Modernin verkkosivun kehittäminen ja optimointi, kun toteutustapana on jälkiasennettava CMS

Teijo Aleksander Virta

Haaga-Helia ammattikorkeakoulu
Tradenomi, tietojenkäsittely
Toiminnallinen opinnäytetyö
2023

Tiivistelmä

Tekijä(t)
Teijo Aleksander Virta

Tutkinto
Tradenomi, tietojenkäsittely

Raportin/Opinnäytetyön nimi
Modernin verkkosivun kehittäminen ja optimointi, kun toteutustapana on jälkiasennettava CMS

Sivu- ja liitesivumäärä
46 + 0

Tämän toiminnallisen opinnäytetyön aiheena oli toteuttaa toimeksiantona uudet, sisältöhallittavat verkkosivut hieronta- ja kauneudenhoitopalveluja tarjoavalle yritykselle, Freesi Fiilis. Toteutustapana jälkiasennettava sisällönhallintajärjestelmä muodosti kehitystyön ydinajatuksen, koska tämä salli staattisen verkkosivuston kehittämisen vapaavalintaisilla frontend-teknologioilla ja -kehyksillä, kuten Bootstrap 5. Opinnäytetyössä perehdytään kehitystyön eri vaiheisiin, sisällönhallintajärjestelmien eri toteutustapoihin ja verkkosivuston optimointiin. Samalla vahvistettiin osaamista perinteisen web-kehityksen parissa.

Opinnäytetyön tietoperustassa tutkitaan sisällönhallintajärjestelmiä toteutustapoina, jotka muodostavat vaihtoehtoisia tapoja verkkosivujen kehitykselle. Työssä erotellaan tarkemmin perinteinen, monoliittinen CMS-järjestelmäarkkitehtuuri ja headless CMS. Tämän jälkeen pohditaan, mikä on jälkiasennettava CMS, ja miten se erottautuu edellä mainituista toteutustavoista. Seuraavaksi tietoperustassa perehdytään verkkosivujen optimointiin suorituskyvyn, hakukonenäkyvyyden, turvallisuuden ja responsiivisuuden näkökulmista. Verkkosivuston kehitystyö tapahtui soveltaen ketteriä menetelmiä ja siten viimeisenä aihekokonaisuutena tutustutaan lyhyesti ketteriin menetelmiin, johon liittyvät käsitteet ketterä vaatimusmäärittely, käyttäjätarinat ja kehitysjono web-kehityksen kontekstissa.

Opinnäytetyön empiirisessä osuudessa syvennytään toimeksiantoon ja kehitystyöhön. Tämä käytännöllinen osuus muodostaa raportin, jossa tarkastellaan tiiviisti kehitystyön vaiheita alusta alkaen verkkosivuston julkaisuun asti, jättäen kuitenkin itse julkaisuprosessin käsittelemättä. Luvussa reflektoidaan kehitysprosessin eri vaiheita ja perehdytään tarkemmin yksityiskohtiin, kuten verkkosivuston optimoinnin osa-alueisiin ja jälkiasennettavaan CMS-toteutustapaan. Opinnäytetyö päättyy loppupohdintaan, jossa arvioidaan toimeksiannon ja tavoitteiden saavuttamista, omaa oppimista, tutkimuseettisyyttä ja työn hyödynnettävyyttä. Tuotoksen, eli julkaistun verkkosivuston osoite on www.freesifiilis.fi.

