Tuomo Vitikainen

Nuxt.js-kirjastolla toteutetun verkkosivuston nopeus ja hakukoneoptimointi

Tietotekniikan pro gradu -tutkielma

15. toukokuuta 2022

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Tuomo Vitikainen

Yhteystiedot: tuomo.m.j.vitikainen@student.jyu.fi

Ohjaajat: Lakanen Antti-Jussi ja Zhidkikh Denis

Työn nimi: Nuxt.js-kirjastolla toteutetun verkkosivuston nopeus ja hakukoneoptimointi

Title in English: Speed and technical search engine optimization on the website implemented with Nuxt.js

Työ: Pro gradu -tutkielma

Opintosuunta: Ohjelmistotekniikka

Sivumäärä: 50+0

Tiivistelmä: Tässä tutkielmassa tutkittiin single-page application -arkkitehtuurilla luodun sivuston suorituskyvyn kahta osa-aluetta: nopeutta ja teknistä hakukoneoptimointia. Valitut kaksi osa-aluetta ovat tärkeitä, sillä niillä on suora vaikutus myös liiketoimintaan. Tutkielma toteutettiin käyttäen tutkimusmenetelmänä suunnittelutiedettä, jonka prosessimallin mukaisesti tutkielmassa kehitettiin artefakti, jotta haluttuja ominaisuuksia voitiin kehittää ja testata. Sivustoille suoritettiin joukko mittauksia, joiden tuloksista selvisi, että erityisesti staattinen sivusto tarjoaa erityisen hyvän suorituskyvyn nopeuden osa-alueella. Lisäksi tutkielmassa selvitettiin, mitkä ovat keskeisimmät tekniset hakukoneoptimointikeinot. Keinot toteutettiin onnistuneesti verkkosivustolle, mikä kertoo, että SPA-arkkitehtuurilla toteutettuun verkkosivustoon pystytään tekemään kaikki teknisen hakukoneoptimoinnin kannalta olennaiset toimenpiteet.

Avainsanat: hakukoneoptimointi, verkkosivuston suorituskyky, verkkosivuston nopeus, single-page application, WordPress

Abstract: This work focuses on the speed and search engine optimization of a website implemented with a singe-page application architecture. These two areas are important, because they have an impact on business. In this paper, we use design science as a research method, which process model we used to develop an artifact to develop selected features and mea-

suring the end result. After that the measurements were performed on the websites, and the results showed that especially static website provides good performance of speed area. In addition, we defined the most important technical search engine optimization configurations and implemented them to the website. The study shows that there are no difficulties when implementing technical search engine optimization techniques to the SPA-architecture based website.

Keywords: search engine optimization, website performance, website speed, single-page application, WordPress

Termiluettelo

SPA	Single-page application
HTTP	Hypertext Transfer Protocol
SSR	Server-side rendering
CSR	Client-side rendering
PLT	Page Load Time
TTFB	Time to First Byte
TTFP	Time to First Paint
ATF	Above the Fold
FCP	First Contentful Paint
TTI	Time to Interactive
LCP	Largest Contentful Paint

Kuviot

Taulukot

Sisällys

1 Johdanto

Verkkosivusto koostuu yksinkertaisimmillaan HTML-dokumentista ja CSS-tyylitiedoston yhdistelmästä. Kiinteä osa verkkosivustoja on myös JavaScript-ohjelmointikieli, joka mahdollistaa interaktiivisten toimintojen luomisen verkkosivuille. JavaScriptillä on luotu myös useita kokonaisia sovelluskehyksiä, jotka mahdollistavat tavan luoda arkkitehtuuriltaan ja toiminnaltaan erilaisia verkkosivustoja, kuin pelkästään HTML ja CSS mahdollistaisi. (Brunelle ym. 2016) Sovelluskehykset mahdollistavat niin sanottujen single-page application *(SPA)* -verkkosivustojen rakentamisen, joita voidaan ajatella sovelluksina, jotka toimivat verkkoselaimessa (Mikowski ja Powell 2013). Koska SPA-verkkosivustojen arkkitehtuuri eroaa perinteisistä sivustoista, tässä tutkielmassa halutaan selvittää, kuinka SPA-menetelmällä toteutettu sivusto suoriutuu kahdesta valitusta osa-alueesta: nopeudesta ja hakukoneoptimoinnista.

Verkkosivuston nopeutta voidaan mitata objektiivisesti useilla tähän käyttöön kehitetyllä mittareilla, joista yleisin on verkkosivuston latausaika (Hora ym. 2018). Tutkimuksissa on havaittu, että sivuston nopeudella on merkittävä vaikutus konversioon (Gallino, Karacaoglu ja Moreno 2018; Smith 2012). Esimerkiksi verkkokauppajätti Amazonin myynti kasvoi prosentin jokaista latausnopeudesta vähentynyttä 0.1 sekuntia kohden (Wang ym. 2013). Sivustojen monimutkaistuessa myös käytössä olevan laitteen merkitys kasvaa. Internetin käyttö matkapuhelimella on yleistynyt huimasti, ja yhä useammat asiat hoidetaan mobiililaitteen kautta. Bröhl ym. (2018) mukaan useita internetistä löytyviä palveluita käytettiin kaikista eniten matkapuhelimella. Tämän takia on tärkeää kiinnittää huomiota verkkosivuston nopeuteen myös mobiililaitteilla.

Kuitenkin jotta käyttäjä löytää sivustolle on tärkeää, että sivusto sijoittuu hyvin hakutuloksissa. Sivuston hakukonenäkyvyyttä voidaan parantaa verkkosivustolle tehtävillä toimilla, joita kutsutaan hakukoneoptimoinniksi. On havaittu, että hakukoneoptimointi auttaa lisäämään yrityksen markkinaosuutta sekä parantamaan markkinoijan brändipääomaa, jolla taas on vaikutusta esimerkiksi tuotetietoisuuteen ja ostopäätöksiin (Bhandari ja Bansal 2018).

Valituilla kahdella osa-alueella on siis merkittävä vaikutus verkkosivun menestykseen se-

kä käyttökelpoisuuteen riippuen liiketoimintamallista. SPA-verkkosivustojen ongelmana on ollut heikko hakukonenäkyvyys, koska hakukoneet eivät ole indeksoineet JavaScriptillä luotua sisältöä (Brunelle ym. 2016; Góralewicz 2017; Smith 2012). Aihetta on syytä tutkia, sillä SPA-arkkitehtuurilla toteutetulla verkkosivustolla voidaan saavuttaa useita etuja verrattaessa perinteisiin verkkosivuihin. Käyttäjän navigoidessa SPA-verkkosivustolla, sivusto päivittää käyttöliittymästä vain tarpeelliset osiot ilman sivulatauksia, mikä nopeuttaa sivuston toimintaa. Lisäksi arkkitehtuurilla voidaan toteuttaa monimutkaisia toiminnallisuuksia sisältäviä verkkosivustoja, jotka ovat käytettävissä millä tahansa laitteella missä on internetyhteys. (Smith 2012)

Tässä tutkielmassa pyritään vastaamaan seuraaviin tutkimuskysymyksiin:

1. Millä menetelmällä SPA-verkkosivusto kannattaa renderöidä, kun otetaan huomioon sivuston nopeus ja hakukoneoptimointi?
2. Pystyykö SPA-verkkosivustoon toteuttamaan samat tekniset hakukoneoptimoinnit kuin perinteisiin verkkosivustoihin?

Tutkielman toisessa luvussa esitellään tutkielman kannalta olennainen teoria SPA-verkkosivuista, verkkosivujen nopeudesta ja hakukoneoptimoinnista. Sen jälkeen luvussa kolme esitellään tutkielmassa käytettävä tutkimusmenetelmä, joka on suunnittelutiede. Luvussa esitellään myös rakennettava artefakti, sekä artefaktin mittaamiseen käytetyt menetelmät. Luvussa neljä artefaktiin toteutetaan ennalta määritellyt optimoinnit ja muutokset. Luvussa viisi esitellään ja analysoidaan mittauksien tulokset. Tutkielma päättyy lukuun kuusi, joka sisältää pohdinnan.

2 Teoriatausta

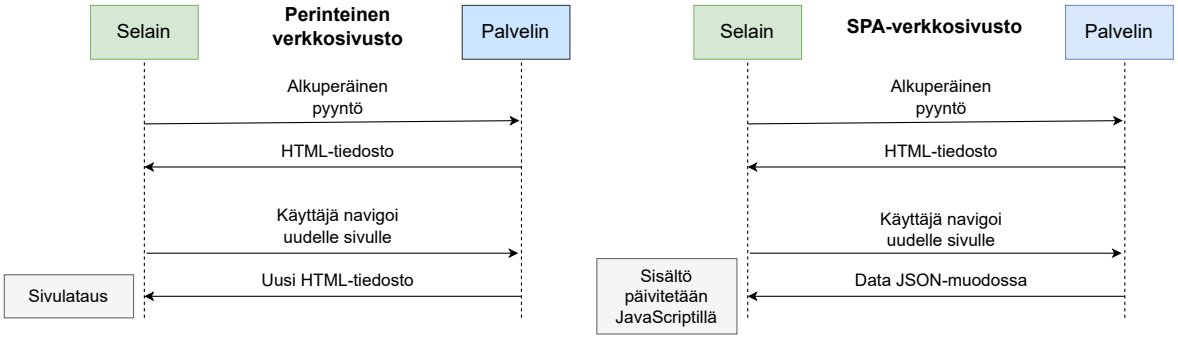
Tässä luvussa esitellään tutkimuksen kannalta olennainen teoria liittyen käytettyyn teknologiaan, verkkosivujen suorituskykyyn ja hakukoneoptimointiin.

2.1 SPA-verkkosivut

Perinteisissä verkkosivustoissa käyttäjän halutessa vaihtaa sivua, selain lähettää pyynnön palvelimelle. Jos haettu resurssi löytyy palvelimelta, palvelin palauttaa onnistuneen pyynnön HTTP-koodin 200 ja haetun verkkosivun. (Fielding, Gettys ym. 1998). Kun selain lopulta muodostaa lopullisen sivunäkymän, se päivittyy käyttäjän selaimeen sivulatauksen myötä (Scott Jr 2015). Yksinkertaisimmillaan verkkosivusto rakentuu HTML-dokumentista, CSS-tyylitiedostosta ja mahdollisesta JavaScript-tiedostosta, jotka selain osaa kääntää verkkosivustoksi (Goodman 2002).

SPA:lla tarkoitetaan verkkosivua, jossa näytettävä sisältö muodostetaan dynaamisesti HTML-pohjaan. Ensimmäiset SPA-toteutukset pohjautuivat Java ja Flash-teknologioihin, mutta nykyään modernit kirjastot toimivat yksinomaan JavaScriptillä (Mikowski ja Powell 2013). Käyttäjän vaihtaessa sivua, uusi sisältö luodaan dynaamisesti samaan aiemmin noudettuun verkkosivun runkoon. Tällöin selaimessa ei myöskään tapahdu sivulatauksia ja samalla verkkosivun toiminta on nopeampaa. SPA-verkkosivuilla haetaankin yleensä sovellusmaista, nopeaa käyttökokemusta. (Scott Jr 2015) Perinteisen verkkosivun ja SPA-verkkosivuston toimintalogiikan ero käy ilmi kuviosta 1.

Riippuen käytetystä kirjastosta, SPA-verkkosivustoissa on usein mahdollisuus valita sivuston renderöintitapa. Yleisesti käytetyistä SPA-kirjastoista esimerkiksi Vue renderöi oletuksena näytettävän sisällön käyttäjän selaimella (Vue.js 2021). Tämä tarkoittaa sitä, että palvelimella on nähtävissä vain lähes tyhjä HTML-dokumentti, jossa on osoitettuna tietyllä tunnisteella paikka, johon dynaaminen sisältö luodaan käyttäen JavaScriptiä. Toinen vaihtoehto renderöidä verkkosivu valmiiksi palvelimella (Iskandar ym. 2020). Koska verkkosivusto sisältöineen muodostetaan jo palvelimella, ryömijät pystyvät käymään sen läpi kuten staattisen verkkosivun. Esimerkiksi Vue-kirjastoon perustuva Nuxt.js tarjoaa mahdollisuuden renderöidä si-



Kuvio 1. Selaimen ja palvelimen väliset pyynnöt ja vastaukset, kun sivulle navigoidaan ensimmäisen kerran ja käyttäjä vaihtaa kerran sivua (Jadhav, Sawant ja Deshmukh 2015)

vusto palvelimella (Vue.js 2021).

Joidenkin kirjastojen tapauksessa on lisäksi mahdollista generoida sivustosta staattinen sivu. Tämä tarkoittaa sisällönhallintajärjestelmää käytettäessä sitä, että sisältöjen muokkauksen tai lisäämisen jälkeen verkkosivusto on generoitava uudestaan. Renderöintitavan valinnalla on vaikutus sivuston hakukonenäkyvyyteen. SPA-verkkosivujen hakukonenäkyvyyden erityispiirteistä kerrotaan lisää luvussa 2.3.

2.2 Verkkosivuston suorituskyky

Nykyisten verkkosivujen ollessa varsin monimutkaisia kokonaisuuksia, niiden nopeuden mittaamiseksi on kehitetty useita eri mittareita. Hora ym. (2018) selvityksen mukaan kaikista yleisin mittari on sivun latausaika (engl. *page load time*). Esimerkiksi Stadnik ja Nowak (2017) ja Wang ym. (2013) mukaan tutkimuksissa sekä käytännön mittauksia tehdessä usein käytetään sivuston latausaikaa ainoana mittarina. Sekä Hora ym. (2018) että Wang ym. (2013) määrittelevät sivun latausajaksi ajan, joka kuluu HTTP-pyynnöstä siihen hetkeen, kun load-niminen tapahtuma toteutuu käyttäjän selaimessa. Load-tapahtuma toteutuu selaimessa silloin, kun sivu on täysin latautunut, sisältäen sivuun liitetyt resurssit, kuten tyylitiedostot ja kuvat (Mozilla 2022).

