukainen näkymä.

Tässä vaiheessa ennen kuin WordPress voidaan asentaa ja sen käyttäjäpaneeliin siirtyä, on projektia varten luotava oma nimetty tietokanta sekä sille käyttäjätunnus ja salasana. WAMP:ssa ja muissa vastaavissa ohjelmistokokonaisuuksissa tietokannan luominen onnistuu vaivattomasti siirtymällä selaimessa osoitteeseen localhost/phpmyadmin. Alustariippumaton PHPMyAdmin on osa WAMP:n ohjelmistopakettia ja sillä pystytään luomaan, hallitsemaan sekä tuhoamaan tietokantoja. Kuvassa esimerkkinä 13 luotuun tietokantaan tulee antaa haluttu käyttäjänimi sekä salasana. Tässä tapauksessa ”hostiksi” asetetaan localhost eli paikallinen palvelin. Salasanan voi

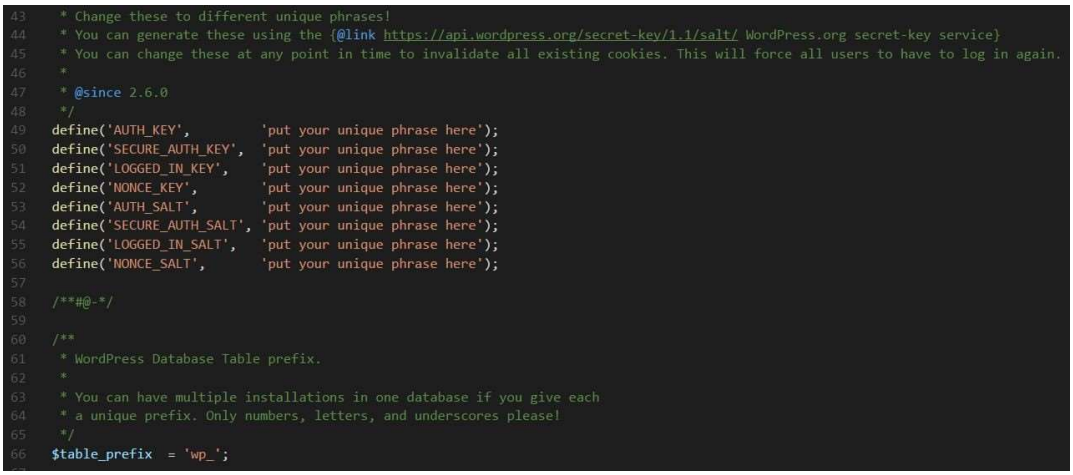
joko generoida tai keksiä itse. Se kuitenkin kannattaa merkitä muistiin jo tässä vaiheessa, sillä sitä tarvitsee myöhemmin WordPressin asetustiedoston muokkaamisessa.



Kuva 13: Käyttäjän luominen tietokantaan

Kun tämä on tehty, voidaan siirtyä ladattuun WordPress-hakemistoon ja valita sieltä ”wp-config-sample.php” halutulla editorilla. Tämä tiedosto sisältää joukon tärkeitä asetuksiin liittyviä määrityksiä, joihin tulee täyttää käsin aiemmin luotu tietokannan nimi sekä ohessa tehdyt käyttäjätunnus ja salasana. Se sisältää myös täytettävät määritteet tietoturvaa lisääville avaimille ja suolille. 14 Näiden on tarkoitus olla ainutlaatuisia ja vaikeasti arvattavissa. [15.] Siksipä niitä varten on tiedostossa valmis linkki, joka ohjaa avaimia ja suolia automaattisesti generoivalle sivulle.

Kuvan 14 pohjalla näkyy vielä yksi asia, joka on syytä täyttää, nimittäin ”table_prefix”. Tämän muuttujan arvo vaikuttaa WordPressin käyttämän tietokannan taulujen nimiin: WordPressin asennusvaiheessa tietokantaan luodaan joukko tauluja, ja prefixi näkyy näiden nimien etuliitteenä. Prefixin arvo on kannattavaa vaihtaa sen valmiiksi asennetusta arvosta johonkin vaikeasti arvattavaan, sillä jos muuttujaan ei kiinnitä huomiota, avaa se hakkereille takaoven välittää SQL-injektioita ja roskapostia.



```
 * Change these to different unique phrases!
 * You can generate these using the {@link https://api.wordpress.org/secret-key/1.1/salt/ WordPress.org secret-key service}
 * You can change these at any point in time to invalidate all existing cookies. This will force all users to have to log in again.
 *
 * @since 2.6.0
 */
define('AUTH_KEY',         'put your unique phrase here');
define('SECURE_AUTH_KEY',  'put your unique phrase here');
define('LOGGED_IN_KEY',    'put your unique phrase here');
define('NONCE_KEY',        'put your unique phrase here');
define('AUTH_SALT',        'put your unique phrase here');
define('SECURE_AUTH_SALT', 'put your unique phrase here');
define('LOGGED_IN_SALT',   'put your unique phrase here');
define('NONCE_SALT',       'put your unique phrase here');

/**#@-*/

/**
 * WordPress Database Table prefix.
 *
 * You can have multiple installations in one database if you give each
 * a unique prefix. Only numbers, letters, and underscores please!
 */
$table_prefix  = 'wp_';
```

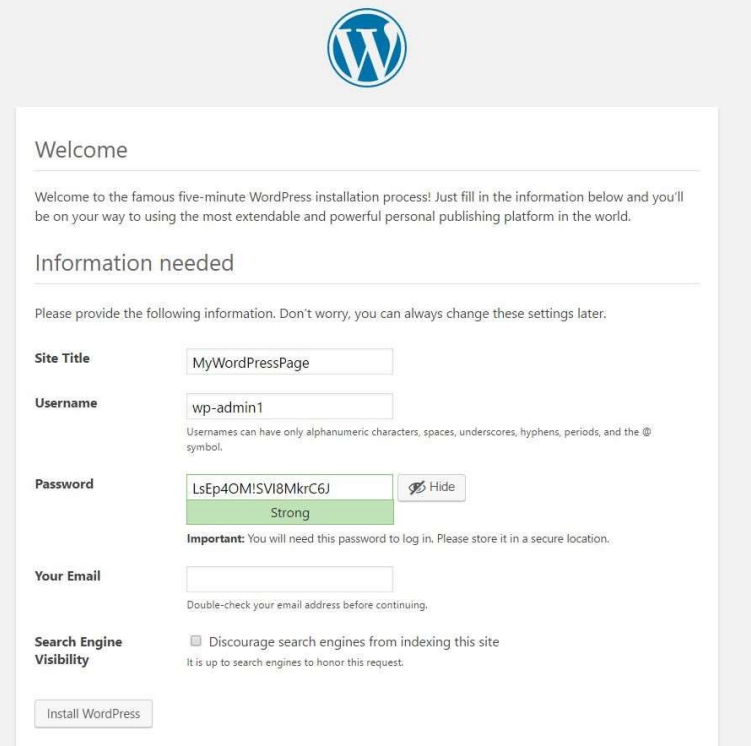
Kuva 14: Konfiguraation suolat ja avaimet