Asiasanat
CMS, bootstrap, jälkiasennettava, headless, optimointi, responsiivisuus, web-kehitys

Sisällys

1 Johdanto 1
2 Tietoperusta 2
2.1 CMS on sisällönhallintajärjestelmä ja toteutustapa 2
2.2 Responsiivinen web-suunnittelu ja Mobile first 5
2.2.1 CSS3 Media Queries eli mediakyselyt 5
2.2.2 Asemointitekniikat - fluid layout: CSS Flexbox ja CSS Grid 7
2.2.3 Flexible media eli joustava media 7
2.2.4 Viitekehyksien käyttämisestä 8
2.3 Optimointia tarkistuslistojen avulla 9
2.3.1 Suorituskyvyn optimointi 9
2.3.2 Turvallisuus 12
2.3.3 SEO eli hakukoneoptimointi 13
2.4 Ketterät menetelmät 15
3 Freesi Fiilis -sivuston kehitystyö 16
3.1 Projektin käytänteet ja ketterä vaatimusmäärittely 16
3.2 Toimeksiantajan lähtötilanne ja alkuperäinen näkökulma heidän ongelmaansa 19
3.3 Kehitysvaiheet 20
3.4 Alkuvaatimusten toteuttaminen 20
3.5 Jälkiasennettava CMS 21
3.6 Kehitystyön kulmakiviä 23
3.7 Optimoinnista 24
3.7.1 Responsiivisuudesta eli mukauttamisesta 24
3.7.2 Sivustorakenne ja SEO 26
3.7.3 Suorituskyvystä 28
3.7.4 Turvallisuudesta 32
4 Loppupohdinta 33
4.1 Tutkimusetiikka 34
4.2 Yhteenveto omasta oppimisesta ja tuotoksen hyödynnettävyydestä 34
Lähteet 39

Sanasto

API, eli ohjelmointirajapinta, mahdollistaa sovellusten välistä kommunikaatiota (Korhonen 2018).

Backend, eli taustahallinta, sisältää kaiken palvelimella suoritettavan osan verkkosovelluksesta.

Bootstrap on avoimen lähdekoodin frontend-viitekehys.

CMS (eng. Content Management System) eli sisällönhallintajärjestelmä tai julkaisujärjestelmä kattaa laajan joukon sisällönhallintajärjestelmiä. Niille ei ole yhtä selkeää määritelmää, koska termillä CMS voidaan viitata eri kokonaisuuksiin keskittyviin tietojärjestelmiin. (Venermo A. s.a.)

Frameworkit eli viitekehykset, ovat ohjelmistoteknologioista muodostuvia moduuleja, joita hyödynnetään web-kehityksessä.

Frontend, eli selainpuoli, on käyttäjälle näkyvä osa verkkosovelluksesta.

Headless CMS, eli päätteetön CMS, termillä tarkoitetaan toteutustapaa, jonka mukaan sivuston käyttöliittymä, eli selainpuoli, on erillään taustahallinnasta (Arola 2022). Sisältö talletetaan taustahallintaan, josta selainpuolen sisältö jaetaan näkyväksi rajapinnan välityksellä (Boyarko 2021).

Jälkiasennettava CMS ei ole vakiintunut termi, mutta sillä tarkoitetaan sisällönhallintajärjestelmää, jolla kovakoodattu, staattinen verkkosivusto muutetaan jälkikäteen dynaamisesti muokattavaksi.

Mobile first eli mobiililähtöinen suunnittelu tarkoittaa sitä, että sivusto rakennetaan ensisijaisesti toimimaan hyvin mobiililaitteilla. Työssä edetään pienistä kohti suuria näyttökokoja.

Ketterät menetelmät, jossa projektin kehitystyö etenee pienemmissä vaiheissa ja suunnittelua tehdään toistuvasti ja jouhevasti työn edetessä. **Kehitysjono** on lista kehitettävän tuotteen tunnistettuja ominaisuuksia, joista muodostuu tehtäviä kehitystiimin toteutettavaksi, jotta tuotos valmistuu. **Käyttäjätarina** on vakiomuotoinen ja selkokielinen lause, joka kuvaa toivotun toiminnallisuuden käyttäjän näkökulmasta ja ohjaa teknistä toteutusta. (Leffingwell 2011.)

Sivuston optimointi, eli sivuston toimivuutta optimoidaan esimerkiksi hakukoneiden, turvallisuuden, suorituskyvyn ja responsiivisuuden näkökulmista.