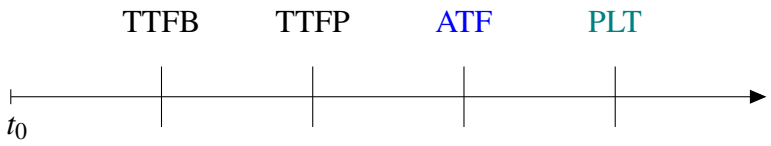
Hora ym. (2018) mukaan viimeaikaisissa tutkimuksissa on alettu kyseenalaistamaan sivun latausajan tärkeyttä sivuston suorituskyvyn mittarina. Käyttäjän näkökulmasta olennaisempaa olisi tarkastella käyttäjälle sillä hetkellä näkyvän verkkosivun osan rakentumisen no-

peutta, sillä verkkosivustoa voi olla mahdollista käyttää, vaikka se ei olisi täysin latautunut. Tähän tarpeeseen on kehitetty useita, yksityiskohtaisempia mittareita. Bocchi, De Cicco ja Rossi (2016) esittelivät jaottelun, jossa jo olemassa olevat mittarit ryhmitellään kahteen eri osa-alueeseen: Mittareihin, jotka mittaavat tietyn hetken tapahtumista (engl. *time-instant metrics*) tai mittareihin, joiden lopputulos muodostuu usean eri mittarin yhteisvaikutuksesta (engl. *time-integral metrics*). Gao, Dey ja Ahammad (2017) taas ehdottavat mittareiden jakoa niiden toimintaperiaatteen perusteella: visuaalisiin tai ei-visuaalisiin mittareihin. Visuaaliset mittarit mittaavat hetkeä, jolloin tietty visuaalinen elementti piirretään näytölle. Ei-visuaalisissa mittareissa taas mitattava tapahtuma on käyttäjälle näkymätön. Tässä luvussa käytetään mittareiden ryhmittelyyn Bocchi, De Cicco ja Rossi (2016) esittelemää jakoperiaatetta.

2.2.1 Aikahetkelliset mittarit

Aikahetkellisillä mittareilla (engl. *Time-instant metrics*) mitataan, milloin tietty yksittäinen tapahtuma tapahtuu. Tämän kategorian mittarit ovat laajalti käytettyjä, koska niiden yksinkertaisuuden vuoksi niiden mittaaminen on helppoa. (Bocchi, De Cicco ja Rossi 2016) Aikahetkelliset mittarit tapahtuvat tietyssä järjestyksessä, kuten kuviosta 2 käy ilmi. Tämän ryhmän mittareissa ajanotto alkaa aina hetkestä, jolloin navigointi verkkosivustolle tapahtuu eli pyyntö lähetetään palvelimelle (Hora ym. 2018). Verkkosivun varsinainen rakentuminen alkaa, kun ensimmäinen osa datasta saapuu selaimelle.

Esimerkki aikahetkellisestä mittarista on ajankohta, jolloin ensimmäinen bitti saapuu palvelimelta selaimeen *(time to first byte)* (Hoßfeld, Metzger ja Rossi 2018) . TTFB mittaa HTTP-pyynnön ja ensimmäisen selaimeen saapuneen bitin välistä aikaa (Hora ym. 2018). TTFB luetaan ei-visuaalisiin mittareihin. Toinen yleisesti käytetty mittari on TTFP-mittari *(time to first paint)*, joka mittaa kestoa, kuinka kauan selaimella kestää piirtää ensimmäinen pikseli verkkosivusta käyttäjän näytölle (Hoßfeld, Metzger ja Rossi 2018) .

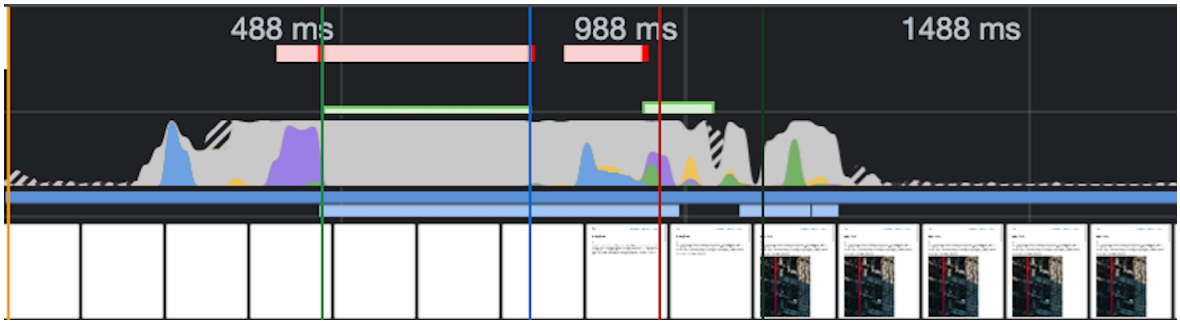


Kuvio 2. Aikahetkellisiä mittareita janalla. t_0 on hetki, kun verkkosivustolle navigoidaan ja pyyntö lähetetään palvelimelle. TTFB-mittari mittaa hetken, jolloin selain vastaanottaa ensimmäisen bitin dataa palvelimelta. TTFP-mittari on hetki, jolloin ensimmäinen pikseli piirtyy selaimeen. ATF-mittari kertoo, milloin verkkosivun käyttäjälle näkyvä osa on visuaalisesti valmis. PLT-mittari kertoo ajankohdan, jolloin verkkosivu on täysin latautunut.

2.2.2 Aikaintegraaliset mittarit

Aikaintegraalisissa mittareissa (engl. *Time-integral metrics*) tunnuspiirteenä on, että niiden laskemiseen liittyy useampi verkkosivun tapahtuma, jonka takia ne ovat kompleksisempia kuin time-instact metrics-ryhmän mittarit. Mittarissa täytyy tällöin määritellä haluttu päätepiste, siihen johtaneet tapahtumat ja niiden kestot sekä aloitusajankohdat (Bocchi, De Cicco ja Rossi 2016).

Esimerkiksi Googlen kehittämä Speed Index -mittari kertoo, milloin selaimessa näkyvä osa verkkosivusta on visuaalisesti muodostettu (Hoßfeld, Metzger ja Rossi 2018). Verkkosivun visuaalisen rakentumisen etenemistä on havainnollistettu kuviossa 3. Työkalu videoi verkkosivun visuaalisen rakentumisen etenemisen, jonka avulla voidaan laskea kesto, jonka jälkeen se on visuaalisesti täysin muodostettu (Hoßfeld, Metzger ja Rossi 2018). Koska mittari mittaa käyttäjälle näkyvän osan muodostumista, päätepisteenä käytetään niin sanottua verkkosivun taitekohtaa (engl. *above the fold*). Gao, Dey ja Ahammad (2017) ovat kehitelleet Speed Index -mittarista version, joka mittaa verkkosivuston visuaalista muodostumista samalla tavalla, kuin ihminen sen kokee. Kun normaali Speed Index -mittari mittaa pikselitason edistymistä, Gao, Dey ja Ahammad (2017) kehittämä mittari yrittää havaita eroja, jotka ihminen oikeastikin havaitsisi.



Kuvio 3. Esimerkki videotallennuksesta, jota käytetään Speed Index -arvon määrittämiseen

2.3 Hakukoneet

Hakukoneella tarkoitetaan web-ohjelmaa, joka hakee käyttäjän hakusanojen perusteella verkkosivuja internetistä. Hakukoneilla on omat tietokantansa, joiden avulla ne palauttavat hakutuloksia. (Levene 2011) Googlen hakukoneella on 91 prosentin markkinaosuus kaikista hakukoneista vuonna 2021, ja toisiksi suosituin hakukone on Bing 3,1 prosentin markkinaosuudella (Statcounter 2021).

Hakukoneen toimintaperiaate perustuu internetissä olevien verkkosivustojen läpikäyntiin ja niistä löytyvän datan indeksointiin. Hakukone koostuu neljästä eri pääkomponentista, jotka ovat:

- Ryömijä (engl. *crawler*)
- Hakuindeksi (engl. *search index*)
- Kyselymoottori (engl. *query engine*)
- Käyttöliittymä

Käyttäjälle näkyvä osa hakukoneesta on käyttöliittymä, johon käyttäjä voi syöttää yhden tai useampia hakusanoja. Kun kysely on tehty, hakutulokset ilmestyvät käyttöliittymään hakukoneen algoritmin päättämässä järjestyksessä (Levene 2011).

Ryömijällä (engl. *crawler*) tarkoitetaan järjestelmää, joka joukkolataa verkkosivustoja. Niillä on monia käyttötarkoituksia, mutta erityisen tärkeitä ne ovat hakukoneissa. Ryömijät toimivat automaattisesti niille annetun algoritmin perusteella, jonka perusajatuksena on ladata annettujen URL-osoitteiden sivut ja sivujen sisältämät hyperlinkit ja jatkaa hyperlinkkien

osoittamille muille verkkosivuille. (Olston ja Najork 2010) Sivuston arkkitehtuurilla on merkitystä siihen, onnistuvatko ryömijät löytämään kaikki sisällöt. On osoitettu, että kaikkien hakukoneiden ryömijät eivät suorita JavaScriptiä, joten sen takana oleva sisältö jää käymättä läpi. (Brunelle ym. 2016; Góralewicz 2017). Vuonna 2022 kuitenkin kaksi suurinta hakukonetta, Google ja Bing, ovat molemmat ilmoittaneet, että heidän hakukoneryömijät pystyvät indeksoimaan myös JavaScriptillä luotua sisältöä (Google 2022c; Bing 2022). Ongelma ei siis ole välttämättä nykypäivänä yhtä relevantti, mutta on ratkaistavissa myös SPA-verkkosivuston puolella. Yksi tapa on käyttää kirjastoa, joka tarjoaa mahdollisuuden renderöidä verkkosivu valmiiksi palvelimella (Iskandar ym. 2020). Koska verkkosivusto sisältöineen muodostetaan jo palvelimella, ryömijät pystyvät käymään sen läpi ilman, että ne joutuvat suorittamaan JavaScriptiä. Esimerkiksi Vue-kirjastoon perustuva Nuxt.js tarjoaa mahdollisuuden renderöidä sivusto palvelimella (Vue.js 2021). Toinen, luvussa 2.1 mainittu tapa, on generoida SPA-verkkosivustosta staattinen versio, jolloin JavaScriptin suorittaminen jää myös pois.

Ryömijöiden lataamien verkkosivujen hakusanat tallennetaan, eli niistä luodaan tietokanta eli hakuindeksi. Hakuindeksia tarvitaan, jotta kyselyt voivat palauttaa relevantteja, parametrien mukaisia hakutuloksia. Hakuindeksi sisältää kaikki verkkosivuilta löytyvät sanat aakkosjärjestyksessä, ja tätä tiedostoa kutsutaan indeksitiedostoksi. Jokainen sana sisältää viitteen sivustoihin, joilta kyseinen sana löytyy. Tätä tiedostoa taas kutsutaan postituslistaksi (engl. *posting list*). Hakuindeksi sisältää myös erillisen linkkitietokannan. Linkkitietokannan avulla voidaan hahmottaa webin rakennetta ja lisäksi päätellä sen kattavuutta. Linkkien avulla tehtävä analyysi voi vaikuttaa sivuston sijoitukseen hakukoneissa. (Levene 2011)

Hakumoottori sisältää algoritmit, jonka avulla verkkosivuja haetaan indekseistä. Niiden tarkat toimintaperiaatteet eivät ole yleisesti julkaistua tietoa, sillä tällöin hakukonesijoitusta voisi manipuloida paremmaksi optimoimalla sivusto erityisesti hakukoneen algoritmia varten (Levene 2011). Googlen käyttämä verkkosivujen arvostelualgoritmi on nimeltään PageRank, jonka on raportoitu sisältävän yli 100 muuttujaa. (Krrabaj, Baxhaku ja Sadrijaj 2017).

Hakukonenäkyvyydellä on tärkeä rooli yritysten markkinoinnissa. Katseentunnistusjärjestelmiä hyödyntäen on tutkimuksissa pystytty osoittamaan, että ihmiset mieluiten klikkaavat ensimmäisiä hakutuloksia (Lewandowski ym. 2018). On todettu, että noin 94 prosenttia ihmi-

sistä katsovat vain ensimmäisen sivun hakutulokset läpi. 63 prosenttia ihmisistä taas katsoo vain 3 ensimmäistä hakutulosta. Ihmiset lähtökohtaisesti ennemmin muuttavat hakusanoja ja hakevat uudestaan, sen sijasta että etsisivät tuloksia toiselta sivulta. (Sharma ym. 2019)

2.4 Hakukoneoptimointi ja toteutustavat

Hakukoneoptimoinnilla tarkoitetaan verkkosivulle tehtyjä toimia, joilla voidaan parantaa sivun sijoitusta ja näkyvyyttä hakukoneissa. Sijoitukseen vaikuttavat monet eri tekijät, joista valtaosaan sivuston ylläpitäjä pystyy itse vaikuttamaan joko sivuston sisällön tai teknisen toteutuksen kautta. (Davis 2006)

Kaikki sivustolle tehtävät muutokset eivät kuitenkaan ole hyväksyttyjä, vaan hakukone saattaa rankaista vääristä toimista. Esimerkiksi Google saattaa poistaa selkeästi vilpillisiä tekniikoita käyttäneen sivuston hakemistostaan, ja lievemmistä rikkeistä antaa varoituksen. Tälläisiä ei-suositeltuja keinoja kutsutaan black hat -tekniikoiksi. (Malaga 2010) White hat -tekniikat ovat sallittuja tekniikoita, jotka ovat kyseisen hakukoneen ohjeiston mukaisia (Davis 2006). Black hat -tekniikoista useimmiten käytetty on avainsanojen viljely, jossa sivustolle sijoitetaan suuri määrä hakusanoja paremman näkyvyyden toivossa. Hakusanat ovat sijoiteltu siten, että ne ovat vain hakukoneen nähtävissä, esimerkiksi jonkin muun elementin takana tai sijoiteltuna kuvien alt-tageihin. Hakukoneet kuitenkin nykyään erottavat sisällön jotka eivät selkeästi ole normaalin käyttäjän nähtävissä. (Malaga 2010) Toinen esimerkki tällaisesta tekniikasta on luoda ylimääräisiä sivustoja vain siinä käyttötarkoituksessa, että niistä lähtee linkkejä kohdesivustolle. (Sharma ym. 2019)

Hakukoneoptimointi jaetaan usein kahteen eri osa-alueeseen, sivun sisäiseen ja sivun ulkoiseen hakukoneoptimointiin. Sivun sisäinen hakukoneoptimointi kattaa kaikki ne keinot, jotka sivuston ylläpitäjä itse pystyy tekemään hakukonenäkyvyytensä hyväksi. Tähän lukeutuu sivuston tekninen toteutus sekä kaikki sisältö, kuten tekstit, kuvat ja hakusanat. (Sharma ym. 2019) Sivuston tekninen toteutus ratkaisee myöskin sivuston nopeuden. Vuodesta 2010 lähtien nopeus on ollut arvostelutekijä Googlen hakukoneella tehdyissä työpöytähauissa, ja vuodesta 2018 lähtien myöskin mobiililaitteella tehdyissä hauissa (Osmani ja Grigorik 2019).