Kun halutut luotua tietokantaa vastaavat muutokset sekä tieturvaa lisäävät määritteet on tehty, niin tulee tiedosto tallettaa. Lisäksi, jotta tämä tiedosto ja tietokanta pystyvät kommunikoimaan tulee ”wp-config-sample.php” nimetä uudelleen nimellä ”wp-config.php”.

Nyt siirryttäessä WordPressin asennushakemistoon tulisi syntyä kuvan 15 mukainen näkymä. Tässä vaiheessa tehtävää sivua varten määritellään nimi, käyttäjätunnus ja salasana sekä myös sähköpostiosoite, johon tulee lähinnä ilmoituksia WordPressin tuoreimman version uusista ominaisuuksista. Käyttäjätunnus-salasana-paria ei tule sekoittaa aiempaan vastaavaan, joka tehtiin tietokannan luomisen yhteydessä. Nyt luotavia tunnuksia tullaan käyttämään, kun WordPressin käyttäjäpaneeliin halutaan kirjautua ja tehdä muutoksia. Kaikkia syötettävät tiedot ovat muokattavissa myöhemmin WordPressin hallintapaneelissa.

Paikallisella palvelimelle nyt luotua sivustoa pystyy tarkastelemaan myös ilman kirjautumista. Kirjautuminen hoidetaan kirjoittamalla selaimen hakuun localhost/”WordPressprojektin nimi”/wp-admin, joka aukaisee selaimeen kirjautumisnäkymän.

Tässä vaiheessa sivusto ei ulkoisesti näyttänyt kovin kummoisilta, sillä front-end näyttää jonkin WordPressin valmiista esimerkki teemoista ja tietokantaan ei ole WordPressin rungon lisäksi tallennettu vielä erityisempiä tietoja. Nyt kuitenkin valmiina oli sekä staat-



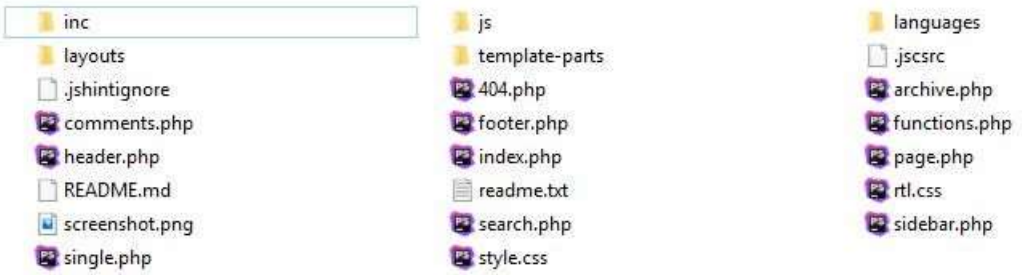
Kuva 15: WordPressin asentamisen viimeistely

tiset sivut että WordPress-alusta niitä varten. Staattisten HTML- ja CSS-koodista koostuvien sivujen siirtäminen ei kuitenkaan onnistu pudottamalla tiedostoja suoraan osaksi WordPress-projektia. Koska WordPress perustuu PHP-ohjelmistokieleen, on staattinen koodi muutettava dynaamiseksi. Tähän on olemassa varmasti monia eri tapoja toimia, mutta nyt päädyttiin käyttämään vapaaseen lähdekoodiin perustuvaa ”Underscoresia”.

4.2 Underscores-teeman lisääminen

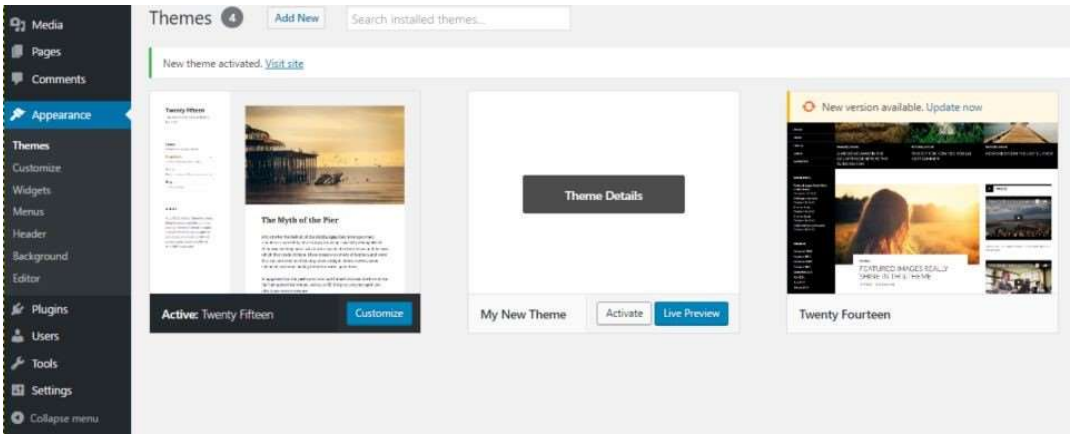
Kuten kappaleessa ”WordPressin teoriaa” mainittiin, WordPress-sivujen front-end määräytyy valitun teeman mukaan. WordPressin asentamisen jälkeen lisättiin projektikansioon ”Underscores”-tiedostopaketti. Vaikka Underscores on käytännössä teema missä muutkin, sen ei ole tarkoitus näyttää valmiilta, vaan se ainoastaan luo perustan kehitettävälle teemalle. Underscoresin sisältämät tiedostot ja hakemistot näkyvät kuvassa 16, se sisältää pakollisten index.php- ja styles.css-tiedostojen lisäksi pohjat mm. yläpalkille ja alapalkille. Lisäksi underscores sisältää runsaan määrän ominaisuuksia, joista osaa tullaan tarkastelemaan ja osaa ei. [16.]