Responsiivisuus eli sisällön mukauttaminen eri päätelaitteille yhtäläisen käyttäjäkokemuksen saavuttamiseksi.

Kotisivukoneella tarkoitetaan yleisesti ohjelmistoa, joka mahdollistaa kotisivujen luomisen ja ulkoasun muokkauksen ilman syvällistä teknistä osaamista.

1 Johdanto

Alkujaan ennen opinnäytetyöhön ryhtymistä pohdin, kuinka voisin koodata verkkosivut alusta alkaen itse haluamillani teknologioilla, oppia samalla mahdollisimman paljon web-kehityksestä ja jälkikäteen muuntaa staattisen sivuston dynaamisesti muokattavaksi. Tämän toiminnallisen opinnäytetyön yksi merkittävä kulmakivi oli sisällönhallintajärjestelmän (CMS) toteutustapa, jonka ymmärtäminen ja valinta ovat keskeisessä osassa työtä. Opinnäytetyön tärkein tavoite oli toteuttaa toimeksiantona uudet, sisältömuokattavat ja modernit kotisivut yritykselle, Freesi Fiilis. Toimeksiannon tarkoituksena oli rakentaa ja julkaista toimeksiantajalle heidän näköisensä, optimoidut ja laadukkaat kotisivut, joita heidän on helppo hallita ja asiakkaiden hyvä käyttää. Tavoitteena oli myös suunnitella sivuston ulkoasu ja käyttöliittymä toimeksiantajan toiveiden mukaisesti. Sopivan sivustoratkaisun tuli olla yksinkertainen ja tarjota asiakkaille laadukas, käyttäjäystävällinen kokemus.

Tutkin erilaisia CMS-ratkaisuja ja käyn opinnäytetyössä läpi prosessia, jonka myötä päädyin valitsemaan jälkiasennettavan CMS-ratkaisun toteutustavaksi. Yhtenä kriteerinä oli keksiä joustava toteutustapa, jonka myötä kehitystyössä voitaisiin ketterästi käyttää mitä tahansa frontend-teknologiaa tai -kehyksiä ilman sitoutumista mihinkään perinteisistä CMS-ratkaisuista. Erilaiset CMS-ratkaisut tarjoavat vaihtoehtoisia toteutustapoja verkkosivustojen luomiseen ja ratkaisuja löytyy runsaasti. Minun oli ensin ymmärrettävä paremmin sisällönhallintajärjestelmiä, taustahallinta- ja frontend-teknologioita, jotta riittävä vaatimusmäärittely saatiin tehtyä ja työhön voitiin ryhtyä. Jälkiasennettava on termi, josta ei löydy verkosta laajasti tietoa CMS-järjestelmätyyppien kontekstissa, mutta ajatus ei ole uusi. Jälkiasennettava CMS toimi itselleni toteutustapana, joka tuntui oman oppimiseni näkökulmasta luonnolliselta, tehokkaalta ja joustavalta tavalta lähestyä tavoitteitani.

Lopulta tämän toiminnallisen opinnäytetyön aihepiiri nivoutuu nykyaikaisen ja käyttäjäystävällisen verkkosivuston kehittämiseen ja optimointiin modernien verkkokehityksen vaatimusten mukaisesti. Optimoinnin alle sijoitan sellaiset kategoriat kuten suorituskyky, hakukonenäkyvyys, käyttäjäystävällisyys, responsiivisuus ja turvallisuus. Opinnäytetyön rajaus muodostuu aihealueista, jotka koin merkittäviksi, kiinnostaviksi tai erityisen hyödyllisiksi kehitystyön aikana. Pyrin samalla kokoamaan opinnäytetyön alle sopivasti hyödyllistä suomenkielistä tietoutta verkkokehityksestä. Tarkoitukseni ei ole vertailla vaihtoehtoisten toteutustapojen tai teknologioiden paremmuutta keskenään, vaan lähestyä sivuston kehitystyötä omasta näkökulmastani ja avata toteutustapaani.

