



Santtu Taskinen

Verkkoblogin rakentaminen modernilla sisällönhallintajärjestelmällä

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

5.4.2021

Tiivistelmä

Tekijä:	SanTTu Taskinen
Otsikko:	Verkkoblogin rakentaminen modernilla sisällönhallintajärjestelmällä
Sivumäärä:	40 sivua
Aika:	5.4.2021
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Mediatekniikka
Ohjaaja:	Lehtori Ilkka Kylmäniemi

Insinööritöön tarkoituksena oli luoda moderni blogisivusto käyttäen SilverStripe 4 -konaisuutta, johon kuuluvat sisällönhallintajärjestelmä ja kehitysrunko, sekä vertailla sitä muihin suositumpiin sisällönhallintajärjestelmiin. SilverStripe ei ole yhtä helppokäyttöinen kuin Wordpress ja Squarespace, ja vaikka se on vähintään yhtä tehokas kuin Joomla, sen dokumentaatio ei ole yhtä laadukasta kuin Joomlailla tai Drupalilla.

Työn alussa SilverStripe-sivusto rakennettiin ensin paikalliseen XAMPP-pohjaiseen kehitysympäristöön, jossa tehtiin myös sivuston pääasiallinen suunnittelutyö tiiviissä yhteistyössä asiakkaan kanssa. Paikallisen kehityksen aikana sivuston värimaailma muuttui täysin asiakkaan alkuperäisestä suunnitelmasta. Lisäksi kehitysvaiheessa huomattiin suunnitellun blogitoiminnallisuuden soveltumattomuus halutun lopputuloksen saavuttamiseen, mikä johti vaihtoehtoisen blogitoiminnallisuuden etsintään ja löytöön.

Asiakkaan hyväksyttyä kehitetyn sivuston se siirrettiin asiakkaan palvelimelle, jolle täytyi asentaa valmiina ollutta Linuxia lukuun ottamatta täysi LAMP-ohjelmistopaketti (Linux, Apache, MariaDB, PHP). LAMP-paketin lisäksi palvelimelle asennettiin Composer-paketinhallintatyökalu, phpMyAdmin-tietokantatyökalu ja FTP-tiedostonsiirto-palvelin. Kun asiakkaan palvelimelle saatiin asennettua tarvittavat ohjelmat, siirrettiin rakennettu SilverStripe-sivusto palvelimelle.

Sivuston palvelimelle siirto pysähtyi kolmeen suurempaan ongelmaan. Ensiksi sivuston käyttämät lisäosat eivät suostuneet asentumaan kaikki yhdellä kertaa, mikä ratkesi asentamalla jokainen erikseen. Seuraava ongelma oli sivuston käyttämiin SCSS-tiedostoihin ja niistä käännettäviin CSS-tiedostoihin liittyvä kohdistusongelma, joka selvisi lisäämällä viittaus SCSS-tiedostoihin sivuston asetustiedostoihin. Lopuksi sivuston julkisiksi tarkoitetut hakemistojen oikeudet oli asetettu väärin, mikä täytyi käydä korjaamassa palvelimella etäyhteyden kautta.

Työn lopussa palvelimella oli täysin asiakkaan vaatimukset täyttävä moderni blogisivusto, jota asiakas voi jatkokehittää omatoimisesti.

Avainsanat: verkkokehitys, SilverStripe, sisällönhallintajärjestelmät

Abstract

Author:	SanTTu Taskinen
Title:	Building a weblog using a modern content management system
Number of Pages:	40 pages
Date:	5 April 2021
Degree:	Bachelor of Engineering
Degree Programme:	Information and communication technology
Professional Major:	Media Technology
Instructor:	Ilkka Kylmäniemi, Senior Lecturer

The goal of this project was to build a modern weblog for the client using the newest version of the SilverStripe content management system and compare it to some of the most popular CMSs.

During the project, the weblog was first built locally in a XAMPP development environment. Most of the design work was done during local development, including adjusting of colours and creating of the site's logo.

After local development was ready, the client's server was prepared for webhosting through the installation of a LAMP stack and an FTP-server, after which the weblog was moved to the server.

The site deployment ran into some difficulties. First the extensions used by the weblog could not be installed in bulk but had to be installed individually. Secondly the site installation could not find the CSS files compiled from the SCSS files used during development which was fixed by adding the appropriate config .yml files. Thirdly the permissions of the public folder of the SilverStripe installation had to be set manually.

During the development and deployment of the weblog SilverStripe was found to not be as easy to work with as Wordpress or services like Squarespace, and while at least on par with Joomla when it came to the power of the CMSs, SilverStripe's documentation leaves something to be desired.

Keywords: web development, SilverStripe, CMS

Sisällys

Lyhenteet

1	Johdanto	1
2	Sisällönhallintajärjestelmät	2
2.1	Wordpress-sisällönhallintajärjestelmä	3
2.2	Joomla-sisällönhallintajärjestelmä	6
2.3	Drupal-sisällönhallintajärjestelmä	10
2.4	Squarespace-verkkosivupalvelu	13
2.5	Wix.com-verkkosivupalvelu	15
2.6	Shopify-verkkokauppalvelu	16
3	SilverStripe-sisällönhallintajärjestelmä	17
3.1	SilverStripe yleisesti	17
3.2	Ominaisuudet	19
3.3	Teemat	20
3.4	Sivupohjat	21
3.5	Sisällönhallinta	23
3.6	Tietoturva	24
4	Sivuston toteutus	27
4.1	Suunnittelu	27
4.2	Sivuston paikallinen kehitys	29
4.3	Palvelimen valmistelu	33
4.4	SilverStripe-sivun asennus palvelimelle	35
5	Yhteenveto	39
	Lähteet	41

Lyhenteet

BSD:	Berkley Software Distribution License. Eräs käytetyimmistä avoimen lähdekoodin lisensseistä. Antaa GPL-lisenssiä suurempia vapauksia.
CSS:	Tietokannan hallintajärjestelmä. Ohjelmisto tiedon tehokkaan hakemisen, säilyttämisen ja päivittämisen toteuttamiseksi.
FTP:	File Transfer Protocol. TCP-protokollaa käyttävä tiedostonsiirtomenetelmä kahden tietokoneen välillä.
FTPS:	FTP Secure. FTP-protokollan jatke, joka lisää FTP-tiedostonsiirtomenetelmään tuen TLS- ja SSL-pohjaiselle salaukselle.
GPL:	Berkley Software Distribution License. Eräs käytetyimmistä avoimen lähdekoodin lisensseistä. Antaa GPL-lisenssiä suurempia vapauksia.
HTML:	Hypertext Markup Language. Verkkosivujen rakentamiseen käytetty merkintäkieli.
MVC:	Model-View-Controller-arkkitehtuuri. Sovellusarkkitehtuurityyppi, jossa sovellus jaetaan nimen mukaisesti kolmeen osa-alueeseen.
PHP:	Hypertext Preprocessor. Tulkittava ohjelmointikieli, jota käytetään web-palvelinympäristöissä.
SaaS:	Software as a Service. Pilvessä sijaitseva ohjelmisto, jota tarjotaan palveluna.
Sass:	Syntactically Awesome Style Sheets. CSS-esikäsittelykieli, jolla on useampia käytössä olevia syntakseja. Käännetään CSS-kieleksi.

- SCSS: Sassy Cascading Style Sheets. Yksi SASS-esikäsittelykielelle olevista syntakseista, joka muistuttaa hyvin paljon CSS-kieltä.
- SSL: Secure Sockets Layer. Vanha salausprotokolla, jolla on mahdollista suojata tietoliikenne IP-verkoissa ja jonka TLS on korvannut.
- TLS: Transport Layer Security. Salausprotokolla, jolla on mahdollista suojata tietoliikenne IP-verkoissa.
- WYSIWYG: What You See Is What You Get. Käyttöliittymä, joka yrittää näyttää lopputulosta mahdollisimman tarkasti vastaavan näkymän jo muokausvaiheessa.

1 Johdanto

Ammattimaisten henkilökohtaisten verkkosivustojen luonti on helpottunut viimeisen vuosikymmenen aikana huomattavasti niiden luontiin kehitettyjen työkalujen paranemisen ansiosta. Tämä yhdistettynä kasvavaan digitalisoitumiseen luo yksityishenkilöille entistä suuremman tarpeen jonkinlaiselle verkkojalanjäljelle.

Tässä kasvavassa tarpeessa sisällönhallintajärjestelmät ovat kasvattaneet suosiotaan tasaisesti ja mahdollistavat valtaosan maailman verkkosivuista [1].

Eräs vähemmälle huomiolle jäänyt mutta tehokas sisällönhallintajärjestelmä on uusiseelantilainen SilverStripe, joka on ollut jopa Uuden-Seelannin hallinnon käytössä.

Työn tarkoituksena oli luoda asiakkaalle oma henkilökohtainen verkkosivusto, jolla tämä voisi pitää yllä blogia ja joka olisi tarpeeksi selkeä, että asiakas pystyy jatkokehittämään sitä tarpeen tullen itse. Työn alussa esitellään ja vertaillaan käytetyimpiä sisällönhallintajärjestelmiä, minkä jälkeen perehdytään tarkemmin itse verkkosivun tuotannossa käytettyyn SilverStripe-sisällönhallintajärjestelmään.

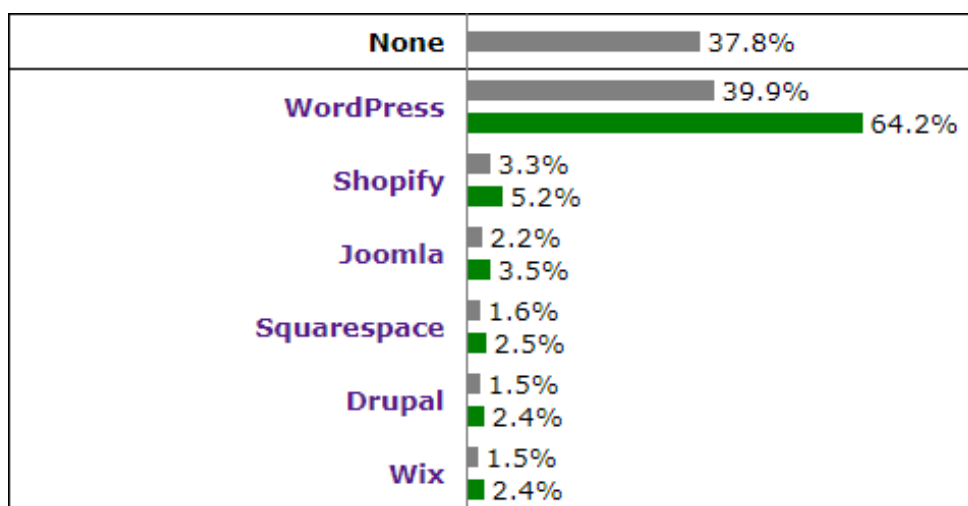
SilverStripen esittelyn jälkeen käydään läpi itse sivuston tuotanto, asiakkaan palvelimen valmistelu SilverStripe-sivustoa varten sekä sivuston vienti tälle palvelimelle.

Työn tarkoituksena on myös vertailla SilverStripea muihin suositumpiin sisällönhallintajärjestelmiin sekä laatia yksinkertainen opas SilverStripe-sivustojen suunnittelusta palvelimelle vientiin asti.

2 Sisällönhallintajärjestelmät

Sisällönhallintajärjestelmä on ohjelma tai ohjelmistokokonaisuus, joka mahdollistaa jonkin järjestelmän, kuten verkkosivuston, sisällön hallitsemisen ilman pääsyä itse järjestelmän lähdekoodiin tai tiedostoihin. Maailman verkkosivustoista 62,2 % käyttää jonkinlaista sisällönhallintajärjestelmää [2]. Sisällönhallintajärjestelmät, kuten hyvin suosittu WordPress, jota käyttää 39,9 % maailman verkkosivuista [3], ovat kasvattaneet suosiotaan 2000-luvun jälkimmäiseltä puoliskolta eteenpäin, ja niitä käyttävien sivujen määrä kasvaa päivittäin.

Torstaina 4.2.2021 käytetyimmät sisällönhallintajärjestelmät osuuksineen ovat nähtävissä kuvasta 1.



Kuva 1. W3techs.com-tilastoja sisällönhallintajärjestelmien osuuksista [4].

Sisällönhallintajärjestelmistä Wordpress, Joomla! ja Drupal ovat palvelimelle asennettavia kokonaisuuksia, joita käytettäessä verkkosivun kehittäjällä on mahdollisuus muuttaa mitä tahansa haluamaansa sivuston osaa. Tämän muokattavuuden kanssa kehittäjän vastuu verkkosivun laadusta on myös huomattavasti suurempi niin sanottuihin verkkosivukonesisällönhallintajärjestelmiin verrattuna. Niissä käyttäjälle annetaan valmiit työkalut, joita käyttäen käyttäjä voi muokata sivusta haluamansa näköisen ja laittaa sivulle haluamansa toiminnot niin kauan, kuin halutut toiminnot löytyvät valmiina sisällönhallintajärjestelmästä

tai sille tarjolla olevista lisäosista. Näihin jälkimmäisiin kuuluvat Shopify, Squarespace ja Wix. Verkkosivukonesisällönhallintajärjestelmät on usein kehitetty ajatellen henkilöitä, joilla ei ole taitoja, aikaa tai halua kehittää sivustojaan itse.

Suosituimmat itse isännöivät sisällönhallintajärjestelmät on rakennettu PHP-ohjelmointikielellä. PHP on verkkopalvelimille tarkoitettu ohjelmointikieli, jolla kirjoitetuista verkkosivuista näytetään käyttäjälle vain tulkinnan jälkeinen, PHP-tiedostosta tulkittu, HTML-koodi. Internetin käytöstä dataa keräävän ja jakelevan W3Techs.comin mukaan 79 % sen tutkimista verkkosivuista käyttää PHP:tä jossain muodossa [5], mikä tekee siitä maailman suosituimman palvelinkielen.

2.1 Wordpress-sisällönhallintajärjestelmä

WordPress on vuonna 2003 julkaistu sisällönhallintajärjestelmä, ja sen on maailman suosituin sisällönhallintajärjestelmä 64,2 %:n [6] osuudellaan sisällönhallintajärjestelmien markkinoista. Sen ovat kehittäneet Matt Mullenweg ja Mike Little, jotka päättivät rakentaa oman järjestelmänsä blogien ylläpitoa varten, kun heidän käyttämänsä B2/cafelog-järjestelmän kehitys loppui. Matt Mullenweg ilmaisi halunsa jatkaa B2/cafelog-järjestelmän kehitystä luomalla järjestelmästä oman versionsa, joka nimettiin WordPressiksi. Tämä oli mahdollista B2/cafelog-järjestelmän käyttämän GPL-lisenssin ansiosta. [7.]

GPL-lisenssi eli GNU General Public License on avoin ohjelmistolisenssi, jonka keskeinen idea on pitää avoin ohjelmisto avoimena. GPL-lisenssi antaa käyttäjille täysin avoimet kädet sen alla julkaistun ohjelmiston käytössä, niin kauan kuin käytöllä ei estetä GPL-lisenssin antamia oikeuksia muilta käyttäjiltä [8]. WordPress on myös GPL-lisenssin alaista ohjelmistoa, ja sen on luvattu pysyvän avoimena ja vapaana myös tulevaisuudessa, koska tämä ohjelmiston vapaus mahdollisti myös WordPressin syntymän [9].