Underscores asennettiin aluksi generoimalla tiedostopaketti teeman omilta kotisivuilta osoitteesta ”https://underscores.me/” ja lisäämällä tiedostopaketti hakemistoon /wp-content/themes.



Kuva 16: Underscores-teeman rakenne alussa

Kun tämä on tehty, voidaan generoitu Underscores-teema ottaa käyttöön WordPressin hallintapaneelissa, jolloin näkymän tulisi muuttua Underscores-teeman mukaiseksi. Generoidun teeman nimi määritellään generoinnin yhteydessä kyseisen teeman kotisivuilla. Kuvassa 17 ollaan hallintapaneelissa näkymässä, jossa valitaan teema.



Kuva 17: Halutun teeman aktivointi

Tässä vaiheessa näkymä näytti vieläkin tylymmältä. Tämä oli kuitenkin odotettua, sillä toisin kuin WordPressin alunperin tarjoama valmis teema, Underscores ei sisällä alussa juurikaan CSS-tyylimääreitä, jotka vaikuttaisivat front-endissä oleviin elementteihin.

4.3 Ohjelmointivaihe

Kun oli asennettu WordPress sekä lisätty siihen valmis pohja teemaa varten, voitiin siirtyä itse koodaamiseen, jossa toimittiin siirtämällä luotujen staattisten sivujen eri paloja osaksi dynaamiseksi saatettavaa sivustoa.

4.3.1 Dynaaminen navigaatio

Aivan aluksi lähdettiin työstämään navigaation saattamista WordPress-mallin mukaiseksi. Aluksi siirrettiin kullekin sivulle kuuluva header-elementin koodi underscoresin mukana tulleeseen header.php-tiedostoon. Kuvassa 18 riveillä 52-59 oleva staattisen menun luova koodi korvattiin riveillä 43-49. ”wp-nav-menu-funktio” saa parametrinaan taulukon, joka sisältää avain-arvo-pareja, mitkä edelleen määrittävät, millaista navigaatiota ollaan luomassa. Funktion avulla luodaan siis vain tyhjä taulukko, jonka sisältö määritellään koodin sijaan WordPressin hallintapaneelin puolella.

```
43      <?php
44        wp_nav_menu( array(
45          'theme_location'    =>  'primary',
46          'container'         =>  'nav',
47          'container_class'   =>  'navbar-collapse collapse',
48          'menu_class'        =>  'nav navbar-nav navbar-right'
49        ) );
50      ?>
51
52      <nav class="navbar-collapse collapse">
53          <ul class="nav navbar-nav navbar-right" id="nav-head">
54              <li><a href="services.html">Palvelut</a></li>
55              <li><a href="about-me.html">Riitta</a></li>
56              <li><a href="prices.html">Hinnasto</a></li>
57              <li><a href="contact.html">Yhteystiedot</a></li>
58          </ul>
59      </nav>
```

Kuva 18: Halutun teeman aktivointi

Jotta navigaatio ja sen sisältö voidaan käytännössä asettaa muokattavaksi hallintapaneelissa, tulee navigaatio myös rekisteröidä sitä varten. Tähän underscores tarjoaa valmiiksi tehdyn

funktion, joka sijaitsi tiedostossa functions.php. Tähän funktioon voidaan tarvittaessa lisätä useampiakin navigaatioita, mikäli työstettävällä sivustolla olisi sellaisia. 19

```
46      register_nav_menus( array(
47        'primary' => esc_html__( 'Primary', 'mynewtheme' ),
48      ) );
49
```

Kuva 19: Halutun teeman aktivointi

Kun navigaation luontiin ja rekisteröintiin liittyvä koodi oli tehty onnistuneesti, niin hallintapaneelista voitiin luoda uusi navigaatiota vastaava menu eli valikko. Tähän valikkoon pystyttiin hallintapaneelista projektin teon aikana lisäämään linkit dynaamisiksi saatettavia sivuja varten.

4.3.2 Viittauksin uudelleenkalibrointi

Jo valmiiksi luodussa staattisessa sivustossa oli riippuvuuksia tyylitiedostoihin sekä javascript-kirjastoihin, head- ja footer-tagien sisällä. Nämä riippuvuudet siirrettiin osaksi WordPress-sivustoa kirjoittamalla ne header.php- ja footer.php-tiedostoihin. Kuvassa 20 olevat, staattisen sivuston riippuvuudet eivät riitä sellaisenaan, sillä polun tulee viitata Teeman juureen. Kirjoittamalla polun alkuun ”<?php bloginfo(’stylesheet’) ?>” saadaan polusta dynaaminen: riippumatta siitä, missä osoitteessa sivusto tulee toimimaan, niin teema tulee löytämään tarvitsemansa tiedostot. Luonnollisesti ne riippuvuudet, joiden lähdettä ei haeta lokaalisti vaan verkosta, voidaan jättää ennalleen.

```
<!-- All necessary style content links -->

<link href="assets/css/bootstrap.min.css" rel="stylesheet">

<link href="assets/css/styles.css" rel="stylesheet">

<link href='https://fonts.googleapis.com/css?family=Noto+Serif' rel='stylesheet' type='text/css'>
```

Kuva 20: Staattisen sivuston tiedostoriippuvuudet

Oma tyylitiedosto, johon määriteltiin kaikki kustomoitu CSS-koodi, voitiin joko siirtää sellaisenaan haluttuun polkuun ja viitata samaan tapaan kuin muihinkin tiedostoihin. Tässä tapauksessa paras tapa oli kuitenkin siirtää luotu tyylitiedoston sisältö Underscores-teemaan juuressa olevaan ”style.css”-tiedostoon.

4.3.3 Sivupohjien luominen

Staattisen sivuston kehityksessä luotiin kutakin sivua varten oma html-tiedosto. Nyt oli aika luoda kullekin sivulle oma ”template” eli sivupohja. WordPress-sivupohjat ovat palikoita, joista sivusto rakentuu. Jo aiemmin tässä luvussa käsitelty header ja footer ovat sivupohjia, joihin liittyy nimenomaan header.php ja footer.php. Näistä kumpikin halutaan tavallisesti nähdä jokaisella sivulla, ja ne implementoidaankin osaksi sisältöä WordPressin omin funktioin.

Itse sisältö, joka kullakin sivulla halutaan näytettävän, siirretään omiin php-tiedostoihinsa, jotka vastaavat sivupohjia. Tässä projektissa jokaiselle sivulle annettiin oma sivupohjansa, mutta WordPress mahdollistaa myös tietyn sivupohjan asettamisen usealle sivulle. Tämä voi olla tarpeellista etenkin suurempia sivustoja tehdessä, sillä kunkin sivupohjan räätälöinti vaatii aikaa.

