SAVONIA

OPINNÄYTETYÖ - AMMATTIKORKEAKOULUTUTKINTO
TEKNIIKAN JA LIIKENTEEN ALA

SISÄLLÖNHALLINTAJÄRJESTELMIEN KÄYTTÖTAPOJEN VERTAILU

TEKIJÄ: Markus Matilainen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Tutkinto-ohjelma Tietotekniikan tutkinto-ohjelma	
Työn tekijä(t) Markus Matilainen	
Työn nimi Sisällönhallintajärjestelmien käyttötapojen vertailu	
Päiväys 30.11.2020	Sivumäärä/Liitteet 27/0
Toimeksiantaja/Yhteistyökumppani(t) Hurja Solutions Oy	

Tiivistelmä

Opinnäytetyön tavoitteena oli testata verkkosivujen latausnopeuksia staattisen verkkosivun ja sisällönhallintajärjestelmien kanssa. Työssä selvitettiin, kuinka staattisen verkkosivun käyttäminen sisällönhallintajärjestelmän kanssa vaikuttaa sivuston latausnopeuteen pelkän sisällönhallintajärjestelmän käyttämisen suhteen. Opinnäytetyössä selvitettiin myös, kuinka helppoa on toteuttaa staattisen verkkosivun ja sisällönhallintajärjestelmän yhdistelmä, ja saavuttaa mahdolliset suorituskyvylliset hyödyt. Opinnäytetyössä tuotetut tiedot tulivat opinnäytetyön tilaavan yrityksen käyttöön omiin projekteihin.

Työ toteutettiin kehittämällä staattinen verkkosivu GatsbyJS-frameworkilla, joka mukailee Salmi Platform -sivustoa. Sivusto julkaistiin Netlify-palvelussa, ja siihen liitettiin sisällönhallintajärjestelmiksi Wordpress, sekä Netlify CMS. Latausnopeuden kannalta testattavia käyttötapoja olivat Wordpress-sivusto, Wordpress staattisen sivun kanssa, sekä Netlify CMS staattisen sivun kanssa. Sivustojen latausnopeuteen liittyvät testit toteutettiin käyttämällä Lighthouse-, Pingdom- ja Gtmetrix-palveluita. Staattisen verkkosivun tuottamat vaikutukset latausnopeuteen testattiin liittämällä Wordpress- ja Netlify CMS-sisällönhallintajärjestelmät vuorotellen staattiseen sivuun.

Opinnäytetyön lopputuloksena voitiin osoittaa, kuinka staattisen verkkosivun käyttäminen parantaa sivuston latausnopeutta pelkän sisällönhallintajärjestelmän käytön suhteen. Latausnopeuden parannus oli selkeämpi mobiiliversiossa kuin työpöytäversiossa. Mobiiliversion testaaminen oli kuitenkin puutteellisempaa kuin työpöytäversion. Sisällönhallintajärjestelmien väliset erot eivät olleet merkittäviä latausnopeuksien suhteen staattisen sivun ollessa liitettynä. Työssä voitiin myös osoittaa, kuinka staattisen verkkosivun toteuttaminen ja liittäminen sisällönhallintajärjestelmään on kannattava sijoitus, joka tuo huomattavia hyötyjä.

Avainsanat
GatsbyJS, Wordpress, Netlify, sisällönhallintajärjestelmä, suorituskyky, latausajat

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport	
Degree Programme Degree Programme in Information Technology	
Author(s) Markus Matilainen	
Title of Thesis Comparison of Usage of Different Content Management Systems	
Date 30 November 2020	Pages/Appendices 27/0
Client Organisation /Partners Hurja Solutions Oy	

Abstract

The aim of this thesis was to test the page load times of websites between a static site and content management systems. In the thesis it was investigated how using a static site with a content management system effects the loading speed of the sites instead of using only a content management system. It was also investigated how easy it is to implement a combination of a static website and content management system, and to reach possible performance benefits. The information gathered in the thesis was used by the client organisation in their own projects.

The thesis was done by developing a static website with the GatsbyJS framework, which is based on the Salmi Platform website. The website was published on Netlify, and the content management systems that were attached to it were Wordpress and Netlify CMS. The different ways of using content management systems that were tested were Wordpress-site, Wordpress with a static site, and Netlify CMS with a static site. The tests related to page load times were done by using Lighthouse, Pingdom and Gtmetrix. The effect of static websites on page load times was tested by attaching Wordpress and Netlify CMS content management systems separately to the static site.

As a result of the thesis, it was pointed out how using a static website improves the page load time of the website instead of using only a content management system. The improvement in page load times was clearer in a mobile version than in a desktop version. The testing of the mobile version was more insufficient than desktop versions. The differences between content management systems related to page load times were not significant when the static site was attached. It was made clear how implementing and attaching a static site to a content management system is a worthwhile investment, which brings notable benefits.

Keywords
GatsbyJS, Wordpress, Netlify, content management system, performance, pageload speeds

ESIPUHE

Tahdon kiittää Hurja Solutions Oy:tä mielenkiintoisesta opinnäytetyöstä, joka oli sekä sopivan haastava, että ammattitaitoa kehittävä. Kiitokset Jarno Airaksiselle ja Jesper Ruuthille ohjauksesta, sekä neuvoista. Kiitokset myös Jukka Kinnuselle opinnäytetyön ohjaajana toimimisesta.

Kuopiossa 30.11.2020
Markus Matilainen

SISÄLTÖ

TERMIT JA LYHENTEET

Sisällönhallintajärjestelmä (CMS) on tietojärjestelmä, jonka tarkoituksena on palvella koko organisaation sisällönhallintaa. Sisällönhallintajärjestelmiä ovat esimerkiksi Wordpress, Drupal ja Joomla

Headless CMS on sisällönhallintajärjestelmä, jossa käyttöliittymä eli front-end on irrotettu back-endistä. Esimerkiksi Contentful ja Netlify CMS

GatsbyJS on Reactiin pohjautuva Javascript framework staattisten verkkosivujen kehittämiseen

Front-end tarkoittaa verkkosivuston ”selainpuolta”, eli verkkoselaimessa ajettavaa, käyttöliittymän tuovaa koodia

Back-end tarkoittaa verkkosivuston ”palvelinpuolta”, eli sivuston palvelimella ajettavaa koodia

JAMStack tarkoittaa web-palvelimesta riippumatonta verkkosivustojen kehitystapaa

Bounce rate tarkoittaa välitöntä poistumisprosenttia, eli osuutta sivustolle saapuvista kävijöistä, jotka poistuvat sivustolta ilman sen alasivujen avaamista

Single-page application (SPA) on web-sovellus, joka uudelleenkirjoittaa avoinna olevan sivun uudella palvelimelta haetulla tiedolla sen sijaan, että avaisi täysin uuden sivun

Application programming interface (API) tarkoittaa ohjelmointirajapintaa, joka mahdollistaa erilaisten ohjelmien keskustelun keskenään tietoja vaihtamalla

Distributed Denial of Service (DDoS) eli hajautettu palvelunestohyökkäys tarkoittaa useista lähteistä tapahtuvaa verkkohyökkäystä, jonka tarkoituksena on estää verkkosivuston käyttö

PHP (PHP: Hypertext Preprocessor) tarkoittaa Perlin tapaista ohjelmointikieltä, jolla luodaan dynaamisia web-sivuja

MySQL on relaatiotietokantaohjelmisto, jota käytetään erityisesti web-palveluissa

1 JOHDANTO

Opinnäytetyön tilaava yritys on Hurja Solutions Oy, joka on Kuopiossa toimiva ohjelmistoalan yritys. Opinnäytetyön toimeksiantajana toimii yrityksen operatiivinen johtaja Jarno Airaksinen.