Haastoin itseni toteuttamaan sivustoratkaisun mahdollisimman alusta alkaen teknisesti itsenäisesti. Oppimistavoitteena oli kehittyä web-kehityksessä ja optimoinnissa. Pyrin vahvistamaan osaamistani web-kehityksen koodaustason rakennuspalikoista, kuten HTML, CSS, JavaScript ja PHP. Hyödynsin toteutuksessa Bootstrap 5 -kirjastoa ja kehitystyö tapahtui soveltaen ketteriä menetelmiä. Tämä on samalla raportti kehitystyöstä, sen värikkäistä vaiheista ja omasta oppimispolustani.

2 Tietoperusta

Käsittelen tietoperustassa aluksi lyhyesti **sisällönhallintajärjestelmiä** toteutustapoina ja osana järjestelmäarkkitehtuuria, koska valitsemani toteutustapa vaati aiheeseen perehtymistä. Tämän jälkeen siirryn käsittelemään **verkkosivuston optimointiin** liittyvää laajempaa kokonaisuutta. Käyn luvussa läpi optimoinnin näkökulmasta **responsiivisuutta** modernissa web-suunnittelussa sekä **suorituskyky**-, **turvallisuus**- ja **hakukoneoptimointia**. Kehitystyöni noudatteli **ketteriä menetelmiä**, josta kerron enemmän luvun lopussa. Lisäksi lukijalta oletetaan perusymmärrystä web-kehityksestä, responsiivisesta web-suunnittelusta ja yleisesti verkkosivuston optimoinnista.

2.1 CMS on sisällönhallintajärjestelmä ja toteutustapa

Tässä opinnäytetyössä termillä perinteinen CMS viitataan verkkosivustojen sisällön hallintaan niin sanotusti monoliittisilla sisällönhallintajärjestelmillä, kuten Wordpress ja Drupal. Monoliittisuudella tarkoitetaan sitä, että verkkosivun sisältöä ja käyttäjälle näkyvää osaa hallinnoidaan yhdestä paikasta (Arola 23.3.2022). Monoliittisessa sisällönhallintajärjestelmässä kaikki komponentit mukaan lukien verkkosivun sisällön renderöinti, tietokerros (tietokanta) ja käyttäjähallinta löytyvät tyypillisesti samasta paketista. Tämä eroaa headless, eli päätteettömästä sisällönhallintajärjestelmästä, jossa renderöinti tapahtuu ulkoisesti ohjelmointirajapinnan kautta. (Wiśniewski 12.10.2021.) Renderöinti tarkoittaa webkehityksessä prosessia, jossa selain muuttaa HTML-, CSS- ja JavaScript-koodin käyttäjälle näkyväksi verkkosivuksi (Kosaka 20.9.2018).

Headless CMS termillä tarkoitetaan sitä, että sivuston käyttöliittymä on erillään taustahallinnasta toisin kuin perinteisissä tavoissa toteuttaa verkkosivustoja, joissa julkaisujärjestelmä ja käyttöliittymä ovat samaa alustaa (Muranen 15.6.2021). Sisällönhallintajärjestelmänä kyse on siitä, että sisältö talletetaan pelkästään taustahallintaan, josta selainpuolen sisältö voidaan jakaa näkyväksi rajapinta-arkkitehtuurin välityksellä mille tahansa laitteelle tai alustalle (Boyarko 2.7.2021).