Vaikka WordPress kehitettiin alkujaan blogien luontia ja ylläpitoa varten ja tällaisten sivujen luonti WordPressillä on yhä hyvin helppoa ja vaivatonta, nykyään

WordPressillä on mahdollista luoda minkälainen sivu tahansa. Oli kyseessä sitten verkkokauppa, tilaajille tarjottava tilaussivu tai Wikipedian tyyppinen tietosivu, WordPress ei estä toiminnallisuudellaan tai sen puutteella sivuston rakennusta. Tästä huolimatta WordPressin blogipohjaisuuden juuret näkyvät yhä selvästi WordPressin hallinnassa ja termistössä. Siinä missä vaikkapa SilverStripe käyttää termiä data-object eli dataobjekti, WordPressissä samanlaisen toiminnallisuuden sisältävä osa on nimeltään post eli posti. Postityyppejä on erilaisia ja omia postityyppejä on mahdollista luoda, mutta WordPressin perustoiminnallisuudelle tärkein postityyppi ja samalla selkein jääne WordPressin blogipohjaisuudesta on artikkeli.

WordPressin suuri lisäosien ja teemojen kirjo tekee mahdolliseksi sen, että henkilö, jolla ei ole ollenkaan ohjelmointitaitoja, voi luoda hyvin toiminnallisia ja visuaalisesti hienoja sivustoja. Tällä lisäosien ja teemojen kirjolla on kuitenkin myös negatiivinen puolensa. Koska melkein mihin tahansa ongelmaan on tarjolla sen ratkaiseva lisäosa ja monessa tapauksessa saman ongelman ratkaisevia lisäosia on useita erilaisia, täytyy ylläpitäjän mahdollisesti tehdä ylimääräistä työtä osataksaan valita juuri sopivan lisäosan, joka ei aiheuta ongelmia sivulle jo asennettujen lisäosien tai sivuston teeman kanssa.

Ajan kuluessa sivustolle asennettujen, tarkkaan valittujen lisäosien kehitys voi lakata monesta erinäisestä syystä ja lisäosan vanhetessa sitä käyttävän sivuston toiminnallisuus kärsii. Lisäksi melkein jokaisen lisäosan kehitys on täysin riippumatonta toisista lisäosista, mikä johtaa tilanteeseen, jossa lisäosaa vaihdettaessa vastaavaan täytyy pahimmassa tilanteessa koko sivun materiaali käydä läpi ja varmistaa sen toimivuus. Joskus lisäosan päivitys voi myös tuhota sivustolla olleet, lisäosasta riippuvaiset materiaalit ilman varoitusta tai tapaa palauttaa poistettua tietoa.

WordPress-sivuston visuaalisen ilmeen määrittää sivuston käyttämä teema. WordPressin käytön yleisyyden ansiosta kehittäjän ei yleensä tarvitse aloittaa aivan tyhjästä, vaan hän voi etsiä jo valmiin teeman, joka vastaa jo valmiiksi suurimmilta osin sitä, miltä valmiin verkkosivun halutaan näyttävän, ja muokata

siitä haluamansa. Nämä valmiiden teemojen muokkaukset suositellaan tekemään luomalla valmiista teemasta niin sanottu lapsiteema, jolloin pääteeman päivittyessä teemaan tehdyt muutokset eivät katoa [10].

Jos tarjolla olevista valmiista teemoista ei löydy sopivaa, on mahdollista kehittää sivustolle aivan oma teemansa. Tämä voidaan tehdä tyhjästä käyttäen hyväksi WordPressin laadukasta dokumentaatiota, ja WordPress tarjoaakin juuri teemojen kehittämiseen kohdistettua ohjekirjaa. Vaihtoehtoisesti oman teeman voi kehittää käyttäen hyväksi tarjolla olevia aloitusteemoja, kuten Astra, Genesis tai Underscores, joiden päälle kehittäjä voi luoda oman teemansa. Aloitusteemojen etu tyhjästä kehitettyihin teemoihin verrattuna on nopea aloitus, koska ne yleensä sisältävät kehitystä auttavia työkaluja, kuten Babel tai Gulp, eikä kehittäjällä kulu aikaa näiden työkalujen käyttöönottoon. Samalla ne ovat täysin yksityisiä ja itsenäisiä teemoja, joita käyttäjä voi muokata mielensä mukaan ilman pelkoa siitä, että pääteeman päivitys rikkoo jotakin sivustossa.

WordPressin teemoja ja lisäosia on mahdollista asentaa sivustolle suoraan WordPressin sisäisestä ohjauspaneelistä tai manuaalisesti lataamalla teeman tai lisäosan koodin sisältävä paketti ja purkamalla se palvelimelle WordPress-asennuksesta löytyviin teema- tai lisäosakansioihin. WordPress-lisäosia ja -teemoja on tarjolla sekä ilmaisia että maksullisia, ja useista suosituista kohteista on olemassa sekä ilmaiseksi käytettävä että maksullinen versio. Maksullisten versioiden lisäarvo luodaan yleensä joko ominaisuuksista, jotka lukitaan pois teeman ilmaisversiosta, tai tarjottavan tuen määrällä. Saadessaan suoraa tukea joutuvat ilmaiskäyttäjät yleensä tyytymään perustason tukeen.

Kooditasolla WordPress antaa tavan muokata mitä tahansa toiminnallisuutta suhteellisen helposti niin sanotuilla toimintakoukuilla, joilla oman koodin voi kiinnittää osaksi jo löytyvää funktiota tai määrittää suoritettavaksi aina tietyssä tilanteessa, kuten sivuston käynnistyessä tai kun sivusto hakee tiettyä tietoa, ilman että kehittäjän tarvitsee koskea ollenkaan WordPressin koodiin. Tämän ansiosta WordPress-teemat voivat erota toiminnallisuudeltaan huomattavasti toisistaan ilman erityisiä asennusohjeita.

WordPress käyttää PHP-ohjelmointikieltä. Virallisesti se tukee sitä versiosta 5.6.20 versioon 7.4 asti, ja uudemman PHP 8 -version tuki on aktiivisessa kehityksessä. WordPress 5.6 -version myötä WordPress tukeekin PHP 8:aa beeta-tasolla [11], mutta se on vielä kaukana täydestä yhteensopivuudesta. Tämän lisäksi lisäosien ja teemojen kehittäjien täytyy päivittää tuotoksensa PHP 8 -yhteensopiviksi, jotta sivusto voi kunnolla hyötyä PHP 8:n tuomista hyödyistä.

WordPressin suosion ansiosta internetistä löytyy paljon kehitystyökaluja, jotka on tarkoitettu juuri WordPress-sivustojen kehitykseen, vaikkapa VVV eli Varying Vagrant Vagrants, joka on valmis kokonaisuus HashiCorpin kehittämälle Vagrant-virtuaalikonehallintaohjelmalle. Se mahdollistaa WordPress-kehitysympäristöjen helpon pystytyksen ja onkin keskittynyt WordPress-kehitykseen.

Toinen WordPress-kehitystä varten kehitetty ohjelma on Local, jolla on mahdollista pystyttää WordPress-kehitysympäristö parilla napin painalluksella. Local myös mahdollistaa jo olemassa olevien WordPress-sivustojen helpon tuonnin ohjelman sisälle, mikä mahdollistaa helpon tavan jatkokehittää tuotannossa olevia verkkosivuja.

Wordpress vaatii toimiakseen verkkopalvelimen, jossa suositusten mukaisesti pitäisi olla PHP-versio 7.4 tai uudempi, MySQL 5.6 -tietokanta tai MariaDB 10.1 -tietokanta tai uudempi versio käytettävästä tietokannasta ja tuki HTTPS:lle.

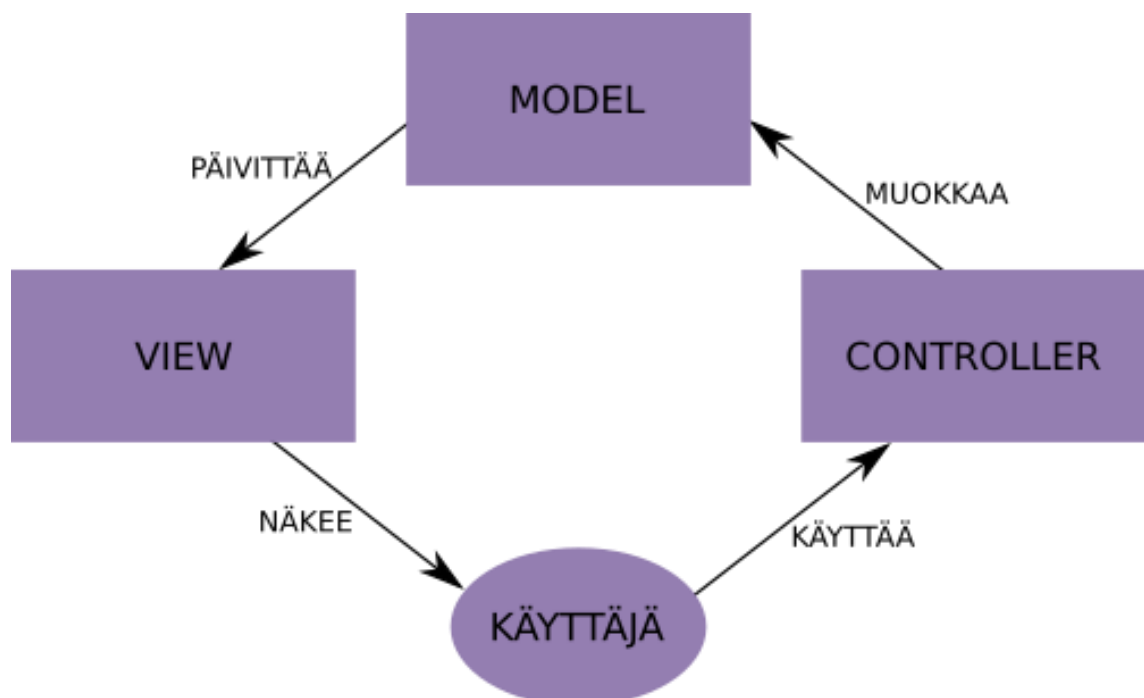
2.2 Joomla-sisällönhallintajärjestelmä

Joomla, myös Joomla!, on käyttäjäkunnaltaan kolmanneksi suurin sisällönhallintajärjestelmä. Sen suomenkielinen yhteisösivu, Joomla.fi, kuvailee sitä yhdeksi "maailman suosituimmista ohjelmapaketeista, jolla voidaan rakentaa, organisoida, hallita ja julkaista sisältöä nettisivustoilla, blogeissa, intraneteissa ja mobiilisovelluksissa." [12]. Joomla sai alkunsa aikaisemman Mambo-nimisen sisällönhallintajärjestelmän pohjalta, Mambo-projektin ylläpitäjien erimielisyyksien ja Mambo-nimeen liittyvän tekijänoikeuskiistan takia [13]. Suurin osa Mambon kehitystiimistä päätti irtisanoutua ja lähti kehittämään omaa versiotaan Mambosta,

jonka nimeksi valittiin ”kaikki yhdessä” tarkoittavan suahilinkielisen sanan jumla äänteellinen kirjoitusmuoto ”joomla”. Joomla’n ensimmäinen versio oli pääasiassa aikaisemman koodin uudelleen brändäys, eikä siinä ollut mitään suurempia muutoksia Mamboon verrattessa. Ensimmäiset suuret muutokset Mamboon verrattuna tulivat Joomlaan kolme vuotta myöhemmin julkaistussa 1.5-versiossa, joka toi sisällönhallintajärjestelmään joukon uusia ominaisuuksia ja parannuksia.

Kuten WordPress, myös Joomla on PHP:llä rakennettu, GLP-lisensoitu avoimen lähdekoodin ohjelmistokokonaisuus, mutta toisin kuin WordPress, Joomla käyttää MVC-arkkitehtuuria. Kuvasta 2 ovat nähtävissä osat, joihin MVC-arkkitehtuurissa sovellus jaetaan:

- malli (model), joka vastaa ohjelman tiedon tallennuksesta, ylläpidosta ja käsittelystä
- näkymä (view), joka kertoo piirrettävän käyttöliittymän ulkoasun ja sen, miten mallista saatava tieto näytetään käyttöliittymässä
- käsittelijä (controller), joka vastaa käyttäjän antamien käskyjen vastaanottamisesta ja käsittelystä sekä näkymän ja mallin muuttamisesta näiden perusteella



Kuva 2. MVC-sovellusarkkitehtuurin osat [14].

MVC-sovellusarkkitehtuuri mahdollistaa sovelluksen jokaisen osan erillisen muokkaamisen irrallaan muista osista. Esimerkiksi sovelluksessa, jossa on näkymä, johon listataan tietueita autoista ja näiden tietueiden pohjalta on mahdollista hakea autoja, muokattaessa tietokannasta löytyvää datamallia lisäämällä siihen uusi tietue ”Tuotannon lopetusvuosi” tulee tämä tietue näkyviin autoja listaavaan näkymään ja hakutoiminnon toteuttavan käsittelijän on myös mahdollista käyttää uutta tietuetta hakutoiminnassaan. Lisäksi mallin ja näkymän erotus toisistaan mahdollistaa sen, että sama data voidaan helposti esittää usealla eri tavalla. Tämä MVC-arkkitehtuurin tuoma joustavuus voi tehdä Joomla-sivustoista ketterästi kehitettäviä ja hyvin skaalautuvia. [15.]

MVC-arkkitehtuurin lisäksi eräs Joomla:n etu verrattuna moniin muihin sisällönhallintajärjestelmiin on sen vakioasennuksen vahva työkalukirjasto. Usea työkalu tai ominaisuus, joka pitää asentaa muihin sisällönhallintajärjestelmiin lisäosana, kuten kattava käyttäjähallinta, kieliversiointi tai laadukas hakutoiminto, löytyy Joomla:sta vakiona.

Laajojen vakio-ominaisuuksien lisäksi Joomlaan on kattavasti lisäosia ja sivustorunkoja, joskin pienemmän käyttäjäkunnan takia tarjontaa ei ole läheskään niin paljon kuin WordPressissä. Silti tarjolla olevista lisäosista löytyvät kaikki olennaiset erinäisistä sosiaalisen media integraatioista ja evästehallinnasta verkkokauppaan ja analytiikkaan. WordPressin tapaan Joomlaan on tarjolla sekä maksullisia että maksuttomia lisäosia. Pienempi tarjonta tekee myös lisäosien etsimisestä helpompaa mutta nostaa riskiä, että kaksi lisäosaa eivät toimi yhdessä eikä kummallekaan ole vaihtoehtoja lisäosaa, jolloin halutun ominaisuuden joutuu ohjelmoimaan itse.

Verkkokeskustelussa Joomla jää yleensä muiden sisällönhallintajärjestelmien jalkoihin, sillä sen sisältämät työkalut ja niiden antamat mahdollisuudet voivat tehdä siitä hieman pelottavan henkilöille, jolla ei ole ohjelmointi- tai tietotekniikkakokemusta, mikä saa heidät siirtymään WordPressin pariin, ja sen rakenteelliset valinnat tekevät siitä myös Drupalia raskaamman ja näin huonomman valin-

nan suuren dataliikenteen sivustoille. Koska markkinoilla on tarjolla erikoistuneempia sisällönhallintajärjestelmiä kuten verkkokauppojen tuottamiseen tarkoitettu Shopify, Joomla:n asema yleissisällönhallintajärjestelmänä henkilölle, joka tykkää puuhastella sivunsa kanssa, mutta haluaa samalla sivuston olevan helpposti hallittavissa ja vakaa hieman vaativammissakin tilanteissa, on hieman kyseenalainen.

