Niki Järvinen

Verkkosivuston käyttöliittymän nopeuden ja suorituskyvyn optimointi

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Mediatekniikan koulutusohjelma
Insinöörityö
8.11.2018

Metropolia

Tekijä Otsikko Sivumäärä Aika	Niki Järvinen Verkkosivuston käyttöliittymän nopeuden ja suorituskyvyn optimointi 33 sivua 8.11.2018
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Mediatekniikka
Suuntautumisvaihtoehto	Digitaalinen media
Ohjaaja	Yliopettaja Kari Aaltonen

Insinöörityön tarkoitus oli tutkia verkkosivuston käyttöliittymän suorituskykyvyn ja nopeuden merkitystä ja niihin vaikuttavia tekijöitä ja tehdä koulutuskäyttöä varten esimerkit hitaasti toimivan verkkosivun optimoinnista. Työssä perehdyttiin erilaisiin verkkosivuston käyttöliittymän suorituskykyä koskeviin suosituksiin, menetelmiin ja työkaluihin. Työtä tehdessä selvisi, että optimointiin on tarjolla useita eri työkaluja ja ohjeita, ja täydellisesti optimoidun verkkosivuston saavuttamiseksi on tehtävä paljon analysointia ja korjaavia toimenpiteitä.

Insinöörityön päämääränä asiakasyrityksen kannalta oli saada koulutuskäyttöön soveltuvat esimerkit suorituskyvyltään hitaasta verkkosivustosta ja optimoidusta sivustosta. Työssä tehtiin esimerkki verkkokaupasta, joka sisälsi yleisiä suorituskykyä heikentäviä tekijöitä ja joiden vuoksi sivuston suorituskyky oli todella heikko. Tästä lähtökohdasta sivustolle toteutettiin optimointityö, jossa tehtiin tarvittavat toimenpiteet suorituskyvyn parantamiseksi. Sivuston suorituskykyä arvioitiin jatkuvasti, ennen optimointia, sen aikana ja sen jälkeen. Koska optimoitava verkkosivusto oli verkkokauppa, oli hyvä suorituskyky ja sulava toimivuus äärimmäisen tärkeää, sillä kaikki lataamiseen käytetty aika on pois myynnistä. Työn tuloksena verkkokaupan latausaikaa saatiin nopeutettua 17 sekunnilla niin, että uusi latausaika oli 0,9 sekuntia. Myös laskennalliset käyttökustannukset saatiin alennettua 2,70 Yhdysvaltain dollarin maksimista 0,90 Yhdysvaltain dollarin maksimiin. Tällainen käyttöliittymän suorituskyvyn parantuminen vaikuttaa suuresti verkkokaupan houkuttelevuuteen ja käyttäjien tyytyväisyyteen. Kyseistä optimointityötä ja -dokumentaatiota tullaan tarvittaessa käyttämään asiakkaan sisäisessä käytössä.

Avainsanat	käyttöliittymä, käyttökokemus, verkkosivuston suorituskyky

Metropolia

Author Title Number of Pages Date	Niki Järvinen Speed and performance optimization of website's user interface 33 pages 8.11.2018
Degree	Bachelor of Engineering
Degree Programme	Media Technology
Specialisation option	Digital Media
Instructor	Kari Aaltonen, Principal Lecturer

The purpose of this thesis was to study the significance and the factors influencing the performance and speed of the website's user interface. The thesis familiarized with various recommendations, methods, and tools for web interface performance.

The aim of the thesis was to get examples of a low performance web page and optimized version of that page for training purposes. An example of an online store featuring general performance-impairing factors was made and the site's performance was poor. From this starting point necessary tasks to improve performance was made to optimize the performance. Site performance was evaluated continuously - before and after optimization.

Avainsanat	User Interface, User Experience, Web Page Performance

Metropolia

Sisällys

Lyhenteet ja käsitteet

UI	User Interface eli käyttöliittymä. Osio, jossa käyttäjän ja käytettävän tuotteen välinen vuorovaikutus tapahtuu – se, minkä kautta käyttäjä käyttää tuotetta.
UX	User Experience eli käyttäjäkokemus. Tuotteen tai palvelun käyttämiseen liittyvä kokemus, joka kattaa tuotteen käyttöliittymän (UI) ja käytettävyyden ohella myös kaikki muut käyttäjän kokemat elämykset, tunteet, uskomukset, mieltymykset.
TCP	Transmission Control Protocol. Tietoliikenneprotokolla, jolla luodaan yhteyksiä tietokoneiden välille, joilla on pääsy Internetiin.
HTTP	Hyper-Text Transfer Protocol eli hypertekstin siirtoprotokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon. Protokollaa käytettäessä asiakasohjelma avaa TCP-yhteyden palvelimelle ja lähettää pyynnön johon palvelin vastaa lähettämällä sopivan vastauksen, tavallisimmin HTML-sivun tai binääridataa kuten kuvia, ohjelmia tai ääntä.
HTML	Hypertext Markup Language eli suoraan suomennettuna hypertekstin merkintäkieli. Standardoitu ja avoin kuvauskieli, jolla voidaan kuvata hyperlinkkejä sisältävää tekstiä eli hypertekstiä. HTML tunnetaan erityisesti kielenä, jolla verkkosivustot on kirjoitettu.
CDN	Content Delivery Network eli sisällönjakeluverkko. Maantieteellisesti hajautetuista välipalvelimista koostuva verkko, jonka tarkoituksena on sisällön tarjoaminen mahdollisimman läheltä loppukäyttäjän päätelaitetta. Näin on pienempi viive sekä parempi toimintavarmuus.
CSS	Cascading Style Sheets eli verkkosivuston tyylimäärittelyt. Verkkoselain pirtää sivuston tyylin CSS-määrittelyjen perusteella.

JS	JavaScript eli dynaaminen komentosarjakieli. Yleisin käyttötarkoitus on lisätä verkkosivustolle erilaisia dynaamisia toiminnallisuuksia. Käytetään tavallisimmin osana verkkoselaimia, joiden toteutukset sallivat asiakaspuolen skriptien interaktion käyttäjän kanssa, selaimen rajoitetun hallinnan, asynkronisen kommunikaation ja käyttäjälle näytettävän dokumenttisisällön muokkaamisen.
AJAX	Akronyymi sanoista Asynchronous JavaScript And XML. Joukko web-sovelluskehityksen tekniikoita, joiden avulla web-sovelluksista voi tehdä vuorovaikutteisempia ja nopeampia. Selainohjelma vaihtaa pieniä määriä dataa palvelimen kanssa taustalla niin, ettei koko verkkosivua tarvitse ladata uudelleen joka kerta käyttäjän tehdessä muutoksen.
DOM	Document Object Model, eli dokumenttioliomalli. Tapa kuvata rakenteisen dokumentin, kuten HTML:n, XML:n tai XHTML:n, rakenne puuna, jonka eri olioita voi hakea, tutkia ja käsitellä esimerkiksi JavaScriptin avulla. DOMin avulla voi toteuttaa vuorovaikutteisia www-sivuja, jotka eivät vaadi jatkuvaa palvelinyhteyttä. DOM:in tarkoitus on määrittää kuinka dokumentissa olevat elementit välittävät tietoa toisillensa ja kuinka näihin elementteihin voidaan viitata.
DNS	Domain Name System eli Internetin nimipalvelujärjestelmä. Muuntaa verkkotunnuksia IP-osoitteiksi. Internetin laitteet kommunikoivat keskenään numeeristen osoitteiden avulla, joiden muistaminen sellaisenaan olisi hankalaa. Nimipalvelun ansiosta niiden sijasta voidaan käyttää helpommin muistettavia nimiä. Nimipalvelun toinen tärkeä tehtävä on sähköpostin reititys.
Front-end	Selaimen tulkitsema ja käyttäjille visuaalisessa muodossa esitettävä osuus verkkosivustosta.
Backend	Palvelimella suoritettava osuus ohjelmistosta.

1 Johdanto

Insinöörityössä tutkitaan verkkosivuston käyttöliittymän suorituskykyyn vaikuttavia tekijöitä ja tehdään verkkosivuston optimointi, jossa tutkitaan lähtökohtaiset syyt, joiden vuoksi sivuston suorituskyky on riittämätön ja selvitetään ja suoritetaan tarvittavat toimenpiteet sen parantamiseksi.

Verkkoyhteydet ja päätelaitteet ovat kehittyneet vuosien kuluessa niin nopeiksi, että verkkosivuista puhuttaessa nopeutta pidetään nykyisin itsestäänselvyytenä. Käyttäjien kärsivällisyys sivuston latautumisen odottamiseen lasketaan sekunneissa, ja hitautta havaittaessa käyttäjät eivät epäröi etsiä nopeammin toimivaa sivustoa. Verkkosivustojen selaaminen mobiililaitteilla on lisääntynyt huomattavasti, joten sivustojen toimivuutta on tärkeää testata myös mobiiliverkoilla ja -laitteilla. Verkkosivuston nopeus herättää luottamusta ja on yhteydessä esimerkiksi käyttäjien ostohalukkuuteen.

Verkkosivuston latautumisnopeuteen vaikuttavat käyttöliittymän optimoinnin lisäksi myös taustapalvelut, kuten esimerkiksi palvelinlaitteiden ja tietokantojen tehokkuus. Tässä insinöörityössä keskitytään kuitenkin käyttöliittymän optimointiin.

Insinöörityön tavoite on selvittää verkkosivun käyttöliittymän suorituskykyyn ja nopeuteen vaikuttavat tekijät ja löydöksien pohjalta tehdä optimointi suorituskyvyltään huonolle sivustolle.

2 Perustietoa verkkosivuston optimointia tekevälle

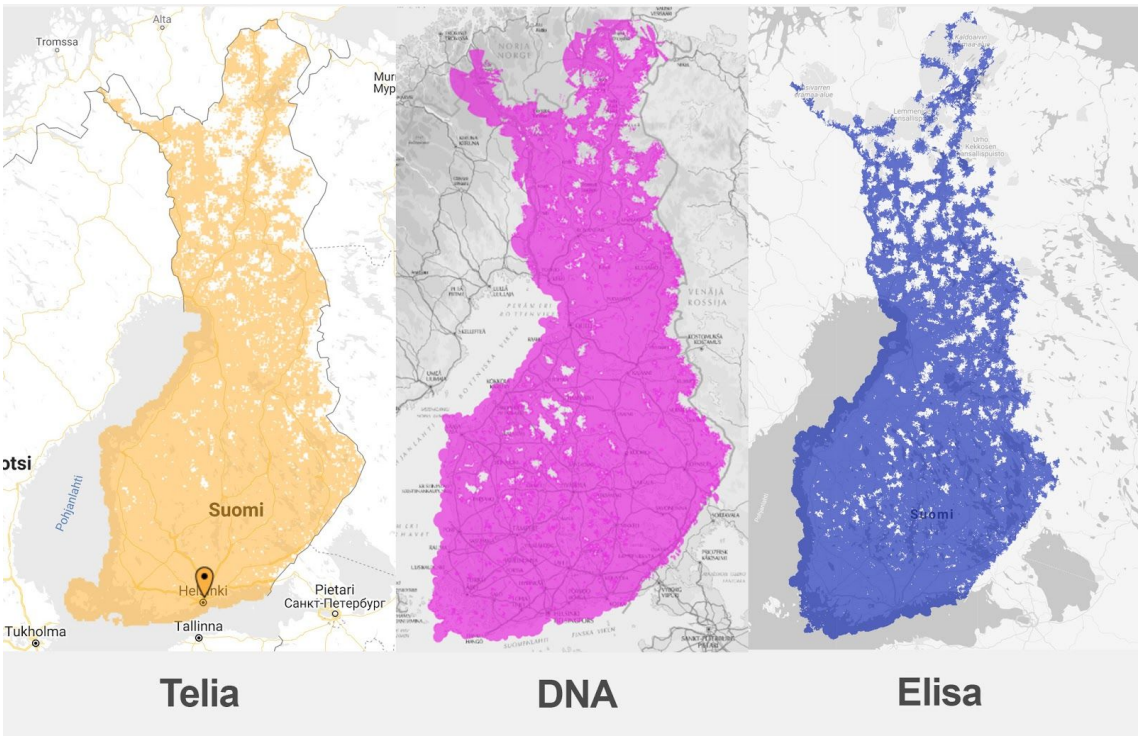
2.1 Internet ja verkkoyhteydet

Internet on maailmanlaajuinen verkkojen verkko eli tietoverkko, joka muodostuu toisiinsa kytketyistä pienemmistä verkoista; tieto kulkee usean reitin kautta, ennen kuin se saapuu käyttäjän päätelaitteeseen tarkasteltavaksi. Internet koostuu palvelun-tarjoajien omista verkoista ja runkoverkosta.