Opinnäytetyön aihe saatiin käynnissä olevan harjoittelujakson aikana. Se nähtiin luonnollisena jatkona harjoittelun aikana opituille asioille, ja sopivana keinona saada tietoa erilaisista web-teknologioista, staattisten verkkosivujen kehityksestä, sekä niiden vaikutuksesta suorituskykyyn ja sivuston latausnopeuteen. Yrityksen tarkoituksena on mahdollisesti hyödyntää saatuja tietoja tulevissa projekteissaan.

Opinnäytetyössä tarkastelun kohteena on sivuston latausnopeus, sillä se on oleellinen osa käyttäjäkokemusta. Sillä on myös suuri vaikutus sivuston bounce rateen, eli osuuteen vierailijoista, jotka poistuvat avaamatta alasivuja. Jamstack-tekniikalla kehitettyjen staattisten verkkosivujen tiedetään latautuvan dynaamisia nopeammin, tuoden näin paremman suorituskyvyn.

Opinnäytetyön tarkoituksena on toteuttaa itse staattinen verkkosivu, joka mukailee Salmi Platformin verkkosivua. Sivusto julkaistaan Netlify-hostingpalvelussa, jonka jälkeen siihen liitetään Wordpress ja Netlify CMS sisällönhallintajärjestelmiksi. Lopuksi suorituskykyä testataan Lighthousen, Pingdomin ja Gtmetrixin avulla. Tarkoituksena on testata, kuinka suuria suorituskyvyllisiä etuja staattisen verkkosivun liittäminen sisällönhallintajärjestelmään toisi sen suhteen, että käytettäisiin pelkkää sisällönhallintajärjestelmää sekä front-endissä, että back-endissä. Tarkoituksena on myös selvittää, kuinka iso työ staattisen verkkosivun ja sisällönhallintajärjestelmän yhdistelmän toteuttaminen on, ja kuinka helposti mahdollisista suorituskyvyllisistä eduista pääsee hyötymään.

2 KÄYTETYT OHJELMISTOT JA TEKNOLOGIAT

2.1 Wordpress

Wordpress on avoimeen lähdekoodiin perustuva, WWW-sisällönhallintaan painottuva sisällönhallintajärjestelmä. Wordpress julkaistiin vuonna 2003 ja se on tämän hetken suosituin WWW-sisällönhallintajärjestelmä. Sen suosiosta kertoo se, että 38% kaikista verkkosivustoista käyttää Wordpressiä sisällönhallintajärjestelmänään (WordPress.com, Wordpress, n.d.). Sen toiminta perustuu julkaisupainotteiseen sisällönhallintaan, ja sitä voidaan käyttää mm. erilaisten blogikirjoitusten ja verkkosivujen julkaisemiseen. Wordpressin toimintaa voidaan laajentaa ja muokata erilaisilla lisäosilla ja teemoilla.

Wordpressin tarkoituksena on olla helppokäyttöinen ja joustava, jolloin uuden blogin tai verkkosivuston perustamiseen ei tarvita paljoa teknistä osaamista, mutta kokeneet käyttäjät voivat muokata sitä lukemattomilla eri tavoilla. Wordpressin pohjana toimivat PHP ja MySQL. (WordPress.org, About Us: Our Mission, n.d.)

Wordpressiin on myös mahdollista luoda omia teemoja. Lisäosilla voidaan laajentaa Wordpressiä esimerkiksi lisäämällä siihen foorumeita, roskapostisuodattimia, gallerioita, lomakkeita ja hakukoneoptimointiin liittyviä ominaisuuksia. Wordpressillä on myös hyvin aktiivinen yhteisö, jolta voi saada tukea ongelmatilanteissa. (WordPress.org, Features, n.d.)

2.2 Netlify CMS

Netlify CMS on avoimeen lähdekoodiin perustuva, staattisten verkkosivujen kanssa käytettävä sisällönhallintajärjestelmä. Se perustuu Jamstack-kehitystyyliin ja mahdollistaa staattisten sivujen nopeuden, turvallisuuden ja skaalautuvuuden. Netlify CMS toimii yhden sivun web-sovelluksen tavoin, joka on kirjoitettu Reactilla. Sisällönhallintajärjestelmän sisältö, eli mm. blogikirjoitukset varastoidaan projektin git-varastoon. (NetlifyCMS, NetlifyCMS, n.d.)

Netlify CMS on headless-tekniikkaan perustuva sisällönhallintajärjestelmä, joka voidaan julkaista esimerkiksi Netlifyssä olevan sivuston kanssa (NetlifyCMS, NetlifyCMS Overview, n.d.).

2.3 Netlify

Netlify on staattisten verkkosivujen julkaisuun tarkoitettu palvelu, joka tarjoaa nopean ja helpon tavan julkaista sekä ylläpitää sivustoja. Sivuston pyörittämiseen ei tarvita palvelimia tai kallista infrastruktuuria, ja se on mahdollista julkaista erilaisten ohjelmavarastojen kautta. Sivusto voidaan julkaista Githubin, Gitlabin tai Bitbucketin ohjelmavarastosta. (Netlify, Netlify, 2020.)

Sivuston uudemman version julkaiseminen tapahtuu helposti antamalla git push -komento liitettyyn varastoon, jonka jälkeen Netlify havaitsee muutoksen varastossa ja rakentaa sekä julkaisee sivuston automaattisesti palvelussaan (Netlify, Netlify Build, 2020).

Netlifyn asiakkaita ovat mm. Google, Facebook ja Samsung. Sen tuotteisiin kuuluu myös Netlify CMS-sisällönhallintajärjestelmä. Myös Jamstack-arkkitehtuuri on Netlifyn kehittämä. (Wikipedia, 2020.)

2.4 GatsbyJS

GatsbyJS on Reactiin pohjautuva, avoimen lähdekoodin Javascript framework staattisten verkkosivustojen luomiseen. Sen keskeisimpiä etuja ovat suorituskyky, skaalautuvuus ja turvallisuus. Gatsbyn nopeus perustuu sen tapaan ladata vain kriittisimmät osat sivustosta. Gatsby noutaa etukäteen muiden sivujen resurssit, jolloin sivulta toiselle siirtyminen käy nopeasti. Gatsbyn sivut julkaistaan keskimäärin 2.5 kertaa nopeammin kuin muilla staattisilla frameworkeilla, sillä se välttää uudelleensuorittamasta tarpeettomia osia sivun rakentamisen aikana. Kaiken tämän tarkoituksena on mahdollistaa sisältökeskeisten sivujen toteuttaminen huolehtimatta suorituskyvystä.

Sivustolle voi integroida dataa useista eri lähteistä, kuten ohjelmointirajapinnoista, tietokannoista, sisällönhallintajärjestelmistä tai staattisista tiedostoista. Gatsbyn turvallisuuteen vaikuttaa olennaisesti palvelimen ja tavoitettavissa olevan tietokannan puuttuminen, jolloin sivulla ei erityisemmin ole hyökkäyspintaa, eikä näin ollen myöskään haitallisia pyyntöjä tai DDoS-hyökkäyksiä pystytä toimittamaan. Gatsbya on mahdollista laajentaa erilaisilla laajennuksilla, joita voi käyttää esimerkiksi datan siirtoon. Niiden avulla voidaan myös liittää sisällönhallintajärjestelmiä.

Toukokuussa 2020 tehdyn tutkimuksen mukaan GatsbyJS oli kaikkein nopeimmin kasvava framework erikokoisten organisaatioiden keskuudessa. (Gatsby, GatsbyJS, 2020.) Gatsbyllä luodut sivustot on mahdollista julkaista esimerkiksi Netlifyssä (Netlify, The fastest way to build the fastest Gatsby sites, 2020). Gatsby tarjoaa myös Gatsby Cloud -palvelua (Gatsby, Gatsby Cloud, 2020).