Sivupohjien tehtävä on näyttää sivun dynaaminen sisältö, jota saattaa olla erilaiset julkaisut, kalenteritapahtumat, kuvat, videot jne. Kustakin sivupohjasta voidaan luoda uniikki kokonaisuus ja tämä nimenomaan mahdollistaa sen, että luodut staattiset sivut pystytään saattamaan dynaamisiksi.

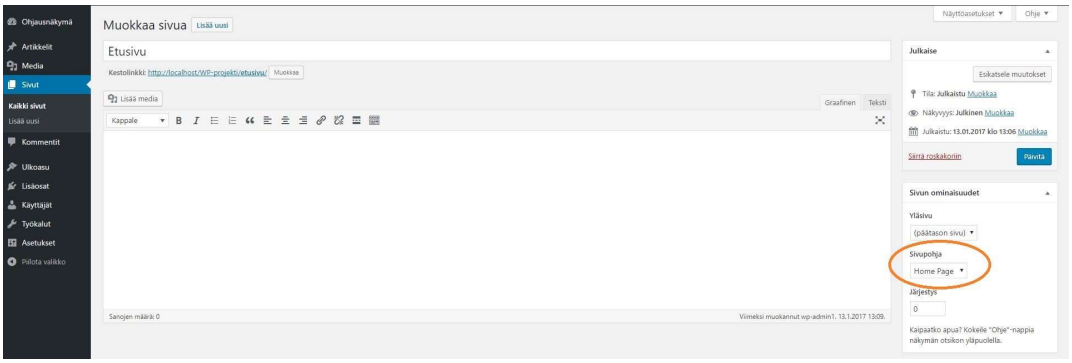
Kunkin sivupohjan tiedosto tulee nimetä ”page-”, jonka perään lisätään vapaasti päätettävä nimi. Tapa, millä sivupohjat saa yhdistettyä hallintapaneeliin, on lisätä sivupohjan nimi kuhunkin sivupohjaan liittyvään tiedoston koodisisältöön. Tämän ei tarvitse olla sama kuin itse tiedoston nimi.

Etusivua koskevan sivupohjan tiedosto nimettiin osuvasti ”page-home”, jonka jälkeen kirjoitettiin sille kuvassa 21 oleva koodi. Riveillä 8 ja 12 olevilla funktioilla lisätään headerja footer-sivupohjat ja näiden väliin voidaan kirjoittaa itse muu sisältö, minkä kirjoitettavan sivupohjan halutaan tarjottavan. Sama kuvan 21 koodipätkä toimi alustana kaikkia muitakin sivupohjia varten, poikkeuksena rivillä 5 esiintyvä sivupohjan nimi. [17.]

```
1
2   <?php
3
4   /*
5       Template Name: Home Page
6    */
7
8   get_header();
9
10
11
12  get_footer();
13
```

Kuva 21: Home Page -sivupohjan aloitusrunko

Kun sivupohjat oli nimetty, tulivat ne käytettäväksi hallintapaneelin puolelta. Tänne kunkin luodun sivun ominaisuuksiin tuli määrittää sille tahdottu sivupohja. Täten kaikki se sisältö, joka hallintapaneelin puolella asetetaan, tulee näkymään yhdistetyssä sivupohjassa.

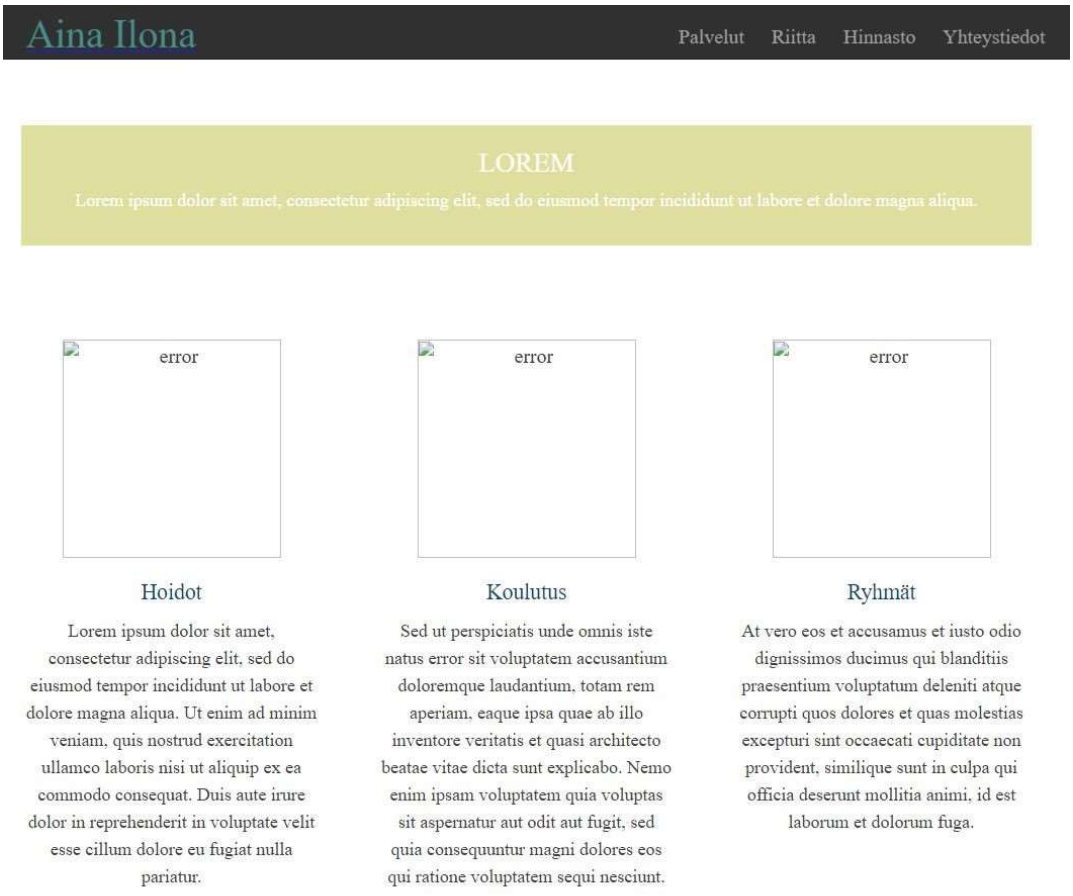


Kuva 22: Sivupohjan aktivointi

Jotta etusivun sivupohja haluttiin näyttää sivustolle tultaessa, tuli kuvassa 22 olevassa hallintapaneelissa tehdä pieni muutos, jonka kautta etusivun voi valita luoduista sivuista. Tavallisesti WordPress näyttää index.php-sivupohjan sisällön, joka tässä projektissa jätettiin tyhjäksi. Toki index.php:stä olisi voitu myös tehdä etusivun sivupohja, mutta esteettisyyden vuoksi pidettiin sivupohjien ja tiedostojen nimet mahdollisimman samanlaisina.