Päätteetön CMS ei ole mikään tietty tuote tai brändi, eikä edusta mitään tiettyä sisällönhallintaratkaisua (Muranen 15.6.2022). Kontent.ai sivuston artikkelin (Kontent.ai 2023) mukaan päätteettömässä toteutuksessa on kyse joustavasta mikropalveluarkkitehtuurista, jonka mukaisesti digitaalinen tuote rakennetaan yhdistämällä erityisiä palveluita eri toimittajien rajapintojen avulla. Sisällönhallintajärjestelmästä tulee vain yksi palvelu lisää digitaalisen kokemuksen pinossa (Kontent.ai 2023). Päätteettömissä ratkaisuissa on mahdollista sovittaa yhteen minkä tahansa lähdekoodin ja ohjelmointikielen yhdistelmiä. Lisäksi ne ovat erittäin yhteensopivia eri teknologioiden kanssa ja joustavia kehityksen aikana tapahtuvien muutosten suhteen. (Darji 2023.) Toteutustavan heikkoudeksi voi muodostua ratkaisujen teknisyys ja monimutkaisuus, jolloin teknisesti taitamattomat eivät kykene hallinnoimaan sivustoa, vaan päivittämiseen tarvitaan kehittäjiä (Darji 2023; Poirier 2022).

Jälkiasennettava CMS ei ole mikään vakiintunut järjestelmätyyppi tai termi, eikä käsitettä tunneta laajasti verkossa toisin kuin esimerkiksi päätteetön tai monoliittinen CMS. Jälkiasennettavuuteen tai samankaltaiseen periaatteeseen rakentuvia sisällönhallintajärjestelmiä on kuitenkin olemassa. Aiheesta on puhuttu reilun vuosikymmenen aikana esimerkiksi verkkokeskusteluissa, joissa keskustelujen avaajat ovat etsineet ratkaisua staattisen sivuston sisällönhallintaan jälkikäteen. Lähestyn jälkiasennettavaa toteutustapaa ensin verkkokeskusteluissa nousseiden tarpeiden kautta ja nostan niistä kaksi CMS-järjestelmää esille.

Käyttäjä etsii Redditissä kevyttä CMS-järjestelmää, jossa on ymmärrettävä käyttöliittymä, jolla voidaan tehdä muutoksia vain tiettyihin osiin staattista verkkosivustoa. Vastauksissa hänelle suositellaan useita jälkiasennettavia CMS-ratkaisuja, mutta hän päätyy valitsemaan päätteettömän WordPress -toteutuksen, jossa selainpuolella ei olisi todellista yhteyttä taustahallintaan. (Meelkor 2014.) Toinen Reddit-käyttäjä etsii CMS-järjestelmää, joka olisi mahdollista liittää olemassa oleviin verkkosivuihin, mutta ei välttämättä kaikkiin. Hänelle suositellaan jälkiasennettavan vaihtoehdon lisäksi Flat File CMS-järjestelmää. (jimii 2014.) Flat File CMS perustuu tietokannattomaan toteutukseen, jossa sisältö talletetaan tiedostoihin (ahadb 2022). Eräs Reddit-käyttäjä pohtii, mikä CMS olisi samaan aikaan intuitiivinen loppukäyttäjälle ja mahdollisimman kivutonta yhdentää hänen koodipohjansa kanssa. Käyttäjälle suositellaan WordPressin ja sen lisäosien lisäksi headless-toteutusta ja jälkiasennettavia ratkaisuja, kuten CouchCMS. (justbooleandawg 2023.)

Maria (Maria 21.9.2017) kertoo blogissaan etsineensä kevyttä CMS-järjestelmää, jotta hänen ei tarvitsisi aloittaa sivuston kehittämistä uudelleen liittääkseen kokonaisuuteen sisällönhallintajärjestelmän. Edelleen hän (Maria 21.9.2017) jatkaa löytäneensä CouchCMS-nimisen jälkikäteen integroitavan CMS-järjestelmän, jolla voidaan kytkeä staattiseen sivustoon dynaamisia osia. Sitepoint-sivustolla nimimerkki (Hlforr 2022) tiedustelee Perch-nimiselle jälkiasennettavalle CMS-järjestelmälle korvaavaa vaihtoehtoa, jossa olisi myös helppo käyttöliittymä. Edellinen keskustelu etenee siten, että eräs käyttäjä (Martyr2 2022) ehdottaa WordPress, Joomla ja Modx CMS-järjestelmiä jälkeenpäin sulautettaviksi vaihtoehdoiksi. Keskustelun kommenttien perusteella jälkiasentaminen ja jälkikäteen sulauttaminen ovat kuitenkin käytännössä hieman erilaisia toimenpiteitä, vaikka lopputulos ja periaate voivat olla lähellä toisiaan.