2.5 Visual Studio Code

Visual Studio Code on ilmainen Microsoftin kehittämä, ohjelmoijille tarkoitettu avoimen lähdekoodin tekstieditori. Sen toiminnallisuutta on mahdollista laajentaa lukuisilla eri laajennuksilla. Niillä voidaan lisätä editoriin esimerkiksi uusia kieliä, teemoja tai virheenjäljittimiä. Editorissa on sisäänrakennettu tuki Git-versionhallinnalle ja komentorivi komentojen suorittamiseen. VSCode on saatavilla Windows-, MacOS- ja Linux-käyttöjärjestelmille. (Microsoft, 2020.)

2.6 XAMPP

XAMPP on ilmainen ja suosittu PHP-kehitysympäristö, jonka nimi tulee sanoista cross-platform, Apache, MariaDB/MySQL, PHP ja Perl (Apache Friends, 2020). Työssä käytetyssä versiossa oli MySQL-tietokanta. XAMPP:ia käytettiin paikallisen Wordpress-asennuksen pyörittämiseen. XAMPP on saatavilla Windowsille, OSX:lle ja Linuxille.

2.7 GraphQL

GraphQL on API:lle, eli ohjelmointirajapinnalle suunniteltu kyselykieli. GraphQL:n ohjelmointirajapinnat voivat toimittaa kaiken tarvittavan datan yhdellä pyynnöllä, ja ne toimivat nopeasti myös hitailla matkapuhelinverkoilla. Rajapinnat organisoidaan tyyppien ja kenttien perusteella, ja käyttäjä voi pyytää vain niitä tietoja, joita tarvitsee. (The GraphQL Foundation, GraphQL, 2020.)

GraphQL ei ole sidoksissa mihinkään tiettyyn tietokantaan, vaan data saadaan jo olemassa olevasta koodista ja datasta (The GraphQL Foundation, Introduction to GraphQL, 2020).

Wordpressille on saatavilla myös WPGraphQL-lisäosa, joka tarjoaa GraphQL-skeeman ja ohjelmointirajapinnan Wordpress-sivustolle (WPGraphQL, 2018).

Toinen hyödyllinen lisäosa Wordpressille on WPGraphiQL, joka lisää ohjelmointiympäristön Wordpressin hallintasivulle, jonka avulla voi testata GraphQL-kyselyjä Wordpressin GraphQL-skeemasta. Sitä käytetään yhdessä WPGraphQL-lisäosan kanssa. (WPGraphiQL, 2019.)

2.8 Git ja Github

Git on versionhallintaohjelmisto, jonka avulla voidaan kirjata ohjelmiston koodiin tehtävät muutokset. Ohjelmassa voidaan palata viimeisimpään toimivaan versioon, mikäli ohjelman toiminnassa havaitaan ongelmia. Git on hyödyllinen erityisesti, kun useampi henkilö työskentelee saman projektin parissa. (Chacon, n.d.)

GitHub on säilytyspaikka Git-versionhallinnan kanssa käytetyille projekteille. GitHub tarjoaa käyttäjälleen lisäksi graafisen käyttöliittymän. (GitHub, 2020.)

2.9 Testaustyökalut

Suorituskyvyn testauksessa käytettiin Lighthouse, Pingdom ja Gtmetrix-työkaluja. Lighthousella voidaan testata esimerkiksi verkkosivuston suorituskykyä ja SEO:ta, sekä antaa niistä raportti käyttäjälle. Sillä on myös mahdollista testata sekä sivuston työpöytä että mobiiliversiota. Pingdom ja Gtmetrix keskittyvät myös suorituskyvyn testaamiseen. Lighthousessa ja Gtmetrixissä testaaminen tapahtuu manuaalisesti, jolloin käyttäjän on itse käynnistettävä testit. Pingdom keskittyy monitorointiin, mikä tarkoittaa sitä, että palvelun asetuksiin määritetään tarkasteltava sivusto, jonka jälkeen palvelu kerää sivustosta suorituskykyyn liittyviä tietoja säännöllisin väliajoin. Ilmaisversiossa tietojen hakeminen tapahtuu puolen tunnin välein. (SolarWinds Worldwide, LLC., 2020.)

Lighthouse on ainut palvelu, johon ei tarvitse rekisteröityä, kun taas Pingdom vaatii rekisteröitymisen, jotta monitorointipalvelua voi käyttää. Gtmetrix ei vaadi rekisteröitymistä, mutta se on suositeltavaa, koska silloin voi muokata suorituskykytestauksissa käytettäviä asetuksia, kuten vaikka testauspalvelimen sijaintia. Rekisteröityminen antaa myös mahdollisuuden suorittaa sivustoille tehtävät suorituskykytestaukset ilman jonottamista. (GTmetrix, 2020.)

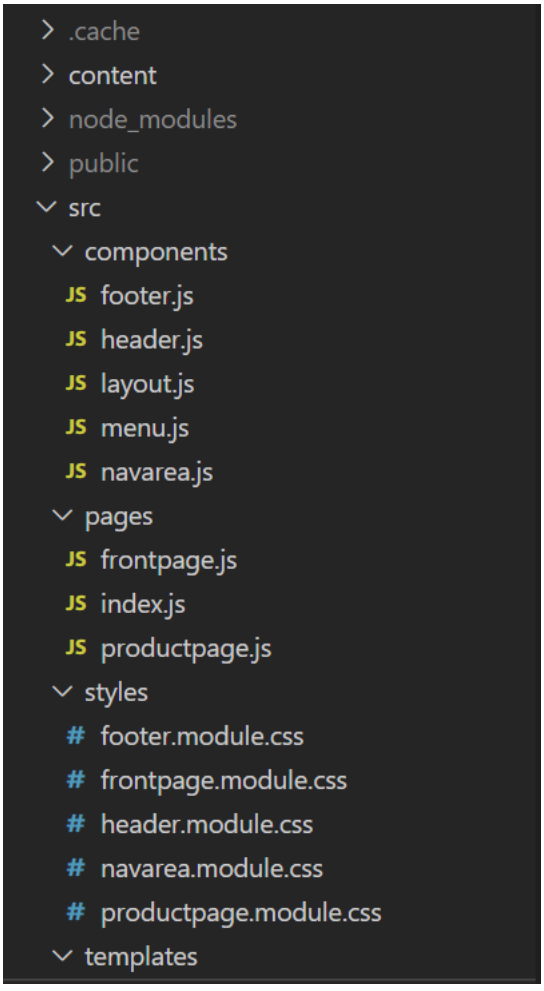
Lighthouse toimii Google Chromen Kehittäjän työkalut -osiossa. Siitä on saatavissa myös komentoriviversio, verkkosivulla toimiva versio, sekä Chrome-laajennus. (Google LLC, 2020.) Pingdom ja Gtmetrix toimivat selaimessa verkkosivun kautta.

3 TYÖN TOTEUTUS

3.1 Gatsby-sivun toteuttaminen

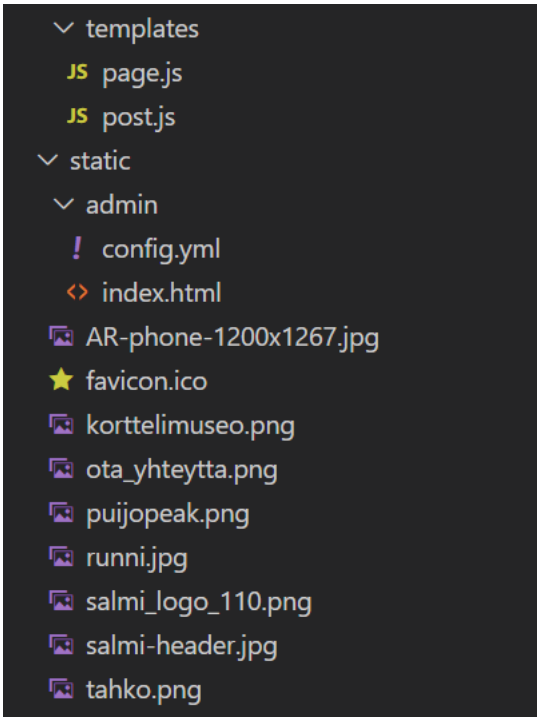
Ensimmäinen työvaihe oli sivuston toteuttaminen GatsbyJS:llä, joka on Reactiin pohjautuva Javascript framework. Toteutuksessa mukailtiin Salmi Platformin sivua, eli siitä tehtiin mahdollisimman samanlainen. Kyseiseltä sivustolta toteutettiin etusivu ja tuotesivu.