Joomla on silti varteenotettava sisällönhallintajärjestelmä, ja sen vakiona sisältämät työkalut mahdollistavat sivustojen nopean rakentamisen ja päivityksen, kunhan sivun kehittäjä on valmis tutustumaan Joomla:n toimintaan hieman tarkemmin. Näin ominaisuuksien, kuten verkkokaupan, tuonti osaksi Joomla-sivustoa ilman sivuston uudelleenrakennusta on yllättävän helppoa.

Joomlatools Vagrant on VVV:n tyyppinen Vagrant-virtuaalikonepaketti, joka on tarkoitettu Joomla-sivustojen kehitystyötä varten. Sen on kehittänyt Joomla-kehitykseen erikoistunut yritys Joomlatools, joka jakaa sitä ilmaiseksi verkkosivuilleen. [16.]

Joomla vaatii toimiakseen verkkopalvelimen, jossa on ainakin PHP 5.3.10, joko MySQL 5.1 InnoDB -tuella tai SQL Server 10.50.1600.1 tai PostgreSQL 8.3.18 -tietokanta, ja Apache 2.0 -tai Nginx 1.0 -tai Microsoft IIS 7 -verkkopalvelinohjelmisto [17]. Nämä ovat kuitenkin vain minimivaatimukset, ja suositellut versiot kyseisistä ohjelmista ovat huomattavasti uudempia.

Joomla:n uusin vakaa versio on 3.9.24. Joomla:n kehittäjät kehittävät tällä hetkellä kahta uudempaa versiota Joomla:sta. Version 3.10 on tarkoitus olla eräänlainen lähtökohta kohti suurempaa 4.0 päivitystä, jotta varmistetaan sulava päivitys versio 3:sta versio 4:ään. Versio 4 tuo paljon uusia ominaisuuksia, kuten nopeamman asennusprosessin, uuden käyttöliittymän sekä AA-tason WCAG 2.1. -saavutettavuuden sivupohjiin. Lisäksi se nostaa palvelimella tarvittavan PHP-version versioon 7.2. [18.]

2.3 Drupal-sisällönhallintajärjestelmä

Drupal on vuonna 2000 keskustelupalstana alkunsa saanut sisällönhallintajärjestelmä. Sen rakensi kaksi belgialaisen Antwerpen yliopiston opiskelijaa, Dries Buytaert ja Hans Snijder, paikaksi, jossa he ja heidän tuttavansa voivat pitää yhteyttä toisiinsa. Vuotta myöhemmin nämä kaksi kehittäjää päättivät avata luomansa järjestelmän lähdekoodin, ja vähitellen Drupalista kasvoi täysimittainen ja tehokas sisällönhallintajärjestelmä. [19.]

Drupal on kehitetty ohjelmistokehittäjät mielessä, ja se on näin mahdollisuuksiltaan laajimpia mutta myös monimutkaisempia sisällönhallintajärjestelmiä. Drupal on Wordpressin ja Joomlaan tapaan ohjelmoitu PHP-ohjelmointikielellä ja julkaistu GPL-lisenssin alaisena. GPL-lisenssin ansiosta Drupalin käyttäjä voi muokata sovellusta ja rakentaa sovelluksen päälle ja pohjalta vapaasti, ja toisin kuin WordPress ja osittain myös Joomla, Drupal olettaa käyttäjällä olevan jo hieman kokemusta verkko- tai sovelluskehityksestä sekä hieman ohjelmointitaitoa. Drupal myös olettaa, että sitä käyttävä henkilö ei jätä asennustaan muokkaamatta vaan tekee sivuista omansa.

Tämän ansiosta Drupalin kattava yhteisö on täynnä hyvin ammattitaitoisia tekijöitä, jotka osaavat auttaa oli ongelma mikä tahansa. Samalla tämä kehittäjälähtökohtainen lähestymistapa antaa Drupalista hieman luotaantyyntävän ja hankalan sisällönhallintajärjestelmän tuntuman. Silti jos Drupalin ominaisuudet jaksa opetella, saa kehittäjä käyttöönsä vahvan sisällönhallintajärjestelmän, jolla voi rakentaa isoja, monimutkaisia ja silti hyvin toimivia verkkosivuja.

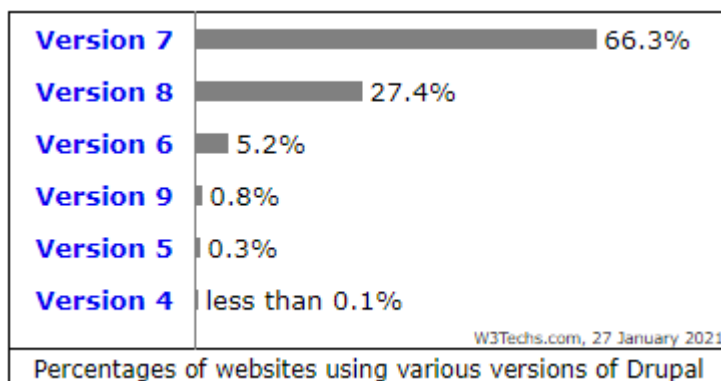
Muokattavuuden ja sitä arvostavan kehittäjäyhteisön tuoman tehokkuuden takia useat suuret yhtiöt ja organisaatiot, kuten NASA, Tesla, NBC, Greenpeace, Pinterest ja useat yliopistot käyttävät Drupalia verkkosivuillaan [20.] Drupal on taustalla myös useiden valtioiden ja kaupunkien virallisilla sivuilla. Näille organisaatioille sivuston pysyvyys käyttäjien saavutettavissa silloinkin, kun sivustoon kohdistuu paljon liikennettä, on hyvin tärkeää, ja Drupal on tässä asiassa myös hyvin ansioitunut. Näillä organisaatioilla on myös henkilökuntaa ja rahaa Drupal-

sivustojen ylläpitoon ja niistä vastuussa olevan henkilökunnan kouluttamiseen. Drupalin suurin heikkous onkin sen vaatima opettelu ja koulutus, joka hidastaa uuden toimijan työtä Drupalin kanssa työskennellessä, eikä se näin sovellu hyvin organisaatiolle, jolla ei ole resursseja tukemaan tätä ylimääräistä työtä verrattuna muihin enemmän ”avaimet käteen” -tyylin ratkaisuihin.

Drupalin luontainen muokattavuus syntyy sen rakenteesta. Drupalin eri osat on jaettu moduuleihin, jotka on mahdollista ottaa pois käytöstä sivuilla, joilla niitä ei tarvita. Drupaliin lisättäviä lisäosia kutsutaan myös moduuleiksi, eikä ohjelma erityisemmin erottele vakiomoduuleja ja erikseen asennettavia lisäosia toisistaan. Tällä on myös varjopuolensa. Koska Drupal lataa aina kaikki moduulinsa käyttötarpeesta riippumatta, Drupal voi hyvin vaatia sitä pyörittävältä palvelinkoneelta enemmän tehoa kuin Wordpress tai Joomla, mutta Drupalin suuret räätälöintimahdollisuudet antavat kehittäjälle vapaat kädet poistaa moduuleja käytöstä tai optimoida sivujaan muilla tavoilla.

Drupalin virallisilla lisäosakirjastosivuilla on tarjolla yli 45 000 erilaista lisäosaa [21], mikä on huomattavasti enemmän kuin Joomla:n hieman yli 6 000 lisäosaa [22] ja hyvä määrä myös verrattuna WordPressin yli 58 000 lisäosaan [23]. Toisaalta vain alle 5 000 tuosta 45 000 on yhteensopivia Drupalin uusimman versio 9 kanssa.

Kuvassa 3 näkyvistä käytössä olevista Drupal-versioista voidaan todeta tämän versio 9:n tukevien lisäosien pienen määrän johtuvan ainakin osittain version 9 pienestä käyttäjämäärästä verrattuna muihin versioihin.



Kuva 3. Eri Drupal-versioiden osuus koko Drupal-käyttäjäkunnasta [24].

Tämän taas voi nähdä johtuvan osittain siitä, että toisin kuin WordPress ja Joomla, joiden kehitystä ohjaa kummankin taustalla vaikuttava organisaatio, Drupalin kehitys on yhteisökeskeisempää, ja näin virallisen aktiivisen tuen alla ovat kaikki Drupal-versiot versiosta 7 eteenpäin. Iso virallisesti tuettujen versioiden määrä ei ohjaa lisäosien kehittäjiä päivittämään lisäosiaan yhteensopiviksi uudempien versioiden kanssa, sillä vanhemmilla, pidempään tarjolla olleilla versioilla on yleensä suurempi käyttäjäkunta ja niin kauan, kun tämä vanhempi versio on virallisesti tuettu, ja saa päivityksiä sekä tietoturvallisuuteen että muuten, ei tämä suurempi käyttäjäkunta siirry uuteen versioon.

Drupalista on tuettuna useampia numeroituja pääversioita samanaikaisesti, mikä tekee sen vaatimusten määrittämisestä hieman vaikeampaa. Kuitenkin Drupalia pyörittävältä palvelinkoneelta suositellaan löytyvän ainakin PHP 7.3, jos käytössä on Drupal versio 7 tai uudempi [25], jokin verkkopalvelinsovellus, kuten Apache 2.0 tai Nginx, ja sopiva tietokanta. Drupal tukee isompaa kirjoa erilaisia tietokantoja kuin WordPress tai Joomla. Se tukee jopa JSON-tiedostoihin pohjautuvaa MongoDB-tietokantaa, joskin suositeltu tietokantaratkaisu on MySQL 5.5.3/MariaDB 5.5.20/Percona Server 5.5.8 tai korkeampi, josta on InnoDB-tuki [26].

2.4 Squarespace-verkkosivupalvelu

Squarespace on amerikkalaisen Squarespace-yhtiön tarjoama SaaS-palvelu verkkosivujen luontia, ylläpitoa ja sisällönhallintaa varten. Sen rakensi Anthony Casalena alkujaan omaan käyttöönsä opiskellessaan Marylandin yliopistossa, ja siitä muovautui myytävä palvelu osana Marylandin yliopiston yrityshautomoa [27]. Se julkaistiin vuonna 2004 ja on kohonnut yhdeksi suurimmista sisällönhallintajärjestelmistä maailmassa 2,5 %:n osuudellaan sisällönhallintajärjestelmiä käyttävistä verkkosivuista.

Squarespace on saanut huomattavasti näkyvyyttä viimeisen vuosikymmenen aikana suurilla markkinointikampanjoillaan, jotka ovat kohdistuneet sekä niin sanottuun perinteiseen mediaan, ostamalla mainosaikaa suurimman amerikkalaisen jalkapallo -tapahtuman Super Bowlin mainostilasta vuosina 2014–2018, että viimeisen vuosikymmen aikana vaikutusvaltaansa kasvattaneeseen, verkko- ja sosiaalisen median vaikuttajiin sponsoroimalla näiden luomia videoita, kanavia ja brändejä.

Toisin kuin Wordpress, Drupal, Joomla ja työssä käytetty SilverStripe, jotka on mahdollista asentaa sivuston omalle palvelimelle, Squarespace-pohjaiset sivustot isännöidään aina Squarespace-yhtiön palvelimilla. Squarespace on maksullinen palvelu, jonka kuukausihinta tammikuussa 2021 oli paketista ja maksuvälistä riippuen 11–42 € [28.]

Halvin, 15 € kuukaudessa maksava, vaihtoehto sisältää teemoilla ja sivupohjilla rakennettavan verkkosivuston ja rajatun määrän sivuston ylläpitäjäoikeuksia, ja jos asiakas ostaa kerralla vuoden tilauksen, saa hän myös verkko-osoitteen ilman lisämaksua tilauksen ajaksi ja paketin hinta laskee 11 euroon kuukaudessa. Tätä pakettia kalliimmat vaihtoehdot sisältävät lisää ominaisuuksia, kuten verkkokauppatyökalut, oikeuden muokata sivustoa omilla CSS-säännöillä ja JavaScript-koodilla teemojen ja sivupohjien lisäksi sekä paremmat analytiikkatyökalut. [29.]

Squarespacea mainostetaan helppona tapana luoda kauniita ja ammattilaismaiselta vaikuttavia sivuja. Tämän luvataan onnistuvan Squarespacen tarjoamien ammattilaisten laatimien teemojen ja sivupohjien avulla. Näitä ”palkintoja voittaneita” sivupohjia on yli 100 erilaista, mikä on huomattavasti vähemmän kuin Squarespacen suurimmalla kilpailijalla WordPressillä. Tämän voi myös nähdä Squarespacen etuna, sillä siinä missä WordPressin valikoima johtuu sen suuren käyttäjäkunnan mahdollisuudesta jakaa omia tuotoksiaan helposti ja näiden tuotosten laatu vaihtelee huomattavasti, kaikki Squarespacen sivupohjat ovat talon sisällä suunniteltuja ja rakennettuja, mikä tekee niistä jokaisesta laadukkaan ja käytännöllisen. [30].

Eräs Squarespacen ongelmista on sillä rakennettujen sivustojen heikompi hakukoneoptimisaatio (HKO). HKO on tärkeää varsinkin toimialoilla, joissa on kova kilpailu ja paljon toimijoita, ja uusille yrityksille, jotta ne tulisivat huomatuksi [31; 32]. Maaliskuussa 2020 julkaistu [websitetooltester.com](https://www.websitetooltester.com/verkkosivun-artikkeli-luettelee-tavallisimpia-squarespace-sivujen-ongelmia-hakukoneoptimoinnin-nakokulmasta)-verkkosivun artikkeli luettelee tavallisimpia Squarespace-sivujen ongelmia hakukoneoptimoinnin näkökulmasta.

Artikkeli mainitsee esimerkiksi, miten tapa asettaa HKO:n kannalta tärkeitä asioita, kuten verkkosivuelementtien alt-attribuutteja, vaihtelee elementin mukaan, mikä tekee niiden asettamisesta sekavaa ja aikaa vievää. Lisäksi artikkeli luettelee Squarespace-sivupohjien vaihtelevat rakenteet ja asetukset, sivujen hitaus yleisesti ja huonot tekstinmuokkausvaihtoehdot, joiden takia sivun luojaan täytyy tehdä HKO:n kannalta huonoja valintoja saadakseen sivunsa näyttämään siltä hän haluaa, muina hakukoneoptimisaation kannalta ongelmallisina asioina. [33.]

Toinen ongelma syntyy erään Squarespacen vahvuuden kautta. Koska Squarespace on sisäisesti kehitetty suljettu kokonaisuus, on sen osien toiminta toisensa kanssa sulavaa ja vaivatonta. Tämä positiivinen ominaisuus kääntyy kuitenkin negatiiviseksi siinä vaiheessa, kun jotain ominaisuutta ei löydy valmiina tai se ei mahdollisesti toimi, niin kuin sivuston omistaja tai ylläpitäjä haluaisi. Squarespace tarjoaa mahdollisuuden oman koodin tuontiin osaksi sivustoa, ja näin Squarespace-palvelusta puuttuvia ominaisuuksia on mahdollista kehittää

osaksi Squarespace-sivustoa. Tämä vaatii kuitenkin kehittäjän palkkaamista ja näin toimii Squarespacen ”Palvelullamme kuka tahansa voi rakentaa kauniin ja toiminnallisen verkkosivun ilman teknistä osaamista” -mainostusta vastaan.