Edellä viitattujen keskusteluiden perusteella jälkiasennettavalta CMS-järjestelmältä toivottuja erityispiirteitä olivat keveys, helppo omaksuttavuus, jälkiasennettavuus/liitettävyys olemassa olevaan sivustorakenteeseen ja vähintään sivuston osittainen muokattavuus sekä yksinkertainen ja käyttäjäystävällinen käyttöliittymä.

Jälkiasennettavia ratkaisuja yhdistää usein sapluunarakenne, jonka ymmärtää paremmin tutustumalla tarkemmin jälkiasennettaviin CMS-ratkaisuihin ja niiden dokumentaatioon. Yleisesti

kotisivukone- ja CMS-alustoilla luodut verkkosivustot perustuvat sapluunoihin (eng. templates), jotka muodostavat pohjan sivuille (Brandon 19.9.2019). Sapluunat sisältävät tyypillisesti HTML-rakenteen sekä tarvittavat skriptit ja CSS-tiedostot sivun toivotun ulkoasun ja tuntuman luomiseksi (Hannon Hill 2023). WordPress-sivustot rakentuvat esimääritetyistä teemoista, jotka ovat kokoelma tiedostoja (esim. tyylitiedostoja ja koodia), jotka määrittävät verkkosivuston kokonaisulkoasun ja toiminnallisuuden. WordPressin sapluunat ovat osa teemaa. Jokaista teeman sisältämää sapluunatiedostoa käytetään tietyn osan näyttämiseen verkkosivustolla. (Maulidina 2023; WPBeginner s.a.)

CouchCMS ja Perch edustavat sisällönhallintajärjestelmiä, joita molempia kuvaillaan jälkiasennettaviksi. Samaan tapaan molempien ratkaisujen erityispiirteeksi kuvataan staattisen verkkosivuston muuntaminen dynaamiseksi jälkikäteen. Kumman tahansa edellä mainitun järjestelmän asentamisen jälkeen staattiset HTML-sivut liitetään sisällönhallintajärjestelmään vaihtamalla niiden tiedostopäätteet .php-päätteisiksi ja lisäämällä tiedostojen koodiin PHP-aloitusmerkintä. Muunnetuista sivuista muodostuu näin CMS-hallinnoituja sapluunoita. (Grabaperch.com s.a.(a).; Grabaperch.com s.a.(b).; Webzyme Softwares 2022; Webzyme Softwares 2017a.) Sisällönhallintajärjestelmissä, kuten CouchCMS ja Perch, hallittava sisältö määritetään tarkemmin sapluunoissa. Tämä tehdään käyttäen XHTML-tyyppisiä tageja, jotka määrittävät mitä sisältöä voidaan hallita järjestelmän kautta. (Grabaperch.com s.a.(b); Webzyme Softwares 2017b.) Sapluunat sisältävät siis sivujen HTML-koodin, johon on yhdistetty PHP-koodia sisällön dynaamista hallintaa ja esittämistä varten.

PHP:n avulla verkkosivuihin voidaan upottaa dynaamisia toimintoja. Sivut saavat pääsyn skriptikielen ominaisuuksiin, kun sivuille annetaan .php-tiedostopääte. Aloitusmerkintä (**<?php**) kertoo verkkopalvelimelle, että sen tulee sallia PHP-ohjelman tulkita kaikki seuraava koodi aina päättävään merkkiin (**?>**) asti. Sivujen HTML:ää voidaan muuttaa suoraan PHP-tiedostoissa, joissa HTML-koodi on. Yhdistelmä muodostaa sujuvan ja joustavan tavan hallita verkkosivujen sisältöä ja toiminnallisuutta. (Nixon 2014, luku 1.)