Gatsby-projektin rakenne oli seuraavanlainen. Kansion ”src” sisällä oleva kansio ”components” sisältää kaikki sivuilla toistuvat elementit. Komponentteina projektissa toimivat ”header”, ”footer”, ”layout”, ”menu” ja ”navarea”. Kansiossa src/pages ovat projektin sivut, eli etusivu ”frontpage”, tuotesivu ”productpage”, sekä aloitussivu ”index” Javascript-tiedostoina. Kansio src/styles sisältää sivujen ja komponenttien tyylit css- muodossa. Alla olevassa kuvassa (KUVA 1) esitetään projektin komponentit, sivut ja tyylitiedostot.



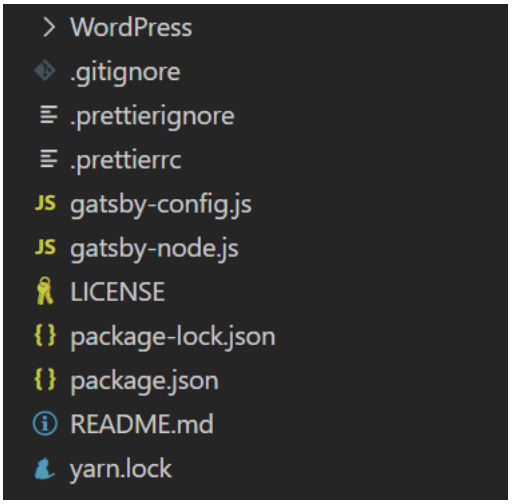
KUVA 1. Ote Gatsbyn projektirakenteesta

Kansio ”templates” sisältää Wordpressistä haettavien sivujen ja postausten pohjat (KUVA 2). Kansiossa static/admin on Netlify CMS-sisällönhallintajärjestelmän liittämiseen kuuluvia tiedostoja. Static-kansio sisältää myös kuvat, joita sivuilla käytetään.



KUVA 2. Ote Gatsbyn projektirakenteesta

Tiedosto ”gatsby-config.js” sisältää projektiin määritellyt lisäosat ja niiden asetukset. Wordpressin sivujen ja blogipostausten hakemiseen käytettävät GraphQL-kyselyt löytyvät ”gatsby-node.js” -tiedostosta. Kaikki projektin riippuvuudet löytyvät ”package.json” -tiedostosta. Alla olevassa kuvassa (KUVA 3) näkyvät kyseiset tiedostot projektirakenteessa.



KUVA 3. Ote Gatsbyn projektirakenteesta

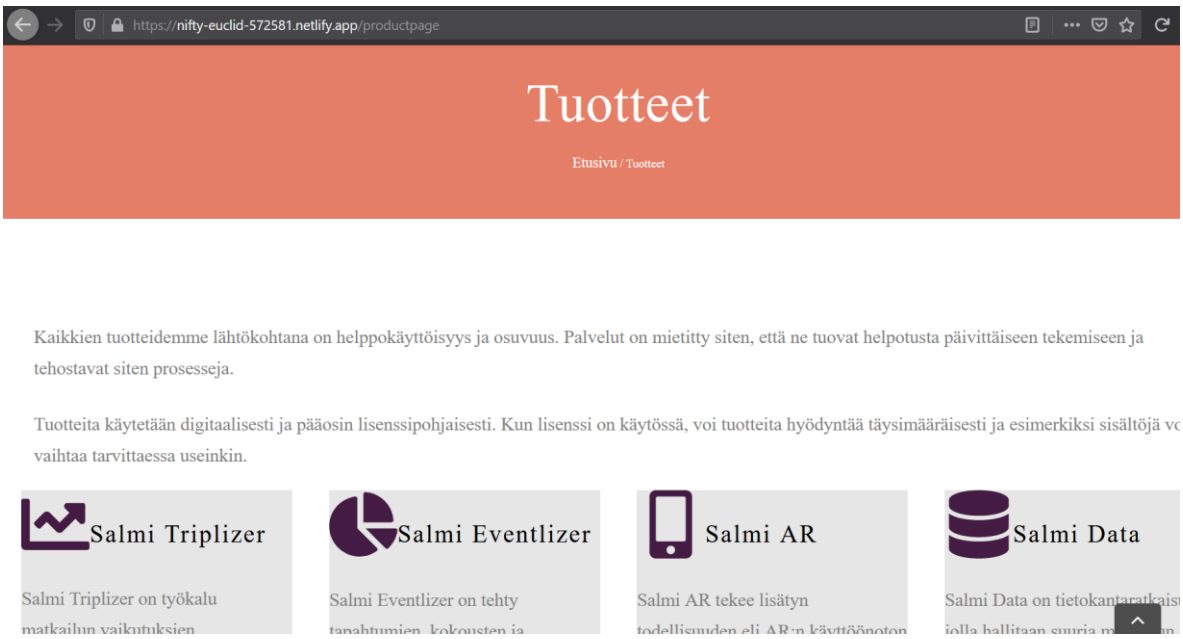
3.2 Sivuston julkaiseminen Netlify-palvelussa

Gatsby-projektille luotiin oma säilytyspaikka GitHub-palveluun, jonka varastoon kaikki projektin työstämisen aikana tehdyt muutokset toimitettiin säännöllisesti tekemällä projektille commit ja push Git-versionhallinnalla. Paikallisen Gatsby-sivuston valmistumisen jälkeen kaikki viimeisimmät muutokset toimitettiin myös GitHubin varastoon. Sivuston julkaisu Netlify-palvelussa toteutettiin kirjautumalla sivulle GitHubin käyttäjätunnuksilla ja valitsemalla vaihtoehto, jonka avulla voimme julkaista suoraan GitHubin ohjelmavarastossa olevan sivuston Netlifyssä.

Seuraavaksi määriteltiin sivustoon liittyvät asetukset, kuten build-komennoksi valittu ”gatsby build”. Netlify suorittaa komennon sivustolle automaattisesti, kun GitHubin ohjelmavarastoon toimitetaan projektin uusimmat muutokset. Näin ohjelmavaraston viimeisimmät muutokset tulivat samalla näkyviin Netlifyssäkin. Alla olevassa kuvassa (KUVA 4) näkyy sivuston etusivu.

KUVA 4. Netlifyssä julkaistun sivun etusivu

Alla oleva kuva (KUVA 5) sisältää näytteen sivuston tuotesivulta.



KUVA 5. Netlifyssä julkaistun sivun tuotesivu

3.3 Wordpressin liittäminen sivustoon

Kun sivusto saatiin julkaistua Netlifyssä, aloitettiin sisällönhallintajärjestelmien liittäminen. Opinnäytetyön tilaavalta yritykseltä, joka hallinnoi Salmi Platformin sivustoa, saatiin kopio sen Wordpress-sivustosta ja tietokannasta. Zip-muodossa ollut Wordpress-sivuston varmuuskopiotiedosto purettiin XAMPP:in htdocs-kansioon, jossa kaikki paikallisesti toimivat sivustot sijaitsevat. Seuraavaksi tietokanta tuotiin sen omasta varmuuskopiotiedostosta menemällä phpMyAdmin-palveluun, ja käyttämällä sen import-toimintoa.