Internet-yhteyttä muodostettaessa Internetiin yhdistettävä laite saa ip-osoitteen ja se voi viestiä muiden Internetiin liitettyjen laitteiden kanssa. Yhteys otetaan yleensä verkko-operaattorin välityksellä, joka ohjaa käyttäjän liikenteen eteenpäin. [1.] Verkko-operaattorit (Suomessa esimerkiksi Elisa Oyj, DNA Oy, TeliaSonera Finland Oyj) omistavat ja hallinnoivat omia verkkojaan ja julkisen hallinnon toimijat (esimerkiksi kunnat) omia verkkojaan. Kotitalouksissa on yleensä omat sisäiset verkot, johon kaikki talouden päätelaitteet on kytketty. Tätä sisäistä verkkoa kutsutaan yksityiseksi Internetin aliverkoksi.

Yhteys verkko-operaattoriin ja Internetiin voidaan ottaa useilla eri tavoilla. Yleensä kiinteää, kaapeleita pitkin otettavaa yhteyttä otettaessa yhteys muodostetaan fyysisestä sijainnista kuparikaapelia pitkin lankapuhelinverkossa tai vaihtoehtoisesti koaksaali-kaapelilla kaapelitelevisioverkossa. Nykyisin myös valokuituverkot ovat yleistyneet laajoilla alueilla, ja uusia rakennuksia kytketään yleisesti valokuituverkkoon, joka mahdollistaa supernopean yhteyden. Nykyinen tiedonsiirron nopeusennätys valokuituverkossa on noin 2 500 terabittiä sekunnissa, joka on siis tavalliseen ADSL-yhteyteen (8 Mbit/s) verrattuna 25 miljoonaa kertaa nopeampi. [2.]

Kiinteän verkon lisäksi Internetiin voidaan yhdistää langattomasti matkapuhelinverkon kautta, jolloin päätelaite ei ole sidottuna yhteen maantieteelliseen sijaintiin, vaan laite kytkeytyy verkkojen verkkoon matkapuhelinverkon välityksellä ja saa ip-osoitteen dynaamisesti verkosta. Mobiililaajakaistan toimivuus riippuu käyttöalueen verkon laadusta, kapasiteetista ja käyttäjien määrästä. Mobiililaajakaistan toiminta perustuu 2G-, 3G-, 4G ja 5G-teknologiaan käyttöalueesta ja saatavasta verkosta riippuen. Parhaat nopeudet saavutetaan 4G- ja uudessa 5G-verkossa. Vaikkakin Suomen tilanne mobiiliverkkojen suhteen on kansainvälisesti tarkasteltuna hyvä, tällä hetkellä nopein yleisessä käytössä oleva teknologia on 4G, ja sekin on saatavilla rajoitetusti suurempien asuinkeskusten ympäristöissä, kuten kuvasta 1 nähdään.



Kuva 1. Suomen verkko-operaattorien 4G-verkon kuuluvuuskartat. Kartoista voidaan havaita, että Suomessa on paljon alueita, johon 4G-verkko ei yllä, ja näillä alueilla verkon nopeus saattaa olla huomattavan hidas. [3; 4; 5.]

3G-kattavuus on lähes koko maan laajuinen. 3G-verkon nopeus on kuitenkin rajallinen, sillä vaikka Suomen verkko-operaattorien teoreettinen maksiminopeus on tällä hetkellä noin 42 Mbps, niin yleensä normaalikäytössä saavutetaan vain noin 2 Mbps:n maksiminopeus. Tämä vaikuttaa myös verkkosivujen toteutukseen; liian raskaat sivut kuormittavat sekä käyttäjän päätelaitetta että verkkoa suuresti, joten verkkosivut tulisi tehdä sekä vallitsevilla nopeusstandardeilla että hitaammilla verkoilla toimivaksi.

2.2 Verkkoselain

Verkkoselain on päätelaitteeseen asennettu ohjelma, joka antaa käyttäjän katsella ja lähettää tekstiä, kuvia ja muita WWW-sivuilta löytyviä tietoja.

Verkkoselaimet kommunikoivat palvelimien kanssa pääasiassa käyttämällä HTTP-protokollaa sivujen hakemiseen. Protokolla perustuu siihen, että asiakasohjelma avaa TCP-yhteyden palvelimelle ja lähettää pyynnön. Palvelin vastaa lähettämällä sopivan vastauksen, tavallisimmin HTML-sivun tai binääridataa, kuten kuvia, ohjelmia tai ääntä. Verkkoselain kääntää HTTP-pyynnön vastauksen käyttäjälle luettavaan muotoon. Verkkoselain pystyy näyttämään myös muita protokolla- ja datapyyntöjen

vastauksia, kuten esimerkiksi HTTPS- ja FTP-pyynnöt sekä tiedostojen avaukset (file:). Verkkoselaimien yleisin käyttötarkoitus on tarkastella World Wide Webissä olevia sivustoja, mutta niitä voi käyttää myös yksityisverkoissa sijaitsevien sivustojen ja tiedostojen tarkastelemiseen. Nykyisin useimmat verkkoselaimet tukevat myös erilasia lisäosia, jotka tukevat aktiivisisältöä, kuten upotettua videota ja ääntä sekä pelejä.

Vaihtoehtoja verkkoselaimen valintaan on tarjolla todella runsas määrä, mutta kaikki selaimet jakavat kuitenkin samoja yleisiä teknologioita. Jokaisessa selaimessa on jokin yleinen selainmoottori eli ohjelmistokomponentti, jonka tehtäviin kuuluu yleensä datan noutaminen verkosta ja HTML/CSS-kielisen koodin tulkitseminen näkyväksi web-sivuksi. Yleisesti käytettyjä vapaita selainmoottoreita ovat Gecko (Mozilla Firefox), KHTML ja siihen pohjautuva WebKit (Apple Safari) sekä WebKitiin pohjautuva Blink (Google Chrome). Muihin, ei-vapaisiin selainmoottoreihin lukeutuvat Microsoftin Trident (Microsoft Internet Explorer), Microsoft EdgeHTML (Microsoft Edge) ja Opera Softwaren Presto (Opera). Verkkosivuja tehdessä on tärkeää testata sivuston toiminta kaikilla yleisimmillä selainmoottoreilla, sillä niiden toimintaperiaatteet ovat erilaiset ja tämän vuoksi testattava sivusto saattaa näyttää erilaiselta eri selainmoottoreihin perustuvilla selaimilla. [6; 7.]

2.3 Verkkosivusto

Verkkosivulla tarkoitetaan maailmanlaajuisessa verkossa eli Internetissä julkaistua hypertekstidokumenttia. Verkkosivut sijaitsevat Internetiin yhteydessä olevilla palvelinkoneilla, joihin käyttäjät voivat ottaa yhteyttä omilta päätelaitteiltaan ja lukea niitä selainohjelmalla. Selain lähettää verkkosivuston palvelimelle pyynnön HTTP-protokollaa käyttäen. Palvelin vastaa pyyntöön lähettämällä HTML- tai XML-kuvaus-kielellä koodatun dokumentin, jonka selain muuntaa käyttäjälle näytettävään muotoon.

Yleensä verkkosivustoon on liitetty sen ulkoasuun ja toiminnallisuuteen vaikuttavia ulkoisia resursseja, kuten CSS-tyylitiedostoja ja Javascript-tiedostoja.

Verkkosivustot voidaan jakaa staattisiin ja dynaamisiin sivuihin. Staattiset sivut sijaitsevat verkkosivuston palvelimen tiedostojärjestelmässä, ja palvelin lähettää sivun käyttäjälle muuttumattomana. Dynaamisten sivujen erona on se, että ne luodaan

käyttäjälle backend-palvelimella. Palvelin luo sille lähetetyn HTTP-pyynnön perusteella käyttäjälle näytettävän sivun ja lähettää sen käyttäjän verkkoselaimeen. [8; 9.]

Dynaamiset sivut ovat suurimmassa osassa tapauksista parempi vaihtoehto, sillä sivuston elementtien tyylien ja sisällön päivitys on näin helpompaa, koska samat yhteiset elementit toistuu sivuston alasivuilla. Staattinen sivu saattaa kuitenkin olla parempi vaihtoehto esimerkiksi väliaikaisille kampanjasivustoille, sillä se on yleensä nopeampi toteuttaa.

2.4 Käyttöliittymä

Käyttöliittymällä tarkoitetaan käytettävän tuotteen osiota, jossa käyttäjän ja käytettävän tuotteen välinen vuorovaikutus tapahtuu. Se on rajapinta, jonka välityksellä käyttäjä käyttää tuotetta [10].

Käyttöliittymään kuuluvat kaikki toiminnot, joiden avulla käyttäjä pyrkii saavuttamaan tietyn päämäärän; esimerkiksi ajoneuvon ilmastoinnin säätöpaneelin painikkeet, joiden avulla käyttäjä pystyy hallinnoimaan ilmastoinnin toimintaa niin, että ajoneuvon sisäilman lämpötila muuttuu.

Kuva 2. Henkilöauton ilmastoinnin käyttöliittymä.

Käyttöliittymä voi olla myös graafinen, esimerkiksi tietokoneen käyttöjärjestelmän laskin. Graafisen käyttöliittymän avulla käyttäjä pystyy tekemään laskutoimituksen, hiiriosoitinta ja/tai näppäimistöä hyödyntäen. Laskimen graafista käyttöliittymää käyttäessään käyttäjä on vuorovaikutuksessa kahden käyttöliittymän (käyttöjärjestelmä ja laskin) ja fyysisten laitteiden (näppäimistö ja hiiri) kanssa.

Käyttöliittymien toimintaan ja laatuun on alettu jatkuvasti enemmän kiinnittää huomiota, ja myös tuotteiden loppukäyttäjät ovat oppineet vaatimaan käyttöliittymän sulavaa toimivuutta. Käyttöliittymän toimivuudella on myös merkittävä vaikutus loppukäyttäjän saamaan kokonaiskuvaan käytettävästä tuotteesta ja tuotetta tarjoavasta yrityksestä.

3 Verkkosivuston käyttöliittymän suorituskyky ja nopeus

Käyttäjien tavat selata verkkosivuja ovat muuttuneet teknologiakehityksen myötä, ja entisen pöytätietokoneella tapahtuvan selailun sijasta verkkosivuja katsellaan yhä useammin erilaisilla mobiililaitteilla. Nämä laitteet useimmiten käyttävät mobiiliverkko-yhteyksiä, jotka ovat ainakin toistaiseksi kiinteitä verkkoja hitaampia. Verkkosivustoja suunnitellessa on tärkeää huomioida, että sivusto latautuu riittävän nopeasti myös mobiiliverkkonopeuksilla.