Myös sillä on merkitystä, kuinka moni muu sivusto linkittää sivuusi. Kuitenkaan pelkkä osoittavien linkkien määrä ei ole tärkein tekijä, vaan hakukoneista ainakin Googlen on todettu ottavan huomioon myös linkkien laatu. Google arvioi verkkosivustoja Page Rank -algoritmilla, ja arvostaa enemmän linkityksiä jotka tulevat verkkosivustoilta joilla on korkea Page Rank -pisteytys. (Malaga 2010). Ulkoisten linkitysten määrä yhdessä muodostavat linkkisuosion, ja suuresta suosiosta voidaan päätellä kyseessä olevan tärkeä sivu. Ulkoiset linkitykset esimerkiksi bannerien tai mainosten muodossa eivät ole algoritmien mielestä niin arvostettavia kuin tekstiviittaukset. (Patil Swati, Pawar ja Patil Ajay 2013)

2.5 Sisäinen hakukoneoptimointi

Sisällönsuunnittelun alkuvaiheissa tulisi ottaa huomioon käytettävät avainsanat. Sivuston tulisi sisältää teemaan ja sivuston aihepiiriin sopivia avainsanoja, sillä käyttäjät käyttävät niitä hakukoneissa verkkosivustoja hakiessaan. Onkin huomioitava, että avainsanat on liitettävä verkkosivuston muuhun sisältöön. Avainsanoja mietittäessä on hyvä kiinnittää huomiota että ne eivät olisi liian laajoja, eivätkä kuitenkaan liian vaikeita. (Hui ym. 2012) Krrabaj, Baxhaku ja Sadrijaj (2017) tekemässä tutkimuksessa saatiin hyviä tuloksia hakukonenäkyvyyden suhteen käyttämällä todennäköisiä hakufraaseja, jotka samalla sopivat sivuston aihepiiriin. Avainsanatiheys määritetään yleensä prosentteina suhteessa muuhun sisältöön, ja Hui ym. (2012) mukaan sopiva tiheys olisi noin 3-8 prosenttia.

Avainsanojen sijoittelulle on myös syytä kiinnittää huomiota. Olennaisia paikkoja hakukoneoptimoinnin kannalta on muun muassa sivuston otsikko, metatiedot, HTML-otsikot ja linkkitekstit. (Hui ym. 2012). Metatietoihin voi sivun lyhyen kuvauksen lisäksi asettaa teemaan sopivia hakusanoja (Davis 2006). HTML-merkintäkielessä on kuusi eri otsikkotyyppiä, joista H1-tagi on sivun pääotsikko, joka on merkitsevin ja joita kuuluu olla vain yksi per sivu. H6-tagilla taas merkataan vähiten merkitsevä otsikko. Otsikkojen käyttö luo myös sivun rakenteeseen hierarkiaa ja selkeän rakenteen. (Gupta ym. 2016) Tärkeä informaatio kannattaa aina kirjoittaa tekstinä kuvien sijasta, jolloin sisältö on myös hakukonebottien luettavissa (Davis 2006).

Yleinen uskomus on, että sivuston sisältöjen päivittäminen auttaa hakukonesijoitukseen. On

kuitenkin todettu, että sivusto voi sijoittua korkealle hakutuloksissa vaikka sisältöä ei aktiivisesti päivitettäisikään. Sisällön uudistuminen voi kuitenkin auttaa sivustoa saavuttamaan paremman hakukonenäkyvyyden, mutta sijoituksen pitäminen ei välttämättä vaadi tuoretta sisältöä. (Malaga 2010)

2.6 Tekninen hakukoneoptimointi

Teknisillä optimointitekniikoilla tarkoitetaan toimia, jotka liittyvät verkkosivuston tekniseen toteutukseen. Laadukkaalla teknisellä toteutuksella varmistetaan muun muassa se, että hakukonebotit löytävät kaikki sivut jotka halutaan indeksoida.

Robots.txt on palvelimelle asetettava tiedosto, joka asetetaan hakemiston juureen. Sen avulla voidaan kertoa hakukoneboteille, mitkä tiedostot niiden tulisi indeksoida ja mitkä ei. Esimerkiksi CSS-tyylitiedostoja, kuvakansioita tai käyttäjähallintasivuja ei usein haluta indeksoitavan, jotta ne eivät tulisi vastaan hakutuloksissa tavallisille käyttäjille. (Kumar 2013)

Sitemap.xml on tiedosto, jonka avulla verkkosivuston ylläpitäjä voi informoida hakukoneille indeksoitavien sivujen osoitteet (Levene 2011). Googlen mukaan sivukartan luominen ja toimittaminen voi olla varsinkin silloin oleellista, kun sivusto on uusi tai kokoluokaltaan poikkeuksellisen suuri, tai sisältää sisältöä joka on eristyksissä tai mihin on vähänlaisesti sisäisiä linkityksiä (Google 2022a).

Sivustokartalla tarkoitetaan XML-muotoista tiedostoa, joka sisältää sivuston sisäisen rakenteen, kuten sivujen väliset linkitykset. Sivustokartan voi itse luoda ja sen jälkeen toimittaa hakukoneelle, jotta sivuston indeksointi nopeutuu. (Kumar 2013)

Sivun alkuperäinen osoite voi olla tarpeen osoittaa, jos sivuun osoittaa useita eri osotteita. Hakukoneryömijät käsittävät silloin sivun olevan sama sivu monistettuna. Jos hakukoneelle ei erikseen kerro sivun alkuperäistä osoitetta, se päättää indeksoitavan osoitteen itse, jolloin hakemistoon voi joutua väärä osoite. Alkuperäisen ja indeksoitavaksi haluttavan sivun voi merkitä erityisellä rel=canonical -elementillä. (*Consolidate duplicate URLs* 2021; Ohye ja Kupke 2012)

Alt-attribuutilla tarkoitetaan kuvan tekstivastinetta, jonka tulisi olla lyhyt selostus kuvan

sisällöstä. Hakukoneet eivät näe kuvia, mutta alt-attribuutit ovat niiden indeksoitavissa. Alt-attribuuttien käyttö on erityisen tärkeää varsinkin silloin, kun kuvaa käytetään linkkinä. Attribuutin käyttö on tärkeää myöskin sivuston saavutettavuuden kannalta, sillä sen avulla myös ruudunlukuohjelmia käyttävät ihmiset saavat tietoonsa kuvan sisältämän informaation. (Slatin 2001)

Sivuston nopeus mahdollistaa sen, että käyttäjä viettää sivustolla mahdollisimman pitkiä aikoja. Lisäksi sivuston nopeus on Googlella sivuston pisteytykseen vaikuttava tekijä (Egri ja Bayrak 2014; Osmani ja Grigorik 2019). Sivuston nopeuteen vaikuttaa suuri määrä tekijöitä, lähtien sivuston toteuttamistavasta. Verkkosivustoa voi optimoida nopeammaksi esimerkiksi käyttämällä vähän tilaa vieviä kuvaformaatteja, minifioimalla HTML-, JavaScript- ja tyylitiedostot ja poistamalla kaikki käyttämättä jäänyt lähdekoodi. (Egri ja Bayrak 2014).

HTTP-statuskoodilla 404 ilmoitetaan, että haettua sivua ei löytynyt palvelimelta. **Pehmeällä 404-virheellä** tarkoitetaan tilannetta, jossa sivulla ei ole sisältöä tai se on esimerkiksi virheviesti joka ilmoittaa sivun puuttumisesta, mutta palvelin palauttaa tästä huolimatta onnistuneen pyynnön HTTP-statuskoodin 200. (Meneses, Furuta ja Shipman 2012; Prieto, Alvarez ja Cacheda 2014) Tällöin sivuston käyttäjä saa tietoonsa että sisältöä ei löytynyt, mutta hakukoneryömijät indeksoivat sivun virheellisestä statuskoodista johtuen ja käyttävän turhaan resursseja. 404-virhesivusta kannattaa tehdä selkeästi sivustoon kuuluva sivu ja lisätä linkki, josta käyttäjä pääsee takaisin etusivulle. (Google 2022b)

3 Tutkimusmenetelmä

Tässä luvussa käsitellään suunnittelutieteen prosessimallia sekä sen käyttöä tässä tutkimuksessa.

3.1 Suunnittelutiede

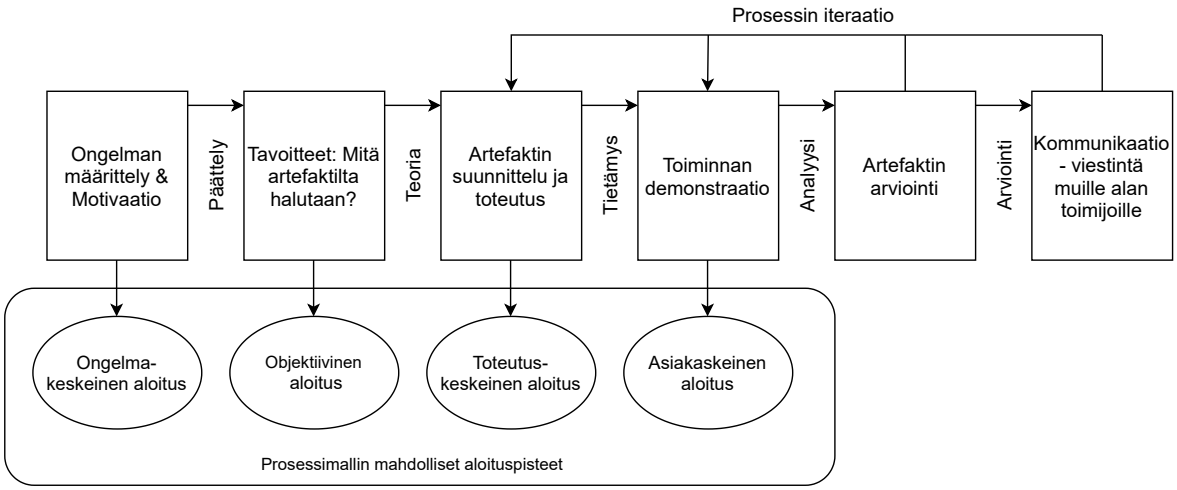
Suunnittelutiede määritellään Peffers ym. (2007, s. 49) mukaan seuraavalla tavalla: "Suunnittelutieteen avulla luodaan ja kehitetään artefakteja, joilla pyritään ratkaisemaan tunnistettuja ongelmia." (Peffers ym. 2007). Artefaktiksi voidaan määritellä lähes mikä tahansa suunniteltu objekti, jonka avulla pyritään ymmärtämään tutkimusongelmaa, jotta siihen voidaan tuottaa ratkaisuja.

Jotta ongelmia voidaan ratkaista, suunnitelmia arvioida ja tuloksia esittää, suunnittelutieteisiin kuuluu ennalta määritelty prosessi. Prosessin eri vaiheiden avulla suunnitteluongelmia voidaan lähestyä järjestelmällisesti tarkastellen asetettuja tavoitteita ja vaatimuksia, jotta voidaan päästä kohti parempia ratkaisuja. Prosessissa hyödynnetään jo olemassa olevia teorioita sekä tutkimuksen aikana ilmennyttä uutta tietoa. (Peffers ym. 2007). Peffers ym. (2007) ovat esittäneet mallissaan neljä eri aloitustuskohtaa, jotka sopivat eri lähtötilanteessa oleville suunnittelutieteen hyödyntäjille.

Tässä tutkielmassa käytetään Peffers ym. (2007) esittämää suunnittelutieteen prosessimallia, jonka toimintaperiaate käy ilmi kuviosta 4. Prosessi on jaettu kuuteen eri aktiviteettiin, jotka on suunniteltu suoritettavan peräkkäisessä järjestyksessä. Aktiviteetit tässä tutkimuksessa määritellään luvussa 3.2.

Ensimmäinen aktiviteetti on sekä ongelman että ratkaisun arvon määrittely. Ongelman määrittely on tärkeää, sillä sen avulla kehitetään artefaktia ja sen ominaisuuksia vastaamaan ongelmaa mahdollisimman hyvin, jonka takia voi olla hyödyllistä jakaa ongelma pienempiin osiin. Ratkaisun arvon määrittely taas motivoi tutkijaa etsimään ratkaisuja, toimien samalla johdantona tutkimuksen yleisölle, jotta he ymmärtävät tutkimuksen motiivit.

Toisena aktiviteettina on määritellä ratkaisun tavoitteet. Tavoitteet tulee johtaa ongelman



Kuvio 4. DSMR-prosessimalli (Peffers ym. 2007)

määrittelyn pohjalta ja konkreettisesta tutkimustilanteesta eli siitä, mitkä ovat käytettävissä olevat resurssit. Resursseihin kuuluu myös mahdolliset valmiit ratkaisut joita voidaan hyödyntää artefaktin kehittämisessä. Tavoitteet voivat olla laadullisia tai määrällisiä.

Kolmantena aktiviteettina on suunnitella ja kehittää artefakti. Tähän aktiviteettiin mennessä tutkijalla tulee olla riittävä teoriatieto hankittuna. Peffers ym. (2007) mukaan artefakti voi olla mikä tahansa suunniteltu objekti, jonka luomiseen on käytetty suunnitteluprosessia, ollen kuitenkin usein erilaisia konstruktioita, malleja, menetelmiä tai ilmentymiä.

Neljäntenä aktiviteettina on demonstroida artefaktin toimintaa. artefakti vastaa silloin tutkimusongelman johonkin kohtaan, olematta kuitenkaan vielä täysin valmis.

Viidentenä aktiviteettina on artefaktin arviointi. Tämän kohdan tarkoituksena on tarkastella, kuinka hyvin se vastaa tutkimuksen alussa määriteltyyn ongelmaan. Arviointimenetelmä eroaakin paljon riippuen artefaktin luonteesta. Arviointi voi olla esimerkiksi simulaatioita, asiakaspalautteita, määrällisiä suorituskykymittauksia tai vertailua toisessa toimenpiteessä määriteltyihin tavoitteisiin. Saadun palautteen perusteella artefaktia voidaan jatkokehittää parempien tulosten saamiseksi, tai jatkaa prosessimallin viimeiseen kohtaan eli viestintään.