Wp-config -tiedostoon muutettiin kaikki asetukset vastaamaan paikallisen asennuksen tarpeita. Tiedostoon määritettiin tietokannan nimi, käyttäjätunnus ja salasana, sekä MySQL-palvelin. phpMyAdmin-palveluun piti vielä päivittää Wordpress-sivuston vanhan osoitteen tilalle uusi osoite. Näiden toimenpiteiden jälkeen sivusto aukeaa verkkoselaimessa osoitteessa

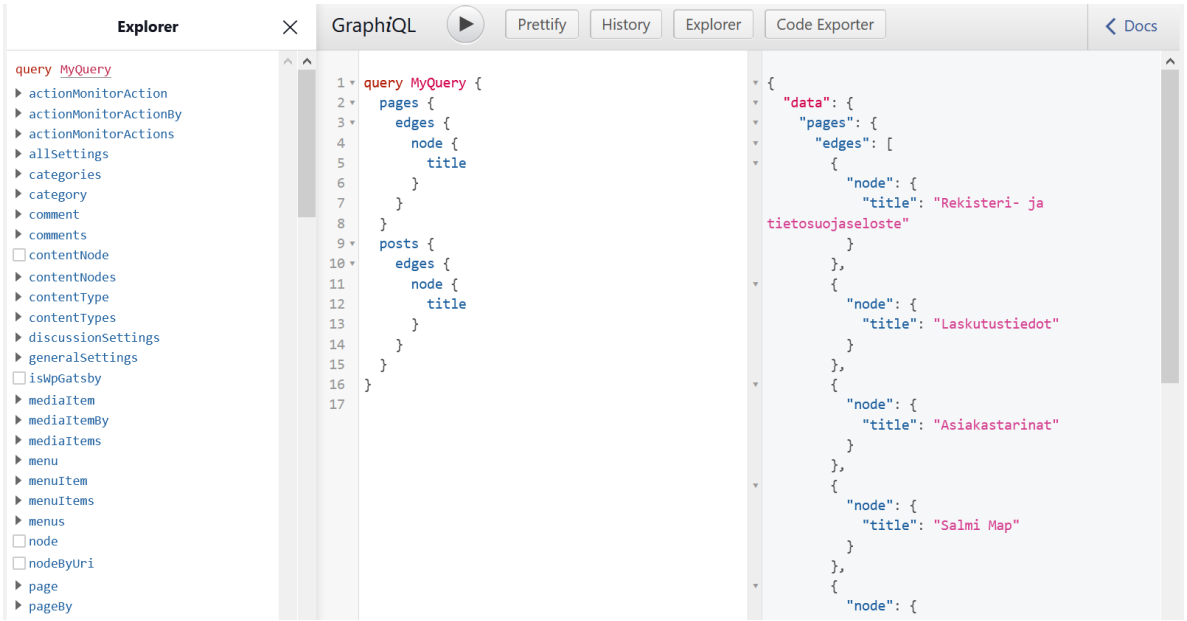
http://localhost/new_wp/ (KUVA 6).

KUVA 6. Paikallisesti asennettu Wordpress-sivusto

Seuraavaksi Wordpress-sivustolle asennettiin kaikki tarvittavat lisäosat, jotta se saataisiin toimimaan Gatsby-projektin kanssa. Työssä tarvittavat lisäosat olivat WPGatsby, WPGraphQL ja WPGraphiQL. Niiden asentaminen tapahtui lataamalla lisäosat zip-tiedostoina, jonka jälkeen ne purettiin projektin plugins-kansioon. Lopuksi ne aktivoitiin Wordpressin hallintapaneelin kautta lisäosat-valikosta. Hallintapaneeliin päästään kirjautumalla sisään osoitteessa http://localhost/new_wp/wp-admin.

WPGatsby säätää Wordpressin toimimaan Gatsbyn kanssa, ja mahdollistaa niiden välisen tiedonvaihdon. WPGraphQL antaa Wordpress-sivustolle GraphQL-skeeman ja ohjelmointirajapinnan, ja WPGraphiQL antaa Wordpressiin integroidun ohjelmointiympäristön, jonka avulla GraphQL-kyselyjä voidaan testata suoraan Wordpressissä (KUVA 7).

Kun lisäosat on aktivoitu, Wordpress on valmis toimimaan lähteenä Gatsbylle.



KUVA 7. Wordpressin GraphQL-skeema

Vielä tehtiin tarvittavat säädöt Gatsby-projektiin, jotta tiedot voitiin hakea Wordpressistä. Gatsbyyn asennettiin ensin lisäosa ”gatsby-source-wordpress” komentorivin avulla, jonka jälkeen se määritettiin gatsby-config.js -tiedostoon tarvittavien asetuksien kanssa.

Asetuksista vaihdettiin baseurl, joka on GraphQL-endpointin osoite. Paikallisessa Wordpress-asennuksessa se sijaitsee osoitteessa http://localhost/new_wp/graphql. Paikallisen asennuksen kyseessä ollessa protokollaksi vaihdettiin http.

Tiedostoon gatsby-node.js määritettiin kyselyt, joiden avulla haetaan Wordpress-sivustosta kaikki sivut ja blogipostaukset (KUVA 8).

```
36
37  const pagesQuery = `
38  query PagesQuery {
39      allWordpressPage {
40        edges {
41          node {
42            id
43            slug
44          }
45        }
46      }
47  }
48  `
49
50  const postsQuery = `
51  query PostsQuery {
52    allWordpressPost {
53      edges {
54          node {
55            id
56            slug
57          }
58        }
59    }
60  }
61  `
62
```

KUVA 8. GraphQL-queryt sivujen ja postausten hakemiseen

Kansiossa src/templates olevat page.js (KUVA 9) ja post.js (KUVA 10) -tiedostot toimivat pohjina, jonne määritetään yksittäisten sivujen ja blogipostausten sisältö, kysely jolla ne haetaan ja elementti, jonka sisällä tiedot näytetään. Projektin riippuvuudet löytyvät tiedostosta package.json.

src > templates > JS page.js > ...

```
43
44  export const pageQuery = graphql`
45      query currentPageQuery($id: String!) {
46          wordpressPage(id: { eq: $id }) {
47              title
48              content
49              date
50          }
51          site {
52              id
53              siteMetadata {
54                  title
55                  subtitle
56              }
57          }
58      }
59  `
```

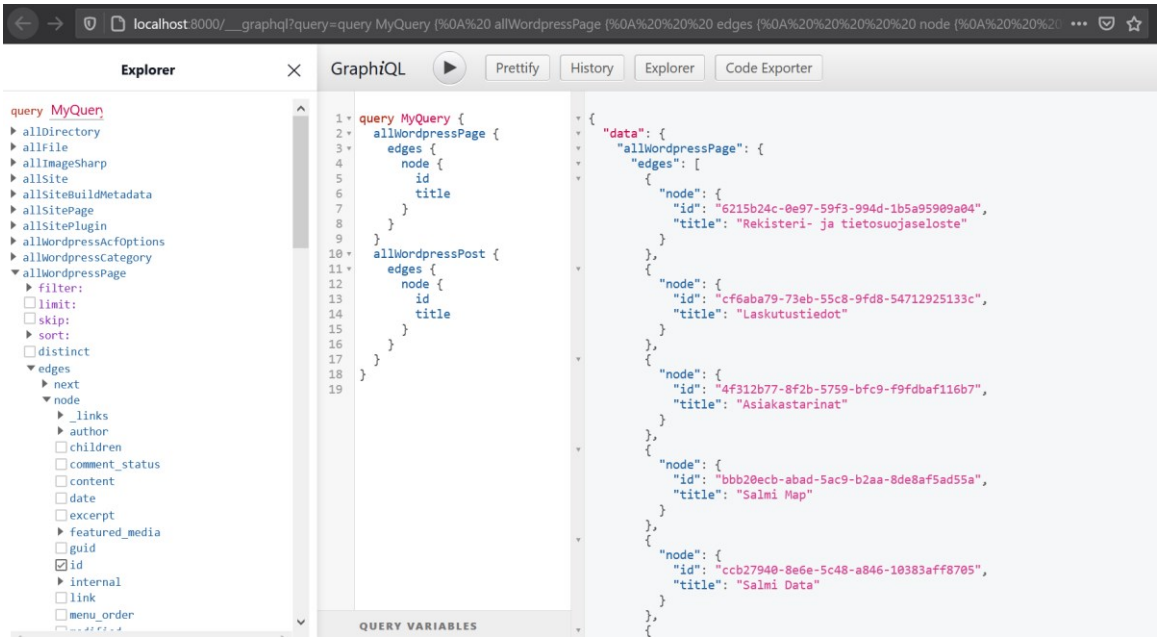
KUVA 9. GraphQL-query yksittäisen Wordpress-sivun tiedoille

src > templates > JS post.js > ...

```
43
44  export const pageQuery = graphql`
45      query currentPostQuery($id: String!) {
46          wordpressPost(id: { eq: $id }) {
47              title
48              content
49          }
50          site {
51              siteMetadata {
52                  title
53                  subtitle
54              }
55          }
56      }
57  `
```