4.3.4 Kustomoitavat kentät

Nyt kun sivupohjat oli luotu, oli aika siirtyä itse sisällön kirjoittamiseen. Esimerkkinä otetaan käsittelyyn etusivu, jonka koko staattinen sisältö kirjoitettiin kuvan 21 funktioiden väliin lukuun ottamatta funktioiden tuomaa head- ja footer-tagien sisältöä. Tuloksena saatiin sivu, joka tyyleiltään näytti jo melko valmiilta, sillä sivupohjaan oli nyt lisätty sekä sivurakenteen luova HTML-rakenne että siihen liittyvä tyylitiedosto. Osa mainitusta sivusta on kuvassa 23.



Kuva 23: Etusivun toistaiseksi staattinen sivupohja

Kuvien puuttuminen johtui luonnollisesti siitä, että koodissa kuvien polut olivat rikkoutuneet aivan kuten muutkin tiedostoriippuvuudet. Polut saatettiin korjata samaan tapaan kuin aiemmin eli kirjoittamalla ”<?php bloginfo(’stylesheet’) ?>” polun alkuun. Tätä ei kuitenkaan tehty sillä kuvien ohella nyt vielä staattisista tekstisisällöistä haluttiin tehdä dynaamisia ja muokattavia.

Dynaamisuuden luominen tehtiin korvaamalla staattiset sisällöt yksi kerrallaan käyttämällä WordPressin funktiokutsuja. Kuvassa 24 on osa tekstisisällöstä korvattu muuttujilla, joihin on sijoitettu aiemmin arvo funktion ”get_post_meta” kutsulla, riveillä 17-24.

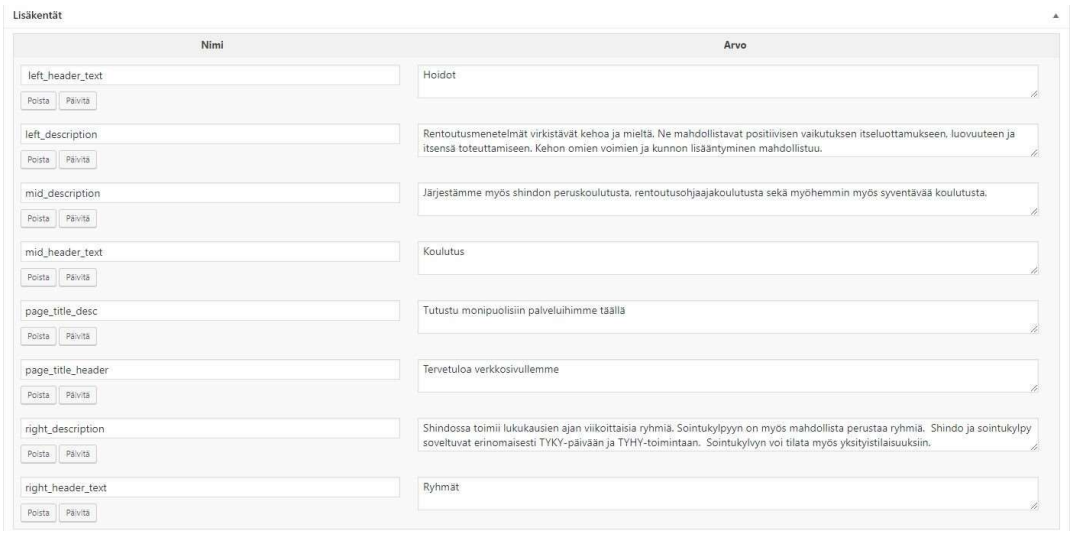
```php
<?php
    $page_title_header = get_post_meta(22, 'page_title_header', true);
    $page_title_desc = get_post_meta(22, 'page_title_desc', true);
    $left_header_text = get_post_meta(22, 'left_header_text', true);
    $left_description = get_post_meta(22, 'left_description', true);
    $mid_header_text = get_post_meta(22, 'mid_header_text', true);
    $mid_description = get_post_meta(22, 'mid_description', true);
    $right_header_text = get_post_meta(22, 'right_header_text', true);
    $right_description = get_post_meta(22, 'right_description', true);
?>
        <section class="hero">
            <div class="hero-background__home">
                <div class="hero-content">
                    <div class="container">
                        <div class="col-sm-12">
                            <div class="hero-text">
                                <h2 class="hero-title"><?php echo $page_title_header; ?></h2>
                                <p><?php echo $page_title_desc; ?></p>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
        </section>
        <section class="about">
            <div class="about-content">
                <div class="container">
                    <div class="row">
                        <div class="about-subject col-sm-4 col-xs-12">
                            <img class="subject__image" src="/assets/img/meditation.jpg" alt="error">
                            <h3 class="subject__header"><?php echo $left_header_text; ?></h3>
                            <p class="subject__description"><?php echo $left_description; ?></p>
                        </div>
                        <div class="about-subject col-sm-4 col-xs-12">
                            <img class="subject__image" src="assets/img/shi-gate.jpg" alt="error">
                            <h3 class="subject__header"><?php echo $mid_header_text; ?></h3>
                            <p class="subject__description"><?php echo $mid_description; ?></p>
                        </div>
                        <div class="about-subject col-sm-4 col-xs-12">
                            <img class="subject__image" src="assets/img/balance.jpg" alt="error">
                            <h3 class="subject__header"><?php echo $right_header_text; ?></h3>
                            <p class="subject__description"><?php echo $right_description; ?></p>
```

Kuva 24: Tiedon hakeminen ja implementointi koodissa funktioiden avulla

Funktioiden avulla tietokantaan voidaan tehdä kutsuja, joista halutut aineistot voidaan hakea. Funktio ”get_post_meta” sisältää 3 eri parametria. Ensimmäinen viittaa ainutkertaiseen tunnukseen, mikä liittyy kuhunkin luotavaan sivuun. Tämä on helposti nähtävissä, kun hallintapaneelista siirtyy ko. sivun kohdalle ja tarkistaa selaimen hakukentästä merkkijonon osan, jossa lukee avaimena post ja sen arvona haettu tunnus. Toinen ja merkityksellisin on avaimen nimi ja kolmas taas on boolean-tyyppinen muuttuja, joka tulee asettaa arvoon "true", kun kutsulla halutaan palauttaa vain avaimeen liittyvä arvo.