Viimeinen aktiviteetti on viestintä. Tutkimuksesta usein tiedotetaan muille alan toimijoille, esimerkiksi julkaisemalla aiheesta tutkimusjulkaisu. Julkaisu voi pohjautua rakenteeltaan suunnittelutieteen prosessimalliin, lähtien ongelman havainnollistamisesta jatkuen artefaktin

kehittämiseen, päättyen arviointiin ja johtopäätökseen. (Peffers ym. 2007)

3.2 Suunnittelutiede tutkimuksessa

Ensimmäisenä aktiviteettina Peffers ym. (2007) prosessimallissa oli ongelman ja ratkaisun arvon määrittely. SPA-verkkosivustojen tavoitteena on sulava ja interaktiivinen toiminta. Koska arkkitehtuuri eroaa perinteisen verkkosivuston arkkitehtuurista, sillä voi olla vaikutuksia verkkosivuston olennaisiin osa-alueisiin. Tutkimuksen tavoitteena oli ymmärtää, millä menetelmällä SPA-verkkosivusto kannattaa renderöidä, sillä renderöintimenetelmällä on vaikutus sivuston nopeuteen ja hakukoneiden indeksoitavuuteen. Lisäksi tutkimuksen toisena tavoitteena oli selvittää, voiko SPA-verkkosivustolle toteuttaa samat tekniset hakukoneoptimointikeinot kuin perinteiselle verkkosivustolle. Hakukonenäkyvyys voi joillekin toimijoille olla tärkeä tekijä, jonka takia asiaa kannatti tutkia. Nopeus on toinen tärkeä tekijä - nopeasti toimivalla sivustolla on havaittu olevan positiivisia vaikutuksia liiketoimintaan (Gallino, Karacaoglu ja Moreno 2018; Smith 2012).

Toisena aktiviteettina oli asettaa artefaktille vaatimukset. Tämän artefaktin päävaatimuksina oli, että artefakti on verkkosivusto, jota pystyy käyttämään verkkoselaimella. Verkkosivustolle pitää pystyä luomaan uusia sivuja sisällönhallintajärjestelmän kautta sekä lisäämään tehtyjä sivuja sivuvalikkoon. Tavoitteena on, että verkkosivusto täyttää toiminnoiltaan vähimmäisvaatimukset, jotta verkkosivuston nopeutta ja teknistä hakukoneoptimointia voidaan tehdä ja testata. Tutkimuksen kannalta olennaista on, että verkkosivustojen sisältöjä pystyy syöttämään sisällönhallinnan kautta, minkä lisäksi sivustojen täytyy sijaita palvelimella, jotta mittaaminen on mahdollista. Artefaktin pää- ja alivaatimukset listattuna:

- Sivuston sisällöt täytyy olla päivitettävissä sisällönhallintajärjestelmän kautta.
  - Tiedot sisällönhallintajärjestelmästä haetaan rajapinnan kautta.
- Sivuston täytyy toimia eri kokoisilla näytöillä responsiivisesti.
- Mittauksia varten sivustosta luodaan kolme eri versiota, jokaiselle eri renderöintimenetelmälle oma. Sivustot sijoitetaan samalle palvelimelle.
- Sivuston metatiedot tulee olla päivitettävissä sisällönhallintajärjestelmän kautta.
- Sivuston tulee olla ryömijöiden luettavissa ja indeksoitavissa.

  - Sivuston täytyy omata robots.txt ja sitemap.xml -tiedostot.
  - Linkit ovat ryömijöiden luettavissa.
- Sivuston tulee noudattaa oikeaoppista HTML-dokumentin rakennetta:
  - Title-tagi
  - Kuvien alt-tekstit

Kun vaatimukset oli määritelty, voitiin toteuttaa itse artefakti. Artefaktin teknisen toteutuksen kuvaus on luvussa 3.3. Luvussa kuvataan, mitä ohjelmistoja, kirjastoja ja lisäosia artefaktin rakentamisessa käytettiin. Artefaktille toteutettiin myös joukko optimointitoimenpiteitä, joiden toteutus on esitelty kokonaisuudessaan luvussa 4. Peffers ym. (2007) prosessimallin mukaan neljäntenä aktiviteettina on demonstroida artefaktin toimintaa ja viidentenä aktiviteettina on arvioida artefaktia. Artefaktin eli verkkosivuston toimintaa demonstroidaan luvussa viisi, jossa tarkastellaan mittausten tuloksia ja arvioidaan nopeusoptimointien vaikutusta verkkosivustoon. Lisäksi samassa luvussa käydään läpi, kuinka toteutetut tekniset hakukoneoptimointimenetelmät onnistuivat. Luvussa kuusi arvioidaan tutkimuksen luotettavuutta, mitä voitaisiin prosessimallin mahdollisella seuraavalla syklillä ottaa huomioon ja pohditaan mitä arvoa tämän tutkimuksen tuloksilla on.

3.3 Artefaktin tekninen toteutus

Verkkosivuston käyttäjälle näkyvä osa eli käyttöliittymälogiikka toteutettiin Nuxt.js-kirjastolla. Nuxt.js on Vue.js-kirjastoon pohjautuva kehys, jonka avulla voidaan toteuttaa dynaamisia web-sovelluksia ja verkkosivustoja. Nuxtia käytettäessä on kolme mahdollista tapaa renderöidä sivusto: palvelimella, käyttäjän selaimessa tai staattisen version generoiminen.

Sisällönhallintajärjestelmänä käytettiin avoimeen lähdekoodiin pohjautuvaa WordPressiä, joka on kirjoitettu PHP-ohjelmointikielellä. Artefaktin rakentamisessa hyödynnettiin muitakin valmiita ratkaisuja, kuten WordPressin valmiita lisäosia perustelluista syistä. Valittu sisällönhallintajärjestelmä vaikuttaa siihen, että millä tavalla tekniset ratkaisut käyttöliittymän yksityiskohtaisesti toteutetaan. Sisällönhallintajärjestelmästä riippumatta hakukoneoptimointiin vaikuttavat toimenpiteet ovat kuitenkin pääosin samat.

Jotta sisällönhallintajärjestelmän kautta syötettyä dataa voidaan näyttää käyttöliittymässä, tuli tiedot hakea rajapinnan kautta. WordPressissä on sisäänrakennettuna REST API -rajapinta, joka tarjoaa haetun datan JSON-muodossa. Vaikka REST API -rajapinnan käyttö olisi ollut ilmeinen valinta sen sisäänrakennettavuuden takia, tutkimuksessa päädyttiin käyttämään sen sijasta GraphQL -rajapintaa. Sen sijasta että käytettäisiin RESTIn päätepisteitä (engl. *endpoint*), GraphQL-kyselyt mahdollistavat tavan hakea tietoja yksityiskohtaisemmin. WordPress ei oletuksena tarjoa kyseistä rajapintaa, vaan tuki on mahdollista saada erikseen ladattavan lisäosan kautta. Tässä työssä käytettiin WPGraphQL -nimistä lisäosaa, joka on ilmainen ja avointa lähdekoodia. Kun kehitetään WordPress-sivustoja, on yleensä järkevää tietyissä tapauksissa käyttää valmiita lisäosia sen sijasta, että tarvittavaa toiminnallisuutta lähtisi itse toteuttamaan ja kehittämään alusta asti. Työssä käytetyt lisäosat ja muut käytetyt ohjelmistot käyvät ilmi kuviosta 1. Muita olennaisia WordPress-kehityksessä tarvittavia lisäosia on Advanced Custom Fields, joka mahdollistaa kustomoitavien lisäkenttien luomisen sisällönhallintajärjestelmään, jotta sisältöjä voidaan syöttää monipuolisemmin kuin WordPressin omien kenttien avulla. Advanced Custom Fieldsiin taas on saatavilla GraphQL -lisäosa, jonka avulla kyselyillä voidaan hakea myös ACF-kenttiin syötettyjä tietoja.

Taulukko 1. Artefaktin rakentamisessa käytetyt ohjelmat ja kirjastot

nimi	versio	selite
WordPress	5.8	Sisällönhallintajärjestelmä
Nuxt.js	2.15.8	SPA-sovelluskehys
WPGraphQL	1.6.6	Luo GraphQL-rajapinnan WordPressiin
Advanced Custom Fields	5.10.2	Kustomoitavat sisältökentät
WPGraphQL for ACF	0.5.3	GraphQL-laajennos ACF-lisäosaan
Yoast SEO	17.8	Hakukoneoptimointi-lisäosa

Tutkielmassa kehitettyyn verkkosivustoon luotiin mittauksia varten esimerkkisisällöt. Esimerkkisisällöt ovat luotu siltä pohjalta, että verkkosivu vastaa keskimääräistä verkkosivua HTTP Archiven keräämän datan mukaan. HTTP Archiven mukaan keskimääräinen verkkosivusto on työpöytäympäristössä 2165.5 kB. ja mobiiliympäristössä 1974.1 kilotavua. Kuvien osuus lukemista ovat työpöytäympäristössä 946 kB ja mobiiliympäristössä 877.8 kB.

(HTTP Archive 2022)

Mittauksissa kohteena oleva testisivu sisältää 11 kappaletta kuvia, joiden koko yhteensä on 985 kB. HTTP Archiven mukaan keskimääräinen sivu sisältää noin 900 kilotavua kuvia, ja noin parikymmentä kuvaa. Testisivussa on siis vähemmän kuvia kuin keskimääräisellä verkkosivulla, mutta yhteiskoko on hieman suurempi. Testisivun kuvat ovat siis kooltaan isompia tai laadultaan parempia.

3.4 Mittausmenetelmät

Tässä tutkielmassa keskityimme mittaamaan verkkosivuston nopeutta sekä teknistä hakukoneoptimointia. Näiden ominaisuuksien mittaamiseksi on kehitetty useita eri sovelluksia. Sivuston suorituskyvyn mittaaminen suoritetaan Googlen Lighthouse -työkalulla. On huomioitava, että nopeusmittauksissa mitataan vain sitä tapahtumaa, kun sivustolle saavutaan ensimmäisen kerran, ja mittauksien kohteena on vain verkkosivuston etusivu. Esimerkiksi tapahtuma, jossa sivusto on jo kerran muodostettu ja käyttäjä navigoi etusivulta toiselle sivulle, ei ole tämän työkalun mitattavissa. Muutenkaan kaikki artefaktin vaatimukset eivät ole todettavissa testaustyökaluilla, vaan ne edellyttävät tekijän omia arvioita ja havaintoja. Esimerkiksi sisällönhallintajärjestelmän toimivuus sisältöjen syötössä on todettavissa vain käyttämisen kautta. Myöskin verkkosivuston responsiivinen toiminta eri näyttökoilla on vaatimus, joka on todettavissa vain silmämääräisesti.

Omien havaintojen kautta todettiin myös hakukoneoptimointitekniikoiden onnistuminen. Hakukoneoptimoinnin onnistumisen tarkasteluun on kehitetty useita eri työkaluja, mutta ne keskittyvät lähinnä verkkosivuston sisältöön. Teknisen hakukoneoptimoinnin tekniikat ovat dikotomisia, eikä niiden onnistuminen ole luvuilla mitattavissa. Mittauksissa käytetty laitteisto ja mittaustyökalu on esitetty taulukossa 2.

Mittauksissa käytettiin Googlen Lighthouse -työkalua, joka on tarkoitettu verkkosivustojen laadun automaattiseen mittaamiseen. Lighthousea pystyy käyttämään usealla eri tavalla: Chrome-selaimeen asennettavana lisäosana, node-moduulina tai komentorivityökaluna. Tässä työssä suorituskykymittaukset tehtiin komentorivin kautta, jotta testauskerrat olisivat mahdollisimman yhteneväisiä toistensa kanssa, minkä lisäksi käytetyt asetukset saadaan tal-

Taulukko 2. Mittauksissa käytetty laitteisto ja ohjelmat

	Selite
Tietokone	MacBook Pro (13", 2017)
Käyttöjärjestelmä	MacOS Monterey 12.1
Prosessori	3,1 GHz Intel Core i5
Muisti	16 Gt 2133 MHz DDR3
Näytönohjain	Intel Iris Plus Graphics 650, 1536 Mt
Mittausohjelmisto	Google Lighthouse CLI 9.5.0

lennettua helposti esitettävässä muodossa. Lisäksi generoidut raportit jäävät talteen automaattisesti HTML-muotoisissa dokumenteissa.

Työkalu sisältää kuusi erilaista suorituskykymittaria, jotka on esitelty taulukossa 3. Lighthousen sisältämistä mittareista useampi mukailee teorialuvussa 2.2 esiteltyjen toimintaperiaatetta, mutta eivät ole identtisiä. Lisäinformaationa verkkosivuista otetaan talteen myös network request time -arvo, jolla tarkoitetaan arvoa, kuinka kauan palvelimelta kestää palauttaa dataa selaimelle ensimmäisen tehdyn pyynnön jälkeen.

Ensimmäisenä suorituskykymittaukset tehtiin verkkosivustoilla, joihin ei ole tehty nopeuteen vaikuttavia optimointeja. Optimointien jälkeen mittaukset toistettiin. Kaksi ensimmäistä optimointitoimenpidettä on valittu sillä perusteella, että ne ovat nykyään yleisesti tehty toimenpide useimmilla sivustoilla HTTP Archiven tarjoaman datan mukaan. Kolmas optimointoimenpide liittyy palautteeseen, jonka mittaustyökalu antoi testimittausten yhteydessä.

1. Sivustolle otetaan käyttöön lazy loading-ominaisuus.
2. Kuvien optimointi, kuvat konvertoidaan WEBP-formaattiin.
3. Sivustoille lisätään rajapinnan esihaku-attribuutti.

Mittauskertoja pidettiin yhteensä neljä kappaletta, ja jokaisella mittauskerralla jokainen verkkosivusto testattiin 15 kertaa. Ennen varsinaisia mittauksia tehdyt testimittaukset osoittivat, että hajonta mittauksen välillä ei ole kovinkaan suurta. Jos hajonta olisi suurempaa, suurem-

Taulukko 3. Google Lighthousen suorituskykymittarit. Vastaavalla mittarilla tarkoitetaan vastaavuutta luvussa 2.2 esiteltyihin mittareihin.