KUVA 10. GraphQL-query yksittäisen Wordpress-postauksen tiedoille

Sivusto käynnistetään gatsby develop -komennolla, jonka jälkeen se löytyy osoitteesta http://localhost:8000. Sivuston skeemaa voi tarkastella tarkemmin osoitteessa http://localhost:8000/___graphql (KUVA 11). Yksittäinen Wordpress-sivu löytyy osoiterakenteella localhost:8000/(sivun slug). Yksittäinen Wordpress-postaus löytyy osoiterakenteella localhost:8000/post/(postauksen slug).



KUVA 11. Wordpressin pohjalta luotu Gatsbyn GraphQL-skeema

Kun paikallinen Wordpress toimi lähteenä Gatsbylle, aloitettiin Wordpressin siirtäminen palvelimelle. Tiedostojen siirtämiseen käytettiin FileZilla-ohjelmaa. Ohjelmalla otettiin yhteys yrityksen omalle palvelimelle, jonne siirrettiin kaikki paikallisen Wordpressin tiedostot kansioon /home/gatsbywp/public_html. Tiedostoon wp-config muokattiin asetukset vastaamaan palvelimella olevan sivuston tarpeita. Wordpress-sivuston osoite piti vielä päivittää kohdilleen phpMyAdmin-palveluun.

Gatsby-projektiin piti vielä tehdä tarvittavat muutokset gatsby-config.js -tiedostoon (KUVA 12), jotta tiedot saatiin haettua palvelimella olevasta Wordpressistä. Baseurliin piti vaihtaa palvelimella olevan Wordpressin osoite ilman /graphql-loppua, ja protokollaksi vaihdettiin https.

```
JS gatsby-config.js > ...
  1  module.exports = {
  2    siteMetadata: {
  3      title: 'GatsbyWP',
  4      subtitle: `Fetch Data From Wordpress`,
  5    },
  6    plugins: [
  7      'gatsby-plugin-react-helmet',
  8      'gatsby-plugin-netlify-cms',
  9      'gatsby-transformer-sharp',
 10      {
 11        resolve: "gatsby-source-wordpress",
 12        options: {
 13          baseUrl: "gatsbywp.server1.hrj.fi",
 14          protocol: "https",
 15          hostingWPCOM: false,
 16          useACF: false,
 17          verboseOutput: true,
 18        }
 19      }
 20    ],
 21  };
 22
```

KUVA 12. Gatsbyn lisäosien määritys

Lopuksi Gatsby-projektille tehtiin commit ja push, jolloin muutokset toimitettiin GitHubin ohjelmavarastoon.

3.4 Netlify CMS:n liittäminen sivustoon

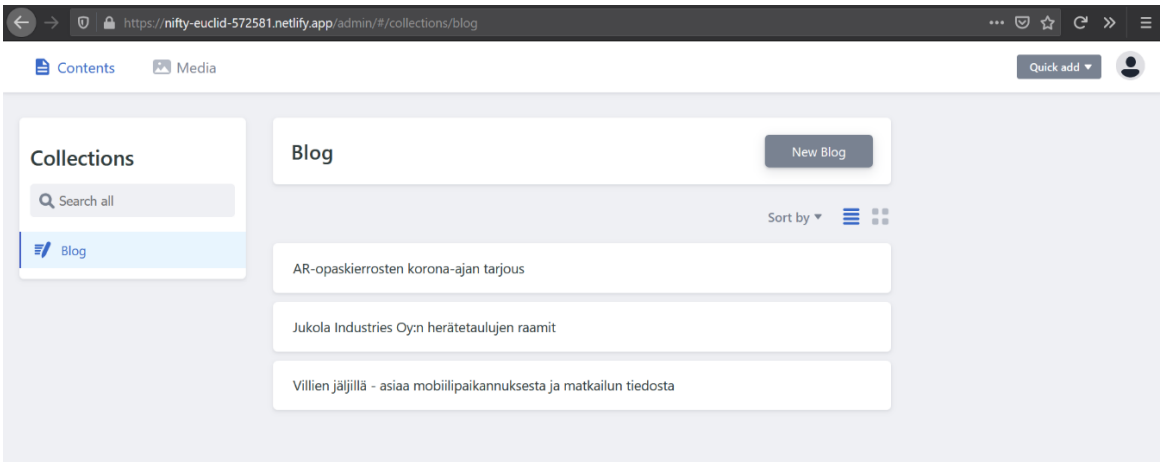
Sivustoon liitettiin vielä Netlify CMS-sisällönhallintajärjestelmä. Ensiksi Netlify CMS ja Gatsbylle tarvittava lisäosa asennettiin projektiin terminaalin avulla komennolla:

npm install --save netlify-cms-app gatsby-plugin-netlify-cms

Projektin static-kansioon täytyi luoda kansio ”admin”, jonne lisättiin tiedosto ”config.yml”. Se sisältää kaikki sisällönhallintajärjestelmän liittämiseen tarvittavat asetukset. Tiedostoon gatsby-config.js oli lisättävä lisäosa ”gatsby-plugin-netlify-cms”, jotta sitä voidaan käyttää osana projektia. Seuraavaksi sallittiin identifikointi Netlifyn asetuksista valitsemalla ”Enable Identity service”. Git Gateway laitettiin vielä päälle valitsemalla Services- osiosta kohta ”Enable Git Gateway”. Tämä mahdollistaa jokaisen uuden blogikirjoituksen luonnin jälkeen siitä luodun markdown-tiedoston automaattisen toimittamisen GitHubin ohjelmavarastoon.

Tämän jälkeen muutokset toimitettiin GitHubin ohjelmavarastoon, jolloin Netlifyn sivusto teki automaattisen buildin, ja muutokset tulivat myös sinne. Netlify CMS:n halllintasivulle päästiin lisäämällä Netlifyssä olevan sivuston osoitteen päätteeseen /admin. Ensin palveluun liitettiin sivuston oma

osoite, jonka jälkeen rekisteröidyttiin. Sähköpostiin tuli vahvistusviesti, jonka linkkiä painamalla saatiin vahvistettua annettu sähköpostiosoite. Sisäänkirjautumisen jälkeen aukeaa hallintapaneeli (KUVA 13), jossa voidaan luoda uusia blogikirjoituksia.



KUVA 13. Netlify CMS:n hallintasivu

3.5 Suorituskykytestaukset

Suorituskykytestaukset toteutettiin käyttämällä kolmea työkalua. Ensimmäinen työkalu on Lighthouse, joka löytyy Google Chromen Kehittäjän työkalut- osiosta. Toinen käytetty työkalu on Pingdom, joka monitoroi verkkosivuja ja luo niistä suorituskykyyn liittyviä raportteja. Lopuksi suorituskyvyn testauksessa käytettiin palvelua nimeltä Gtmetrix. Kaikilla näillä palveluilla testattiin sivuston latausnopeutta.