2.5 Wix.com-verkkosivupalvelu

Wix.com on israelilaisen samannimisen yhtiön tuottama ja tarjoama verkkosivun rakennus- ja isännöintipalvelu. Wix.com-yrityksen perustivat vuonna 2006 Avishai Abrahami, Nadav Abrahami ja Giora Kaplan. Liiketoimintaidea oli synnytt heidän rakentaessaan verkkosivua toiselle idealleen ja he huomasivat, kuinka työlästä verkkosivujen luonti itse asiassa on. Tämän oivaltaneena he alkoivat rakentaa palvelua, jolla kuka tahansa voisi rakentaa oman verkkosivunsa ilman ohjelmointi- tai suunnittelutaitoja. [34.]

Wix-pohjaiset sivut ja rakennustyökalut perustuivat pitkään Adobe Flash -teknoologiaan, mikä teki siitä huonon valinnan Adobe Flashiin liittyvien ongelmien takia. Adobe Flash oli tapa luoda interaktiivisia elementtejä ja tuoda videoita ja animaatioita verkkosivustoille, mikä teki siitä hyvin suosittu osan verkkosivuja ja -mediaa. Mutta Adobe Flash sisälsi myös paljon ongelmia. Se oli tehosyöppö, ja elinkaarensa aikana siitä löytyi useita suuria tietoturva-aukkoja. Näiden ongelmien takia, kun 2010-luvun alkupuolella HTML5-, CSS3- ja JavaScript-tekniikat alkoivat mahdollistaa asioita, kuten median toiston ja verkkoelementtien animoimisen, vanheni Adobe Flash nopeasti, mikä laski sitä käyttävien verkkosivujen suosiota. Tunnetusti Steve Jobs julkaisi vuonna 2010 kirjeen, jossa hän kritisoi Adobe Flashiä ja kertoi uskovansa HTML5:n ja sen kaltaisten avoimien teknologioiden syrjäyttävän sen [35.] Wix julkaisi HTML5-pohjaiset versiot työkaluistaan vuonna 2015, ja se oli ensimmäinen askel kohti Wixin irtautumista Flashistä.

Nykypäivänä Wix-sivustot perustuvat moderneihin HTML5-, CSS3- ja JavaScript-tekniikoihin. Sivut rakentuvat Squarespacen tapaan sivupohjista, mutta toisin kuin Squarespacessa, Wixin sivupohjat eivät ole täysin responsiivisia ja niitä täytyy hioa erikseen mobiilipuolella.

Kuten Squarespace, Wix on kuukausimaksullinen palvelu, jossa on eri maksutasoja, mutta toisin kuin Squarespace, Wix tarjoaa myös maksuttoman palvelun. Tämä maksuton palvelu ei rajoita käytettävissä olevia työkaluja, mutta sivustolla näytetään mainoksia ja sivuston osoite päättyy aina ".wixsite.com". Jos käyttäjä haluaa käyttää omaa verkko-osoitetta tai rakentaa verkkokaupan, täytyy tämän maksaa palvelusta.

Eräs toinen Wixin etu Squarespaceen verrattuna on sen ylläpitämä Wix App Market, jossa on tarjolla paljon lisäosia ja josta voi etsiä ja kiinnittää haluamiaan ominaisuuksia osaksi omaa sivuaan. Näin toisin kuin Squarespacen kohdalla, jossa jos ominaisuutta ei löydy palvelusta, Wixissä on mahdollista löytää haluttu ominaisuus sen lisäosamarketista.

2.6 Shopify-verkkokauppapalvelu

Shopify on kanadalainen yritys, joka tarjoaa samannimistä, verkkokauppojen tuottamiseen ja hallintaan tarkoitettua palvelua. Shopify-palveluun kuuluu myös sisällönhallintajärjestelmä, joka nousi maailman toiseksi suosituimmaksi sisällönhallintajärjestelmäksi toukokuussa 2020. W3techs kuitenkin muistuttaa, että heidän laskevat tähän mukaan kaikki verkkosivut, jotka käyttävät Shopifyta jossain sivustollaan eikä vain täysin Shopifylla tuotettuja sivuja. [36.]

Shopify syntyi sen luojaan tarpeesta pystyttää verkkokauppa lumilautojen myyntiä varten. Tobias Lütke ja Scott Lake eivät aluksi halunneet rakentaa omaa verkkokauppa-alustaansa, mutta päätyivät ratkaisuun todettuaan, että he eivät olleet tyytyväisiä jo tarjolla olleisiin ratkaisuihin. Tobias Lütken ja Scott Laken alkuperäinen suunnitelma oli vain luoda verkkokauppa mutta siitä kehittyi lopulta kokonainen sovellusala, jota tarjoavan yrityksen, Shopify, he ja Daniel Weinand perustivat vuonna 2006.

Toisin kuin WordPress, joka syntyi verkkoalustaksi blogien luontia ja ylläpitoa varten ja laajeni siitä mahdollistamaan kaikenlaisia verkkosivuja, Shopify on aina keskittynyt verkkokauppoihin. Tämän takia Shopify on oikein tehokas,

mutta myös erikoistunut työkalu. Shopify-sivulle voi käytetyistä teemasta ja lisäosista riippuen saada tavanomaisen verkkosivuston ominaisuuksia kuten blogitoiminnon, mutta Shopify-verkkosivustot ovat ensisijaisesti verkkokauppoja, ja näiden toimintojen räätälöinti ja hallinta on rajallisempaa kuin enemmän yleiskäyttöön tarkoitetuissa sisällönhallintajärjestelmissä, kuten WordPress tai Joomla, ja verkkosivukonepalveluissa, kuten Squarespace tai Wix.

Silti tämän Shopifyn selkeän käyttötarkoituksen ansiosta sen käytöstä on saatu selkeää ja helppoa, mikä mahdollistaa verkkokaupan rakentamisen viikonlopussa. Lisäosat, jotka tuovat verkkokauppaan lisäominaisuuksia, kuten tuen tuotearvosteluille, on helppo asentaa ja ottaa käyttöön Shopifyn hallinnasta ilman sivuston hetkellistä alasvetoa.

3 SilverStripe-sisällönhallintajärjestelmä

3.1 SilverStripe yleisesti

SilverStripe on uusiseelantilaisen SilverStripe Oy:n kehittämä sisällönhallintajärjestelmä, joka on käyttäjäkunnaltaan pieni, ja toimii alustana alle 0,1 %:ssa maailman verkkosivuista [37]. SilverStripe oli alun perin kaupallinen tuote, jota SilverStripe Oy oli kehittänyt viisi vuotta, kun se vuonna 2006 päätti rakentaa sen kokonaan uudelleen, tarkoituksena saavuttaa kolme päämäärää: tukea suurempia asiakkaita tehokkaammilla ja monimutkaisemmilla sivuilla, käyttää hyväksi aikaisemman kehitystyön tuomaa kokemusta ja luoda uusi koodipohja julkaistavaksi avoimena ohjelmistona. Uusi SilverStripe 2.0 julkaistiin vuonna 2007, ja samalla sen lähdekoodi vapautettiin BSD-lisenssin alaisena [38]. SilverStripen uusin pääversio, versio 4, julkaistiin vuonna 2017, ja sen uusin aliversio 4.7 julkaistiin tammikuussa 2021.

Vuoden 2013 elokuusta SilverStripe Oy:llä on ollut sopimus Uuden-Seelannin hallituksen kanssa [39]. Sopimuksessa sovitaan, että SilverStripe Oy tuottaa ja ylläpitää järjestelmää, nimeltä Common Web Platform eli yleinen verkkoalusta,

jolla Uuden-Seelannin hallituksen eri virastot ja elimet voivat luoda yhteensopivia verkkosivustoja, joiden välistä koodia on helppo jakaa, mikä nopeuttaa sivustojen kehitystä ja päivitystä. Tämä palvelu on toteutettu käyttäen SilverStripe-sisällönhallintajärjestelmää ja Solr Enterprise -hakuteknologiaa [40]. Sopimuksen oli tarkoitus päättyä vuonna 2020, mutta covid-19-viruksen aiheuttaman pandemian vuoksi sopimusta jatkettiin 12 kuukaudella [41].

BSD-lisenssit ovat GPL-lisenssien tapaan avoimia lisenssejä, mutta toisin kuin GPL-lisenssi, joka on niin sanottu käyttäjänoikeuslisenssi ja joka vaatii GPL-lisenssin alaisena olevien ohjelmistoihin tehtyjen muutosten julkaisun GPL-lisenssin alaisena, BSD-lisenssi on puhdas avoin lisenssi, joka asettaa vain minimaaliset rajoitukset lisensoidun koodin tai ohjelmiston käytölle, muunnolle ja jalkalle. Siinä missä GPL-lisenssi estää sen alaisen koodin tai ohjelmiston käytön suljetussa tuotteessa johtuen sen vaatimuksesta, koska kaikki versiot jaettava koodista tulee jakaa saman tai korkeamman GPL-lisenssin alaisena, BSD-lisenssi ei tällaista vaadi, ja näin sen alaiset avoimen lähdekoodin ratkaisut on mahdollista sisällyttää osaksi suljetun lähdekoodin kokonaisuuksia. SilverStripe on kehittäjien mukaan ainoa BSD-lisenssin alainen sisällönhallintajärjestelmä [42].

Käyttämänsä avoimen sovelluslisenssin ja pienen mutta lojaalin käyttäjäkunnan ansiosta SilverStripelle on tarjolla yli 3 000 erilaista lisäosaa, joista yli 1 000 on yhteensopivia SilverStripen uusimpien 4.5–4.7-versioiden kanssa. Tämä on huomattavasti vähemmän kuin monissa suosituimmissa sisällönhallintajärjestelmissä, mikä johtuu mahdollisesti hieman siitä, miten toisin kuin GPL-lisenssi, BSD-lisenssi ei vaadi jaettavan koodin tai sovelluksen jakoa. Näistä yli tuhannesta lisäosasta löytyy kuitenkin paljon hyödyllisiä lisäosia, ja hyödyllisimmät, kuten Fluent, joka tuo SilverStripen hallintapuolet mahdollisuuden luoda kieli-versioita sivustosta, on tuotu osaksi SilverStripen oletusasennuspakettia.

BSD-lisenssin tuoma vapaus on yksi SilverStripen vahvuuksia, sillä se antaa kehittäjälle täydet valtuudet tehdä SilverStripella, mitä tämä haluaa, ilman pel-

koa käyttöoikeuden poistamisesta tai käyttöön liittyvästä oikeushaasteesta. Kehittäjä voi rakentaa oman palvelunsa SilverStripen päälle ja palvelun valmistuttua alkaa myydä sitä ilman suurempia lisenssihuolia. Samaan aikaan sen pieni käyttäjäkunta ja siitä johtuva valmiiden toteutusten pieni määrä tekevät SilverStripesta sisällönhallintajärjestelmän, jolla on melko vaikea saada haluamansa kaltaista sivustoa ilman kehittäjää.

SilverStripe vaati toimiakseen palvelimen, jossa on PHP 7.1 tai uudempi, MySQL 5.6 tai uudempi tietokanta ja jokin verkkopalvelinohjelmisto. Saatavilla on myös SilverStripe-yhteisön ylläpitämät moduulit, joiden avulla SilverStripe voi käyttää myös Post-greSQL-, SQL Server- tai SQLite-tietokantoja. [43.]

3.2 Ominaisuudet

Silverstripe-pohjaisten sivustojen kehitys perustuu Silverstripe Frameworkiin. Se on MVC-pohjainen kehys, jossa Drupalin tapaan kehitys jaetaan kolmeen pääosa-alueeseen:

- Sivupohjat määrittävät näytettävän tiedon ulkomuodon ja verkkosivun ulkonäön yleisesti. SilverStripe käyttää sivupohjiin omaa tiedostomuotoaan, jonka tiedostolyhenne on .ss, esimerkiksi Page.ss.
- Kontrollerit määrittävät käytössä olevat toiminnot ja funktiot sekä ovat vastuussa käyttäjän antamien käskyjen tulkitsemisesta ja näiden pohjalta MVC-mallin muiden osien muokkaamisesta. Näihin kontrolleihin viitataan usein -Controller päätteellä tiedosto- ja PHP luokkien -nimissä.
- Dataobjektit määrittävät sivuston käyttämän datan muodon, tallennuksen ja käsittelyn. Dataobjektit voivat olla joko sivupohjia vastaavia tai omia tiedostojaan, jotka nimetään niiden sisältämän dataobjektin mukaan, esimerkiksi Page.php.

On tärkeää, että jokaisella sivupohjalla on sitä vastaava kontrolleri, sillä järjestelmä olettaa näiden sivupohja-kontrolleriparien olemassaolon, ja jos se ei onnistu löytämään niitä, ei sivusto toimi.

SilverStripe on rakennettu PHP-ohjelmointi kielellä, mutta se käyttää myös omaa .ss-päätteistä tiedostomuotoa, joka on SilverStripe-sivupohjatiedosto, ja

perustuvat HTML:ään. SilverStripen .ss-tiedostot mahdollistavat dataobjektien sisältämään dataan ja sivupohjaa vastaavan kontrollerin funktioihin viittauksen suoraan sivupohjassa, mikä tekee sivupohjista helposti luettavia ja nopeasti muokattavia.

3.3 Teemat

SilverStripe-sivujen päärakenteet perustuvat teemoihin, jotka toimivat pitkälti samalla tavalla kuin muidenkin sisällönhallintajärjestelmien teemat. Teema on koelma tyylisääntöjä, JavaScript-koodia, sivupohjia ja muita sivuston ulkonäön ja käyttäjän käytettävissä olevien toimintojen kannalta tärkeitä osia. Sivustolla voi olla useita eri teemoja, esimerkiksi eri teemat työpöytä- ja mobiiliversioille verkkosivusta. Kuten WordPressiin ja useimpiin muihin sisällönhallintajärjestelmiin, myös SilverStripen on tarjolla valmiita teemoja, mutta tämä tarjonta on suhteellisen pientä. Onneksi oman teeman rakentaminen on helppoa ja vaivatonta. Uutta teemaa luodessa luodaan SilverStripe-asennuksen themes kansioon uuden teeman nimellä varustettu kansio.

SilverStripe sisältää sivupohjille tarkoitetun templates-kansion, joka sisältää Layout- ja Includes-kansiot. Nämä kansiot sisältävät SilverStripen käyttämät .ss-päätteiset tiedostot, ja templates-kansiosta täytyy aina löytyä sivun perusrakenteen määrittävä Page.ss-tiedosto. Tämän lisäksi Layout-kansiossa on yleensä myös Page.ss-tiedosto, joka määrittää sivun sisällön rakenteen, jos SilverStripe ei löydä mitään muuta sivupohjaa. Näiden lisäksi teemaan on mahdollista lisätä kuvia sekä tyyli-, HTML- ja JavaScript-tiedostoja. Toimiakseen kunnolla SilverStripe vaatii näiden tiedostojen sijaitsevan niille tarkoitetuissa kansioissa, toisin sanoen jos scripts.js-tiedosto ei ole teeman javascript-nimisessä kansiossa, ei SilverStripe löydä kyseistä tiedostoa, kun ohjain sitä pyytää. Näin ollen on hyvin tärkeää, että teeman kansiorakenne on selkeä ja vastaa SilverStripen oletuksia.