Google Ligthousen mittari	Selite	Teoriassa esitelty mittari
First Contentful Paint	Kuinka kauan selaimella kestää renderöidä ensimmäinen osa verkkosivun rakenteesta.	Time to First Paint
Time to Interactive	Mittaa aikaa, jonka jälkeen sivu on täysin interaktiivinen.	Page Load Time
Speed Index	Kesto verkkosivun visuaalisen ulkoasun rakentumiseen.	Above the Fold
Largest Contentful Paint	Kesto, minkä jälkeen näkymän suurin elementti on renderöity.	–
Total Blocking Time	Aika, joka kuluu First Contentful Paintin ja Time to Interactiven välillä.	–
Cumulative Layout Shift	Sivurakenteen muutos sivulatauksen aikana.	–
Network Request Time	Aika, joka palvelimelta menee vastata pyyntöön.	Time to First Byte

pi määrä mittauksia voisi olla tarpeen. Näin ollen 15 kertaa valikoitui testikertojen määräksi lopulliseen tutkimukseen, tuoden riittävän luotettavuuden mittausten tuloksiin, mutta pitäen kuitenkin samalla manuaalisen työn määrän järkevissä rajoissa. Itse testit ajettiin automaattisesti, mutta niiden siirto taulukkolaskentaohjelmaan tapahtui käsin.

Lighthousen komentorivityökalu asennetaan seuraavalla komennolla:

```
$ npm install -g lighthouse
```

Testaustyökalu mahdollistaa lukuisten eri asetusten käytön. Tässä tutkimuksessa käytettiin

samoja asetuksia, kuin HTTP Archive -projekti käyttää heidän omissa mittauksissaan. HTTP Archive on avoimen lähdekoodin projekti, joka tutkii ja seuraa internetin sivustojen tilaa. (HTTP Archive 2022). Samojen asetusten käyttö mahdollistaa vertailun heidän keräämäänsä dataan.

Mittauksissa käytetyt asetukset:

```
lighthouse --throttling.cpuSlowdownMultiplier=1 \
--throttling.downloadThroughputKbps=1600 \
--throttling.uploadThroughputKbps=768 \
--throttling.rttMs=150 \
<url>
```

4 Tutkimus

4.1 Suorituskyvyn optimointi

Verkkosivustojen nopeutta voidaan parantaa useilla eri tavoilla. On olemassa yleispäteviä toimenpiteitä, jotka on järkevää toteuttaa kaikkiin verkkosivustoihin. Sellaisia ovat esimerkiksi lazy loading-attribuutin käyttöönotto ja kuvien muuntaminen kevyempään formaattiin, jotka toteutettiin myös tämän tutkimuksen verkkosivustolle. Lisäksi on olemassa toimenpiteitä, jotka vaativat tarkempaa kohdesivuston tuntemusta, ja riippuvat pitkälti sivuston ominaisuuksista ja arkkitehtuurista. Sellaisena toimenpiteenä toteutettiin esilataus-attribuutin käyttöönotto.

WordPress on automaattisesti käyttänyt kuvien latauksessa lazy loading-attribuuttia versiosta 5.4 lähtien. Tämän takia ensimmäisiin mittauksiin ominaisuus otettiin kokonaan pois käytöstä. Viscomi ja Arntz (2021) havaitsivat mittauksissaan, että vain näyttöalueen ulkopuolella olevissa kuvissa kannattaa käyttää lazy loading-ominaisuutta, jonka johdosta ominaisuus päätettiin toteuttaa muokattuna.

Käytössä olevan WordPress-teeman `functions.php` tiedostoon lisättiin ominaisuus, joka poistaa loading-attribuutin ensimmäiseltä kuvalta.

```
// Toiminnallisuus yksinkertaistettuna
function add_responsive_class($content){
  if ( is_page()) {
    ..
    $imgs = $document->getElementsByTagName('img');
    $img = $imgs[0];
    if ($imgs[0] == 1) {
    $img->removeAttribute( 'loading' );
    $html = $document->saveHTML();
    return $html;
    ..
}
```

Kuvien optimointi toteutettiin muuntamalla käytössä olevat JPG-kuvat WEBP-formaattiin. Kuville ei tehty muita optimointeja tai pakkauksia.

Viimeisenä optimointitoimenpiteenä oli rajapinnan osoitteen lisääminen sivuston otsakkeeseen esilataus-attribuutin kanssa. Esilataus-attribuutti ilmoittaa selaimelle, että osoitettu resurssi on tärkeä ja se halutaan asettaa tärkeysjärjestyksessä korkealle, eli sen lataus halutaan aloittaa mahdollisimman pian.

```
link: [
    { rel: 'preconnect', href: 'https://www.tuomovitikainen.
      fi', crossorigin: true },
],
```

4.2 Hakukoneoptimoinnin toteutus

4.2.1 Sitemap ja robots -tiedostojen luominen

Verkkosivuston hakukoneoptimointia varten sivustolle lisättiin kaksi eri hyödyllistä tiedostoa. Robots.txt tiedoston avulla voidaan esimerkiksi kieltää ryömijöitä indeksoimasta tiettyjä sivuja, kuten esimerkiksi ylläpitäjille tarkoitettu kirjautumissivu. Sitemap.xml taas on tiedosto, joka sisältää listan verkkopalvelun eri sivujen osoitteista.

Nuxtilla näiden tiedostojen luominen onnistui sen omien moduulien avustuksella. Ensimmäisenä paketit asennettiin komentorivityökalulla NPM-pakettienhallintaa käyttäen:

```
$ npm install @nuxtjs/sitemap @nuxtjs/robots
```

Kun paketit oli asennettu onnistuneesti, ne otettiin käyttöön `nuxt.config.js` tiedostossa seuraavalla tavalla:

```
modules: [
  '@nuxtjs/sitemap',
  '@nuxtjs/robots',
],
```

Nuxtin sivukartta-moduuli osaa ottaa automaattisesti huomioon vain staattiset sivut, jotka on luotu sivustohierarkian pages-kansioon. Koska sovelluksessa kaikki sivut luodaan dynaamisesti sisällönhallintajärjestelmässä, joutuu sivukartan eteen tekemään manuaalista työtä. Jos sivuja on todella paljon tai ne muuttuvat usein, uskoisin olevan mahdollista tehdä erillinen funktio, joka osaa tehdä sivukartan automaattisesti ottaen huomioon myös dynaamiset sivut. Sivustolla lisättiin kuitenkin etusivun lisäksi vain kaksi muuta sivua, joten ne oli helppo lisätä manuaalisesti.

```
sitemap: {
  routes: [
    '/yhteydenotto',
    '/referenssit',
  ]
},
```

Molemmat moduulit sisältävät lukuisan määrän eri asetuksia, jotka on nähtävillä niiden dokumentaatioissa. Esimerkiksi seuraavalla asetuksella voidaan kieltää Googlebottia indeksoimasta kaikkia sivuja tai tiedostoja, jotka sijaitsevat `/users/`-hakemiston sisällä.

```
robots: [
  {
    UserAgent: 'Googlebot',
    Disallow: () => '/users/'
  }
]
```

Kun halutut asetukset oli asetettu, sovellus generoi automaattisesti molemmat tiedostot verkkosivun juureen.

4.2.2 Metatietojen tuominen sisällönhallintajärjestelmästä

Metatiedoilla tarkoitetaan tietoja, jotka eivät näy suoraan verkkosivulla, vaan hakukone sekä selain hyödyntää niitä. Esimerkiksi metatietojen otsikkoa käytetään selaimen välilehdissä informoidakseen käyttäjälle, mikä sivu on kyseessä. WordPressin tapauksessa sivun meta-

tiedot olisi voinut luoda lisäkenttien avulla. Tässä tapauksessa päätettiin kuitenkin hyödyntää Yoast SEO -lisäosaa, joka on suosituimpia hakukoneoptimointilaajennuksia WordPressiin, sekä yleisesti käytössä kun luodaan verkkopalveluita yrityksille. Yoast tukee GraphQL -rajapintaa, joka mahdollistaa metatietojen haun sovellukseen. Ensimmäisenä luotiin kysely:

```
const METADATA_QUERY = gql`
query MyQuery {
  page(id: "ID", idType: DATABASE_ID) {
    seo {
      metatitle: title,
      metadesc: metaDesc,
    }}}`;
```

Kyselyssä haettiin sivuston id-tunnisteella kyseisen sivun metaotsikko ja -kuvaus. Oletuksena sivuston metatiedot sijaitsevat `nuxt.config.js` -tiedostossa. Metatiedot voidaan asettaa head-objektin avulla jokaiselle verkkosivuston sivulle erikseen:

```
head() {
      return {
        title: this.page.seo.metatitle,
        meta: [
          {
            content: this.page.seo.metadesc,
}]}}
```

4.2.3 Alkuperäisen sisällön osoittaminen

Verkkosivustoissa samaan sivuun voi osoittaa usea eri osoite. Rel=canonical -elementin avulla voidaan osoittaa alkuperäinen sivu, joka myös halutaan indeksoitavaksi hakukoneryömijöiden toimesta. Canonical-url tulee asettaa sivuston otsakkeeseen. Nuxtilla tämä tehtiin `layouts/default.vue` -tiedostossa asettamalla linkki seuraavasti:

```
export default {
  head() {
```

```
    return {
      link: [
        {
          rel: 'canonical',
          href: 'baseUrl' + this.$route.path
}]}}}
```

4.2.4 404-tapahtuman käsittely

SPA-sovelluksissa 404-tapahtuman käsittely täytyy huomioida ja käsitellä erikseen. Koska sivuja ei haeta erikseen palvelimelta vaan sisältö renderöidään sivupohjaan, ei sivusto palauta 404-statuskoodia vaikka haettua sivua ei olisi olemassa. 404-statuskoodin palauttaminen on tärkeää hakukoneoptimoinnin kannalta, jotta ryömijät eivät tuhlaa resursseja olemattomiin sivuihin, ja tällöin myös vältytään niiden turhalta indeksoinnilta. Tässä sovelluksessa tämän tapaus ratkaistiin lausekkeella, jossa tarkistetaan, onko haettua sivua olemassa.

```
if (store.state.posts.post === null) {
      error({statusCode: 404, message: "Sivua ei löydy"})
    }
```

Jos sivua ei löydy, sovellus palauttaa nyt HTTP-statuskoodin 404. Virheilmoitussivu on kuitenkin Nuxtin oma, muun sivuston graafisesta ilmeestä täysin poikkeava eikä sisällä esimerkiksi navigaatiota. Esimerkiksi Google neuvoo kehittäjäohjeissaan tekemään virheilmoitussivusta muun sivuston kaltaisen ja ohjaamaan käyttäjää eteenpäin virheilmoitussivulta (Google 2022b). Tällä voi olla vaikutusta siihen, jatkaako sivuston käyttäjä selailua vai lähteekö hän sivustolta. Jotta voimme tehdä räätälöidyn virheilmoitussivun, luotiin kansioon `layouts` tiedosto nimeltä `error.vue`. Kyseessä on sivu, joka näytetään virheen sattuessa. Sivun voi räätälöidä näyttämään eri sisältöjä eri virheilmoituksien tapauksissa.

```
<template>
  <div>
    <h1 v-if="error.statusCode === 404">Sivua ei
       valitettavasti löydy!</h1>
```

```
    <h1 v-else>Virhe tapahtui: - {{ error.statusCode }} </h1
        >
    <NuxtLink to="/">Palaa etusivulle</NuxtLink>
  </div>
</template>

<script>
  export default {
    props: ['error'],
    layout: 'error'
  }
</script>
```

Virhekoodin ollessa 404 kerromme käyttäjälle, että sivua ei löytynyt. Muun virheen sattuessa ilmoitamme virheestä ja näytämme HTTP-statuskoodin. Molemmissa tapauksissa käyttäjälle näytetään linkki, josta pääsee takaisin etusivulle.

5 Tulokset

5.1 Verkkosivuston nopeus

Mittauskertoja pidettiin yhteensä 4 kertaa, joissa jokaiselle renderöintitavalle mittaukset suoritettiin 15 kertaa. Mittauksissa parhaiten menestyivät palvelimella renderöity ja staattinen sivusto. Ensimmäisen mittauskerran tulosten keskiarvot ovat esitettynä taulukossa 4.

Palvelimella renderöity että staattinen sivu saivat myös saman arvon First Contentful Paint -mittauksesta, mikä oli ennakoitavissa, sillä molemmat sivustot ovat valmiiksi renderöityjä selaimelle saapuessaan.

Taulukko 4. Suorituskykymittausten tulokset ilman optimointeja. SSR = Server-side rendering, CSR = Client-side rendering. Taulukoiden arvot ovat esitetty sekunteina ja ovat keskiarvoja. Lisäksi taulukossa on ilmaistuna mittauskertojen keskihajonnat.

	SSR (σ)	CSR (σ)	Staattinen (σ)
First Contentful Paint	1,15 (0,06)	2,83 (0,09)	1,03 (0,05)
Time to Interactive	3,89 (0,09)	3,82 (0,07)	3,95 (0,09)
Speed Index	3,14 (0,92)	5,03 (0,92)	1,74 (0,62)
Total Blocking Time	0,06 (0,03)	0,06 (0,01)	0,04 (0,62)
Largest Contentful Paint	1,15 (0,06)	4,22 (0,36)	1,03 (0,02)
Cumulative Layout Shift	0 (0)	0 (0)	0 (0)
Network request time	1,03 (0,22)	0,11 (0,01)	0,12 (0,02)

Time to Interactive -mittarin tulokset vaikuttavat ensinäkemältä epäloogisilta. Staattinen sivusto sai tästä mittauksesta huonoimman tuloksen, vaikka sivusto muuten on selkeästi mittauksien suorituskykyisin sivusto. Tämä johtuu mittaustyökalun teknisestä toteutustavasta. Vaikka Nuxtilla generoitu staattinen sivusto on rakenteen puolesta staattinen, hyödyntää se JavaScriptiä esimerkiksi sivujen esilatauksissa. Sivusto on siis käyttövalmis, mutta taustalla tehdään vielä työtä, joka vaikuttaa mittauksen tulokseen negatiivisesti.