Verkkosivuston toiminnan sulavuus ja nopeus herättää luottamusta ja vaikuttaa suuresti käyttäjien halukkuuteen vierailla sivustolla.

Vanhan sanonnan mukaan aika on rahaa, ja se pätee myös verkkosivustoilla. Nopeus on erityisen tärkeää sivustoilla, joilla tapahtuu myyntiä. Kaikki aika, jonka asiakas käyttää sivuston lataamisen odottamiseen vähentää myyntiä, ja hitaus saattaa lopulta johtaa myös käyttäjän poistumiseen sivulta. Suorituskyvyn optimointi vaikuttaa myös sivuston hakukonenäkyvyyteen, sillä esimerkiksi Google asettaa suorituskyvyltään paremmat sivustot ensimmäisiksi hakutuloksissa.

Optimoimalla verkkosivuston käyttöliittymän voi myös suuresti vaikuttaa sen ylläpitokustannuksiin. Esimerkiksi jos sivustoa ei ole optimoitu käyttämään selaimen välimuistia, käyttäjän jokainen palvelinpyyntö lähetetään aina uudelleen palvelimelle asti. Välimuistia hyödyntäen vastaus pidetään tallessa käyttäjän selaimessa ja palvelinta ei rasiteta kuin kerran välimuistin määritetyn keston aikana. Näin palvelimen kuormitus kevenee ja palvelinkustannukset vähenevät.

Laskemalla sivuston resurssit mahdollisimman pieniksi voidaan vaikuttaa myös siihen, kuinka paljon sivuston käyttäminen maksaa loppukäyttäjälle. Varsinkin prepaid-liittymiä

ja rajoitetulla datamäärällä varustettuja liittymiä käytettäessä sivuston käyttökustannukset nousevat esille.

Sivuston käyttökustannusten arviointiin on erilaisia menetelmiä, ja yksi helppokäyttöinen verkkosivupohjainen työkalu on What Is My Site Cost, https://whatdoesmysitecost.com.

3.1 Suorituskyvyn mittausmenetelmät

Verkkosivuston suorituskykyä voidaan mitata monilla erilaisilla menetelmillä, tunnetuimpina Google PageSpeed Insights, WebPageTest, Pingdom, GTMetrix ja YSlow. Näistä WebPageTest, GTMetrix ja Pingdom tarjoaa myös backend-palvelinosuuden suorituskyvyn mittaustyökalut, mutta samalla ne ovat maksullisia, toisin kuin PageSpeed ja YSlow, joka on avoimen lähdekoodin työkalu, eli käyttäjä pystyy muokkaamaan sen toimintaa mieleisekseen.

Helppo tapa suorituskyvyn arviointiin on käyttää näiden työkalujen selainversioita tai -lisäosia. Myös esimerkiksi Google Chromeen on liitetty Google PageSpeed -ominaisuudet, joita selaimen käyttäjät voivat käyttää suoraan selaimesta. PageSpeed tarkistelee suorituskykyä sekä mobiilipäätelaitteiden että pöytäkoneiden kannalta, ja havaintojen perusteella se antaa korjauskehotuksia ja apua korjaukseen. Se esimerkiksi luo optimoidut versiot tarkistettavalla sivustolla olevista suuriksi havaituista kuvista. [11.]

Google PageSpeed Insights

Google Pagespeed Insights on työkalu, joka antaa neuvoja ja suosituksia sivuston suorituskyvyn optimointiin. Sen suositukset perustuvat vallitseviin parhaisiin käytäntöihin. PageSpeedillä voi tarkistaa, ovatko sivuston optimoinnin kannalta oleellisimmat asiat kunnossa. Työkalu tarkistaa tehokkaasti esimerkiksi palvelimen automaattisen gzip-pakkauksen tilan ja lisäksi se muistuttaa kuvista ja asioista, joihin käyttäjä voi vaikuttaa.

Sen heikkous on kuitenkin se, että se analysoi ainoastaan sivuston nopeuteen vaikuttavia tekijöitä eikä suoranaisesti nopeutta. Se ei siis tiedä tarkkaan, millainen palvelimen tai verkon nopeus on. Jos palvelin latautuu poikkeuksellisen hitaasti, PageSpeed varoittaa tästä, mutta tulos on aina suuntaa antava. PageSpeed myös vain emuloi laitetta eikä käytä oikeaa laitetta sivuston testaamiseen. Tämän vuoksi optimoitava sivusto tulisi testata manuaalisesti yleisimmillä ja tärkeimmillä laitteilla toimivaksi.

Google on julkaissut myös palvelimelle asennettavan PageSpeed-moduulin. Tämä moduuli pakkaa esimerkiksi kuvat ja HTML-koodin loppukäyttäjälle automaattisesti. PageSpeedin voi myös asettaa esimerkiksi WordPress-alustalle tarkistamaan sivuston automaattisesti valitun aikajakson välein (esimerkiksi päivittäin tai viikoittain). [11.]

YSlow

YSlow analysoi verkkosivuja ja syitä siihen, miksi ne ovat liian hitaita, Yahoo!-sivuston korkeiden suorituskykystandardien mukaan. Kuten Google PageSpeed Insights, YSlow näyttää sivuston latausajat ja ongelmakohdat sekä tarjoaa ehdotuksia suorituskyvyn parantamiseen.

Yahoo!-sivuston suorituskykyä tarkkaileva tiimi on löytänyt yhteensä 34 erilaista sääntöä, jotka vaikuttavat sivuston suorituskykyyn. YSlow on asetettu testaamaan näistä 23:a sääntöä, eli kaikkia, jotka voidaan uskottavasti testata; suluissa säännön painoarvo, tärkein määrittely ensin:

1. Vältä tyhjiä src- ja href-määrityksiä (painoarvo 30).
2. Lisää Expires- tai Cache-Control-tunniste (painoarvo 10).
3. Minimoi HTTP-pyyntöjen määrä (painoarvo 8).
4. Käytä GZip-pakkausta (painoarvo 8).
5. Käytä hajautettua sisällönjakeluverkkoa (painoarvo 6).
6. Aseta sivuston tyylit lähdekoodin yläosioon (painoarvo 4).
7. Aseta toiminnallisuusfunktiot lähdekoodin alaosioon (painoarvo 4).
8. Lataa JavaScript- ja CSS-määrittelyt ulkoisista tiedostoista (painoarvo 4).
9. Minifioi JavaScript- ja CSS-määrittelyt (painoarvo 4).
10. Vältä uudelleenohjauksia (painoarvo 4).

11. Poista tuplamäärittelyt (painoarvo 4).
12. Varmista, että AJAX-pyynnöt hyödyntävät välimuistia (painoarvo 4).
13. Vältä 404:aa eli ei löydy-vastauksia (painoarvo 4).
14. Vältä hahmonnuksen estäviä suodattimia (painoarvo 4).
15. Vältä suoria CSS-määrittelyjä lähdekoodissa (painoarvo 3).
16. Vähennä DNS-hakuja (painoarvo 3).
17. Käytä GET-pyyntöä AJAX-kutsuissa (painoarvo 3).
18. Minimoi DOM-elementtien määrä (painoarvo 3).
19. Vähennä evästeiden määrää (painoarvo 3).
20. Käytä evästeettömiä osoitteita komponenttien hakuun (painoarvo 3).
21. Älä skaalaa kuvia HTML-määritteessä (painoarvo 3).
22. Konfiguroi ETagit (painoarvo 2).
23. Tee favicon.ico-tiedostosta pieni ja välimuistista ladattava (painoarvo 2). [12.]

YSlow tutkii dokumenttiolomallin läpi löytääkseen kaikki sivuston komponentit (kuvat, skriptit, tyylimäärittelyt jne.). Sen jälkeen YSlow käy läpi Firebug Net Panel-komponentit ja lisää ne jo löydettyihin komponentteihin. Firebug Net Panelia hyödyntäen YSlow saa kattavat tiedot jokaisesta komponentista: koon, gzip-pakkauksen tilan, tunnisteet jne. Jos komponentin informaatio ei ole saatavilla Net Panelista (jos esimerkiksi komponentti on ladattu välimuistista), YSlow tekee XMLHttp-pyynnön saadakseen komponentin ja jäljittääkseen sen tietoja. YSlow tekee näin kaikille komponenteille ja tutkii jokaista komponenttia yllä mainittujen sääntöjen kannalta ja muodostaa näin sekä komponenttikohtaisen arvosanan että kokonaisarvosanan. [12.]

WebPageTest

WebPageTestin käyttötarkoitus ja toimintaperiaate on hyvin samankaltainen kuin edellä mainituilla työkaluilla, sitä käytetään verkkosivujen suorituskyvyn arviointiin. Sen alun perin kehitti AOL sisäiseen käyttöönsä, mutta vuonna 2008 se julkistettiin avoimella lähdekoodilla. WebPageTest-työkalua kehitetään aktiivisesti GitHub-yhteisössä. Työkalusta on myös saatavilla jatkuvasti päivitettävä ladattava versio, jonka voi asentaa omalle päätelaitteelle tai palvelimella suoritettavaksi.

Kuten PageSpeed Insightsista, myös WebPageTestistä on tarjolla verkkoversio, jonka testiympäristön tarjoavat yhdessä useat yritykset maailmanlaajuisesti. Testaaminen verkkosivulla on ilmaista, ja itse voi valita palvelimen lokaation, josta sivustoa testataan. Työkalu testaa sivustoasi oikeilla selaimilla (IE:llä ja Chromella) sekä oikeilla kuluttajakäytössä olevilla yhteysnopeuksilla. Yksinkertaisia testejä voidaan ajaa tai suorittaa edistyksellisen testin sisältäen monivaiheisia vuorovaikutuksia, videonauhoitusta, sisällön estämistä ja muita kehittyneitä ominaisuuksia. WebPageTest tarjoaa testaustuloksista kattavan diagnostiikan, joka sisältää resurssien latausta kuvaavat vesiputouskaavat, sivuston nopeuden optimoinnin tilan tarkistukset sekä korjausehdotukset. [13; 14.] Se voidaan myös asettaa suoritettavaksi toistuvasti niin, että se tarkistaa välimuistin toimintaa eli sitä, tallentaako selain tiedot välimuistista haettavaksi [15, s. 31].

Yellow Lab Tools

Yellow Lab Tools on avoimen lähdekoodin helppokäyttöinen työkalu, joka analysoi sivuston ja havaitsee frontendin suorituskyvyn ja koodin laatuongelmat. Se toimii Node.js-palvelimella ja tekee kehittäjille mahdolliseksi testata omaa tuotettaan vain syöttämällä URL-osoitteen Yellow Lab Toolsin verkkosivustolla. Yellow Lab Tools vierailee testattavalla sivustolla ja kerää siitä erilaisia tietoja ja tilastoja PhantomJS-työkalua hyödyntäen. [16.]

Yellow Lab Tools antaa arvosanat (A–F) eri osioille, kuten DOM-rakenteelle, DOM-manipulaatiolle, HTTP-pyyntöjen määrälle, CSS-monimutkaisuudelle ja niin edelleen. Siinä on myös kattava JavaScript-toteutuksen aikajanan analysointi JavaScript-profiloijalla, jonka avulla voidaan nähdä, miten JS-skriptejä on suoritettu, mikä niiden lopputulos oli ja miten ne vaikuttivat sivun tehokkuuteen. [16.] Työkalua on helppo käyttää, ja se auttaa korjaamaan JS:n ja CSS:n ongelmia, vähentämään latausnopeuksia ja parantamaan koodin yleistä suorituskykyä.

Pingdom

Kuten edellä mainittujen, myös Pingdomin päätehtävä on arvioida verkkosivun latutumisaikaa ja löytää pullonkaulat, jotka heikentävät käyttökokemusta. Pingdomilla on noin 60 palvelinta ympäri maailman, ja niillä voidaan testata toimiiko sivu kaikkialta.