Teeman käyttöönotto tapahtuu muuttamalla SilverStripe-asennuksen `_config-`kansiosta löytyvää `theme.yml`-tiedostoa, josta löytyy lista teemoista, joita sivuston on määritetty käyttävän. SilverStripe käy listan teemat läpi järjestyksessä, ja jos se ei löydä oikeaa tiedostoa ensimmäisen teeman teemakansiosta, siirtyy se seuraavaan teemaan. Kun luodun tai asennetun teeman nimi lisätään listan ensimmäiseksi, tulee kyseinen teema käyttöön pääteemaksi. Esimerkkikoodissa 1 on nähtävissä projektissa käytetyn `theme.yml`:n sisältö.

```
---
Name: mytheme
---
SilverStripe\View\SSViewer:
  themes:
    - 'staskinen-theme'
    - 'simple'
    - '$public'
    - '$default'
```

Esimerkkikoodi 1. Työssä käytetyn `themes.yml`-tiedoston sisältö.

3.4 Sivupohjat

SilverStripen sivupohjat määrittävät, miten tietokannasta löytyvä tieto esitetään sivuston käyttäjälle. Ne käyttävät SilverStripen omaa `.ss`-tiedostotyyppiä, joka mahdollistaa yleisimpien verkko-ohjelmointikielien, kuten XML, HTML, JSON ja JavaScript, käytön samassa tiedostossa. Tavallisesti sivupohja sisältää kuitenkin vain HTML-koodia, jossa on mukana SilverStripen syntaksin mukaisia elementtejä, jotka merkitään `<%` avaus- ja `%>` sulkutageilla.

Koska sivupohjat vastaavat SilverStripen käyttämän MVC-arkkitehtuurin View osaa, on jokaisella sivupohjalla sitä vastaavat tiedostot, jotka täyttävä MVC-arkkitehtuurin kaksi muuta osaa. Esimerkiksi jokaisesta teemasta löytyvän `Page.ss` (View) -tiedoston muut osat ovat `Page.php` (model) ja `PageController.php` (Controller).

Sivupohjien sisällä on mahdollista viitata suoraan sisällönhallintajärjestelmän tietokantaan tallentamaan tietoon sekä kutsua sivupohjaa vastaavien ohjainten

sisältämiä funktioita. Eli Page.ss-sivupohjatiedoston sisällä on mahdollista kutsua sitä vastaavan ohjaimen, PageController, sisältämiä funktioita.

Nämä viittaukset ja kutsut kulkeutuvat alas sivumallipuuta. Tällä tarkoitetaan sitä, että esimerkiksi projektissa, jossa on Page.ss- ja Homepage.ss-sivumallit pääsivua ja muita sivuja varten ja kummallekin löytyvät oikeat data- ja ohjaintiedostot, voi Homepage.ss-sivupohjassa kutsua dataa, joka löytyy Page.php-tiedostosta, tai funktiota, joka löytyy PageController.php-tiedostosta. Tämä on mahdollista, sillä kuten esimerkikoodista 2 on nähtävissä, HomePage.php-tiedostossa luotava HomePage-luokka määrittellään Page.php-tiedostossa määritellyn Page-luokan jatkeeksi. Näin HomePage-luokalla on kaikki Page-luokan ominaisuudet. Samoin HomeController määritetään jatkamaan PageController-luokkaa.

```
*** HomePage.php ***
<?PHP

namespace SilverStripe\Lessons;

use Page;

class HomePage extends Page
{
}
*** Page.php ***
<?PHP

use SilverStripe\CMS\Model\SiteTree;

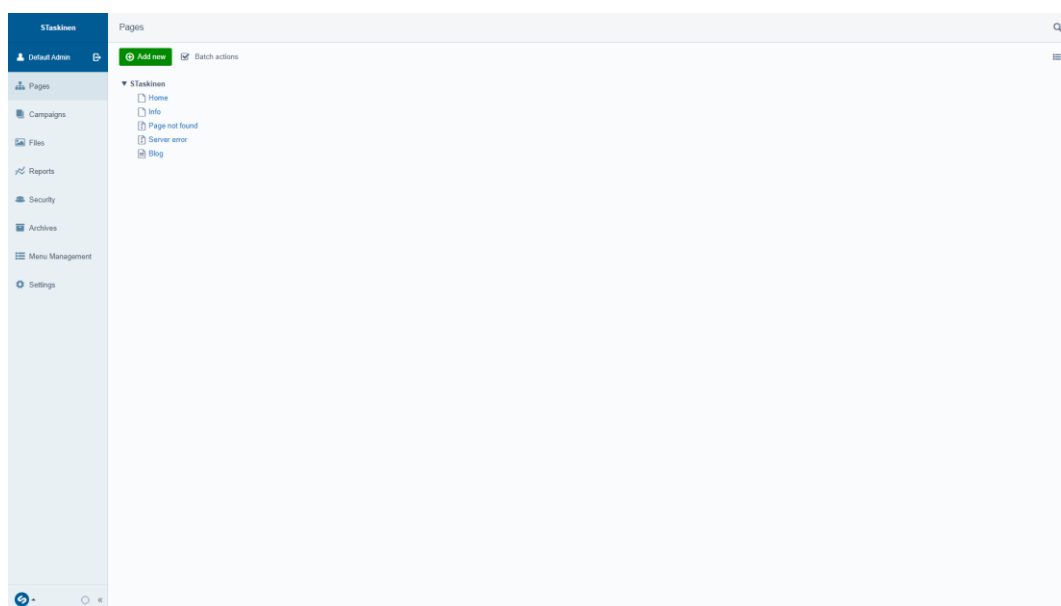
class Page extends SiteTree
{
    private static $db = array(
    );

    private static $has_one = array(
    );
}
```

Esimerkkikoodi 2. Esimerkkiversiot mahdollisista HomePage.php- ja Page.php-tiedostoista.

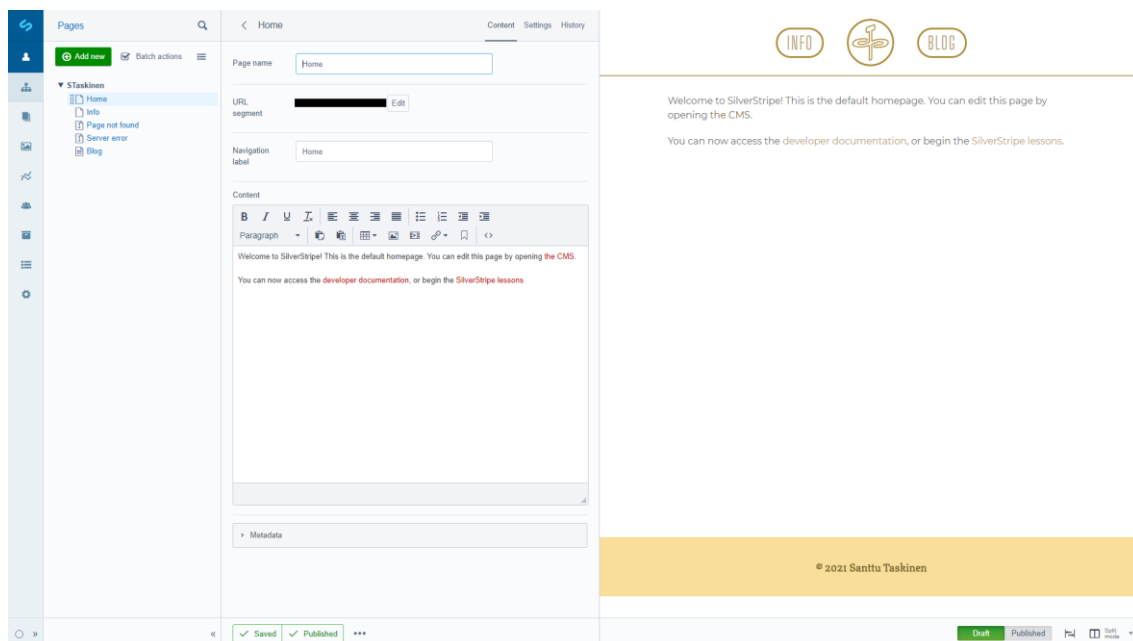
3.5 Sisällönhallinta

SilverStripen sisällönhallinta tapahtuu monen muun sisällönhallintajärjestelmän tapaan verkkosivusta erillään olevasta, kuvassa 4 näkyvässä, hallintanäkymästä, johon pääsee kirjautumaan sisään osoitteesta <https://{verkkosivuston osoite}/admin>. Sisäänkirjautumisen jälkeen käyttäjälle avautuu näkymä, jossa tämä voi luoda uusia sivuja, muokata niiden sisältöjä ja sekä yksittäisten sivujen että koko sivuston asetuksia.



Kuva 4. SilverStripen hallintanäkymän oletusnäkymä.

Kuvassa 5 nähtävä sisällönmuokkausnäkymä sisältää myös niin kutsutun WYSIWYG- eli "What You See Is What You Get" -näkymän. Tällä tarkoitetaan näkymää, joka yrittää vastata mahdollisimman tarkasti sitä, miltä siihen syötetty sisältö tulee näyttämään itse sivulla, johon sisältö on syötetty. WYSIWYG tekee muokkauksen ja uusien sivujen teosta helpompaa, koska se mahdollistaa sivun sisällön asettelun helposti haluttuun tapaan ja paljastaa nopeasti, tarvitseeko sivupohjaan tehdä muutoksia, jotta uusi sisältö toimii tai näkyy halutulla tavalla.



Kuva 5. SilverStripen hallintanäkymän jaettu sivunäkymä, jossa oikealla nähtävissä valittuna oleva sivu WYSIWYG-näkymän piirtämänä.

3.6 Tietoturva

Tietoturva-aukoista tietoa keräävän cvedetails.comin mukaan SilverStripesta on vuodesta 2007 eteenpäin löydetty 48 tietoturva-aukkoa [44]. Samalla aikavälillä Wordpressistä on löytynyt 314 [45], Drupalista 289 [46] ja Joomlaista 327 [47] tietoturva-aukkoa. Tämä saa SilverStripen tietoturvatilanteen vaikuttamaan oikein hyvältä, mutta pitää ottaa huomioon, että cvedetails listaa sovelluksen kaikkien osien tietoturva-aukot saman nimikkeen alle. Koska SilverStripe sisältää vain sisällönhallinnan ja kehitysrungon, on sillä vähemmän mahdollisia listauskohtia ja oikeampi vertailukohta onkin vertailtavien sisällönhallintajärjestelmien ydinkoodi. Silti jos vertailukohdiksi valitaan kunkin sisällönhallintajärjestelmän ydin, huomataan Wordpressistä löytyneen 266 [48], Drupalista 162 [49] ja Joomlaista 137 [50] tietoturva-aukkoa vuodesta 2007 eteenpäin. Joomlaan tietoturva-aukkojen lukumäärä nousee 201:een, kun mukaan lasketaan vielä Joomla!-nimikkeen alla olevien lisäksi Joomla-nimikkeen alta löytyvät tietoturva-aukot [51]. Näin huomaa, että tässäkin valossa SilverStripe vaikuttaa tietoturvan puolesta hyvältä valinnalta.

Tietoturva-aukoille ja riskeille on olemassa CVSS- eli Common Vulnerability Scoring System -luokitus. Se on avoin ja ilmainen luokitusstandardi, joka on tarkoitettu tietoturva-aukkojen vaarallisuuden luokitteluun, mikä mahdollistaa kehitysresurssien paremman keskittämisen. CVSS-luokituksessa tietoturvariskille annetaan arvo nolasta kymmeneen, joka kuvaa luokitellun tietoturva riskin suuruutta. Mitä suurempi CVSS-luku on, sitä pahempi tietoturvariski. Näin ollen on tärkeää verrata myös löytyneiden tietoturva-aukkojen vakavuutta niiden määrän lisäksi. Keräämällä cvedetailsin listaamien tietoturva-aukkojen CVSS-luokitukset ja sijoittamalla ne kymmenpisteiseen asteikkoon, jossa pienin CVSS-arvo on 0–1 ja suurin CVSS-arvo on 9–10, saadaan aikaan taulukon 1 mukainen taulukko.

Taulukko 1. Vuodesta 2007 eteenpäin löydettyjen tietoturva-aukkojen määrä eri CMS-järjestelmissä [52; 53; 54; 55].

CVSS-luku	0–1	1–2	2–3	3–4	4–5	5–6	6–7	7–8	8–9	9–10
WordPress	0	0	9	14	104	63	40	26	1	9
%	0 %	0 %	3 %	5 %	39 %	24 %	15 %	10 %	0 %	3 %
Drupal	0	0	7	16	51	49	23	14	1	1
%	0 %	0 %	4 %	10 %	31 %	30 %	14 %	9 %	1 %	1 %
Joomla!	0	0	1	5	53	37	12	29	0	0
%	0 %	0 %	1 %	4 %	39 %	27	9 %	21	0 %	0 %
SilverStripe	0	1	1	2	17	15	5	5	0	1
%	0 %	2 %	2 %	4 %	39 %	32 %	11 %	11 %	0 %	2 %

Taulukosta 1 on nähtävissä, ettei SilverStripe erityisemmin erotu tietoturva-aukkojensa hajonnalla suosituimmista sisällönhallintajärjestelmistä. Näin ollen voidaan olettaa, että SilverStripeistä löytyvillä tietoturva-aukoilla ei ole taipumusta olla yhtään sen suurempia, kuin mitä on löydettävissä Wordpressistä tai muista sisällönhallintajärjestelmistä.

Koska SilverStripen tietoturva on yleisesti hyvällä tasolla avoimen kommunikation ja nopean reaktion ansiosta, on SilverStripella vielä yksi etu verrattuna suosittuihin sisällönhallintajärjestelmiin, ja se on sen tuntemattomuus. Tuntematon ja pienen käyttäjäkunnan järjestelmä on paremmin turvassa laaja-alaisilta ja ”kaupanhylly”-hyökkäyksiltä, sillä nämä yleensä kohdistetaan joko suurimpaan tai pariin kolmeen suurimpaan kohteeseen. Verkosta löytyy nopealla haulla paljon oppaita, miten testata tai kaapata Wordpress-sivustoja. Samanlaisia on vaikeata löytää vähemmän tunnetuille järjestelmille, ellei jokin versio järjestelmästä sisällä tai sisältänyt huomattavaa tietoturva-aukkoa, ja siksi SilverStripen tuntemattomuus on sen etu tietoturvan näkökulmasta.

4 Sivuston toteutus

4.1 Suunnittelu

Insinööriyön asiakas halusi itselleen hieman valtavirrasta eroavan henkilökohtaisen blogin, jonka kaikkia ominaisuuksia hän voisi halutessaan muokata. Nämä vaatimukset rajoittivat työhön sopivia sisällönhallintajärjestelmiä. Tarve erota valtavirrasta poisti soveltuvien sisällönhallintajärjestelmien listalta WordPressin sen huomattavasti suurimman käyttäjäkunnan takia, ja tarve voida hallita mahdollisimman paljon sivuston eri ominaisuuksia poisti listalta Squarespacen ja Wix.comin kaltaiset verkkosivukoneet. Joomla ja Drupalin maine hieman monimutkaisina ja pienten sivujen kohdalla hieman tehosyöppöinä epäilytti asiakasta, sillä hän piti näitä hieman ylimitoitettuina ratkaisuinä yksinkertaiselle blogisivustolle.