Edellä mainittujen kahden CMS-ratkaisun näkökulmasta jälkiasennettava CMS ilmenee vaihtoehtona, joka tarjoaa asentamisen jälkeen sisällön hallintaan käyttöliittymän, joka upotetaan olemassa olevaan sivustorakenteeseen. Jälkiasennettavassa toteutustavassa selainpuoli ja taustahallinta ovat löyhästi kytköksissä, sillä lähestymistapa erottaa sisällönhallinnan sivuston esitystavasta hieman, mutta ei täysin kuten päätteettömässä toteutuksessa, jossa on kyse pelkästään taustahallintaisesta CMS-järjestelmäarkkitehtuurista.

Perinteisessä, monoliittisessa CMS-järjestelmässä selainpuoli ja taustahallinta ovat melko tiukasti yhteydessä toisiinsa. Monoliittinen CMS yhdistää verkkosivuston selainpuolen ja taustahallinnan

esimääritettynä koodipohjana, johon sisältyy kaikki sisältö tietokannasta aina esityskerrokseen saakka. (Meshen 2021.)

2.2 Responsiivinen web-suunnittelu ja Mobile first

Käsittelen luvussa hyviä peruskäytänteitä responsiiviseen eli mukauttavaan web-suunnitteluun. Mukauttava web-suunnittelu on sisällön esittämistä käytettävien laitteiden kuvaporteissa (eng. viewport) kaikkein asianmukaisimmassa muodossa (Frain 2022, luku 1). Se on myös verkkosivuston optimoinnille merkittävä osa-alue, koska hyvin toteutettuna sillä voidaan parantaa hakukoneoptimointia (Valencia 7.2.2022).

Responsiivisuuden toteuttaminen voidaan aloittaa lisäämällä seuraava metatagi HTML-sivulle:

```
<meta name="viewport" content="width=device-width, initial-scale=1>
```

Käyttämällä edellä kuvattua **viewport**-metatagin arvoa **width=device-width**, ohjeistetaan sivua vastaamaan näytön leveyttä laitekohtaisissa pikseleissä. Tämä mahdollistaa sivun sisällön uudelleenvirtauksen eri näyttökokojen mukaisesti, olipa kyseessä sitten älypuhelin tai pöytätietokoneen näyttö. (LePage & Andrew 2022.)

Mobile first, eli mobiili ensin -lähestymistavan ydinajatus piilee siinä, että suunnittelu aloitetaan mobiililaitteiden kapeista näytöistä ja edetään kohti yhä leveämpiä näyttöjä. Lähestymistapa korostaa mukauttavaa web-suunnittelua, jossa sivusto suunnitellaan ensisijaisesti mobiililaitteille ja sitten mukautetaan suuremmille näytöille (Clancey 2022). Pienimpien mobiililaitteiden kapeille näytöille suunnitellaan vakio asettelu, jonka päälle kerrostetaan asteittain yhä leveämpien kuvaporttien yksityiskohtia (Mozilla Corporation 2023b.). Perinteinen mobiili ensin -lähestymistapa CSS-kehityksessä perustuu tyylimäärityksien ylikirjoittamiseen: työ aloitetaan CSS:n oletustyylimäärityksillä, joita muokataan tai sitten lisätään uusia tyylejä määrittelemällä leveyden raja-arvot mediakyselyillä isompia näyttöjä varten (Clancey 2022).

Mukauttavan web-suunnittelun ydindogmeihin kuuluvat olennaisesti seuraavat kolme tekniikkaa: *fluid layout*, *media queries*, ja *flexible media* (Frain 2022, luku 3). Nykyaikana web-suunnittelussa voidaan hyödyntää yhä enemmän web-kehitykseen kehitettyjä viitekehyksiä ja kirjastoja, jotka voivat helpottaa ja nopeuttaa suunnittelu- ja kehitystyötä. CSS-viitekehyksissä tyypillisesti hyödynnetään aiemmin mainittuja kolmea tekniikkaa. Esittelen seuraavissa aliluvuissa lyhyesti näitä tekniikoita ja otan huomioon Bootstrap 5 -kirjaston ominaisuuksia.