Tämä testi voidaan asettaa suoritettavaksi jopa minuutin välein. Pingdomilla voidaan myös varmistaa tärkeimpien interaktioiden toimivuus. Se lähettää automaattisen viestin, mikäli toiminto on rikki. Se pyrkii myös löytämään vian alkusyyn. Pingdom tarjoaa myös rajapinnan, jonka kautta käyttäjä voi käyttää Pingdomin toimintoja. [17.]

GTMetrix

GTMetrixin päätarkoitus on testata sivuston suorituskykyä ja luoda käyttäjälle raportti suorituskyvyn tilanteesta sekä antaa suosituksia kuinka sitä voidaan parantaa. GTMetrix yhdistää edellä mainuttujen Google PageSpeedin ja YSlow'n ominaisuudet ja säännöt, ajaa molempien testit ja näyttää ne käyttäjälle omiin osioihin eroteltuna.

GTMetrix näyttää sivuston kokonaislatausajan, sivun kokonaiskoon ja HTTP-pyyntöjen yhteismäärän. Se analysoi sivuston kuormituksen 28 palvelimelta 7 eri alueelta ympäri maailmaa. Saatava raportti näyttää myös testattavan sivuston suorituskyvyn suhteessa GTmetrixin kaikkien analysoitujen sivustojen keskiarvoon. GTMetrixin raporttisivulla on sivuston suorituskyvystä tiivis yhteenveto, joka perustuu sivunopeuden avain-indikaattoreihin. Yhteenvetosivulla on myös selkeät suositukset, kuinka suorituskykyä voidaan parantaa. Siinä on myös mobiilitestaus, joka suoritetaan oikealla Android-laitteella mahdollisimman realistisen informaation saamiseksi. [14.]

GTMetrix voidaan asettaa seuraamaan sivun tehokkuutta säännöllisen seurannan avulla ja visualisoi ne vuorovaikutteisilla kaavioilla. Sen hälytykset ilmoittavat, kun sivusto toimii odottamattomalla tavalla. Sivuston testaus voidaan asettaa suoritettavaksi päivittäin, viikoittain tai kuukausittain optimaalisen suorituskyvyn varmistamiseksi. [14; 18.]

4 Verkkosivuston optimointimenetelmät

4.1 Käyttöliittymän lähdekoodin optimointi

4.1.1 Lataustavat ja -järjestys

Verkkosivuston tulee ladata kunkin sivun näkyvä pääsisältö ensimmäisenä. Sivuston koodin rakenteessa tulee siis olla tärkeimmät asiat ensimmäisenä ja ei-niin-kriittiset voidaan ladata kriittisten jälkeen. Tämä saattaa tarkottaa sitä, että CSS-tyylitiedosto tulee jakaa kahteen osaan: heti näkyvien elementtien tyylit ja alkunäkymän ulkopuolisten elementtien tyylit.

Jos esimerkiksi blogisivustolla kirjoitukset on jaettu kahteen sarakkeeseen niin, että vasemmalla puolella on sivupalkki, jossa on sosiaalisen median toiminnot ja linkit aiheeseen liittyviin kirjoituksiin, ja toisella sivun sisältö, on suositeltavaa, että kirjoituksen sisältö ladataan ennen sivupalkkia, sillä käyttäjän oletetaan tutustuvan kirjoitukseen, ennen kuin tarvitsee sivupalkin elementtejä.

Sivulla tarvittavat CSS-tyylit ja JavaScript-määrittelyt voidaan ladata kolmella eri tavalla:

- suoralla, tunnistekohtaisella määrittelyllä, jossa elementin tyylit määritellään suoraan elementin omaan tunnisteeseen style-tunnisteella ja JS-määrittelyt esimerkiksi onclick-tunnisteella
- HTML-tiedostoon sulautetulla määrittelyllä, jossa edellä mainitun menetelmän tapaan määrittelyt tehdään suoraan HTML-tiedostoon, sillä erotuksella, että sulautetussa menetelmässä ne ovat elementtien sijasta oman tunnisteen sisällä: tyylit style-tunnisteessa ja JS-määrittelyt script-tunnisteessa
- määrittelyjen lataaminen ulkoisista tiedostoista.

Kahdessa ensimmäisessä vaihtoehdossa pyyntöjen vähentymisestä huolimatta HTML-dokumentin koko kasvaa. Tämä voi kuitenkin olla hyödyllistä niissä tapauksissa, joissa ladattavat resurssit ovat pienet ja pyyntöjen määrällä on suuri merkitys. Optimoidun nopeuden saavuttamiseksi tulisi suorittaa testejä säännöllisesti ja tarkkailla todellisia muutoksia nopeudessa. Näitä testejä ajettaessa on tärkeää huomata

testattavan sivun tarkoitus ja käyttäjäryhmät: jos suurin osa käyttäjistä vierailee sivulla suurella todennäköisyydellä jatkuvasti, on selaimen välimuistin hyödyntämisellä suuri merkitys, kun taas esimerkiksi väliaikaisilla kampanjasivuilla, joissa käydään todennäköisesti vain kerran, ovat välimuistin hyödyntämisen edut pienet.

Kolmas vaihtoehto ei vain paranna koodin järjestystä, vaan mahdollistaa myös selaimen välimuistin hyödyntämisen. CSS-tyylimäärittelyt ja Javascript-tiedostot tulisi aina ladata erillisestä tiedostosta, jotta selain pystyy tallentamaan nämä määritteet välimuistiin latausajan nopeuttamiseksi. Tämä vaihtoehto on useimmiten paras optimointivaihtoehto, varsinkin silloin, kun tiedostojen koot ovat suuria ja sivustolla on paljon alasivuja. Lataaminen ulkoisista tiedostoista siis nopeuttaa sivun lataamista suurimmassa osassa tapauksista ja täten myös parantaa sivuston hakukone-näkyvyyttä, sillä hakukoneet suosivat nopeasti lataavia sivustoja.

Ulkoinen tyylitiedosto voidaan ladata kahdella tapaa: joko link-tunnisteella (kuva 3) tai @import-tunnisteella tyylimäärittelyjen sisällä (kuva 4).

```
<link rel="stylesheet" href="style.css">
```

Kuva 3. Tyylitiedoston lataaminen link-tunnisteella.

```
@import url('style.css');
```

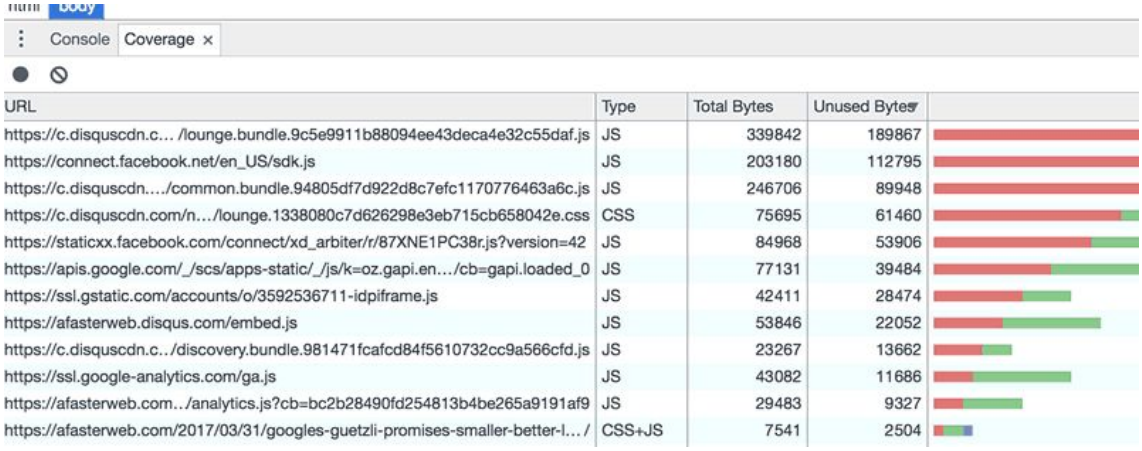
Kuva 4. Tyylitiedoston lataaminen @import-tunnisteella.

Jälkimmäinen tapa on usein merkittävästi hitaampi, sillä selain ei pysty lataamaan resurssia taustalla, vaan saattaa estää muiden resurssien lataamisen, kunnes import-tunnisteen resurssi on ladattu.

4.1.2 Käyttämättömien määrittelyjen poisto

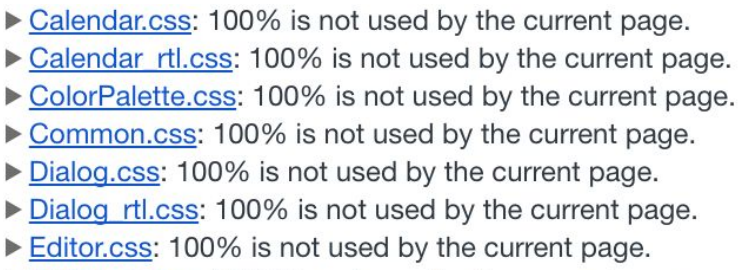
Sivuston CSS-tyylitiedostoihin ja Javascript-tiedostoihin jää usein sellaisia määrittelyjä ja funktioita, jotka eivät ole sivustolla käytössä. Erityisesti valmiit, verkkosivuston kehittämistä nopeuttavat käyttöliittymärakenteet, kuten Bootstrap ja Foundation, sisältävät usein runsaan määrän verkkosivustolla käyttömättömiä CSS-tyylimäärittelyitä ja Javascript-funktioita. Nämä määrittelyt lisäävät turhaan ladattavien tiedostojen kokoa ja näin myös sivuston latausaikaa.

Käyttämättömiä määrittelyjä voidaan etsiä ja poistaa monilla eri työkaluilla. Esimerkiksi Google Chrome -selaimen kehitystyökalujen avulla pystyy tarkastelemaan tiedosto-kohtaisesti, kuinka paljon kukin tiedosto sisältää sivustolla käyttämättömiä määrittelyitä (kuva 5).



Kuva 5. Google Chromen kehitystyökalujen tiedostokohtaiset määrittelyjen käyttöasteet.

Käyttämättömien CSS-tyylimäärittelyjen poistossa voidaan käyttää esimerkiksi PurifyCSS-työkalua, joka poistaa tyylimäärittelyt, jotka eivät ole käytössä verkko-sivustolla. Se ei kuitenkaan muokkaa alkuperäisiä CSS-tiedostoja, vaan se voidaan määrittää luomaan uusi, puhdistettu tiedosto. Se voidaan asettaa myös minifioimaan määrittelyt samalla. [19.] Erityisesti kolmannen osapuolen lisäosia käytettäessä käyttämättömiä määrittelyitä tulee usein todella runsaasti, kuten kuvan 6 esimerkistä voidaan huomata.



Kuva 6. Esimerkki kolmannen osapuolen lisäosan lataamista tyylitiedostoista, joiden määrittelyt ovat kokonaan käyttämättä sivustolla.

4.1.3 JavaScript- ja tyylimäärittelyjen tehokkuus

Raskaita, suurta laskentatehoa käyttäviä funktioita tulee välttää, sillä ne voivat todella merkittävästi hidastaa verkkosivua tai jopa seisauttaa sen kokonaan. Yleisimpiä funktioiden tehokkuutta laskevia ja laskenta-aikaa lisääviä tekijöitä ovat seuraavat:

1. Liikaa vuorovaikutteisia tapahtumia isäntäpalvelimen kanssa

   Jokainen interaktiivinen tapahtuma isäntäpalvelimelle tai käyttäjän selaimeen kasvattaa sivuston ennalta-arvaamatonta käyttäytymistä ja johtaa DOM-objektien hitaaseen hahmonnukseen ja suorituskyvyn laskuun. Näitä interaktioita ei voi välttää, mutta niiden määrä tulisi pitää mahdollisimman vähäisenä.