Lopulta päädyttiin SilverStripeen, koska insinööriyöntekijällä oli kokemusta SilverStripeestä aiempien työtehtävien pohjalta, BSD-lisenssin ja yksinkertaisen hallintakäyttöliittymän takia. Asiakas näki SilverStripella sivuston ylläpidon hyvänä oppimismahdollisuutena, joka ei kuitenkaan tuottaisi liikaa vaivaa sen selkeän ja yksinkertaisen käyttöliittymän sekä MVC-sovellusarkkitehtuurin ansiosta.

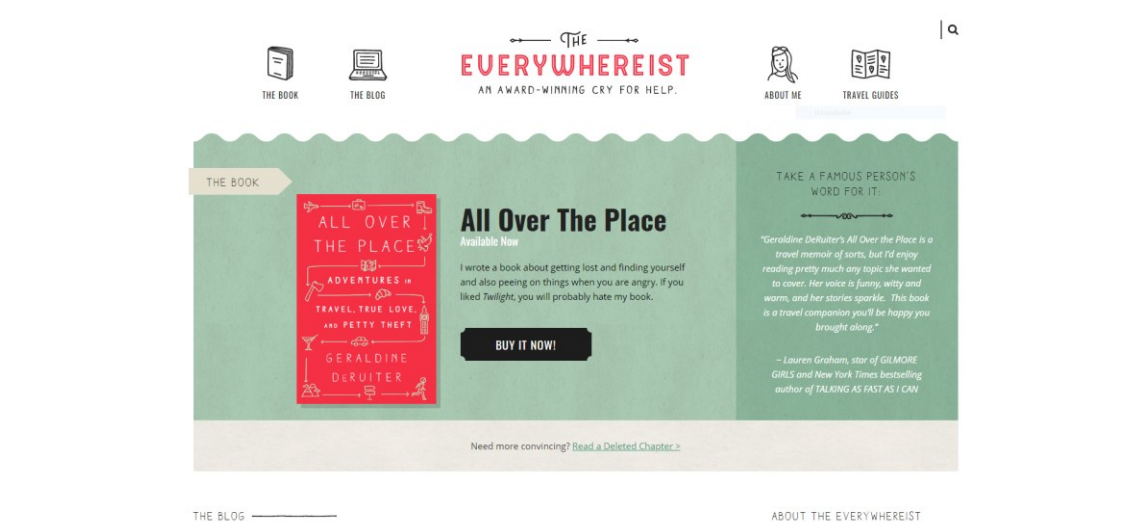
Asiakas halusi sivuston ulkonäön olevan yksinkertainen, mielellään vain yläpalkki, alapalkki ja keskitetty välitila, johon sivun tai blogiartikkelin sisältö sijoitettaisiin. Projektin alussa asiakas oli suunnitellut projektin pääväreiksi tummia sävyjä, kuten tummaa sinistä ja violettiä, mutta ei lopulta ollut tyytyväinen sävyjen tuomaan lopputulokseen. Sivuston lopullinen valkoinen, keltaisen ja kullan väriskeema syntyi puolivahingossa sivuston ST-yhdistelmälogoa suunnitellessa. Sivuston väriskeemaa oli työstetty verkon Paletton.com-työkalulla, joka mahdollistaa erilaisten väripalettien luonnin automaattisesti siihen syötetyn värin mukaan. Asiakas oli luonut tällä työkalulla väripaletin, jota halusi sivulla käytettävän. Mutta sivuston logoa työstettäessä valittiin värilistasta vahingossa korostusväriksi tarkoitettu ruskea logoon pääväriksi ja lopputuloksen nähdessään asiakas

päätti vaihtaa sivuston väriteeman valkoisiin, keltaisiin ja haaleisiin ruskeisiin. Kuvassa 6 on lopullinen sivustolla käytetty logo.



Kuva 6. Asiakkaan verkkosivua varten luotu logo.

Tuotettu logo sijoitettiin sivulla keskelle yläpalkkia, ja sen kummallekin kyljelle asiakas halusi mahdollisuuden lisätä linkkejä samankaltaisesti, kuin miten sivu-linkit asettuvat kuvassa 7 nähtävällä sivustolla <https://everywhereist.com>.



Kuva 7. Everywhereis.com-verkkosivusto 5.2.2021.

Sivuston rakenteelliseksi pohjaksi valittiin Ulkit, suosittu Bootstrapin kaltainen avoimen lähdekoodin HTML-, CSS- ja JavaScript-sovelluskehys, joka sisältää paljon valmiita CSS-luokkia ja JavaScript-komponentteja. Näillä valmiilla elementeillä pystyy toteuttamaan verkkosivujen tavallisimpia elementtejä, kuten navigaatiovalintoja, painikkeita ja pudotusvalikkoja, helposti, mikä nopeuttaa kehitystyötä huomattavasti.

4.2 Sivuston paikallinen kehitys

Sivuston toteutus aloitettiin rakentamalla paikallinen kehitysympäristö. Tämä oli tarkoitus tehdä käyttäen Hashicorpin Vagrant-ohjelmaa, jolla on helppoa pystyttää virtuaalikoneita erilaisissa kokoonpanoissa. Ohjelmalle on tarjolla valmiita kokoonpanoja, joita kutsutaan nimellä vagrantbox. Toivoin voivani käyttää tätä kehitysympäristönä, sillä SilverStripe Oy jakelee itse sisäisesti käyttämäänsä vagrantboxia, ja uskoin tämän olevan hyvä kehitysympäristö. Valitettavasti sen käyttöönotto ei teknisistä syistä onnistunut, ja koska en halunnut kuluttaa siihen turhaan aikaa, päättyi kehitysympäristöksi yksinkertainen XAMPP-asennus. XAMPP on valmis sovelluspaketti, jolla on mahdollista pystyttää paikallinen palvelin, jolta löytyy kaikki verkkosivukehitykselle tarpeellinen, kuten PHP ja tietokannat, nopeasti.

XAMPP:n asennuksen jälkeen asensin kehitysympäristöön Composer-ohjelman, joka on eräänlainen hallintatyökalu, joka mahdollistaa erilaisten PHP-pakettien asentamisen, hallitsemisen ja poistamisen. Composer täytyy asentaa osaksi PHP-asennusta, joten tässä tilanteessa se asennettiin kiinni XAMPP:n PHP-asennukseen. Composer tekee SilverStripen ja sen lisäosien asennuksesta huomattavasti helpompaa, kuin se olisi manuaalisesti.

Kun Composer oli asennettu, luotiin SilverStripe-projekti XAMPP-verkkopalvelimen htdocs-kansioon navigoimalla siihen komentokehotteessa ja ajamalla komennon "Composer create-project silverstripe/installer SilverStripe". Tämän jälkeen avasin komennon luoman SilverStripe-kansion Visual Studio Codessa ja

tein kansion juureen .env-ympäristötiedoston johon kopioin .env.example-tiedoston tiedot ja muunsin ne vastaamaan kehitysympäristöä. Varmistettuani, että kehitysympäristö oli asetettu oikein, ajoin komentokehoteessa vielä komennon "vendor/bin/sake dev/build" luodakseni SilverStripe-asennuksen käyttämän tietokannan.

Asennettuani SilverStripe-asennuksen paikalliseen kehitysympäristöön, loin asennukseen oman teeman, jolle annoin nimeksi STaskinen-theme luomalla asennuksen themes-kansioon staskinen-themes-kansion. Seuraten SilverStripe-teemojen luonnin ohjeistusta [56] loin teemakansion. Teeman luonnin jälkeen muutin theme.yml-tiedostoa saadakseni sivuston käyttämään luotua staskinen-theme-teemaa.

Halusin käyttää SilverStripe-teemassa Sassia, koska tiedän puhtaan CSS-kielen kanssa työskentelyn olevan hankalaa pidemmällä aikavälillä. "Syntactically awesome style sheets" eli Sass on komentokieli, jota käyttäen on mahdollista luoda monimutkaisia CSS-tyylitiedostoja huomattavasti helpommin kuin kirjoittamalla puhdasta CSS-kieltä. Tämä helppous tulee Sassin tuomista ominaisuuksista, jotka puuttuvat CSS-kielestä, kuten muuttujista, funktioista ja kyvystä asettaa tyylisääntöjä sisäkkäin.

Sass-komentokieltä on mahdollista kirjoittaa kahdella eri syntaksilla eli muotokenteella. Sassin alkuperäinen syntaksi oli sisennetty syntaksi, ja sillä kirjoitetuilla tiedostoilla on pääte .sass, ja tätä syntaksia kutsutaankin välillä myös Sassiksi. Vaihtoehtoinen SCSS-syntaksi on CSS-kielen niin sanottu yläjoukko, sillä SCSS sisältää kaiken, mitä CSS:kin ja ylimääräistä. SCSS muistuttaa hyvin pitkälti CSS-kieltä, ja tämä on tehnyt siitä suosituimman Sass-syntaksin. Sekä Sass- että SCSS-syntaksilla kirjoitetut tiedostot käännetään lopulta CSS-tiedostoiksi käyttöä varten.

Voidakseni käyttää Sassia kehittämässäni teemassa, asensin sivustolle SilverStripen verkkosivuilta löytämäni "ScssPHP compiler for Silverstripe" -lisä-

osan. Lisäosa mahdollisti käyttämieni .scss-tiedostojen kääntämisen .css-tiedostoiksi aina, kun ajoin SilverStripen keskusmuistin tyhjentävän flush-komennon. Asennettuani ScssPHP compiler for Silverstripe -lisäosan loin teemakansion scss-kansioon seuraavat kansiot:

- Base, johon luotuihin `_body.scss`- ja `_typography.scss`-tiedostoihin asetettiin sivuston perusrakenteiden kuten kirjaisimien ja HTML-dokumentin kannalta olennaiset määrittäykset
- Helpers, johon luotuihin `_colors.scss`-, `_mixins.scss`-, `uikit-mixins.scss`- ja `_variable.scss`-tiedostoihin asetettiin muissa tyylitiedostoissa käytettävät muuttujat, jotta jokaista samaa väriä käyttävää tyylisääntöä ei tarvitse käydä muuttamassa erikseen värin vaihtuessa
- Layout, johon luotuihin `_blog.scss`-, `_content.scss`-, `_footer.scss`-, `_header.scss`- ja `_navigation.scss`-tiedostoihin asetettiin kyseisten elementtien tyylisäännöt
- Vendor, johon tuotiin UIKitin lähdekoodin SCSS-syntaksiset Sass-versiot UIKitin integraatiota varten.

UIKitin Sass-lähdekoodin sisään tuonti osaksi laadittavaa CSS-tiedostoa mahdollisti sen, että sivusto lataa vain yksittäisen CSS-tiedoston, joka sisältää sekä UIKitin ominaisuudet että omat tyylisääntöni. Tarve ladata vain yksi tyylitiedosto useamman sijasta nopeuttaa sivuston latautumista, varsinkin ensilatauksella.

SilverStripe ei suoraan tue sivupuusta poikkeavaa valikkoa eikä useita valikkoja, joten saadakseni aikaan asiakkaan haluaman yläpalkin, jossa logon kummallakin puolella on muokattavissa oleva valikko, täytyi sivustolle asentaa Heyday Menu Manager -lisäosa ja sen vaatiman SilverStripe Grid Field Extensions Module lisäosa. Heyday Menu Manager mahdollistaa uusien, hallintapuolella editoitavien valikoiden luonnin (1) lisäämällä halutut valikot sivuston `_config`-kansioon joko omana yml-tiedostonaan tai jo olemassa olevaan tiedostoon.

Tämän työn puitteissa päätin luoda oman `menusets.yml`-tiedoston, johon lisäsin asiaankuuluvat rivit luodakseni "Header_Right"- ja "Header_Left"-nimiset valikot sivustolle. Suoritettuani `/dev/build`-komennon saadakseni valikot käyttöön, lisäsin `Header.ss`-tiedostoon logoa ennen ja sen jälkeen toistorakenteet, jotka käyvät läpi `Header_Right`- ja `Header_Left`-valikot. Käytetty toistorakenne vastaa navigaatiovalikon luontiin SilverStripessa tavallisesti käytettävää toistorakennetta,

joidenkin muuttujien nimet vain vaihdettiin Heyday Menu Managerin käyttämiin ja hallintanäkymästä asetettaviin muuttujiin. Erot toistorakenteessa ovat nähtävissä esimerkkikoodista 3 ja 4.

```
<% loop $menu(1) %>
    <li class="$LinkingMode">
        <a href="Link" title="Title.XML">MenuTitle.XML</a>
    </li>
<% end_loop %>
```

Esimerkkikoodi 3. SilverStripe-valikon oletustoistorakenne.

```
<% loop $MenuSet('Header_Left').MenuItems %>
    <li class="MenuItem">
        <a class="$LinkingMode" href="$Link" title="Go to the $Menu-
Title page">$MenuTitle</a>
    </li>
<% end_loop %>
```

Esimerkkikoodi 4. Heyday Menu Manager -lisäosan mukainen SilverStripe-valikon toistorakenne.

Kun valikot on asetettu paikalleen, on niihin helppo asettaa sivuston sisäisiä ja ulkoisia linkkejä hallintanäkymästä käsin.

Toteutin sivustolle halutun blogitoiminnon ensin seuraamalla SilverStripen virallista tutoriaalia ja luomalla sivulle ArticleHolder + ArticlePage + ArticleCategories -kokonaisuuden. Tämä ratkaisu vaikutti aluksi hyvältä, mutta pienellä testailulla se osoittautui olevan kestämaton pitkällä aikavälillä. Ratkaisu soveltuu hyvin tilanteisiin, joissa artikkeleja on rajallinen määrä, sillä jokainen artikkeli on oma sivunsa, jotka ArticleHolder-sivu kerää. Isolla artikkelimäärällä tämä johtaa nopeasti hyvin sekavaan hallintajärjestelmään, sillä jokainen artikkeli näkyy sivupuussa erikseen.

Esitettyäni huomion asiakkaalle, päätin tarkistaa, löytyisikö SilverStripen virallisista lisäosista paremman blogiomaisuuden tuovaa lisäosaa. Pienen etsinnän jälkeen löysin "SilverStripe Blog Module"-lisäosan, joka vaikutti juuri siltä, mitä etsin ja mitä asiakas halusi. Asensin tämän lisäosan, ajoin dev/build-komennon ottaakseni sen käyttöön ja kävin muuttamassa jo luodun Blog-sivun lisäosan blog-sivutyyppiin. Tämä blogiratkaisun muutos johti lisätyöhön tyylien kanssa,

sillä olin jo tyytellyt aikaisemman ratkaisun sivut sivun teemaan sopiviksi. Tein tämän muokkaamalla `_blog.scss`-tiedostoa lisäämällä lisäosan luomiin elementteihin sopivat tyyllisäännöt ja ajamalla sisällönhallintajärjestelmän välimuistin tyhjentävän `flush`-komennon. Sama komento saa `ScssPHP` compiler for SilverStripe -lisäosan kääntämään `SASS`-tiedostot `CSS`-tiedostoiksi.