Suurimmat erot mittauksissa löytyivät Speed Index -arvon kohdalta. Arvo perustuu siihen, että kuinka nopeasti verkkosivun visuaalinen ilme rakentuu sivulatauksen yhteydessä. Työkalu ottaa sivulatauksesta videokuvaa, jonka perusteella arvo lasketaan. Speed Index -arvo ei ota kantaa siihen, onko sivu valmis vuorovaikuttamaan käyttäjän kanssa. Staattinen sivu sai samasta mittauksesta selkeästi parhaimman tuloksen. Staattisen sivuston nopeuden huomaa myös sivustoa käytettäessä, navigoinnin ollessa lähes välitöntä. Selaimessa renderöity verkkosivusto sai lähes sekuntin huonomman tuloksen, kuin palvelimella renderöity. On kuitenkin huomioitava, että tulos ei sisällä pyynnön lähettämisen ja vastaanottamisen välistä aikaa. Tosiasiassa palvelimella renderöity sivusto on hieman hitaampi, sillä sivun generointi palvelimella vie aikaa, ja keskimäärin palvelimen vastaus kesti noin 1,03 sekuntia. Sivusto generoidaan kuitenkin palvelimella vain kerran, sivulle ensimmäisen kerran tultaessa tai sivun uudelleenlatauksen yhteydessä. Kun sivustolla jatketaan navigointia linkkien kautta, muodostetaan sisältö selaimessa kuten selaimessa renderöitävän sivun tapauksessa.

Largest Contentful Paint -mittauksen arvot olivat hyvin lähellä toisiaan sekä staattisella sivustolla että palvelimella renderöidyllä. Selaimessa renderöidyn sivuston heikohko tulos ei ole yllätys, koska mittari muistuttaa tyypiltään Speed Index -mittaria. Testeissä käytetyllä näyttökoolla suurin ruudulla näkyvä elementti on ensimmäinen tekstikappale.

Lazy loading-ominaisuuden lisääminen sivustoille toi selkeää parannusta kaikkien sivustojen suorituskykyyn. Mittauskerran tulokset ovat esitettynä taulukossa 5. Staattisessa sekä palvelimella renderöidyssä sivustoissa Speed Index -arvo parani eniten. Palvelimella renderöidyssä muutos oli 15,9 prosenttia ja staattisessa 26,4 prosenttia. Selaimella renderöidyssä sivustossa taas LCP-arvo parani eniten, kesto lyheni 8,3 prosenttia.

Ennen lazy loading-ominaisuuden lisäystä staattisen sivuston ja esirenderöidyn sivuston koot olivat 2.5 megabittiä. Selaimessa renderöidyn sivuston koko taas oli 2.2 megabittiä. Ominaisuuden lisäyksen jälkee staattisen sekä palvelimella renderöidyn sivuston koot ovat molemmat 1.7 megabittiä, ja selaimessa renderöidyn sivuston koko 1.4 megabittiä. Se, että kuvat ladataan jälkikäteen, ei ole käyttäjän havaittavissa. Lazy loading-ominaisuus osaa ennakoida, ja lataa kuvan hieman ennen kuin se näkyy käyttäjän näytöllä.

Toisella optimointikerralla sivuston kuvat vaihdettiin JPG-formaatin kuvista WEBP-formaatin

Taulukko 5. Suorituskykymittausten tulokset lazy loading-attribuutin lisäämisen jälkeen.

	SSR (σ)	CSR (σ)	Staattinen (σ)
First Contentful Paint	1,11 (0,03)	2,79 (0,09)	0,97 (0,05)
Time to Interactive	3,85 (0,05)	3,93 (0,12)	3,75 (0,06)
Speed Index	2,64 (0,69)	4,73 (0,66)	1,28 (0,55)
Total Blocking Time	0,06 (0,02)	0,05 (0)	0,05 (0,02)
Largest Contentful Paint	1,10 (0,03)	3,87 (0,21)	0,98 (0,04)
Cumulative Layout Shift	0 (0)	0 (0)	0 (0)
Network request time	0,94 (0,15)	0,09 (0,02)	0,12 (0,03)

kuviin, jonka vaikutus on nähtävissä taulukosta 6. Kuvien koko putosi yhteensä 235 kilobittiä, eli 23 prosenttia. Jälkeenpäin ajateltuna kuvaformaatin vaihtamisen vaikutuksen suhteessa sivuston suorituskykyyn olisi huomannut tekemällä mittaukset ennen lazy loading-ominaisuuden lisäämistä. Valitettavasti niiden yhteistä vaikutusta suorituskykyyn on hankala mitata, mutta 23 prosentin pudotus kuvien kokoon tekee kuvien formaattivaihdoksesta ehdottomasti kannattavan optimointitoimenpiteen.

Taulukko 6. Suorituskykymittausten tulokset kuvien optimoinnin jälkeen.

	SSR (σ)	CSR (σ)	Staattinen (σ)
First Contentful Paint	1,12 (0,04)	2,8 (0,07)	1,01 (0,14)
Time to Interactive	3,64 (0,1)	3,71 (0,08)	3,55 (0,08)
Speed Index	2,77 (0,53)	4,73 (0,44)	1,38 (0,51)
Total Blocking Time	0,06 (0,03)	0,05 (0,01)	0,06 (0,03)
Largest Contentful Paint	1,12 (0,04)	3,72 (0,11)	1,01 (0,14)
Cumulative Layout Shift	0 (0)	0 (0)	0 (0)
Network request time	0,99 (0,25)	0,09 (0,02)	0,18 (0,02)

Viimeinen tehtävä optimointitoimenpide oli rajapinnan osoitteen lisäys sivuston otsakkeeseen esilataus-attribuutin kanssa. Mittauskerran tulokset ovat esitettynä taulukossa 7. Ennakko-

oletus oli, että tällä voisi olla oikeastaan vaikutusta vain selaimessa renderöityyn sivustoon. Staattiseen sivuston suorituskykyyn attribuutin lisäämisellä ei tulisi olla vaikutusta, ja palvelimella renderöityyn sivustoon vaikutus oli ennalta vaikeasti arvioitavissa.

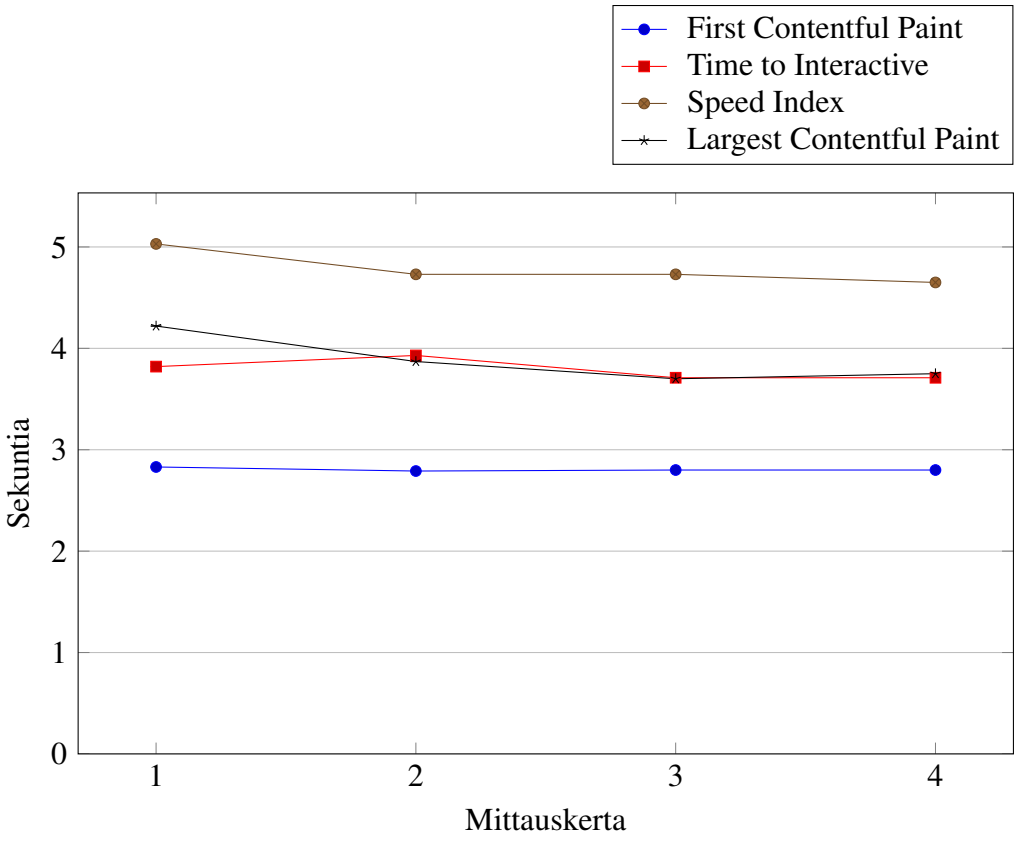
Tuloksista on havaittavissa, että ainoat havaittavat parannukset ovat Speed Index -arvoissa. On kuitenkin huomattava, että myöskin staattisen sivuston Speed Index -arvo on kaikkien testauskertojen matalin. Erot mittauksien tuloksissa voivat olla myös vaihtelua mittauskertojen välillä, eikä esilataus-attribuutin hyötyä voi yksiselitteisesti todeta tämän testauskerran perusteella.

Taulukko 7. Suorituskykymittausten tulokset esilataus-attribuutin lisäämisen jälkeen.

	SSR (σ)	CSR (σ)	Staattinen (σ)
First Contentful Paint	1,11 (0,04)	2,81 (0,08)	1,03 (0,03)
Time to Interactive	3,67 (0,06)	3,69 (0,09)	3,51 (0,08)
Speed Index	2,41 (0,52)	4,62 (0,6)	1,17 (0,35)
Total Blocking Time	0,05 (0,03)	0,05 (0,01)	0,05 (0,02)
Largest Contentful Paint	1,11 (0,04)	3,75 (0,12)	1,00 (0)
Cumulative Layout Shift	0 (0)	0 (0)	0 (0)
Network request time	0,89 (0,19)	0,14 (0,02)	0,12 (0,01)

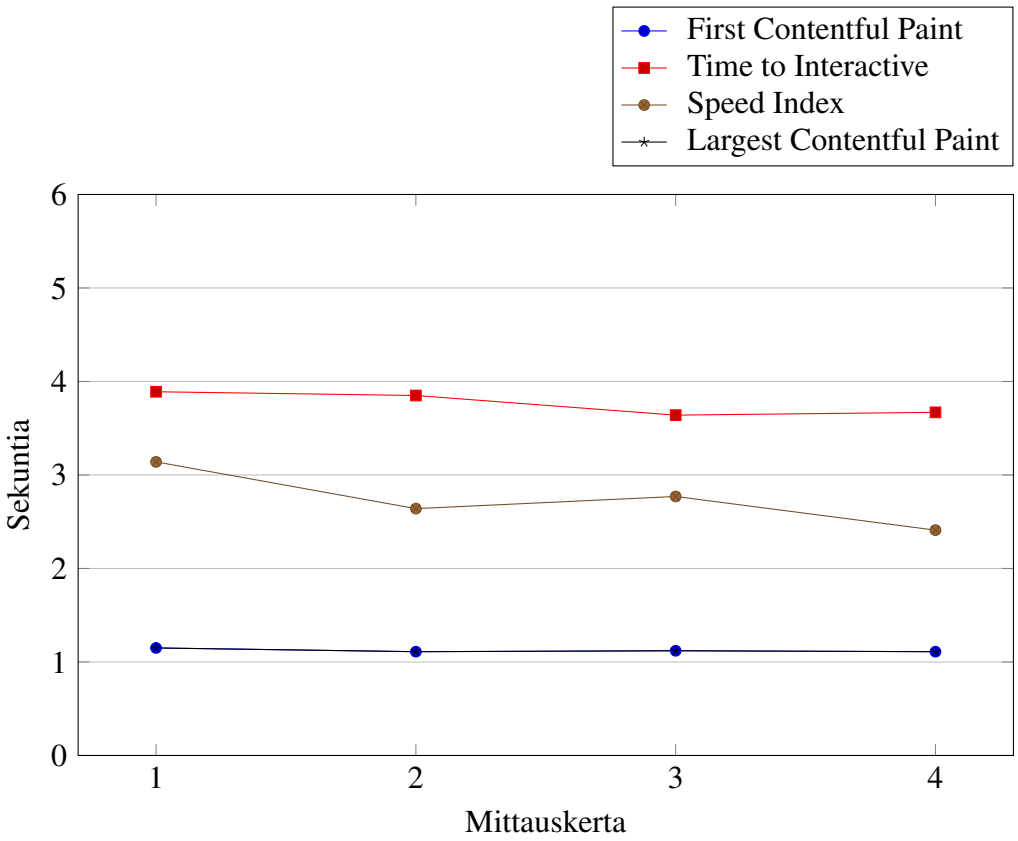
Kuviossa 5 on nähtävillä selaimessa renderöidyn sivuston mittauskerrat kootusti. Lisäksi taulukkoon on sisällytetty mittausten keskihajonnat. Kuten kuviosta voimme havaita, selkeästi suurin hajonta mittausten välillä oli Speed Index -arvossa. Keskihajonta on suurimmillaan ensimmäisellä mittauskerralla, jolloin se oli 0,92 sekuntia. Hajontaa kasvattaa mittauskerta 14., jolloin Speed Index -mittarin arvo oli poikkeuksellisesti 7,4 sekuntia, mikä on useita sekunteja yli keskiarvon.

Silmiinpistävää tuloksissa oli matala Time to Interactive -arvo suhteessa kahteen muuhun renderöintitapaan. Mittari on määritelty siten, että sivusto voi olla interaktiivinen ennen kuin visuaalinen sisältö on täysin muodostunut, mutta odotus ennen mittauksia oli, että sivuston täysi interaktiivisuus olisi tapahtuma joka tapahtuisi viimeisenä.



Kuvio 5. Selaimessa renderöidyn sivuston neljä mittauskertaa, ja neljän tärkeimmän mittarin arvot. Ensimmäinen mittauskerta on suoritettu sivustolle, jolle ei ole tehty optimointeja.