2. Liikaa riippuvuuksia

   Jos sivuston JavaScript-määrittelyjen riippuvuudet ovat laajat ja huonosti hallitut, sivuston suorituskyky kärsii huomattavasti ja sivuston käyttäjät joutuvat odottamaan pidempää objektien hahmonnusta.

3. Heikko tapahtumien käsittely

   Tapahtumien käsittelijöiden oikea käyttö voi parantaa suorituskykyä huomattavasti. Kiinnittämällä huomiota käsittelijöihin ja niiden toimintaan voidaan keventää kutsujonoja. Esimerkiksi jos sivustolla olevan lähetyspainikkeen toiminto on estetty, koska lähetyssivu sisältää virhesanomia, painikkeen tapahtumien käsittelijä voidaan poistaa kokonaan, jolloin painike ei turhaan reagoi tapahtumiin, ja sivuston tapahtumien käsittelijöiden määrä vähenee.

4. Tehottomat iteraatiot

Iteraatiot vaativat paljon prosessointiaikaa, ja siksi ne ovat hyvä kohde aloittaa koodin optimointi. Usein tarpeettomien funktiosilmukoiden poisto parantaa sivuston suorituskykyä huomattavasti.

5. Organisoimaton koodi

JavaScriptin armollinen luonne tekee siitä helppokäyttöisen ja nopean työkalun frontendin funktioiden luontiin, mutta juuri sen vuoksi se saattaa olla epäloogisesti organisoituna todella tehoton. [20.]

CSS-tyylimäärittelyt on hyvä pitää mahdollisimman yksinkertaisina ja viitata haluttavaan elementtiin lyhintä mahdollista termiä käyttäen.

CSS-tyylimäärittelyjä tehdessä on tärkeää huomata, että selaimet lukevat niitä oikealta vasemmalle, mikä tarkoittaa, että esimerkiksi määrittelyssä ul > li a[title=”palvelut”] selain tulkitsee ensimmäisenä a[title=”palvelut”]-määritteen. Tämä ensimmäisenä tulkittu osa on siis avain, joka on määrittelyn haluttu elementti, mutta ennen elementin hahmontamista selaimen on luettava koko määrittely oikealta vasemmalle ja tulkittava, löytyykö koko määrittelyä vastaavaa elementtiä sivustolta (kuva 7).

```
ul.nav.navbar-nav li a[href$="#contact-us"] {
    font-weight: bold;
}

#contact-us-nav-link {
    font-weight: bold;
}
```

Kuva 7. Ylemmässä määrittelyssä selain joutuu tulkitsemaan kuusi erilaista valintaperiaatetta, kun taas alemmassa määrittelyssä vain yhden. Näin ollen alempi vaihtoehto on huomattavasti kevyempi ja nopeampi vaihtoehto tyylimäärittelylle.

Suorituskykyä ajatellen tehokkain keino on määritellä elementtien tyylit niiden ID:llä. Jos elementtiin ei ole asetettu id:tä, seuraavaksi paras metodi luokan lisäämiseen on tunniste. Tunnisteen käyttämisessä on kuitenkin se ongelma, että se lisää tyylit kaikkiin kyseisiin tunnisteisiin, joten jos halutaan vaan esimerkiksi jokin tietty li-elementti, on käytettävä jotakin tarkempaa määrittelyä, ja tällöin taas ei tulisi käyttää tunnistetta

ollenkaan, koska se hidastaa suorituskykyä (luku oikealta vasemmalle, tunniste turha tässä vaiheessa). Kuvasta 8 nähdään selektoreiden väliset nopeuserot.

Yksi selektori

Selektori	Chrome
1. id	271.9ms
2. tunniste	267.2ms
3. (sama) luokkanimi	355.8ms
4. (eriävä) luokkanimi	263.8ms
5. attribuutit	1069.7ms
6. attribuutti arvolla	1062.9ms

Kuva 8. Esimerkki CSS-selektoreiden nopeuseroista satunnaisella verkkosivustolla testattuna.

4.1.4 Minifiointi

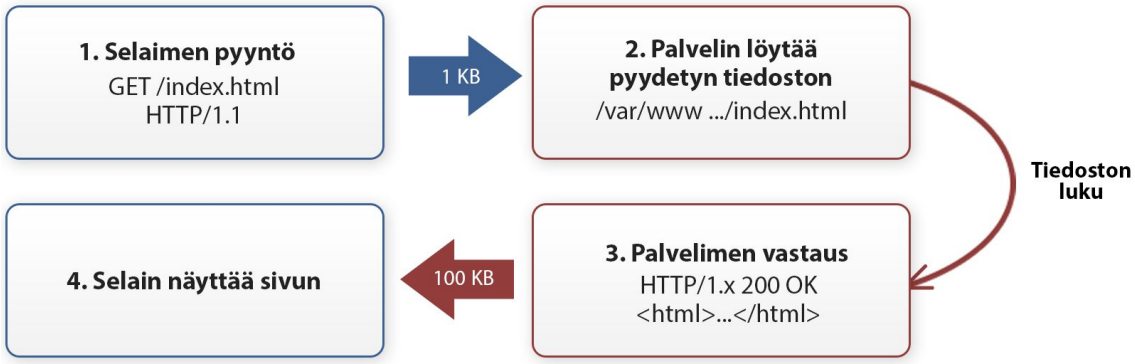
Sivuston lähdekoodit (CSS, HTML ja Javascript) on suositeltavaa minifioida eli poistaa koodista sen toiminnan kannalta turhat merkit, kuten välilyönnit, rivinvaihdot ja kommentit, jolloin tiedostokoko pienenee. Tämä parantaa verkon vastetta, ja pienemmät tiedostot hahmontuvat selaimessa nopeammin. Parhaan suorituskyvyn saavuttamiseksi voidaan myös pitkät funktioiden ja muuttujien nimet korvata geneerisillä kirjaimilla (a, b, c..) merkkien vähentämiseksi. Varsinkin tässä tapauksessa minifioidun koodin lukeminen on erittäin hankalaa, joten alkuperäinen tiedosto tulee pitää tallessa ja suorittaa muokkaukset aina ensin alkuperäiseen ja tehdä minifioitu versio muutoksien jälkeen.

Lähdekoodin minifiointiin on tarjolla lukuisia eri työkaluja. Minifiointityökalua voi käyttää tekstieditorissa, verkkoselaimessa tai komentoriviltä. Yleensä valmiissa frontend-

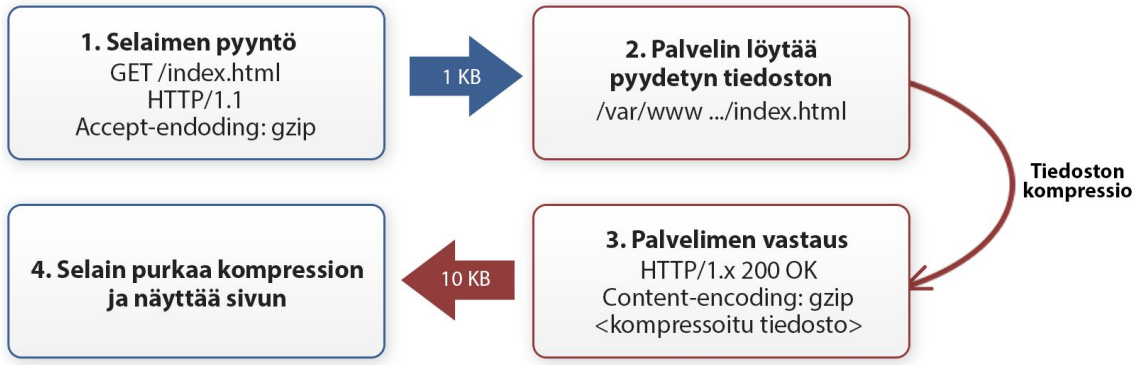
kehitysympäristöissä on jokin minifiointityökalu liitetty osaksi rakennetta ja ajetaan aina automaattisesti tuotantoversiota luotaessa.

4.1.5 Palvelindatan pakkaaminen

Palvelindata on suositeltavaa pakata GZIP-menetelmällä aina kun mahdollista, sillä se vähentää lähetettävän datan määrää (kuva 10).



Kuva 9. HTTP-pyyntö ja vastaus ilman GZIP-kompressiota [20].

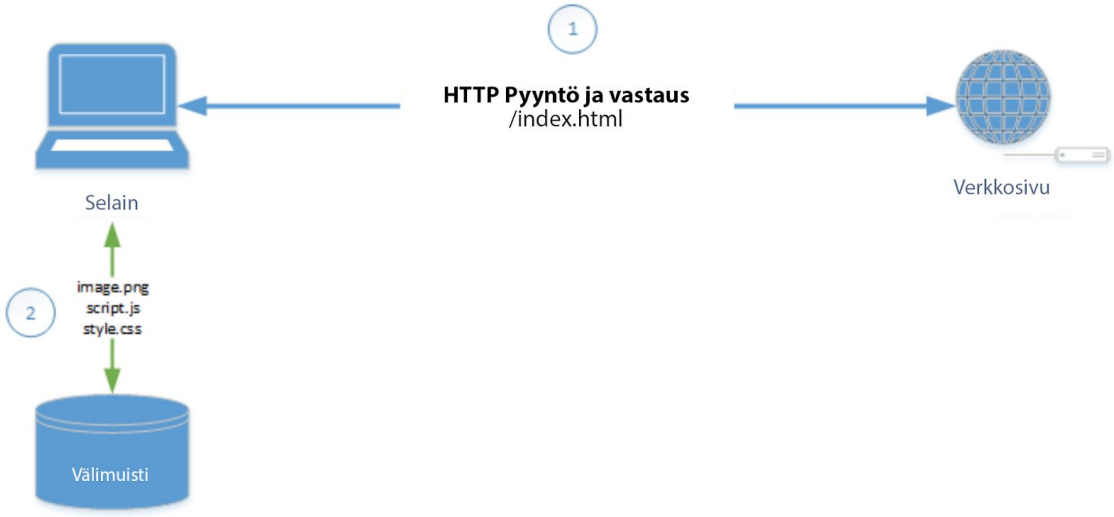


Kuva 10. HTTP-pyyntö ja vastaus GZIP-kompression kanssa [21].

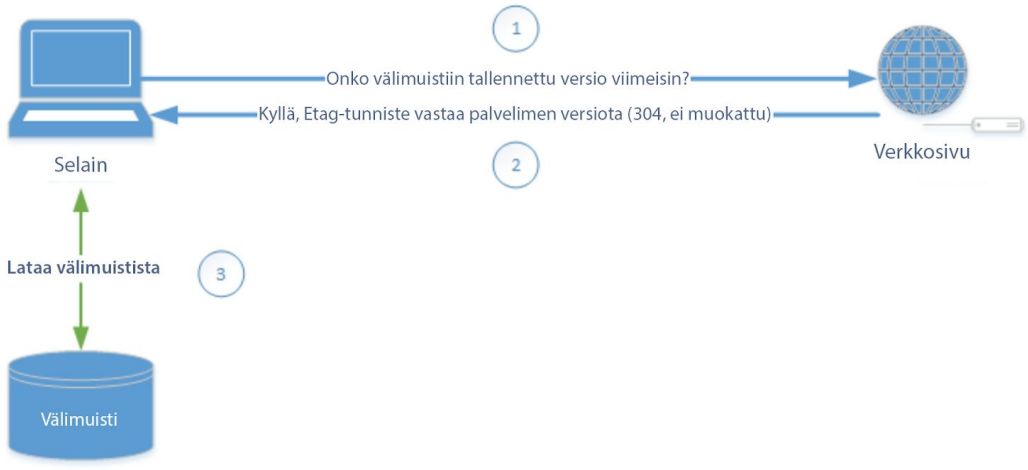
4.1.6 Selaimen välimuistin hyödyntäminen

Selaimen välimuistin hyödyntäminen on yleensä suositeltava toimenpide jokaiselle sivustolle, ja sillä saadaan huomattavat säästöt sivuston latausnopeudessa.