4.3 Palvelimen valmistelu

Kun asiakkaan kanssa oli tultu yhdenmielisyyteen siitä, että verkkosivusto on valmis asiakkaan virtuaalipalvelimelle vietäväksi, asiakas antoi tarvittavat oikeudet palvelimelleen ja vapaat kädet tarvittavien muutosten tekemiseen. Virtuaalipalvelimen ympäristö oli `Ubuntu 20.04`, jolle asiakas oli aikoinaan asentanut `Node.js`:n ja sitä käyttävän ohjelman, joka mahdollistaa klassisten pöytäroolipeliä pelaamisen verkossa, eikä asiakas ollut koskenut palvelimeensa sen jälkeen, sillä se toimi tämän tarkoitukseen oikein hyvin. Tämän takia ennen kuin lähdin asentamaan yhtäkään `SilverStripen` tarvitsemista ohjelmistoista, kuten verkkopalvelinta, hain ohjelmistopakettien uusimmat tarjolla olevat versiot `"apt-get update"`-komennolla ja asensin löydettyt päivitykset `"apt-get upgrade"`-komennolla. Seuraavaksi asensin palvelimelle tiedostojen siirtoa varten `vsftpd-ftp`-palvelimen ja asetin sen käyttämään `FTPS`-yhteyttä tiedoston siirtoon.

`FTPS`-yhteydessä `FTP`-yhteys salataan käyttäen `TLS`- tai `SSL`-sertifikaattia. Kyseinen sertifikaatti on mahdollista saada luotetun tahon allekirjoittamana pyytämällä se varmenneviranomaiselta tai luomalla sellainen itse. Sillä, onko sertifikaatti virallinen vai itse luotu ja allekirjoitettu ei ole `FTPS`-salauksen toiminnan kannalta merkitystä, joskin jotkin ohjelmat eivät mahdollisesti anna luoda yhteyttä kohteeseen, jonka sertifikaatti ei ole luotetusta lähteestä. Tulevaisuudessa tämän projektin tiimoilla luodut sertifikaatit onkin tarkoitus vaihtaa yleisesti luotettujen lähteiden kuten `Let's Encryptin` jakamiin sertifikaatteihin.

Asennettuani `VSFTPD`-palvelimen ja testattuani, että asettamani `FTPS`-yhteys toimi, lähdin asentamaan palvelimelle `Apache`-verkkopalvelinta, joka on maail-

man suosituin verkkopalvelin. Apachen asennus sujui helposti ajamalla palvelimella komento "apt-get install apache2", minkä jälkeen asennuksen luoman /var/www/html-kansion sisältö oli nähtävissä palvelimen osoitteesta. Asiakas ei ollut nähnyt aikaisemman käyttönsä puitteissa verkkotunnuksen hankintaa tarpeelliseksi ja päädyimme keskustellessa siihen tulokseen, että asiakas varaa verkkotunnuksen myöhemmin projektin päättyttyä, kunhan hän on ensin saanut tuotettua sivustolle hieman materiaalia.

Apachen asennuksen jälkeen asiakkaan kanssa mietittiin palvelimelle asennettavaa tietokantaa. SilverStripen vaatimussivulla mainitaan MySQL ainoana oletusasennuksen tukemana tietokantana, ja aluksi oli tarkoitus asentaa se palvelimelle, mutta pienen pohdinnan jälkeen se vaihdettiin MySQLin korvaavaan MariaDBhen. Pääasialliset syyt vaihtoon olivat verkosta löytyvät artikkelit, kuten hackr.io:n vertailu [57], joissa MariaDB:n kerrotaan olevan tehokkaampi kuin MySQL. MariaDB:n asennus tapahtui ajamalla "apt-get install mariadb-server"-komento palvelimella.

MariaDB:n asennuksen jälkeen on suositeltavaa ajaa "mysql_secure_installation"-komento, jolla asetetaan tietokannan root-käyttäjän salasana, ja muuttaa erinäisiä oletusasetuksia, mikä tekee tietokanta-asennuksesta turvallisemman. Säästyäkseni ylimääräiseltä työltä päädyin ajamaan tämän komennon vasta myöhemmin, koska halusin asentaa phpMyAdmin-työkalun auttamaan tietokantojen hallintaa. Tämä vaatii toimiakseen PHP:n, jota palvelimella ei vielä ollut asennettuna.

PHP:n asennus palvelimelle onnistui helposti ajamalla "apt-get install php libapache2-mod-php". Komento asentaa PHP:n ja useamman PHP:n lisäosan sekä varmistaa, että libapache2-mod-php, Apache-verkkopalvelimen moduuli, joka mahdollistaa PHP:n käytön Apache-palvelimella, on asennettu. Asennuksen päättyttyä ajoin "php -m"-komennon, joka näyttää listan asennetuista PHP:n lisäosista, varmistaakseni, että siitä löytyvät kaikki SilverStripen vaatimat lisäosat. Huomasin, että listasta ei löytynyt intl-lisäosaa, joten asensin sen ajamalla komennon ""apt-get install PHP-intl".

Viimeiseksi, ennen kuin aloin asentaa phpMyAdmin-työkalua, asensin palvelimelle vielä Composer-ohjelman. Composer on PHP-pakettienhallintatyökalu, joka mahdollistaa erinäisten PHP-pakettien asennuksen, päivityksen ja poistamisen PHP-projekteissa. Tämä asentui komennolla `"apt-get install composer"`.

Lopuksi asensin phpMyAdminin komennolla `"apt-get install phpmyadmin"`. PhpMyAdminin asennuksen pyytäessä valitsemaan asennetun verkkopalvelimen, valitsin asetuksen `apache2` ja annoin asennusohjelman luoda tietokannan ja tietokantakäyttäjätunnuksen. Asennuksen mentyä läpi käynnistin verkkopalvelinohjelmiston vielä uudelleen ajamalla komennon `"systemctl restart apache2"`.

Asennettuani PHP:n, composerin ja phpMyAdminin palasin suojaamaan MariaDB-asennuksen ja ajoin `"mysql_secure_installation"`-komennon, joka ensimmäiseksi pyysi asettamaan root-tietokantakäyttäjän salasanan, ja sen asetettuani ohjelma kysyi, haluanko poistaa käytöstä erinäisiä oletusasetuksia, joihin vastasin aina kyllä. Tämän jälkeen varmistin tietokannan toimivuuden käyttäen phpMyAdminia. Varmistettuani tietokannan toimivan oikein kirjauduin sisään tietokantaan root-käyttäjänä käyttäen `"mysql -u root -p"`-komentoa ja juuri asettamaani salasanaa. Päästyäni sisään tietokantohallintaan loin SilverStripe-asennusta varten `ss_user`-käyttäjän ajamalla komennon `"CREATE USER käyttäjänimi@'localhost' IDENTIFIED BY 'haluttusalasana';"` MariaDB:n sisällä.

4.4 SilverStripe-sivun asennus palvelimelle

Saatuani palvelimen verkkopalvelinasennuksen kuntoon ja varmistettuani vielä kaiken toimivan aloin asentaa palvelimelle SilverStripen oletusasennusta, johon pystyisin sitten tuomaan paikallisessa kehitysympäristössä kehittämäni staskinen-theme-teeman. SilverStripen asennus tapahtui Composeria käyttäen navigoimalla palvelimella kansioon `/var/www/HTML` ja ajamalla komento `"composer create-project silverstripe/installer."` Komennon lopussa olevalla pisteellä komento ohjataan kohdistamaan toimintonsa kansioon, jossa komento ajetaan. Komennon ladattua ja luotua HTML-kansioon SilverStripen pohja-asennuksen

tiedostot täytyi sille luoda erinäisiä ympäristöasetuksia, kuten käytettävän tietokannan ja tietokantakäyttäjän tiedot.

SilverStripen sivuilta löytyvästä virallisesta dokumentaatiosta on mahdollista löytää kaikki mahdolliset sen käyttämät ympäristöasetukset, mutta projektin verkkosivulle riittivät hyvin pohja-asennuksen `.env.example`-tiedostosta löytyvät perusasetukset kahdella lisärivillä. Lisäämällä rivit `SS_DEFAULT_ADMIN_USERNAME` ja `SS_DEFAULT_ADMIN_PASSWORD` `.env.example`-tiedostosta kopioituun `.env`-tiedostoon ja lisäämällä niille arvot pystyin asettamaan SilverStripen admin-käyttäjän tunnuksen ja salasanan. Varmistettuani ympäristöasetusten olevan kunnossa ajoin komennon `"vendor/bin/sake dev/build"`, joka luo SilverStripen käyttämän tietokannan käyttäen `.env`-tiedostoon asetettuja tietoja. Tietokannan luonnin jälkeen uuteen SilverStripe-sivustoon oli mahdollista yhdistää menemällä selaimella palvelimen IP-osoitteeseen.

Projektin seuraava vaihe, itse paikallisesti kehitetyn sivuston siirto palvelimelle, osoittautui haastavammaksi kuin olin odottanut. Ensimmäinen vastaan tullut ongelma oli sivuston kehityksessä käyttämieni lisäosien asennus. Olin olettanut, että tämä onnistuisi helposti siirtämällä paikallisen version sivuston `composer.json`-tiedostosta palvelimelle FTP-yhteyden kautta ja ajamalla `"composer update"`-komento palvelimen `/var/www/html`-kansiossa, mutta jostain syystä tämä sekoitti palvelimen SilverStripe-asennuksen.

Asennuksen sekoittumisen vuoksi se täytyi poistaa palvelimelta, mikä onnistui tyhjentämällä palvelimen HTML-kansio ja sen luoma ja käyttämä tietokanta, jonka poisto onnistui phpMyAdminista käsin. Tämän jälkeen loin uuden SilverStripen projektin palvelimelle, tein samat muutokset `.env`-tiedostoon kuin aikaisemmin ja asensin SilverStripen ajamalla `"vendor/bin/sake dev/build"`-komennon. Saatuaani SilverStripen taas toimintakuntoon katsoin paikallisen projektin `composer.json`-tiedostosta projektissa käyttämieni lisäosien pakettinimikkeet, jotta pystyin asentamaan ne yksitellen palvelimelle käyttäen Composerin `"composer require pakettinimi"`-komentoa.

Seuraava ongelma oli saada luotu teema toimimaan. Tämän pitäisi onnistua helposti siirtämällä teeman sisältävä kansio palvelimen teemakansioon ja muuttamalla `~/app/_config`-kansioista löytyvää `theme.yml`-tiedostoa lisäämällä uusi teema teemalistan ensimmäiseksi. Näin helposti teeman siirto ei kuitenkaan tapahtunut, mikä johtui pääasiassa käyttämästäni ”ScssPHP compiler for Silverstripe”-lisäosasta, sillä se muuttaa CSS-tyylitiedostojen käytöstä ja tapaa käyttää SilverStripessa.

Lisäosan tekemä muutos johti siihen, että SilverStripe-sivusto ei onnistunut löytämään hakemiaan tyylitiedostoja. Tämän sai ratkaistua tuomalla paikallisesta toteutuksesta `scss.yml`-tiedosto SilverStripen kokoonpanokansioon ja ajamalla ”`vendor/bin/sake dev/build`”-komento. `Scss.yml`-tiedostossa määritetään teemakansion sijainti, jotta käännettyssä CSS-tiedostossa tiedostoihin osoittavat osoitteet osoittavat oikeaan paikkaan.

Samalla muistin käyttämäni SilverStripe Menumanager -lisäosan, joka mahdollistaa useamman navigaatiovalikon SilverStripe-sivuilla, tarvitsevan `menusets.yml`-tiedoston. `Menuets.yml`-tiedostossa määritetään lisäosan hallitsevat valikot, jotka luodaan `dev//build`-komento ajettaessa. Toin tämänkin tiedoston palvelimelle paikallisesta toteutuksesta, minkä jälkeen annoin SilverStripen tehdä tarvittavat muutokset ajamalla uudelleen ”`vendor/bin/sake dev/build`”-komennon. Lisäksi ajoin vielä SilverStripen ”`composer vendor-expose`”-komennon, jonka tarkoitus on tehdä Silverstripe-asennuksen `public`-kansiossa olevista staattisista tiedostoista julkisia, jotta selain pääsee niihin käsiksi sivustolle navigoidessa.

Kun olin saanut komennot suoritettua, siirryin kirjautumaan sivuston sisällönhallintaan tarkoituksena luoda tarvittavat sivut ja asettaakseni ne luotuihin Menumanager-valikoihin. Huomasin kuitenkin hyvin nopeasti sivustolle päästyäni unohtaneeni jotain. SilverStripe käyttää MVC-sovellusarkkitehtuuria ja ainoastaan View-osa tästä kokonaisuudesta oli mukana aikaisemmin siirtämissäni teematiedostoissa. SilverStripen Model- ja Controller-osat sijaitsevat erillään `app`-kansion alakansiossa `src`. Huomattuani, että en ollut vielä siirtänyt `src`-kansion

sisältöjä, siirsin ne ja ajoin "vendor/bin/sake dev/build"-komennon saadakseni muutokset näkymään.

Siirryin tekemään viimeisiä muutoksia sivulla, ennen kuin luovuttaisin sivuston asiakkaalle. Törmäsin kuitenkin sivujen luonnissa ja muokkaamisessa taas uuteen ongelmaan, sillä SilverStripe ei antanut muokata sivuja ja kieltäytyi avaa-
masta yhtäkään sivua yrittäessäni avata niitä sisällönhallintapuolella. SilverStripe ilmoitti ainoana virheilmoituksena "Internal Server Error" ilman tarkennuksia, ja saadakseni tarkempaa tietoa kohdatusta ongelmasta, yritin saada SilverStripen vianetsintätyökalut käyttööni seuraamalla virallista ohjeistusta. Tämä ei kuitenkaan onnistunut. Lopulta kokeiltuani useita erilaisia ratkaisuja ongelmaan, mukaan lukien verkkopalvelin- ja tietokantaohjelmistojen uudelleenasetaminen, ajattelin, että ongelma saattaisi johtua SilverStripen public-kansion käyttöoikeuksista. Päätin kokeilla muuttaa public-kansion oikeudet muotoon 775 komennolla "chmod 775 public". Suoritettuani komennon avautui sivujen muokausnäkyvä tavalliseen tapaan ja sivujen luonti onnistui ilman ongelmia. Kävin vielä sivuston läpi varmistaakseni sen toimivan kunnolla, ennen kuin luovutin sivuston asiakkaalle.

5 Yhteenveto

Insinööriyössä tutustuttiin suosittuihin verkkosivujen luontiin ja hallintaan tarkoitettuihin sisällönhallintajärjestelmiin sekä niiden eroihin ja suunniteltiin ja toteutettiin blogin pitoa varten verkkosivusto käyttäen SilverStripe-sisällönhallintajärjestelmää. Ennen sivuston suunnittelun aloitusta täytyi löytää sisällönhallintajärjestelmä, joka sopi asiakkaan vaatimuksiin ja jonka kanssa asiakas tuntisi olonsa mukavaksi ilman, että tulevaisuuden mahdollisuudet sivustolle tuntuisivat rajoitetuilta.

Asiakas oli tarkka siitä, että hän ei halua Wordpress-sivustoa, vaikka tiesikin sen sopivan hyvin haluamansa sivuston tarpeisiin. Osittain tämä johtui Wordpressin suuresta suosiosta ja osittain käyttöliittymän ulkonäöstä. Lopulta ehdotettuani työn alkuvaiheessa vielä työpaikallani käytettyä SilverStripe-sisällönhallintajärjestelmää ja esiteltyäni erästä sillä tehtyä sivustoa, asiakas sai päätettyä haluavansa SilverStripe-pohjaisen verkkosivun.