Koodin puolella tehdään siis tietokantakutsuja avaimilla, mutta missä ovat sitten avaimiin liittyvät arvot? Nämä löytyvät ja ovat määritettävissä hallintapaneelin puolella. Kuvassa 25 on etusivua varten luotu 8 eri lisäkenttää, joista kukin koostuu avaimen ja arvon muodostamasta

parista. Kuvan 25 vasemmalla puolella ovat avaimet, jotka esiintyivät siis myös aiemmassa kuvassa 24 ja oikealla puolella ovat taas niiden arvot, joihin voidaan dynaamisesti täyttää hallintapaneelissa halutut sisällöt.



Kuva 25: Hallintapaneelin lisäkenttien avain-arvo -parit

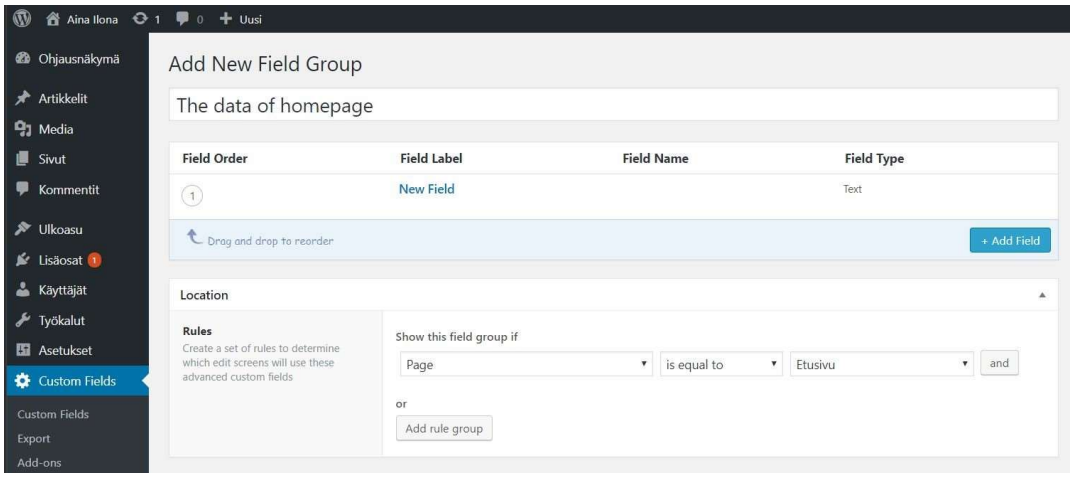
WordPressin tarjoamien lisäkenttien avulla voi sivuista vastaava taho siis halutessaan määrittää sivujensa sisältöjä uudelleen ja helposti, aluksi kirjautumalla hallintapaneeliin, ja täyttää, muokata tai tyhjentää valitsemiansa sisältöjä uusiksi. Perinteiset lisäkentät ovatkin oivallinen ominaisuus sivuille, joita jatkuvasti päivitetään, mutta ovatko ne lopulta kovinkaan käyttäjäystävällisiä? Jos tarkastellaan esimerkiksi kuvan 25 näkymää uudelleen. Voidaan havaita, että oikealla sijaitsevat kentät, joihin sisällöt laitetaan, ovat varsin suppeita. Myös kuvien käyttäminen arvoina on näillä työkaluilla hankalaa, sillä kuvien lataamisen jälkeen on arvona kenttään annettava kuvan sijainti url-tunnisteena, mikä jo tietojenkäsittelyn ammattilaisellekin on hidasta. Kolmantena erityisen merkittävänä seikkana on huomioitava, että jos käyttäjä täyttääkin vahingossa, jonkin vasemmalla olevan kentän uudelleen ja unohtaa alkuperäisen arvon. Tämä hyvin inhimillinen virhe johtaa suoraan siihen, etteivät avaimet hallintapaneelin ja koodin puolella enää vastaa toisiaan ja haluttua sisältöä ei päästä hakemaan tietokannasta selaimen näkymään. Paremman ja turvallisemman käyttäjäystävällisyyden varmistamiseksi WordPressin lisäosat ovat varteenotettava vaihtoehto.

4.3.5 WordPressin Lisäosat

WordPressiin liittyvien lisäosien kirjo on erittäin laaja. Jotkut sopivat erityisesti bloggaajille ja saattavat suodattaa esim. artikkeleihin liitettyjä kommentteja, kun taas toisilla voidaan saada käyttöön joitakin näyttäviä graafisia elementtejä, joiden toteuttaminen olisi itse tehtynä hankalaa ja aikaa vievää. Omia WordPress teemoja kehittäessä ja lisäosia valittaessa onkin tärkeää määrittää, mitkä ”pluginit” todella tarjoavat huomattavaa hyötyä ja varmistaa myös se, että lisäosa on luotettu ja suosittu. Laajassa käytössä olevat lisäosat ovat usein paitsi toimivia, niin myös turvallisempia. Lisäksi ongelmatilanteissa apua saa yhteisöltä tehokkaammin ja varmemmin verrattaessa tuntemattomampiin vastikkeisiin.

Yksi eräs suositummista lisäosista tunnetaan nimellä ”Advanced Custom Fields”, mikä oli mitä erinomaisin vaihtoehto korjamaan edellisen kappaleen lopussa eriteltyä ongelmaa. ”Advanced Custom Fields” tarjoaa helpot työkalut kehittäjälle saattaa hallintapaneelissa olevat näkymät käyttäjäystävällisempään muotoon. Tämä kyseenomainen lisäosa käsitellään seuraavaksi, sillä se on sekä mainio esimerkki lisäosien hyödyntämisestä että kuinka sivusta pystytään tekemään käyttäjäystävällisempi sivujen ylläpitäjälle.