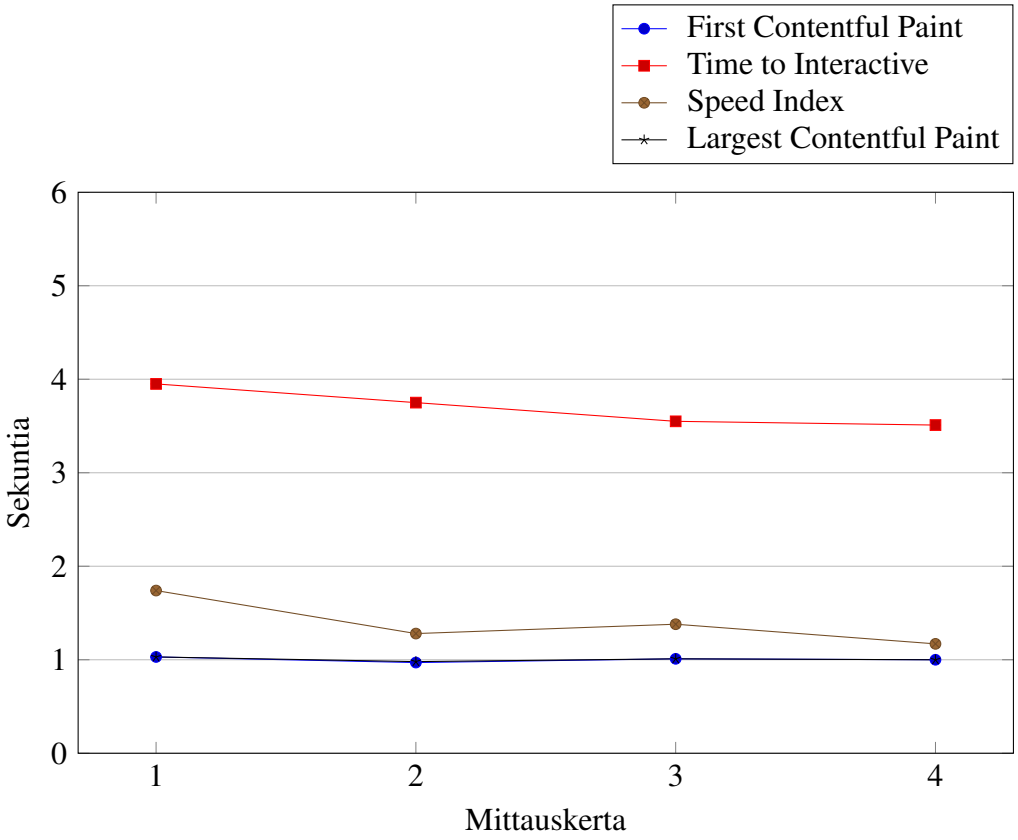
Kuviossa 6. on nähtävillä palvelimella renderöidyn sivuston mittauskerrat kootusti. Myöskin palvelimella renderöidyllä sivustolla selkeästi eniten hajontaa oli Speed Index -arvossa. Muiden arvojen mittausten tulokset taas olivat hyvin yhteneväisiä ja hajonta pientä. Selkeää syytä sille, miksi Speed Index -arvossa esiintyy eniten hajontaa, ei ole tiedossa. Yhtenä tekijänä voitaisiin pitää sitä, että kyseinen mittari on monimutkaisempi rakenteeltaan ja käyttää tuloksen laskemisessa videokuvaa. Tuloksen laskennassa on useampia vaikuttavia tekijöitä ja siten vaihtelua tuloksissa syntyy enemmän. Kuviosta on havaittavissa, että First Contentful Paint -mittarin ja Largest Contentful Paint -mittarin tulokset ovat samat. Tämä johtuu siitä, että molempien mittareiden kohteena on sama elementti. Lighthousen tallentamasta videokuvasta näkee, että kaikki tekstit ilmestyvät kerralla, ja näkymän isoin elementti on ensimmäinen tekstikappale.



Kuvio 6. Palvelimella renderöidyn sivuston neljä mittauskertaa, ja neljän tärkeimmän mittarin arvot. Ensimmäinen mittauskerta on suoritettu sivustolle, jolle ei ole tehty optimointeja.

Kuviossa 7. on nähtävillä staattisesti generoidun sivuston mittauskerrat kootusti. Aiemmis-

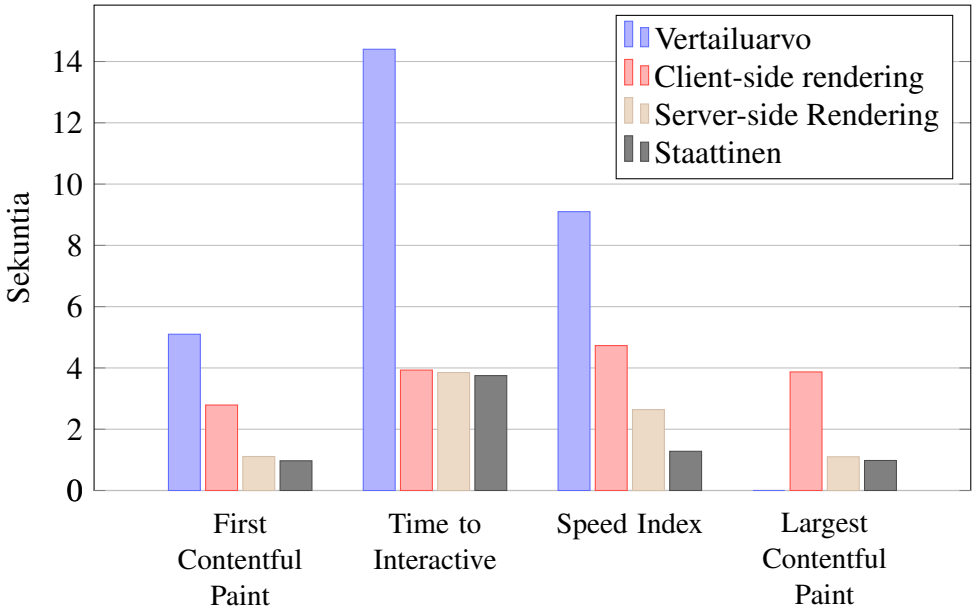
sa tuloksissa esiintynyt suurehko hajonta on nähtävissä myös tämän sivuston mittauksissa. Kolmannella mittauskerralla 5. mittaus antoi LCP-mittarille tuloksen 1,5 sekuntia, mikä luo muista testikerroista eroavan hieman suuremman hajonnan. Visuaalinen sisältö taas piirtyy ruudulle erittäin nopeasti: alle puolen sekuntin sisällä siitä, kun ensimmäinen pikseli piirtyy ruudulle, ovat myös muut sisällöt latautuneet sivuston taitekohdan yläpuolelta. Myös staattisen sivuston kanssa käy samoin First Contentful Paint ja Largest Contentful Paint -mittareiden kanssa kuin palvelimella renderöidyssä, jonka takia kuviot ovat päällekkäin.



Kuvio 7. Staattisen sivuston neljä mittauskertaa, ja neljän tärkeimmän mittarin arvot. Ensimmäinen mittauskerta on suoritettu sivustolle, jolle ei ole tehty optimointeja.

Mittauksissa käytettiin samoja asetuksia, kuin HTTP Archive-projekti käyttää heidän omissa mittauksissaan. Tämä mahdollistaa epämuodollisen vertailun tässä tutkimuksessa saatujen tulosten ja heidän tulostensa välillä. Tulokset koostettuna ovat esillä kuviossa kahdeksan. Vaikkakin mittauksissa käytetyt verkkosivustot sisältävät suurin piirtein keskiarvon verran kuvia ja resursseja kuin HTTP Archiven mukainen keskimääräinen verkkosivusto, on verk-

kosivustoissa usein eroja, jotka vaikuttavat sen nopeuteen. Verkkosivustoilla on usein käytössä erilaista analytiikkaa, jota käytetään käyttäjädatan keräämiseen. Analytiikkaa voi olla useasti eri lähteestä, ja esimerkiksi verkkokaupan tapauksessa voidaan haluta tietää, minkä sosiaalisen median kautta juuri ostoksen tehnyt käyttäjä on alun perin saapunut sivuille. Analytiikka on usein skriptillä ladattava resurssi, joten sillä voi olla vaikutus latausnopeuteen. Luotu testisivusto ei myöskään sisältänyt upotuksia, kuten esimerkiksi sosiaalisen median kuvaseiniä tai mainoksia.



Kuvio 8. Tärkeimmät mittarit ja vertailuarvot HTTP Archivesta. HTTP Archivessa ei ole saatavilla largest contenful paint-mittausta.

5.2 Hakukoneoptimointi

Sivustolle toteutettiin luvussa 4.2 joukko teknisen hakukoneoptimoinnin toimenpiteitä, ja tässä alaluvussa tarkastellaan, kuinka toimenpiteet onnistuivat. Teknisen hakukoneoptimoinnin mittaaminen on tyypiltään erilaista, kuin verkkosivun nopeuden: joko ominaisuus löytyy sivustolta tai on löytymättä. Toteutustavat voivat erota sivuston arkkitehtuurista riippuen, mutta tehtävät toimet ovat pääosin samat. Kun käytössä on Nuxt.js-kirjasto, edellytti toimenpiteiden tekeminen Nuxtin kehittämien moduulien lataamista ja asetusten laittoa oikeiksi. Tätä voidaan pitää hyvänä toimintatapana, sillä tällöin itse kirjasto ei sisällä mitään tarpee-

tonta, vaan sovellusta voi täydentää tarpeen mukaan.

Erikseen ladattavilla moduuleilla sivustolle luotiin sitemap.xml- ja robots.txt-tiedostot. Näiden kahden tiedoston olemassaolo on helppo todeta selaimen kautta, lisäämällä kyseisten tiedostojen nimet verkkosivuston osoitteen perään. Moduulien käyttö on mutkatonta, mutta sivukartan luominen saattaa teettää ylimääräistä työtä. Moduuli osaa nimittäin lisätä sivukarttaan vain staattiset, sivuston koodissa manuaalisesti luodut sivut. Sisällönhallinnan kautta luodut sivut ovat dynaamisia sivuja, joita moduuli ei osaa automaattisesti lisätä sivukarttaan, vaan ne täytyy itse käsin lisätä. Ongelma ei ole suuri jos sivusto on pienehkö tai tarvittavat sivut ovat ennalta helposti määriteltävissä. Jos kyseessä on suurempi verkkopalvelu, voi olla tarpeen kehittää ratkaisu, jonka avulla myös dynaamiset sivut saataisiin lisättyä automaattisesti sivustokarttaan.

Sivuston metatietoja hallitaan sisällönhallintajärjestelmän kautta Yoast SEO-lisäosan avulla. Yoast tarjoaa tuen myös GraphQL-kyselyille, jonka avulla metatiedot haettiin. Metatiedot pystyy asettamaan sekä sivusto- että sivukohtaisesti, joka oli tavoitteena. Kaikki Yoast SEO-lisäosan tarjoamat ominaisuudet eivät kuitenkaan ole käytössä ilman lisätyötä. Lisäosaa käytetään usein esimerkiksi murupolkujen muodostamiseen, jotka helpottavat navigointia laajemmilla sivustoilla. Ominaisuus olisi kuitenkin mahdollista toteuttaa myös sivustollemme.

Alkuperäisen sivun osoittaminen onnistui suoraviivaisesti ja oli nopea toteuttaa. Ominaisuus toimii suoraan sisällönhallinnan kautta luoduilla sivuilla, joiden lisääminen sivukarttaan täytyi tehdä käsin. Alkuperäisen sisällön sijainti merkitään verkkosivustoilla <head>-tagien sisään. Palvelimella sijaitsevalla verkkosivustolla voimme navigoida referenssi-sivulle, ja tarkistaa selaimen kehittäjäkonsolista sivuston rakennepuusta osoitteen:

```
<link data-n-head="ssr" rel="canonical" href="https://nuxt-
    tuomovit.vercel.app/referenssit">
```

404-tapahtuman käsittely on asia, joka SPA-verkkosivustoja tehdessä kannattaa ottaa huomioon, vaikkei kovin kiinnostunut hakukoneoptimoinnista muuten olisikaan. Kirjaston tarjoama geneerinen virheilmoitussivu saattaa hämmentää sille joutunutta sivuston käyttäjää, eikä tarjoa linkkiä takaisin etusivulle. Selaimen kehittäjätyökalun avulla pystyy todentamaan, että verkkosivusto todella palauttaa HTTP-statuskoodin 404, kun käyttäjä on yrittä-

mässä sivulle jota ei ole olemassa. Näytettävän virheilmoituksen voi muokata haluamakseen statuskoodin mukaan.

Verkkosivuston yhtenä päävaatimuksena oli, että sivusto täytyy olla hakukoneiden ryömijöiden läpikäytävissä. Tutkielman teoriaosuudessa selvisi, että hakukoneryömijät pystyvät vaihtelevasti indeksoimaan JavaScripillä luotua sisältöä. Koska kaikkien eri hakukoneiden ryömijöillä sivuston testaaminen olisi työlästä eikä siten mahtuisi tämän tutkielman puitteisiin, päätettiin sivustojen indeksoitavuus testata Googlen ja Bingin hakukoneilla. Valitettavasti ilmeisesti siitä syystä, että verkkosivustoissa käytetään palvelimen tarjoajan ilmaista domainia, ei Bing hyväksy sivustoja indeksoitavaksi. Bing kertoo sivuillaan, että JavaScript-sisällön renderöinti ei ole ongelma. Kuitenkin Sharp (2018) on havainnut, että Bing ei lupauksistaan huolimatta indeksoi JavaScript-sisältöä. Jos hakukonenäkyvyys kaikissa hakukoneissa on tärkeää ja sivusto renderöidään selaimessa, kannattaa asiasta varmistua.

Verkkosivuston kaikki kolme versiota lisättiin Googlen Search Console -palveluun. Search Console on hakukoneoptimointiin liittyvä palvelu, joka sisältää useita eri toimintoja. Sen avulla voidaan esimerkiksi lähettää sivuston sivukartta tai pyytää hakukonetta indeksoimaan verkkosivusto uudestaan. Testi osoittaa, että palvelimella ja selaimella renderöidyt sekä staattinen sivusto ovat kaikki Googlen hakukonebotin indeksoitavissa.

6 Pohdinta

Tässä tutkimuksessa oli kaksi tutkimuskysymystä, jotka käsittelivät SPA-verkkosivuston nopeutta ja hakukoneoptimointia. Tutkimuksen alkuperäisenä kiinnostuksenkohteena oli tekninen hakukoneoptimointi, minkä lisäksi tarkoituksena oli käyttää ainoastaan yhtä renderöintimenetelmää. Tutkimusprosessin edetessä päätettiin ottaa mukaan laajemmin verkkosivuston eri renderöintitavat sekä nopeusmittaukset, joista muodostui toinen tutkimuskysymys. Eri renderöintitapojen mukaanotto mahdollisti myös niiden välisten nopeuserojen mittaamisen, mikä toi uutta tietoa renderöintitapojen nopeuseroista. Lisäksi se mahdollisti sivustojen indeksoitumisen testaamisen käytännössä.