Sivuston suunnittelu tehtiin tiiviissä yhteistyössä asiakkaan kanssa, ja sivustosta suunniteltiin yksinkertainen, jotta asiakkaan olisi helppo palata sen pariin myöhemmin. Suunnitteluprosessi oli nopea ja vaivaton. Sivusto rakennettiin ensin paikallisesti, jolloin sen värimaailma muuttui täysin tummista sinisen ja purpuran sävyistä kirkkaisiin keltaisiin ja kultaan. Samalla syntyi sivuston logo ja sivuston elintärkeä blogiominaisuus päätettiin toteuttaa valmiina olevalla lisäosalla.

Asiakkaan todettua sivuston siirtovalmiiksi alettiin sitä siirtää asiakkaan palvelimelle. Tätä varten asiakkaan annettua oikeudet palvelimelleen asennettiin palvelimelle verkkosivun isännöinnin mahdollistavat ja kehitystyötä helpottavat ohjelmistopakettit. Sivuston siirrossa esiintyi hieman odottamattomia ongelmia, kun sivustolla käytettävät lisäosat täytyi asentaa yksitellen eivätkä palvelimella sijaitsevien julkisiksi tarkoitettujen hakemistojen oikeudet muuttuneet SilverStripen komennoilla vaan ne täytyi käydä muuttamassa manuaalisesti palvelimella.

Työn aikana SilverStripe osoittautui hyväksi vaihtoehdoksi henkilölle, joka ei pelkää itse käydä käsiksi sivuston taustalla olevaan koodiin, sillä pienen käyttäjäkunnan ja siitä johtuvan valmiiden ratkaisujen pienen määrän vuoksi SilverStripe-sivuston kehittäjä saattaa hyvin joutua rakentamaan haluamiaan ominaisuuksia itse. Näin ollen SilverStripesta ei ole suoraksi tai ainakaan kovin voimakkaaksi kilpailijaksi Wordpressin kaltaiselle hyvin helppokäyttöiselle sisällönhallintajärjestelmälle, eikä varsinkaan Squarespacen ja Wixin tyyppisille kotisivukoneille.

Toisaalta SilverStripe soveltuu hyvin tilanteisiin, joissa tavallisesti saatettaisiin käyttää Joomlaa tai Drupalia, eli projekteissa, joissa on aikaa kehittää jotain uutta ja erikoista. SilverStripen MVC-arkkitehtuuri ja selkeä tiedostorakenne tekevät siitä nopeasti kehitettävän, joskin tässäkin tilanteessa valmiiden ratkaisujen puute on ongelma, vaikkakin pienempi kuin verratessa Wordpressiin.

SilverStripen suurimmat ongelmat ovat kuitenkin sen dokumentaation määrä ja laatu, joka kävi selväksi, kun sivuston siirron loppupuoolella SilverStripen sisällönhallintaosio ei toiminut, sekä sen käyttäjäkunnan koko. SilverStripen dokumentaatio on kehittäjäkunnan vastuulla, ja koska kehittäjäkunta on pieni, on dokumentaatio vajavaista, mikä tekee sen kanssa työskentelystä ajoittain työläämpää kuin mitä sen tarvitsisi olla. SilverStripen käyttäjäkunnan pienuuden takia on epäuskottavaa, että vastaus ilmenneeseen ongelmaan löytyy nopealla verkkohauulla.

SilverStripe on silti täysin varteenotettava sisällönhallintajärjestelmä, joka saattaa hyvin lähteä nousemaan suosiossa tai käytössä SilverStripe Oy:n Uuden-Seelannin hallinnon kanssa tekemän sopimuksen kaltaisten sopimusten vaikutuksesta, jolloin hallintoelimien vaatimukset ja käyttäjäkunnan kasvu saattavat tuoda helpotusta SilverStripen dokumentaatio-ongelmaan.

Lähteet

- 1 Historical yearly trends in the usage statistics of content management systems. Verkkoaineisto. W3techs. <https://w3techs.com/technologies/history_overview/content_management/all/y>. Luettu 15.3.2021.
- 2 Usage statistics of content management systems. Verkkoaineisto. W3techs. <https://w3techs.com/technologies/overview/content_management>. Luettu 4.2.2021.
- 3 Usage statistics of content management systems. Verkkoaineisto. W3techs. <https://w3techs.com/technologies/overview/content_management>. Luettu 6.2.2021.
- 4 Usage statistics of content management systems. Verkkoaineisto. W3techs. <https://w3techs.com/technologies/overview/content_management>. Luettu 4.2.2021.
- 5 Usage statistics of server-side programming languages for websites. Verkkoaineisto. W3techs. <https://w3techs.com/technologies/overview/programming_language>. Luettu 5.2.2021.
- 6 Usage statistics of content management systems. Verkkoaineisto. W3techs. <https://w3techs.com/technologies/overview/content_management>. Luettu 25.1.2021.
- 7 The History of WordPress, its Ecosystem and Community. 2021. Verkkoaineisto. Kinsta Incorporated. <<https://kinsta.com/learn/wordpress-history/>>. 19.1.2021. Luettu 25.1.2021.
- 8 GNU General Public License v3. 2007. Verkkoaineisto. Free Software Foundation. <<https://www.gnu.org/licenses/gpl-3.0.html>>. Luettu 25.1.2021.
- 9 Democratize Publishing. Verkkoaineisto. Wordpress.org. <<https://wordpress.org/about/>>. Luettu 25.1.2021.
- 10 Koster, James. Why child themes are so important. Verkkoaineisto. Woocommerce. <<https://woocommerce.com/posts/why-child-themes-matter/>> Luettu 28.03.2021
- 11 Desrosiers, Jonathan. 2020. WordPress and PHP 8.0. Verkkoaineisto. Wordpress.org. <<https://make.wordpress.org/core/2020/11/23/wordpress-and-PHP-8-0/>>. Luettu 25.1.2021.
- 12 Mikä Joomla? Verkkoaineisto. Joomla.fi. <<https://www.joomla.fi>>. Luettu 26.1.2021.

- 13 Thompson, Lewis. 2019. History of Joomla (Full-time Line of the CMS). Verkkoaineisto. <<https://webaroo.com.au/history-of-joomla/>>. 12.2.2019. Luettu 26.1.2021.
- 14 Model-view-controller (MVC) -arkkitehtuuri. Verkkoaineisto. Jyväskylän yliopiston informaatioteknologian tiedekunta. <<http://appro.mit.jyu.fi/web-sovellukset/luennot/mvc/>> Luettu 28.3.2021
- 15 Mikä Joomla? Verkkoaineisto. Joomla.fi. <<https://www.joomla.fi>>. Luettu 26.1.2021.
- 16 Joomla!tools Vagrant. Verkkoaineisto. Joomla!tools.com. <<https://www.joomla!tools.com/developer/vagrant>> Luettu 26.1.2021
- 17 Technical Requirements. Verkkoaineisto. Joomla.org. <<https://downloads.joomla.org/technical-requirements>> Luettu 28.3.2021
- 18 Technical Requirements. 2019. Verkkoaineisto. Open Source Matters Incorporated. <<https://webaroo.com.au/history-of-joomla/>>. 22.2.2019. Luettu 26.1.2021.
- 19 Our history. Verkkoaineisto. Drupal.org. <<https://www.drupal.org/about/history>>. Luettu 3.3.2021.
- 20 Who Uses Drupal? Verkkoaineisto. Drupal.org. <<https://www.drupal.com/showcases>>. Luettu 3.3.2021.
- 21 Download & Extend. Verkkoaineisto. Drupal.org. <https://www.drupal.org/project/project_module>. Luettu 27.1.2021.
- 22 Joomla! Extensions Directory™. Verkkoaineisto. Open Source Incorporated. <<https://extensions.joomla.org>>. Luettu 27.1.2021.
- 23 Plugins. Verkkoaineisto. Wordpress.org. <<https://wordpress.org/plugins/>>. Luettu 27.1.2021.
- 24 Usage statistics and market share of Drupal. 2021. W3Techs. <<https://w3techs.com/technologies/details/cm-drupal>>. 5.2.2021. Luettu 27.1.2021.
- 25 PHP requirements. Verkkoaineisto. Drupal.org. <<https://www.drupal.org/docs/system-requirements/PHP-requirements>> Luettu 27.1.2021.
- 26 Database server requirements. Verkkoaineisto. Drupal.org. <<https://www.drupal.org/docs/system-requirements/database-server-requirements>> Luettu 27.1.2021.
- 27 Lohr, Greg. 2004. University of Maryland student in a class by himself. Verkkoaineisto. <<https://www.bizjournals.com/washington/stories/2004/04/26/focus3.html>>. 22.4.2004. Luettu 27.1.2021.

- 28 Squarespace hinnasto. Verkkoaineisto. Squarespace.com
<<https://www.squarespace.com/pricing/>> Luettu 27.1.2021
- 29 Squarespace hinnasto. Verkkoaineisto. Squarespace.com
<<https://www.squarespace.com/pricing/>> Luettu 27.1.2021
- 30 Garcia, Josep. 2021. Your Ultimate Guide to Squarespace Templates. Verkkoaineisto <<https://www.websitetooltester.com/en/blog/squarespace-templates/>>. 29.2.2021. Luettu 28.1.2021.
- 31 Penttilä, Erno. 2020. HKO: Mitä, miten, onko tärkeää? Verkkoaineisto. <<https://www.verkkokauppablogi.fi/hakukoneoptimointi-mita-miten-onko-tarkeaa/>>. 21.11.2020. Luettu 28.1.2021.
- 32 Miksi hakukoneoptimointi on Startup-yritykselle tärkeää. Verkkoaineisto. Suomen Digimarkkinointi Oy. <<https://www.digimarkkinointi.fi/blogi/hakukoneoptimointi-startup-yritykselle/>>. Luettu 28.1.2021.
- 33 Garcia, Josep. 2020. Is Squarespace Actually SEO Friendly? Verkkoaineisto. <<https://www.websitetooltester.com/en/blog/squarespace-seo/>>. 12.3.2020. Luettu 28.1.2021.
- 34 The Leader in Website Creation. Verkkoaineisto. Wix.com Incorporated. <<https://www.wix.com/about/us>>. Luettu 29.1.2021.
- 35 Jobs, Steve. 2010. Thoughts on Flash. Verkkoaineisto. <<https://medium.com/riow/thoughts-on-flash-1d1c8588fe07>>. Luettu 29.1.2021.
- 36 Gelbmann, Matthias. 2020. Shopify is now the second most popular content management system. Verkkoaineisto. <https://w3techs.com/blog/entry/shopify_is_now_the_second_most_popular_content_management_system>. 25.2.2020. Luettu 4.2.2021.
- 37 Usage statistics and market share of SilverStripe. Verkkoaineisto. W3Techs. <<https://w3techs.com/technologies/details/cm-silverstripe>>. Luettu 1.2.2021.
- 38 History. Verkkoaineisto. SilverStripe Limited. <<https://www.silverstripe.org/software/history/>>. Luettu 1.2.2021.
- 39 Common Web Platform. Verkkoaineisto. New Zealand Government Procurement. <<https://www.procurement.govt.nz/contracts/common-web-platform/>>. Luettu 1.2.2021.
- 40 About CWP. 2020. Verkkoaineisto. Common Web Platform. <<https://www.cwp.govt.nz/about-cwp/>>. 9.9.2020. Luettu 1.2.2021.
- 41 O'Neill, Rob. 2020. Silverstripe wins one-year extension on government Common Web Platform contract. Verkkoaineisto. <<https://www.reseller.co.nz/article/682852/silverstripe-wins-one-year-extension-government-common-web-platform-contract/>>. 11.9.2020. Luettu 1.2.2021.

- 42 BSD License. Verkkoaineisto. SilverStripe Limited. <<https://www.silver-stripe.org/software/bsd-license/>>. Luettu 1.2.2021.
- 43 Requirements. Verkkoaineisto. SilverStripe Limited. <https://docs.silver-stripe.org/en/4/getting_started/server_requirements/>. Luettu 1.2.2021
- 44 Silverstripe: Vulnerability Statistics. Verkkoaineisto. CVE Details. <<https://www.cvedetails.com/vendor/6513/Silverstripe.html>> Luettu 10.3.2021.
- 45 Wordpress: Vulnerability Statistics. Verkkoaineisto. CVE Details. <<https://www.cvedetails.com/vendor/2337/Wordpress.html>> Luettu 10.3.2021.
- 46 Drupal: Vulnerability Statistics. Verkkoaineisto. CVE Details. <<https://www.cvedetails.com/vendor/1367/Drupal.html>> Luettu 10.3.2021.
- 47 Joomla: Vulnerability Statistics. Verkkoaineisto. CVE Details. <<https://www.cvedetails.com/vendor/3496/Joomla.html>> Luettu 10.3.2021.
- 48 Wordpress » Wordpress: Vulnerability Statistics. Verkkoaineisto. CVE Details. <https://www.cvedetails.com/product/4096/Wordpress-Wordpress.html?vendor_id=2337> Luettu 10.3.2021.
- 49 Drupal » Drupal: Vulnerability Statistics. Verkkoaineisto. CVE Details. <https://www.cvedetails.com/product/2387/Drupal-Drupal.html?vendor_id=1367> Luettu 10.3.2021.
- 50 Joomla » Joomla: Vulnerability Statistics. Verkkoaineisto. CVE Details. <https://www.cvedetails.com/product/6129/Joomla-Joomla.html?vendor_id=3496> Luettu 10.3.2021.
- 51 Joomla » Joomla!: Vulnerability Statistics. Verkkoaineisto. CVE Details. <https://www.cvedetails.com/product/16499/Joomla-Joomla-.html?vendor_id=3496> Luettu 10.3.2021.
- 52 CVSS Scores For Wordpress Wordpress Between 2007-01-01 and 2021-03-10. Verkkoaineisto. CVE Details. <https://www.cvedetails.com/cvss-score-charts.php?fromform=1&vendor_id=&product_id=4096&startdate=2007-01-01&enddate=2021-03-10> Luettu 10.3.2021
- 53 CVSS Scores For Drupal Drupal Between 2007-01-01 and 2021-03-10 Verkkoaineisto. CVE Details. <https://www.cvedetails.com/cvss-score-charts.php?fromform=1&vendor_id=&product_id=2387&startdate=2007-01-01&enddate=2021-03-10> Luettu 10.3.2021
- 54 CVSS Scores For Joomla Joomla! Between 2007-01-01 and 2021-03-10. Verkkoaineisto. CVE Details. <https://www.cvedetails.com/cvss-score-charts.php?fromform=1&vendor_id=&product_id=16499&startdate=2007-01-01&enddate=2021-03-10> Luettu 10.3.2021

- 55 CVSS Scores For Silverstripe Products Between 2007-01-01 and 2021-03-10. Verkkoaineisto. CVE Details. <https://www.cvedetails.com/cvss-score-charts.php?fromform=1&vendor_id=6513&product_id=&start-date=2007-01-01&enddate=2021-03-10> Luettu 10.3.2021
- 56 Themes. Verkkoaineisto. silverstripe.org. <https://docs.silverstripe.org/en/4/developer_guides/templates/themes/> Luettu 24.1.2021
- 57 Goel, Adam. 2021. MariaDB vs MySQL: [2021] Everything You Need to Know. Verkkoaineisto. hackr.io. <<https://hackr.io/blog/mariadb-vs-mysql>> Päivitetty 8.1.2021. Luettu 13.3.2021