Selain asettaa tiedoston omaan muistiinsa, ja tämän jälkeen se tallentaa referenssin tästä muistissa olevassa tiedostosta ja liittää referenssin jokaiseen HTTP-kyselyyn. Palvelin vastaa käyttäjän selaimelle onko tallennettu versio yhä viimeisin. Jos version on edelleen viimeisin, selain lataa sen omasta välimuististaan eikä pyydä sitä uudelleen palvelimelta. Kuvat 11 ja 12 havainnollistavat perinteisen sekä välimuistiin asetettavan HTTP-pyynnön eroavaisuudet.



Kuva 11. HTTP-pyyntö ja vastaus, välimuistia ei hyödynnetä.



Kuva 12. HTTP-pyyntö ja vastaus, välimuistia hyödynnetään.

Selaimen välimuistin hyödyntäminen tulee kuitenkin asettaa verkkosivuston palvelimella, joten käyttöliittymässä sitä ei voi suoraan määrittää. Esimerkiksi mikäli käytössä on Apache-palvelin, voidaan välimuisti asettaa yksinkertaisella htaccess-tiedoston määrittelyllä kuvan 13 mukaisesti:

```
## EXPIRES CACHING ##
<IfModule mod_expires.c>
ExpiresActive On
ExpiresByType image/jpg "access plus 1 year"
ExpiresByType image/jpeg "access plus 1 year"
ExpiresByType image/gif "access plus 1 year"
ExpiresByType image/png "access plus 1 year"
ExpiresByType text/css "access plus 1 month"
ExpiresByType application/pdf "access plus 1 month"
ExpiresByType text/x-javascript "access plus 1 month"
ExpiresByType application/x-shockwave-flash "access plus 1 month"
ExpiresByType image/x-icon "access plus 1 year"
ExpiresDefault "access plus 2 days"
</IfModule>
## EXPIRES CACHING ##
```

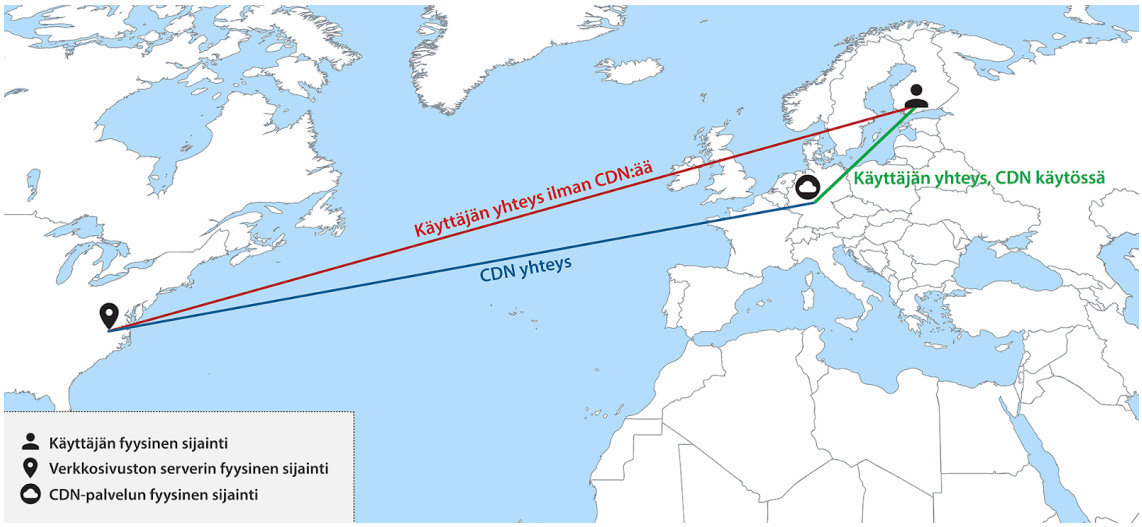
Kuva 13. Välimuistin hyödyntäminen htaccess-tiedostoa käyttäen. Kuvatiedostot on asetettu säilymään vuoden. Tyyli-, JavaScript- ja PDF-tiedostoille on asetettu kuukauden välimuisti. Muille, määrittelyn ulkopuolisille tiedostoille on asetettu kahden päivän välimuisti.

4.1.7 Hajautettu sisällönjakelu

Hajautettu sisällönjakelu eli CDN (Content Delivery Network) on maantieteellisesti hajautettu joukko palvelimia, jotka tarjoavat sivustojen resurssit, kuten sivuston tyyli- ja Javascript-tiedostot sekä kuvat ja videot, kutakin vierailijaa fyysisesti lähimmältä mahdolliselta palvelimelta. Palvelimen läheisellä sijainnilla saadaan lyhennettyä resurssien latausaikaa merkittävästi. Hajautettua sisällönjakelua tulisi käyttää erityisesti silloin, kun tarjottavaa sisältöä on runsaasti ja verkkosivustolla on käyttäjiä monista maista. Jotta sivustolla käytettävä sisältöverkko olisi mahdollisimman tehokas, on varmistuttava, että verkko tarjoaa toimivat palvelimet lähellä sivuston yleisimpiä käyttöpaikkoja.

Hajautetun sisällönjakelun etuna on myös palvelinresurssien käytön väheneminen. Pilvipalveluiden hinta useimmiten määräytyy laskenta-ajan mukaan, joten hajautetulla sisällönjakelulla voidaan säästää palvelinkustannuksia.[22.] Kuvassa 14 huomataan

käyttäjän yhteyden lyhenevän merkittävästi, kun käytetään lähintä mahdollista sisällönjakelukanavaa.



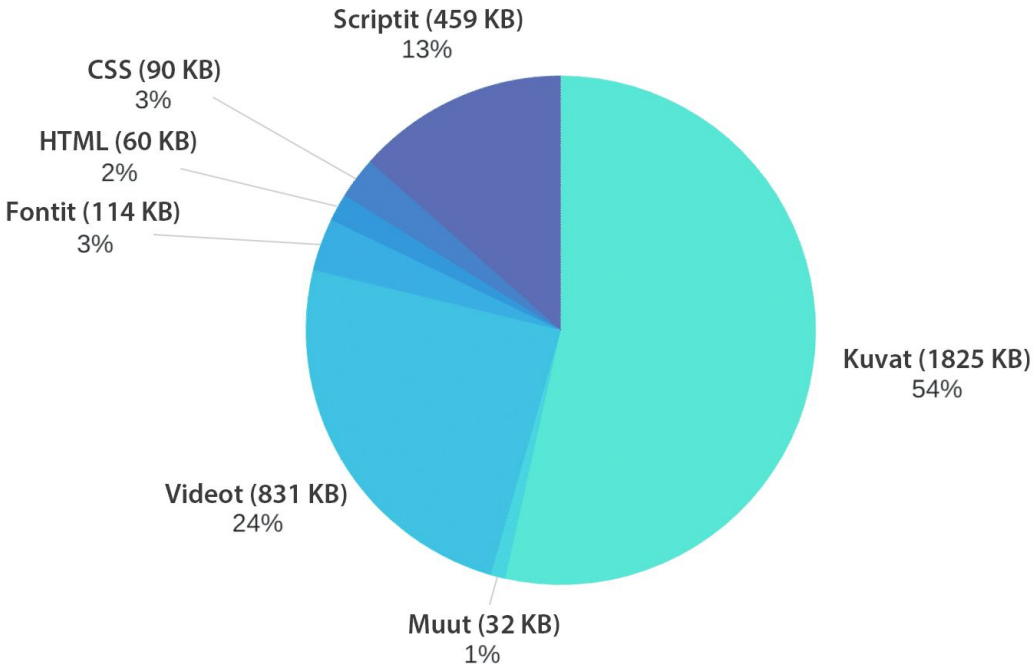
Kuva 14. Käyttäjän yhteys, hajautettu sisällönjakeluverkkoa hyödyntäen ja ilman.

Hajautettu sisällönjakelu mahdollistaa sulavamman käyttökokemuksen lisäksi myös paremman luotettavuuden sivuston ruuhka-aikoina, sillä kuormaa jaetaan eri palvelimille.

Useat suuret verkkoyritykset kuten Facebook, Netflix, Google ja Amazon ylläpitävät omia sisällönjakeluverkkojaan. Näiden yritysten palvelin- ja verkkoinfrastruktuurit ovat jo itsessään maantieteellisesti hajautettuja eri maissa sijaitseviin konesaleihin, ja liiketoiminnan mittakaava on erittäin laaja. Amazon (tarkemmin Amazon Web Services) ja Google myyvät infrastruktuuriaan myös kapasiteettipalvelujen, niin kutsuttujen pilvipalvelujen, eri ohjelmistojen ja palvelujen kehittäjille.

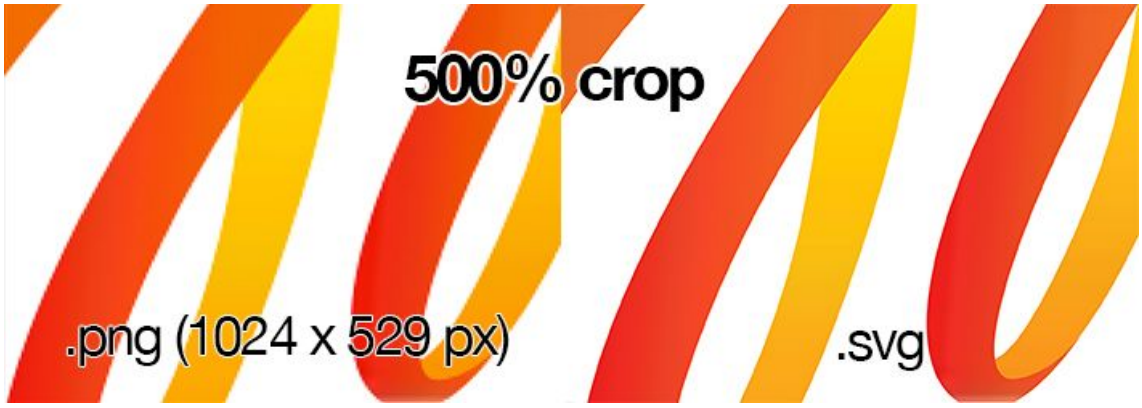
4.2 Kuvien optimointi

Kuten kuvasta 15 nähdään, suurin osa, eli yli puolet, verkkosivun bittimäärästä koostuu kuvista. Tästä syystä on tärkeää optimoida kuvat mahdollisimman pieniksi kooltaan.



Kuva 15. Keskimääräinen bittimäärä verkkosivulla [23].

Sivustolla voidaan käyttää sekä vektorigrafiikkaa että rasterikuvia. Vektorigrafiikka muodostetaan viivoja, pisteitä ja monikulmioita käyttäen, kun taas rasterikuva muodostuu purkamalla ja esittämällä yksittäisten pikseleiden arvoja suorakaiteen muotoisesta elementistä. Vektorigrafiikkaa tulisi käyttää kaikissa mahdollisissa tilanteissa, sillä toisin kuin rasterikuva, se on rajattoman tarkka kaikilla resoluutioilla tarkasteltuna. Rasterikuvia tulee käyttää ainoastaan tilanteissa, joissa kuvaa ei voida esittää vektorimuodossa, esimerkkinä tavallinen valokuva, koska valokuvaa ei pystytä esittämään vektorimuodossa tämän hetken tekniikan avulla. Kuvasta 16 huomataan vektorigrafiikan ja rasterikuvan ero tarkkuudessa.