Testattavia sisällönhallintajärjestelmien käyttötapoja olivat:

1. Wordpress-sivusto

2. Wordpress Gatsby-sivuston kanssa

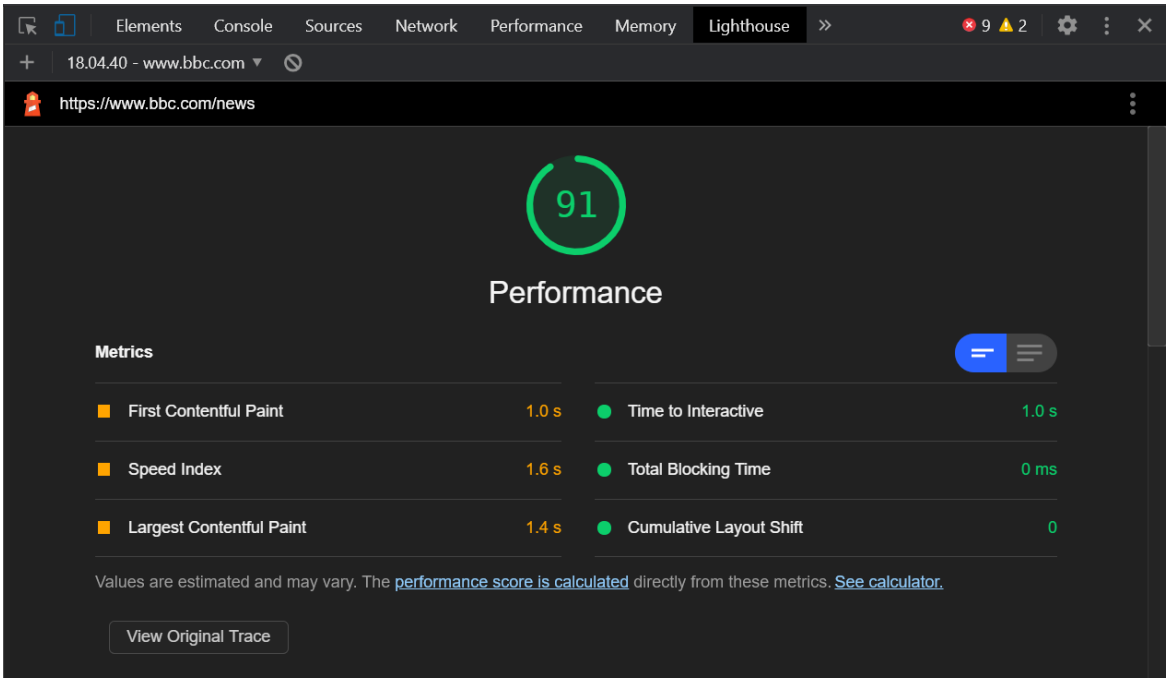
3. Netlify CMS Gatsby-sivuston kanssa

Kaikkia käyttötapoja testattiin kolme kertaa jokaisella työkalulla, jotta tulokset olisivat luotettavampia. Lighthousen avulla testattiin sivustosta työpöytäversion lisäksi myös mobiiliversio. Lighthousella tehdystä mobiiliversion testaamisesta raportoitiin vain yksi testi jokaisesta skenaariosta, koska tulokset eivät muuttuneet seuraavilla testauskerroilla.

Wordpress-sivustosta testattiin yrityksen palvelimella oleva versio, Gatsby-sivustosta testattiin Netlifyssä oleva versio. Gatsby-sivusto testattiin liittämällä siihen yksi sisällönhallintajärjestelmä kerrallaan. Ensin sivusto testattiin Wordpressin kanssa, jolloin Netlify CMS irroitettiin siitä kommentoimalla pois sen lisäosa gatsby-config.js -tiedostosta, sekä tiedostossa config.yml olevat asetukset. Sitten varmistettiin, että Wordpressin liittämiseen tarvittava lisäosa gatsby-source-wordpress, ja sen asetukset ovat kohdillaan gatsby-config.js -tiedostossa, ja gatsby-node.js -tiedostosta löytyvät kaikki tarvittavat kyselyt. Lopuksi tarkistetaan templatet page.js ja post.js, ja tehdään projektille commit sekä push, jonka jälkeen sivusto on liitettynä vain Wordpressiin.

Kun sivustoa haluttiin testata Netlify CMS:n kanssa, varmistettiin että gatsby-config.js -tiedostoon on määritetty lisäosa gatsby-plugin-netlify-cms, ja config.yml -tiedoston asetuksia ei ole kommentoitu pois. Sitten kommentoitiin lisäosa gatsby-source-wordpress pois gatsby-config.js -tiedostosta ja kaikki kyselyt gatsby-node.js -tiedostosta, sekä templatet page.js ja post.js. Lopuksi muutokset toimitettiin GitHubin ohjelmavarastoon, jonka jälkeen ainut sivustoon liitetty sisällönhallintajärjestelmä oli Netlify CMS.

Lighthousella testaaminen tapahtui valitsemalla ”performance” mitattavaksi muuttujaksi testaamista varten. Kun työpöytäversio oli testattu, ajettiin testit vielä mobiiliversioon valitsemalla asetuksista vaihtoehto ”mobile”. Kaikki mobiiliversioiden testit tehtiin Lighthousella. Alla olevassa kuvassa (KUVA 14) näytetään esimerkki Lighthousen testistä testaamalla BBC:n sivustoa.



KUVA 14. Esimerkki Lighthousen testistä

Pingdomiin täytyi ensin rekisteröityä, jotta palvelua pystyi käyttämään. Sen jälkeen asetuksista määritettiin monitoroitaviksi sivuiksi yrityksen palvelimelle siirretty Wordpress-sivu, ja Netlifyssä oleva Gatsby-sivu, johon sisällönhallintajärjestelmät liitettiin. Testauspalvelimen sijanniksi asetettiin vaihtoehto ”Europe”. Pingdomin monitoroidessa ja luodessa suorituskykyyn liittyviä raportteja puolen tunnin välein, on odotettava joitain tunteja, jotta dataa on riittävästi.

Gtmetrix-palveluun on suotavaa rekisteröityä ensin, jotta päästään muokkaamaan testeissä käytettäviä asetuksia. Oletuksena olevan testauspalvelimen sijainnin ollessa Kanadan Vancouver, vaihdettiin se Lontooseen, koska se on kaikista mahdollisista vaihtoehdoista lähimpänä nykyistä sijaintia. Lopuksi palvelulle annettiin testattavan verkkosivuston osoite, jonka jälkeen testit oli mahdollista ajaa. Alla olevassa kuvassa (KUVA 15) näytetään esimerkki Gtmetrixin testistä testaamalla BBC:n sivustoa.

KUVA 15. Esimerkki Gtmetrix-palvelun testistä

4 TULOKSET

Seuraavaksi esitetään testaustyökaluilla saadut tulokset, joista ensimmäinen taulukko (TAULUKKO 1) kuvaa sivuston työpöytäversion latausajat erilaisten sisällönhallintajärjestelmien ja staattisen sivun yhdistelmien kanssa. Taulukkoon on listattu kaikilla kolmella työkalulla tehdyt kolme testiä jokaista käyttötapaa kohden.

TAULUKKO 1. Sivuston latausajat työpöytäversiossa

Sivuston latausajat työpöytäversiossa			
Testauspalvelut	**Wordpress**	**Wordpress + Gatsby**	**Netlify CMS + Gatsby**
Lighthouse 1	2.8s	0.5s	0.5s
Lighthouse 2	2.9s	0.3s	0.4s
Lighthouse 3	2.6s	0.5s	0.4s
Pingdom 1	1.56s	282ms	251ms
Pingdom 2	1.7s	295ms	272ms
Pingdom 3	1.5s	210ms	268ms
Gtmetrix 1	1.8s	0.6s	0.7s
Gtmetrix 2	2.1s	462ms	0.5s
Gtmetrix 3	1.9s	0.7s	0.6s

Seuraavassa taulukossa (TAULUKKO 2) kuvataan sivuston mobiiliversion latausajat erilaisilla sisällönhallintajärjestelmän ja staattisen sivun yhdistelmillä.