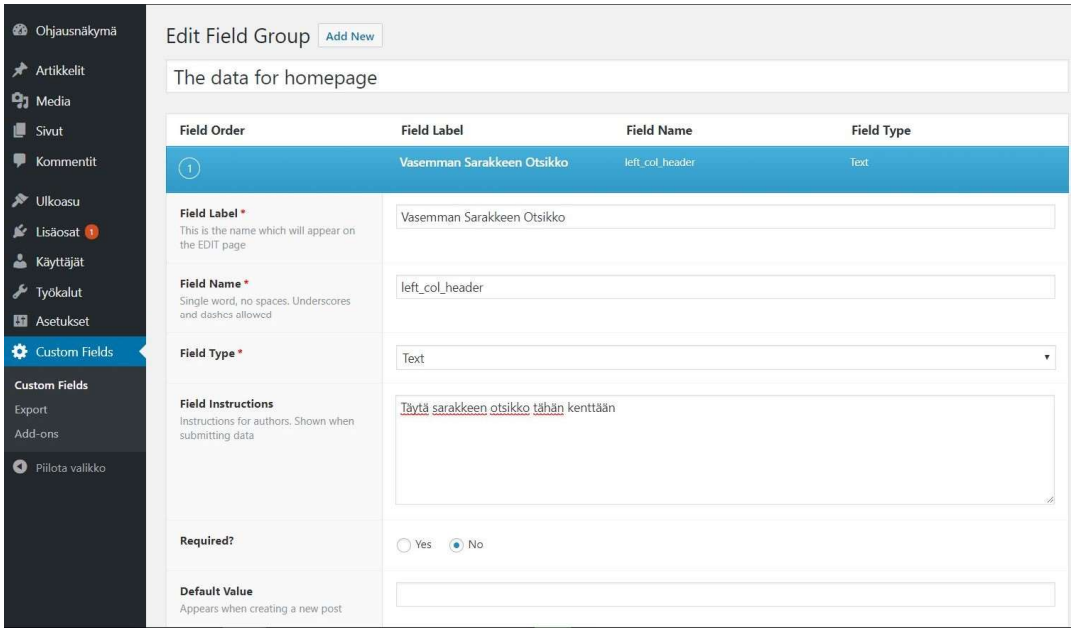
Lisäosan lataamisen jälkeen ilmestyy WordPressin hallintapaneelin vasempaan valikkoon kohta ”Custom Fields”, josta päästään luomaan uusia kenttäjoukkoja. Kuvassa 26 on luotu uusi kenttäjoukko, joka voidaan nimetä mielivaltaisella tavalla.



Kuva 26: Lisäkenttäjoukon nimeäminen ja säännön asettaminen

Tärkeämpänä kohtana kuvassa voidaan pitää ”location”-osiota, millä määritetään se että mitkä näkymät liittyvät luotavaan kenttäjoukkoon. Kun tähän asetetaan ”etusivu”, tulee luotava

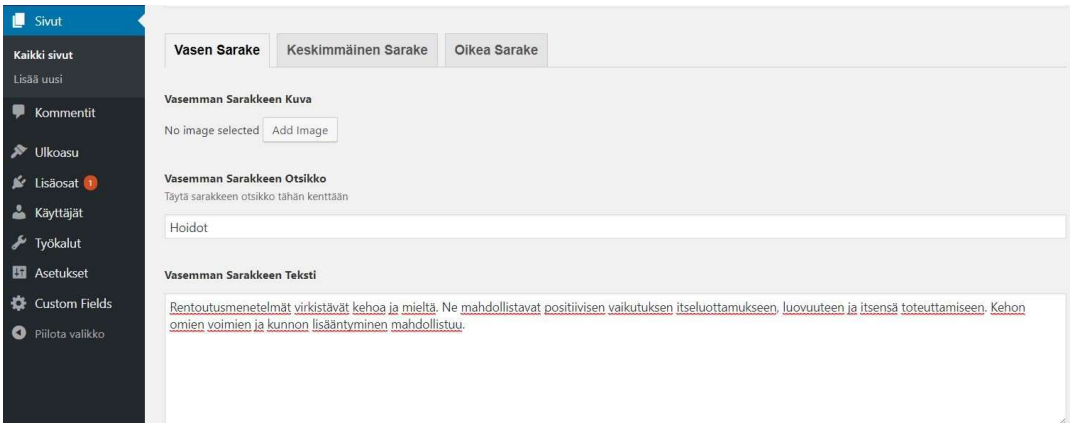
kenttäjoukko siirtymään hallintapaneelissa näkymään, jossa etusivun sisältö määritellään. Nyt voidaan alkaa valmistelemaan kenttäjoukon kenttiä yksi kerrallaan painamalla kuvassa 26 ympyröityä ”Add Field” -painiketta. Kenttää luotaessa kuvan 27 mukaisessa näkymässä voidaan monipuolisesti suunnitella, minkälainen kenttä tulee olemaan.



Kuva 27: Yksittäisen kentän ominaisuuksien määrittäminen

Kentälle valitaan mm. tunnus, avainta vastaava nimi sekä tyyppi, joka määrittää sen, millainen kenttä tulee olemaan. Se voi olla esimerkiksi perinteinen yhden rivin tekstikenttä, kuva tai suureen tekstisisältöön tarkoitettu WYSIWYG-editorilla varustettu kenttä. Kenttiä voitiin nyt luoda halutun sisällön määrän mukaisesti. Etusivulle oltiin aiemmassa kappaleessa yhdistetty ohjauspaneeli ja osa selaimessa olevasta näkymästä perinteisin lisäkentin. Nyt tämä tehtiin ”Advanced Custom Fields” -lisäosan avulla. [18.]

Kun halutut kentät oli luotu ja julkaistu oli hallintapaneelin etusivun muokkaus näkymään luotu kuvassa 28 esiintyvä osio.



Kuva 28: Sisällön asettelu Advanced Custom Fields -lisäosan luomassa näkymässä

Perinteisiin lisäkenttiin, jotka esiintyivät kuvassa 25, verrattuna ”Advanced Custom Fields” tarjoaa paljon selkeämmän ja käyttäjäystävällisemmän ratkaisun. Kuvasta 28 voidaan huomata, kuinka muokattava sisältö on vielä jaettu eri välilehtien alle. Täten voidaan todeta, että sivun hallinnan ei tarvitse vaikeutua merkittävästi riippumatta siitä, kuinka rikas sivun sisältö tulee olemaan. Sivun sisältö pystytään jakamaan välilehtien kesken, jolloin kentät eivät keräänny allekkain yhteen näkymään pitkäksi listaksi. Välilehdet voivat olla hyödyllisiä muillakin tavoin, sillä niillä sisältöä pystytään asettamaan sivun näkymän mukaisesti. Esimerkiksi, jos sivulla esiintyy vaakatasossa vierekkäin olevia elementtejä, niin ovat välilehtien ansiosta sisällöt pystytään asettamaan hallintapaneelin puolella ja selaimen näkymässä identtisemmin toisiinsa nähden.