Kuva 16. Rasterikuva (vasemmalla) ja sama kuva vektorigrafiikkana (oikealla).

4.2.1 Rasterikuvien optimointi

Ennen rasterikuvien optimointia on hyvä pohtia sopivin tiedostomuoto. Suosituimpien tiedostotyyppien, eli PNG:n, JPEG:n ja GIF:n, välillä on eroja, jotka on hyvä tiedostaa kuvia tallentaessa:

PNG – Tuottaa hyvälaatuisia kuvia, mutta tiedostokoko saattaa olla suuri, sillä PNG käyttää vain häviötöntä pakkausmenetelmää.
JPEG – Käyttää sekä häviöllistä että häviötöntä pakkausmenetelmää, ja tämän ansiosta on helppo säätää optimaalinen tasapaino laadun ja tiedostokoon välillä.
GIF – Käyttää vain 256:ta väriä. Paras vaihtoehto animoiduille kuville. Käyttää vain häviötöntä pakkausmenetelmää. [23.]

Latausprosessin nopeuttamiseksi, kuvien tulee olla vain tarvittavan kokoisia, ei siis suuria kuvia pieneksi skaalattuina. Jos esimerkiksi galleriasivulla kuvista on pieniresoluutioiset versiot, jotka saa täysikokoisiksi klikkaamalla, tulee niissä käyttää kahta eri kuvaa: galleriassa pienikokoista esikatselukuvaa ja klikkaamalla auki saatavaa isoresoluutioista ja hyvälaatuista alkuperäiskuvaa.

4.2.2 Vektorigrafiikan optimointi

SVG-tiedostoissa, varsinkin niissä, jotka ovat usealla ohjelmalla tallennettuja – on paljon tiedoston toimivuuden kannalta tarpeettomia tietoja, kuten esimeriksi käytetyn ohjelman metatietoja, kommentteja, piilotettuja elementtejä ja ylikirjoitettuja arvoja. Näiden ylimääräisten tietojen poisto on mahdollista, ja jopa suotavaa esimerkiksi SVGO-nimisellä työkalulla. SVGO sisältää lukuisia erilaisia metodeja SVG-tiedoston koon pienentämiseen, ja näitä metodeja voi käyttää kaikkia yhdessä tai erikseen halutun lopputuloksen saamiseksi. [24.]

5 Esimerkki verkkokaupan optimoinnista

5.1 Lähtökohta

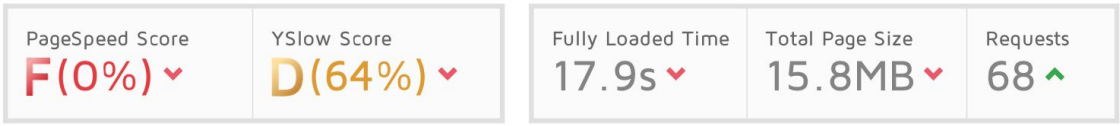
Insinöörityön osana tehtiin oppimistarkoituksiin käytettävä esimerkki huonosti optimoidusta ja suorituskyvyltään heikosta verkkokaupasta ja sille tarvittavat optimoinnit suorituskyvyn parantamiseksi. Sivustolle tehtiin tarkoituksella yleisimpiä virheitä, jotka heikentävät suorituskykyä.

Mahdollisimman luotettavien testaustuloksien takaamiseksi mittaamisessa käytettiin kaikkia aiemmin mainittuja ilmaisia työkaluja, eli WebPageTestiä, Yellow Lab Toolsia ja GTMetrixiä, joka yhdistää PageSpeedin ja YSlow'n. Testit suoritettiin työkalujen maksuttomasti tarjolla olevilla menetelmillä.

Jokainen tehtävä testi tehtiin kolme kertaa, jotta testikertojen satunnaisten vaihtelujen mahdollisuus voitiin eliminoida.

Mittaus aloitettiin GTMetrix-verkkotyökalulla, joka siis arvoi sivustoa sekä Googlen (PageSpeed Insights) että Yahoon (YSlow) listaamien sääntöjen mukaisesti. Testaus suoritettiin Vancouverin palvelinta käyttäen.

Testien tulos oli selkeä – Googlen sääntöjen pisteytys F (0 %) ja YSlow'n vain 64 %. Latausaika oli peräti 17,9 sekuntia, kuten kuvasta 17 nähdään.



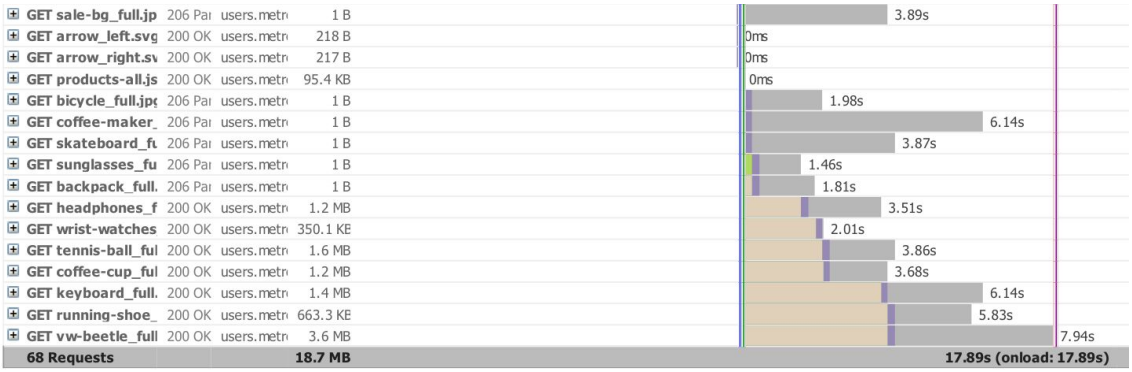
Kuva 17. GTMetrixin tulos. PageSpeed-mittausmenetelmien tulos oli F eli huonoin mahdollinen. YSlow-mittausmenetelmien tulos oi 64 % / 100 %. Latausaika oli jopa 17,9 s ja sivuston koko 15.8 MB. HTTP-pyyntöjä oli yhteensä 68.

PageSpeed-sääntöjen mukaiset korjauskehotukset:

1. Optimoi kuvat.
2. Käytä GZIP-pakkausta.
3. Minifioi JavaScript, CSS ja HTML.
4. Hyödynnä selaimen välimuistia.
5. Poista virheelliset HTTP-pyynnöt.

Näiden lisäksi seuraavat korjaukset olivat YSlow-sääntöjen mukaisesti:

6. Lisää Expires-headerit.
7. Käytä hajautettua sisällönjakeluverkkoa.



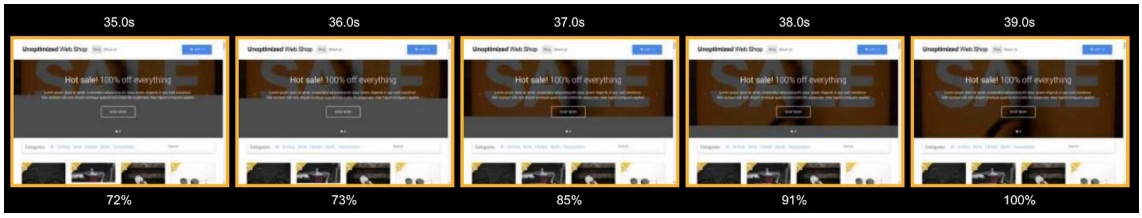
Kuva 18. Erityisesti kuvat lisäsivät latausaikaa huomattavasti. Suurimman kuvan lataus vei peräti 7,94 sekuntia.

Testausta jatkettiin Yellow Lab Toolsin verkkotyökalulla, joka antoi sivulle arvosanaksi F (0/100), kuten kuvasta 19 nähdään.

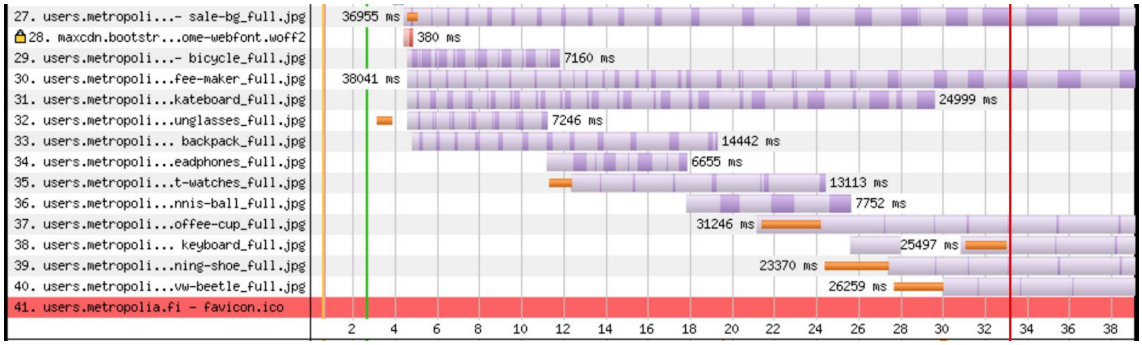
F	Bad CSS	●	CSS syntax error	0
		●	Uses of @import	0
		●	Duplicated selectors	11
		●	Duplicated properties	17
		●	Empty rules	0
		●	CSS expressions	0
		●	Uses of !important	1826 ⚠⚠⚠
		●	Old IE fixes	5
		●	Old prefixes	120
		●	Redundant body selectors	0
		●	Redundant tags selectors	13
F	Web fonts	●	Webfonts number	5
		●	Overweighted webfonts	104 KB
		●	Unused Unicode ranges	25 ⚠⚠

Kuva 19. Yellow Lab Toolsin parannusehdotukset.

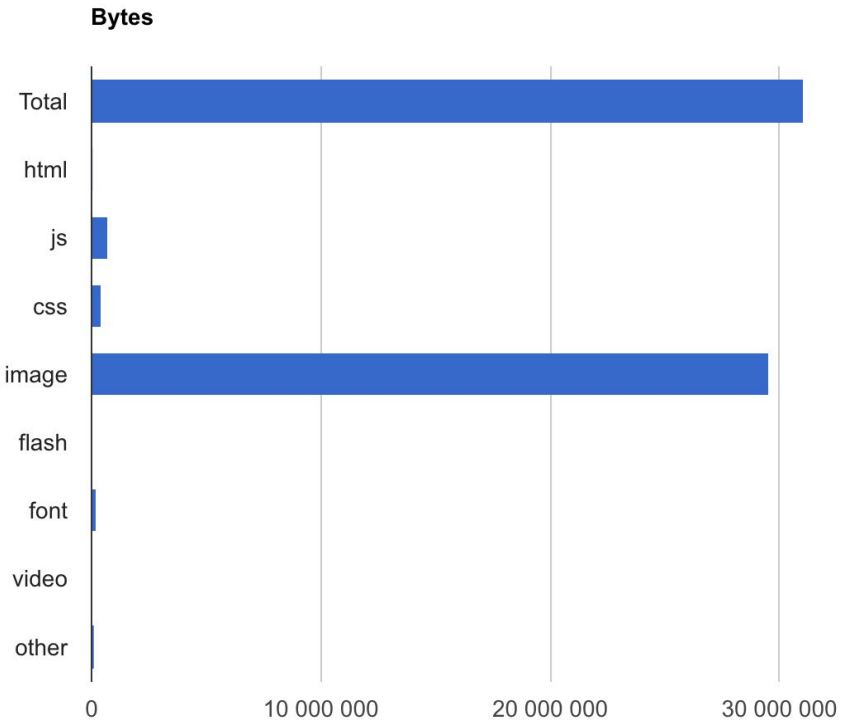
WebPageTest-sivuston testit suoritettiin Tukholman palvelinta käyttäen. Kolmen testikerran keskimääräinen latausaika oli jopa 38 sekuntia. Kuvasta 20 nähdään, että testattavalla sivustolla olevien kuvien lataus oli vielä kesken viimeisten sekuntien kohdalla.