TAULUKKO 2. Sivuston latausajat mobiiliversiossa

Sivuston latausajat mobiiliversiossa	
Teknologiat	**Tulokset**
Wordpress Mobile	15.2s
Wordpress + Gatsby Mobile	1.3s
Netlify CMS + Gatsby Mobile	1.4s

Frontendin toteuttaminen erillisenä sivuna paransi suorituskykyä huomattavasti verrattuna siihen, että käytettäisiin pelkkää Wordpressiä (TAULUKKO 1). Erityisesti tämä näkyi sivuston mobiiliversiossa, jossa sivuston latausnopeus parantui jopa yli kymmenellä sekunnilla (TAULUKKO 2). Työpöytäversiossakin ero oli huomattava. Wordpressin ja Netlify CMS:n kesken ei havaittu suurempia eroja latausajoissa staattisen sivun ollessa liitettynä.

5 YHTEENVETO

Opinnäytetyön tarkoituksena oli vertailla erilaisia sisällönhallintajärjestelmien käyttötapoja suorituskyvyn suhteen. Tämä tehtiin toteuttamalla Salmi Platformia mukaileva testisivu GatsbyJS-frameworkillä, joka liitettiin Wordpress- ja Netlify CMS -sisällönhallintajärjestelmiin. Salmi Platformin Wordpress-version suorituskykyä verrattiin Gatsby-sivustoon, johon oli liitetty Wordpress ja Netlify CMS.

Lopputuloksena voitiin osoittaa, kuinka front-endin toteuttaminen erillisenä staattisena sivuna parantaa sivuston suorituskykyä ja latausnopeutta. Erot suorituskyvyssä olivat selkeämpiä mobiiliversiossa. Yllätyksenä tuli, kuinka suuren eron sivuston mobiiliversion latausnopeuteen staattisen sivuston käyttäminen tuo.

Työn toteutus sujui mielestäni melko hyvin. Olin työharjoittelun aikana tutustunut jo osaan teknologioista, joita työn toteuttamisessa tarvittiin. Haastavinta työssä oli mielestäni Wordpressin liittäminen sisällönhallintajärjestelmäksi, sillä sen kanssa tuli usein vastaan sekä pienempiä, että isompia ongelmia. Myös sivustojen mobiiliversioiden testaaminen jäi puutteelliseksi, sillä testaaminen onnistui pelkästään Lighthousella. Testaaminen olisi ollut hyvä toteuttaa useammalla työkalulla, jotta mobiiliversioiden latausajoista olisi ollut enemmän tietoa saatavilla. Opinnäytetyön tulokset kuitenkin vahvistavat yleisen käsityksen siitä, miten Jamstack-tekniikalla toteutetut staattiset sivustot latautuvat nopeammin.

Yhteistyö opinnäytetyön tilaavan yrityksen kanssa sujui hyvin. Pidimme yhteyttä Slack-viestintäsovelluksen avulla, ja apua oli tarvittaessa saatavilla.

Front-end -sivujen toteuttaminen staattisena on kannattava sijoitus, sillä se tuo huomattavia hyötyjä projekteja ajatellen. Eniten aikaa vievä osuus työssä tulee olemaan front-end -sivuston kehittäminen jollain staattisen sivuston luomiseen tarkoitetulla teknologialla, varsinkin jos sivusto on melko iso. Sisällönhallintajärjestelmien liittäminen ei tule olemaan kovinkaan työlästä, mikäli tietää mitä tekee. Itselläni niiden liittämiseen meni hieman enemmän aikaa, mikä johtuu siitä, etten ollut liittänyt niitä ennen mihinkään sivustoihin, enkä näin ollen osannut välttyä vastaantulevilta ongelmilta. Olen kuitenkin tyytyväinen, että ne tulivat tutuksi tämän opinnäytetyön myötä.

Työn aikana opin paljon lisää ohjelmistokehityksestä, sekä erilaisista teknologioista ja tavoista kehittää sivustoja. Opin myös miten sisällönhallintajärjestelmiä voidaan hyödyntää. Olen kokonaisuudessaan tyytyväinen työhön ja lopputulokseen. Aihe oli kiinnostava ja sitä oli mukava tehdä. Työ oli myös sopivan haastava ja ammattitaitoa kehittävä.

LÄHTEET

Apache Friends. (2020). *Apache Friends*. Haettu 23. 9. 2020 osoitteesta https://www.apachefriends.org/index.html

Chacon, S. (n.d.). *Git*. Haettu 19. 9. 2020 osoitteesta https://git-scm.com/

Gatsby, I. (2020). *Gatsby Cloud*. Haettu 25. 9. 2020 osoitteesta https://www.gatsbyjs.com/cloud/

Gatsby, I. (2020). *GatsbyJS*. Haettu 25. 9. 2020 osoitteesta https://www.gatsbyjs.com/

GitHub, I. (2020). *GitHub*. Haettu 19. 9. 2020 osoitteesta https://github.com/

Google LLC. (2020). *Lighthouse*. Haettu 3. 10. 2020 osoitteesta https://developers.google.com/web/tools/lighthouse

GTmetrix. (2020). *How fast does your website load? Find out with GTmetrix*. Haettu 3. 10. 2020 osoitteesta https://gtmetrix.com/

Microsoft. (2020). *Visual Studio Code*. Haettu 23. 9. 2020 osoitteesta https://code.visualstudio.com/

Netlify. (2020). *Netlify*. Haettu 22. 9. 2020 osoitteesta https://www.netlify.com/

Netlify. (2020). *Netlify Build*. Haettu 22. 9. 2020 osoitteesta https://www.netlify.com/products/build/

Netlify. (2020). *The fastest way to build the fastest Gatsby sites*. Haettu 25. 9. 2020 osoitteesta https://www.netlify.com/with/gatsby/

NetlifyCMS. (n.d.). *NetlifyCMS*. Haettu 20. 9. 2020 osoitteesta https://www.netlifycms.org/

NetlifyCMS. (n.d.). *NetlifyCMS Overview*. Haettu 20. 9. 2020 osoitteesta https://www.netlifycms.org/docs/intro/

SolarWinds Worldwide, LLC. (2020). *Pingdom*. Haettu 3. 10. 2020 osoitteesta https://www.pingdom.com/

The GraphQL Foundation. (2020). *GraphQL*. Haettu 28. 9. 2020 osoitteesta https://graphql.org/

The GraphQL Foundation. (2020). *Introduction to GraphQL*. Haettu 24. 9. 2020 osoitteesta https://graphql.org/learn/

Wikipedia. (2020). *Netlify*. Haettu 23. 9. 2020 osoitteesta https://en.wikipedia.org/wiki/Netlify

WordPress.com. (n.d.). *About Us*. Haettu 15. 9. 2020 osoitteesta https://wordpress.com/about/

WordPress.com. (n.d.). *Wordpress*. Haettu 15. 9. 2020 osoitteesta https://wordpress.com/

WordPress.org. (n.d.). *About Us: Our Mission*. Haettu 15. 9. 2020 osoitteesta https://wordpress.org/about/

WordPress.org. (n.d.). *Features*. Haettu 15. 9. 2020 osoitteesta https://wordpress.org/about/features/

WPGraphiQL. (2019). *WPGraphiQL: GraphiQL IDE in the WP-Admin*. Haettu 27. 9. 2020 osoitteesta https://github.com/wp-graphql/wp-graphiql

WPGraphQL. (2018). *WPGraphQL*. Haettu 27. 9. 2020 osoitteesta https://www.wpgraphql.com/