Koodin puolella ”Advanced Custom Fields” toimii varsin samankaltaisesti kuin lisäkenttien kanssa. Lisäosa sisältää joukon funktioita, jotka saa käyttöön asennuksen yhteydessä. Kuvassa oleva koodiesimerkki 29 kutsutaan tietoa kannasta ”get_field” -funktioilla. Voidaan myös huomata, että lisäosan käyttäminen on myös kehittäjille helpompaa johtuen pakollisten parametrien pienemmästä määrästä.

```
24    $left_col_image = get_field('left_col_image');
25    $left_col_header = get_field('left_col_header ');
26    $left_col_text = get_field('left_col_text');
27    $mid_col_image = get_field('mid_col_image');
28    $mid_col_header = get_field('mid_col_header');
29    $mid_col_text = get_field('mid_col_text');
30    $right_col_image = get_field('right_col_image');
31    $right_col_header = get_field('right_col_header');
32    $right_col_text = get_field('right_col_text');
```

Kuva 29: Tietojen hakeminen Advanced Custom Fields -funktiokutsuin

Lisäosana ”Advanced Custom Fields” on erinomainen esimerkki siitä, kuinka sivuston hallinnan näkökulmasta käyttäjäystävällisyyttä voidaan ehostaa. Se ei myöskään missään nimessä sulje muiden lisäosien käyttöä osaksi verkkosivuprojektia, mutta jo pelkästään senkin avulla työn alla ollut verkkosivuprojekti olisi onnistunut.

5 Yhteenveto

Insinöörityön tarkoituksena oli käsitellä verkkosivukehitystä, sen nykyisistä mahdollisuuksista ja standardeista. Merkittävinä tekijöinä esiintyvät nykyisin mobiiliystävällisyys ja käyttäjälähestyttävä sisällönhallinta. Insinöörityössä havainnollistettiin eri kehityksen vaiheita useista perspektiiveistä kuten informaatioarkkitehtuurin, käyttäjäystävällisyyden sekä myös koodin kautta.

Työssä käytettiin erilaisia teknologioita kuten Bootstrapia ja WordPressiä. Raportti eteni lineaarisesti sivun suunnittelun ensiaskeleelta aina sisällönhallintajärjestelmän käyttöönottoon asti. Aluksi kuvattiin informaatioarkkitehtuurin mallia ja sen saattamista staattisiksi sivuiksi. Tästä siirryttiin työn teknisempään osaan eli WordPressin asentamiseen, käyttöönottoon sekä dynaamisuuden luomiseen. Lopuksi käsiteltiin vielä, miten WordPressin lisäosiakin voidaan käyttää parhaan mahdollisen käyttäjäkokemuksen saavuttamiseksi.

Työn tärkein tehtävä olikin valitun sisällönhallintajärjestelmän ohella käyttäjäystävällinen suunnittelu ja määrittely. Samalla todistettiin, että käyttäjäystävällisyyttä voidaan harjoittaa ajatellen sekä sivun sisällöstä vastaavaa tahoa että sivulla vierailevaa henkilöä. Kokonaisuudessaan tämä insinöörityö ohjastaa, millaisin menetelmin ja millaisin vaihein verkkosivuprojektin kehityksessä kannattaa edetä.

Lähteet

[1] CMS - Comparison; Verkkodokumentti. <http://websitesetup.org/cmscomparison-wordpress-vs-joomla-drupal/>. Luettu 9.5.2017.

[2] WordPress General Info; Verkkodokumentti. <https://en.wikipedia.org/wiki/ WordPress>. Luettu 10.5.2017.

[3] History of WordPress; Verkkodokumentti. <https://premium.wpmudev.org/blog/ wordpress-history/>. Luettu 18.11.2017.

[4] Old Security Leak in WordPress; Verkkodokumentti. <https://www.geek.com/ news/free-image-utility-renders-many-wordpress-websites-easy-hackingtargets-1408599/>. Luettu 18.11.2017.

[5] WordPress Security; Verkkodokumentti. <https://wordpress.org/about/ security/>. Luettu 18.11.2017.

[6] WordPress Requirements; Verkkodokumentti. <https://wordpress.org/about/ requirements/>. Luettu 10.5.2017.

[7] HTML Introduction; Verkkodokumentti. <https://developer.mozilla.org/en-US/ docs/Web/Guide/HTML/Introduction>. Luettu 14.5.2017.

[8] CSS Introduction; Verkkodokumentti. <https://developer.mozilla.org/en-US/ docs/Web/Guide/CSS/Getting_Started/What_is_CSS>. Luettu 14.5.2017.

[9] PHP Introduction; Verkkodokumentti. <http://www.thesitewizard.com/htmltutorial/what-is-html.shtml>. Luettu 14.5.2017.

[10] Sinkkonen, Kuoppala, Parkkinen, Vastamäki. 2006. Käytettävyyden Psykologia. Edita;.

[11] How to choose colors; Verkkodokumentti. <http://www.avangate.com/avangateresources/article/color-web-site.htm>. Luettu 6.6.2017.

[12] Common Navigation Mistakes; Verkkodokumentti. <https:// blog.kissmetrics.com/common-website-navigation-mistakes/>. Luettu 21.6.2016.

[13] IA Importance; Verkkodokumentti. <https://www.poutapilvi.fi/artikkelit/ mitä-on-informaatioarkkitehtuuri-ja-miksi-se-on-tärkeää >. Luettu 9.6.2017.

[14] What is Lorem Ipsum; Verkkodokumentti. <http://www.lipsum.com/>. Luettu 6.9.2017.

[15] Why WordPress Authentication Unique Keys and Salts Are Important; Verkkodokumentti. <https://codeseekah.com/2012/04/09/why-wordpress-authenticationunique-keys-and-salts-are-important/>. Luettu 2.9.2017.

[16] What are WordPress themes; Verkkodokumentti. <https://developer.wordpress.org/themes/getting-started/what-is-a-theme/>. Luettu 17.9.2017.

[17] Stepping Into Templates; Verkkodokumentti. <https://codex.wordpress.org/Stepping_Into_Templates>. Luettu 19.10.2017.

[18] Creating a field group; Verkkodokumentti. <https://www.advancedcustomfields.com/resources/creating-a-field-group/>. Luettu 27.10.2017.