Kuva 20. Kuuvien lataus on vienyt huomattavan paljon aikaa.



Kuva 21. Aikajanakuvasta huomataan, että sivuston latausaika koostui suurimmaksi osaksi yhden kuvan latauksesta.



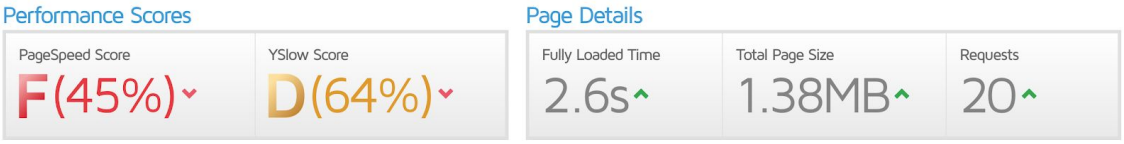
Kuva 22. Sivuston bittimäärä osioittain.

WhatDoesMySiteCost.com-sivuston testin perusteella tämän verkkosivuston lataaminen voi tulla todella kalliiksi, sillä esimerkiksi Kanadasta ladattuna sivusto saattaa maksaa käyttäjälle jopa 2,70 Yhdysvaltain dollaria.

5.2 Optimointi ja tulokset

Sivuston optimointityö aloitettiin eniten latausaikaa lisäävästä tekijästä eli kuvien optimoinnista. GTMetrix tarjoaa liian suureksi havaituista kuvista automaattisesti optimoidut versiot.

Pelkästään kuvien optimoinnilla saavutettiin jo todella merkittävä parannus suorituskykyvyssä; PageSpeed-mittaus parantui 45%, latausaika putosi 2,6 sekuntiin ja sivuston koko 1,38 MB:iin (kuva 23).



Kuva 23. GTMetrixin tulokset kuvien optimoinnin jälkeen. Sivuston kokonaiskoko laski alle 1,4 megabittiin ja latausaika alle kolmeen sekuntiin.

Kuvien optimoinnin jälkeen Yellow Lab Tools antoi tulokseksi 26/100 ja WebPageTest antoi kaikista muista A-arvosanan paitsi tiedostojen pakkauksesta, jota ei siis ollut vielä otettu käyttöön.

Seuraavaksi tehtiin useita muutoksia kerralla, koska ne vaikuttava toisiinsa. Frontend-palvelimelle asetettiin GZIP-pakkaus, Etagit ja Expires-tunnisteet .htaccess-tiedostoon voimaan kaikille HTML, CSS ja JS-tiedostoille. Lisäksi hajautettu sisällönjakeluverkko otettiin käyttöön kaikkiin mahdollisiin paikkoihin, eli kolmannen osapuolen lisäosien tyyli- ja toiminnallisuusmäärittelyihin.

Näiden muutoksien jälkeen Yellow Lab Tools antoi tulokseksi jo 72/100, ja latausaikakin oli laskenut 0.9 sekuntiin alkuperäisestä 17,9 sekunnista. Sivuston käyttökustannukset olivat pienentyneet aiemmasta maksimiarviosta uuteen maksimiarvioon, joka oli 0,09 Yhdysvaltain dollaria, eli sivusto oli lähes 29-kertaisesti halvempi käyttää.

5.3 Jatkuva optimointi

Verkkosivuston käyttöliittymän toimivuutta ja suorituskykyä tulisi arvioida jatkuvasti automaattisilla hyväksymistesteillä. Näiden, yleensä kerran päivässä ajettavien testien tarkoitus on varmistaa, että loppukäyttäjä pystyy tekemään haluamansa toiminnot verkkosivulla. Testit määritetään suorittamaan testattavan verkkosivun toiminnot tavalla, jolla myös käyttäjä ne tekisi.

Optimoinnista tulee pitää huoli jatkuvasti myös uutta sisältöä lisättäessä, jotta sivusto mpysyy optimoituna eikä sisältöön lisätä esimerkiksi isoja optimoimattomia kuvia.

Esimerkiksi WebPageTest voidaan asettaa pyörimään jatkuvasti Jenkins-palvelimelle päivittäin ajettavaksi [15, s.126]. Myös GTMetrix tarjoaa työkalun, jolla sivuston testaus voidaan asettaa suoritettavaksi päivittäin, viikoittain tai kuukausittain. Tällöin voidaan varmistua sivuston optimaalisesta suorituskyvystä ja huomataan suorituskykyä heikentävät resurssit.

Jatkuvan suorituskyvyn optimoinnin lisäksi on hyvä tutkia sen vaikutuksia niin kutsuttuihin verkkosivun onnistumismetriikoihin, joita ovat esimerkiksi uniikkejen käyttäjien määrä, keskimääräinen vierailuaika sivustolla ja vierailun laajuus eli se, kuinka monella sivulla käyttäjä on vieraillut sivustolla. [25, s. 298.]

6 Yhteenveto

Käyttöliittymän suorituskyvyn optimointi on tärkeää sekä palveluntarjoajan että loppu-käyttäjän kannalta. Huonosti optimoidun sivun latausajat voivat koitua kohtuuttoman suuriksi, kuten insinöörityön esimerkissä huomattiin.

Loppukäyttäjän kannalta suorituskyky merkitsee haluttujen toimintojen nopeampaa loppuunsaattamista, kun taas palveluntarjoaja hyötyy paremmasta suorituskyvystä alhaisempina kustannuksina sekä myös parempina tuloina, sillä parempi suorituskyky tuo myös enemmän asiakkaita hakukonenäkyvyyden parantuessa ja käyttäjien välittömän sivustolta poistumisen vähentyessä.

Tehdyssä esimerkissä oli kyseessä verkkokauppa, joille on erityisen tärkeää sivuston hyvä suorituskyky ja sulava toimivuus, sillä kaikki lataamiseen käytetty aika on poissa myynnistä. Optimointityön tuloksena verkkokaupan latausaikaa saatiin pienennettyä 17,9 sekunnista 0,9 sekuntiin, käyttökustannukset saatiin 2,70 Yhdysvaltain dollarin maksimista 0,90 Yhdysvaltain dollarin maksimiin. Tällainen käyttöliittymän suorituskyvyn parantuminen vaikuttaa suuresti verkkokaupan houkuttelevuuteen ja käyttäjien tyytyväisyyteen. Lähtökohtana ollut verkkokauppa oli erittäin raskas käyttää, ja ostoksien tekeminen oli erittäin hidasta. Noin 20 kertaa nopeammaksi hiottu

käyttöliittymä on omiaan tuomaan lisää ostostapahtumia verkkokaupalle lähtötilanteeseen verrattuna.

Tehtyä optimointiesimerkkiä tullaan tarvittaessa käyttämään tilaajan sisäisessä käytössä verkkosivuprojekteissa ja koulutustilaisuuksissa.

Lähteet

1 What is the Internet? Vangie Beal. Verkkoaineisto. Webopedia. <https://www.webopedia.com/TERM/I/Internet.html>. Luettu 7.11.2018.

2 Miksi valokuitu? Verkkoaineisto. Kuitu 16. <https://kuitu16.fi/index.php/miksi-valokuitu/>. Luettu 7.11.2018.

3 Telian Verkkokartta. Verkkoaineisto. Telia. <https://www.telia.fi/asiakastuki/verkko/verkko/verkkokartta>. Luettu 7.11.2018.

4 Peittokartta. Verkkoaineisto. DNA. <http://kartat.dna.fi/Peittokartta/>. Luettu 7.11.2018.

5 Kuuluvuus. Verkkoaineisto. Elisa. <https://elisa.fi/kuuluvuus/> Luettu 7.11.2018.

6 Web Browser. Verkkoaineisto. Techopedia. <https://www.techopedia.com/definition/288/web-browser>. Luettu 7.11.2018.

7 Web Browser. Verkkoaineisto. Tech Terms. <https://techterms.com/definition/web_browser>. Luettu 7.11.2018.

8 Website. 2018. Verkkoaineisto. Computer Hope. <https://www.computerhope.com/jargon/w/website.htm>. Päivitetty 3.8.2018. Luettu 7.11.2018.

9 What exactly is a website? 2018. Verkkoaineisto. Ionos. <https://www.ionos.com/digitalguide/websites/website-creation/what-exactly-is-a-website/>. 3.4.2018. Luettu 7.11.2018

10 User interface definition. Verkkoaineisto. PCMag. <http://www.pcmag.com/encyclopedia/term/53558/user-interface>. Luettu 16.9.2018.

11 About PageSpeed Insights. 2018. Verkkoaineisto. Google. <https://developers.google.com/speed/docs/insights/about>. Päivitetty 9.1.2018. Luettu 4.11.2018.

12 YSlow. Verkkoaineisto. YSlow. <http://yslow.org/>. Luettu 4.11.2018.

13 About WebPageTest.org. Verkkoaineisto. WebPageTest. <https://www.webpagetest.org/about>. Luettu 4.11.2018.

14 Morey, Raelene. 2018. Pingdom vs GTmetrix vs WebPagetest: How Are They Different? Verkkoaineisto. WP Rocket. <https://wp-rocket.me/blog/pingdom-vs-gtmetrix-vs-webpagetest-different/>. 13.2.2018. Luettu 4.11.2018.

15 Viscomi, Rick; Davies, Andy & Duran, Marcel. 2015. Using WebPageTest. O´Reilly Media.

16 What is Yellow Lab Tools? 2014. Verkkoaineisto. GitHub. <https://gmetais.github.io/yellowlabtools/2014/11/20/what_is_yellow_lab_tools.html>. 20.11.2014. Luettu 4.11.2018.

17 Product. Verkkoaineisto. Pingdom. <https://www.pingdom.com/product/>. Luettu 4.11.2018.

18 Features. Verkkoaineisto. GTMetrix. <https://gtmetrix.com/features.html>. Luettu 4.11.2018.

19 Drastically Reduce your CSS File Size with PurifyCSS and webpack. Verkkoaineisto. ChrisCourses. <https://www.chriscourses.com/blog/purifycss-and-webpack>. 10.6.2018. Luettu 4.11.2018.

20 Arsenault, Cody. 2018. 20 Best Practices for Improving JavaScript Performance. Verkkoaineisto. KeyCDN. <https://www.keycdn.com/blog/javascript-performance>. Päivitetty 8.3.2018. Luettu 4.11.2018.

21 How To Optimize Your Site With GZIP Compression. Verkkoaineisto. Better Explained. <https://betterexplained.com/articles/how-to-optimize-your-site-with-gzip-compression/> Luettu 4.11.2018.

22 Why use a Content Delivery Network (CDN)? 2017. Verkkoaineisto. GTMetrix. <https://gtmetrix.com/why-use-a-cdn.html>. 23.2.2017. Luettu 4.11.2018.

23 How to Optimize Images for Web and Performance. Brian Jackson. 19.9.2018. Kinsta. Verkkoaineisto. <https://kinsta.com/blog/optimize-images-for-web/>. Luettu 4.11.2018.

24 Perna, Maria Antonietta. 2017. Three Ways of Decreasing SVG File Size with SVGO. Verkkoaineisto. SitePoint. <https://www.sitepoint.com/three-ways-decreasing-svg-file-size-svgo/>. 4.5.2017. Luettu 4.11.2018.

25 King, Andrew B. 2008. Website Optimization. O´Reilly Media.

1