Suunnittelutiede osoittautui toimivaksi tutkimusmenetelmäksi tämänkaltaiseen tutkimukseen, varsinkin kun sivustoa haluttiin kehittää, eikä vain käyttää valmista lähdemateriaalia. Artefakti, eli mittauksien kohteena ollut verkkosivusto, onnistui odotusten ja tavoitteiden mukaisesti. Se täytti vaatimukset jotka sille oli asetettu, jotta sivustojen nopeutta ja teknistä hakukoneoptimointia voitiin testata. Huomioitava asia liittyy artefaktin toteutuksen loppupuoliskoon: kuinka päättää, mitä sisältöjä sivustolle tulisi asettaa, jotta se olisi keskiverto? Tässä tapauksessa käytettiin apuna sivustoa, joka kerää dataa internetin tuhansista eri verkkosivustoista. Toinen kohde, joka voitaisiin ottaa huomioon tulevassa tutkimuksessa, liittyy sivustojen domaineihin. Kuten luvussa 5.2 todettiin, Googlen ja Bingin hakukoneista vain Google hyväksyi ilmaisia domaineja käyttävät sivustot indeksoitavaksi. Siitä huolimatta, teoriassa useammassa paikkaa ilmennyt huoli SPA-verkkosivuston huonosta hakukonenäkyvyydestä osoittautui turhaksi. Jos sivuston ylläpitäjälle riittää näkyvyys Googlen hakukoneessa, voidaan käyttää normaalia renderöintiä selaimessa. Ei kuitenkaan ole syytä olla hyödyntämättä palvelimella renderöityä ja staattista sivustoa, sillä niissä indeksointiongelmia ei ole ja lisäksi ne osoittautuivat nopeammiksi menetelmiksi.

Koska verkkosivut ovat nykyään hyvin monimutkaisia kokonaisuuksia, ei niiden suorituskyvyn mittaaminenkaan ole yksinkertaista. Vaikka mittaustyökalu antaa tulokseksi tarkan arvon, sivuston toteutustapa voi vaikuttaa arvoon tavalla, jonka jälkeen se ei enää kuvasta totuutta. Tuloksissa huomattiinkin epäloogisuus, joka johtui SPA-arkkitehtuurin toimintatavasta. Mittauksissa saatujen tulosten keskihajonta oli hyvin pientä lukuun ottamatta Speed

Index -arvon vaihtelua, jonka syytä ei onnistuttu osoittamaan. Virheen mahdollisuus on olemassa myös mittausten manuaalisessa työssä: tulokset on käsin kirjattu taulukkolaskentaohjelmaan, jonka jälkeen ne on tuotu tutkimusraporttiin.

Jatkotutkimuksissa voitaisiin selvittää, millainen vaikutus nopeuteen olisi analytiikalla tai dynaamisilla upotuksilla, joita ei sisällytetty tähän tutkimukseen. Lisäksi nopeusmittauksissa voitaisiin käyttää vertailukohteena myös perinteistä verkkosivustoa, jotta arkkitehtuurien välisiä nopeuseroja päästäisiin tarkemmin selvittämään. Mielenkiintoista olisi myös tutkia palvelimen vaikutusta verkkosivuston renderöintinopeuteen, jotta palvelimien väliset mahdolliset nopeuserot saataisiin selville.

Tulokset osoittavat, että SPA-arkkitehtuurilla toteutettu verkkosivusto on varteenotettava vaihto, kun käytössä on Nuxt.js-sovelluskehys. Sivusto on erityisesti palvelimella renderöitynä ja staattisena versiona nopea, ja eri renderöintitavat tuovat myös vaihtoehtoja riippuen siitä, mitä verkkosivustolta halutaan. Kun SPA-verkkosivustoon on yhdistetty moderni sisällönhallintajärjestelmä, on myös sisältöjen päivittäminen helppoa, tuoden mukanaan SPA-arkkitehtuurin edut. Näkisin, että tämän yhdistelmän yleistymistä rajoittaa lähinnä valmiiden ratkaisujen puute. WordPress on useimmille tuttu, turvallinen ja toimiva, ja niistä syistä maailman käytetyin verkkosivualusta, ja sen takia tässä tutkimuksessa haluttiin käyttää nimenomaan sitä sisällönhallintajärjestelmänä. Kyse on myöskin kustannuksista – jos verkkopalveluprojektia lähdetään toteuttamaan uudella teknologialla, se näkyy suoraan työmäärissä ja sitä kautta laskussa. SPA-arkkitehtuuria ei myöskään kannata sivuttaa siinä uskossa, etteikö käyttäjät löytäisi sivustolle hakukoneiden kautta. SPA-toteutuksia on yleisesti käytetty paikoissa, joihin liikenne ohjataan joltain muulta alustalta, eikä hakukonenäkyvyys ole ollut niin kriittinen tekijä. Kuitenkin kun on valittu tilanteeseen sopiva renderöintimenetelmä ja toteutettu tarpeelliset tekniset hakukoneoptimointitekniikat, vastuu hakukonenäkyvyydestä siirtyy sisällöntuottajan harteille.

Lähteet

Bhandari, Ravneet Singh, ja Ajay Bansal. 2018. “Impact of search engine optimization as a marketing tool”. *Jindal Journal of Business Research* 7 (1): 23–36.

Bing. 2022. *Bing Webmaster Guidelines.* https://www.bing.com/webmasters/help/webmasters-guidelines-30fba23a. Viitattu 1. toukokuuta 2022.

Bocchi, Enrico, Luca De Cicco ja Dario Rossi. 2016. “Measuring the quality of experience of web users”. *ACM SIGCOMM Computer Communication Review* 46 (4): 8–13.

Brunelle, Justin F, Mat Kelly, Michele C Weigle ja Michael L Nelson. 2016. “The impact of JavaScript on archivability”. *International Journal on Digital Libraries* 17 (2): 95–117.

Bröhl, Christina, Peter Rasche, Janina Jablonski, Sabine Theis, Matthias Wille ja Alexander Mertens. 2018. “Desktop PC, tablet PC, or smartphone? An analysis of use preferences in daily activities for different technology generations of a worldwide sample”. Teoksessa *International Conference on Human Aspects of IT for the Aged Population,* 3–20. Springer.

Consolidate duplicate URLs. 2021. https://developers.google.com/search/docs/advanced/crawling/consolidate-duplicate-urls. Viitattu 8. tammikuuta 2022.

Davis, Harold. 2006. *Search engine optimization.* "O’Reilly Media, Inc."

Egri, Gokhan, ja Coskun Bayrak. 2014. “The role of search engine optimization on keeping the user on the site”. *Procedia Computer Science* 36:335–342.

Fielding, R, Jim Gettys ym. 1998. “Hypertext Transfer Protocol-HTTP/1.1”.

Gallino, Santiago, Nil Karacaoglu ja Antonio Moreno. 2018. “Why retailers should care about net neutrality: The impact of website performance on online retail”. *Available at SSRN 3260203.*

Gao, Qingzhu, Prasenjit Dey ja Parvez Ahammad. 2017. “Perceived Performance of Top Retail Webpages In the Wild: Insights from Large-Scale Crowdsourcing of Above-the-Fold QoE”. Teoksessa *Proceedings of the Workshop on QoE-Based Analysis and Management of Data Communication Networks.*

Goodman, Danny. 2002. *Dynamic HTML: The definitive reference: A comprehensive resource for HTML, CSS, DOM & JavaScript.* "O'Reilly Media, Inc."

Google. 2022a. *Learn about sitemaps.* https://developers.google.com/search/docs/advanced/sitemaps/overview. Viitattu 8. tammikuuta 2022.

———. 2022b. *Soft 404 errors.* https://developers.google.com/search/docs/advanced/crawling/soft-404-errors. Viitattu 3. helmikuuta 2022.

———. 2022c. *Understand the JavaScript SEO basics.* https://developers.google.com/search/docs/advanced/javascript/javascript-seo-basics. Viitattu 1. toukokuuta 2022.

Góralewicz, Bartosz. 2017. *Going Beyond Google: Are Search Engines Ready for JavaScript Crawling & Indexing?* https://moz.com/blog/search-engines-ready-for-javascript-crawling. Viitattu 1. toukokuuta 2022.

Gupta, Swati, Nitin Rakesh, Abha Thakral ja Dev Kumar Chaudhary. 2016. "Search engine optimization: Success factors". Teoksessa *2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC),* 17–21. https://doi.org/10.1109/PDGC.2016.7913146.

Hora, Diego Neves da, Alemnew Sheferaw Asrese, Vassilis Christophides, Renata Teixeira ja Dario Rossi. 2018. "Narrowing the gap between QoS metrics and Web QoE using Above-the-fold metrics". Teoksessa *International Conference on Passive and Active Network Measurement,* 31–43. Springer.

Hoßfeld, Tobias, Florian Metzger ja Dario Rossi. 2018. "Speed index: Relating the industrial standard for user perceived web performance to web qoe". Teoksessa *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX),* 1–6. IEEE.

HTTP Archive. 2022. *Methodology.* https://almanac.httparchive.org/en/2021/methodology. Viitattu 1. huhtikuuta 2022.

Hui, Zhou, Qin Shigang, Liu Jinhua ja Chen Jianli. 2012. "Study on Website Search Engine Optimization". Teoksessa *2012 International Conference on Computer Science and Service System,* 930–933. https://doi.org/10.1109/CSSS.2012.236.

Iskandar, Taufan Fadhilah, Muharman Lubis, Tien Fabrianti Kusumasari ja Arif Ridho Lubis. 2020. “Comparison between client-side and server-side rendering in the web development”. Teoksessa *IOP Conference Series: Materials Science and Engineering,* 801:012136. 1. IOP Publishing.

Jadhav, Madhuri A, Balkrishna R Sawant ja Anushree Deshmukh. 2015. “Single page application using angularjs”. *International Journal of Computer Science and Information Technologies* 6 (3): 2876–2879.

Krrabaj, Samedin, Fesal Baxhaku ja Dukagjin Sadrijaj. 2017. “Investigating search engine optimization techniques for effective ranking: A case study of an educational site”. Teoksessa *2017 6th Mediterranean Conference on Embedded Computing (MECO),* 1–4. https://doi.org/10.1109/MECO.2017.7977137.

Kumar, Arunjay. 2013. “Search engine optimization (SEO): technical analysis concepts”. *International Journal of Emerging Technology and Advanced Engineering* 3 (3): 123–128. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.413.4659&rep=rep1&type=pdf.

Lewandowski, Dirk, Friederike Kerkmann, Sandra Rümmele ja Sebastian Sünkler. 2018. “An empirical investigation on search engine ad disclosure”. *Journal of the Association for Information Science and Technology* 69 (3): 420–437.

Levene, Mark. 2011. *An introduction to search engines and web navigation.* John Wiley & Sons.

Malaga, Ross A. 2010. “Search engine optimization—black and white hat approaches”. Teoksessa *Advances in Computers,* 78:1–39. Elsevier.

Meneses, Luis, Richard Furuta ja Frank Shipman. 2012. “Identifying “Soft 404” error pages: analyzing the lexical signatures of documents in distributed collections”. Teoksessa *International Conference on Theory and Practice of Digital Libraries,* 197–208. Springer.

Mikowski, Michael, ja Josh Powell. 2013. *Single page web applications: JavaScript end-to-end.* Simon / Schuster.

Mozilla. 2022. *Window: load event.* https://developer.mozilla.org/en-US/docs/Web/API/Window/load_event. Viitattu 20. maaliskuuta 2022.

Ohye, Maile, ja Joachim Kupke. 2012. *The Canonical Link Relation.* RFC 6596, 6596, huhtikuu. https://doi.org/10.17487/RFC6596. https://www.rfc-editor.org/info/rfc6596.

Olston, Christopher, ja Marc Najork. 2010. *Web crawling.* Now Publishers Inc.

Osmani, Addy, ja Ilya Grigorik. 2019. *Using page speed in mobile search ranking.* [Verkkosivu; Luettu 8.11.2021]. https://developers.google.com/web/updates/2018/07/search-ads-speed.

Patil Swati, P, BV Pawar ja S Patil Ajay. 2013. "Search engine optimization: A study". *Research Journal of Computer and Information Technology Sciences* 1 (1): 10–13. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.1070.1729&rep=rep1&type=pdf.

Peffers, Ken, Tuure Tuunanen, Marcus A. Rothenberger ja Samir Chatterjee. 2007. "A Design Science Research Methodology for Information Systems Research". *Journal of Management Information Systems* 24 (3): 45–77. https://doi.org/10.2753/MIS0742-1222240302. eprint: https://doi.org/10.2753/MIS0742-1222240302. https://doi.org/10.2753/MIS0742-1222240302.

Prieto, Vıctor M, Manuel Alvarez ja Fidel Cacheda. 2014. "Soft-404 Pages, A Crawling Problem." *J. Digit. Inf. Manag.* 12 (2): 73–92.

Scott Jr, Emmit A. 2015. *SPA Design and Architecture: Understanding single-page web applications.* Simon / Schuster.

Sharma, Dushyant, Rishabh Shukla, Anil Kumar Giri ja Sumit Kumar. 2019. "A brief review on search engine optimization". Teoksessa *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence),* 687–692. IEEE.

Sharp, Dan. 2018. *Is Bing Really Rendering & Indexing JavaScript?* https://www.screamingfrog.co.uk/bing-javascript/. Viitattu 1. toukokuuta 2022.

Slatin, John M. 2001. "The art of ALT: toward a more accessible Web". *Computers and Composition* 18 (1): 73–81. ISSN: 8755-4615. https://doi.org/https://doi.org/10.1016/S8755-4615(00)00049-9. https://www.sciencedirect.com/science/article/pii/S8755461500000499.

Smith, Peter G. 2012. *Professional Website Performance: Optimizing the Front-End and Back-End.* John Wiley & Sons.

Stadnik, Wiktor, ja Ziemowit Nowak. 2017. “The impact of web pages’ load time on the conversion rate of an e-commerce platform”. Teoksessa *International Conference on Information Systems Architecture and Technology,* 336–345. Springer.

Statcounter. 2021. *Desktop Search Engine Market Share Worldwide.* https://gs.statcounter.com/search-engine-market-share/desktop/worldwide. Viitattu 30. marraskuuta 2021.

Wang, Xiao Sophia, Aruna Balasubramanian, Arvind Krishnamurthy ja David Wetherall. 2013. “Demystifying Page Load Performance with {WProf}”. Teoksessa *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13),* 473–485.

Viscomi, Rick, ja Felix Arntz. 2021. *The performance effects of too much lazy-loading*. https://web.dev/lcp-lazy-loading/. Viitattu 10. huhtikuuta 2022.

Vue.js. 2021. *Vue.js Server-Side Rendering Guide.* https://ssr.vuejs.org. Viitattu 21. tammikuuta 2022.