2.2.1 CSS3 Media Queries eli mediakyselyt

CSS3 Media Query on keino, jolla voidaan asettaa CSS-määrityksiä asettamiemme ehtojen täsmätessä (Mozilla Corporation 2023c). Eniten käytetyt mediakyselyt liittyvät yleensä kuvaportin

leveyteen ja satunnaisemmin korkeuteen tai resoluutioon (Frain 2022, luku 3). Mediakyselyt ovat muotoutuneet avainosaksi mukauttavaa web-suunnittelua ja mahdollistavat eri asetteluiden luomisen kuvaportin koon mukaan, mutta niitä voidaan myös käyttää tunnistamaan muita seikkoja laiteympäristöstä, jolla sivustoa käytetään (Mozilla Corporation 2023c).

Jotta mediakyselyt saadaan kohdennettua laajaan skaalaan erikokoisia näyttöjä, apuna käytetään pysäytyspisteitä (eng. breakpoints). Jokainen pysäytyspiste toimii kuin napsahduskohtana uudelle asettelulle. (Walker 9.3.2023.) Nykyaikana pysäytyspisteitä luodaan tarpeen mukaan tukemaan verkkosivuston suunniteltua asettelua, mutta niitä ei kannata ylikäyttää. Lähestymistapaa nykyaikaiseen web-suunnitteluun voidaan ymmärtää siten, että sisällön tulisi mukautua käytettävissä olevan tilan mukaan, jonka vuoksi voimme saavuttaa responsiivisuuden ilman mediakyselyjä käyttämällä moderneja CSS-ominaisuuksia, kuten flexbox ja grid (Nanz 30.10.2021).

Pysäytyspisteinä käytetään usein sellaisia arvoja, joita pidetään erikokoisten laitetyyppien kuvaporttien vakiotyyppisinä leveysväleinä erittäin pienistä hyvin suuriin näyttöihin. Eri määrityksiä voidaan myös yhdistää ja ketjuttaa käyttämällä sanaa **and** (Mozilla Corporation 2023c). Alla yksinkertainen tapa kirjoittaa mediakysely pysäytyspisteillä, joiden arvoina on suuntaa antavasti eri laitteiden leveysvälit:

```
// maksimileveys 480px – pienet näytöt;
@media (max-width: 480px) { /* CSS-määritykset tähän */};
// minimileveys 480px ja maksimileveys 1024px – Keskikokoiset näytöt;
@media (min-width: 480px) and (max-width: 1024px) {/* CSS-määritykset tähän */};
// minimileveys 1024px – Suuret näytöt;
@media (min-width: 1024px) { /* CSS-määritykset tähän  };
```

Kyselyt voidaan myös kohdentaa tarkemmin koskemaan tiettyjä mediatyyppejä (eng. media types) ja mediaominaisuuksia (eng. media features), kuten leveys, korkeus ja resoluutio. Mediatyypit määrittävät, mihin laitekategorioihin (all, print, screen) mediakysely pätee, mutta sen asettaminen on vapaavalintainen ja oletuksena on kaikki laitteet. Mediaominaisuudet kuvaavat käyttäjän, tulostimen tai laiteympäristön erityispiirteitä. (Mozilla Corporation 2023b.)

Mediakyselyillä voidaan testata ja poissulkea ominaisuuksia. Monenlaiset vuorovaikutteiset ominaisuudet, jotka toimivat pöytäkoneen selaimessa eivät nimittäin välttämättä toimi kosketusnäytöllisillä laitteilla. (Mozilla Corporation 2023b.) Tällaisia vuorovaikutukseen liittyviä mediakyselyjen ominaisuuksia ovat esimerkiksi *pointer media feature* ja *hover media feature. Pointer media featurella* voidaan antaa määrityksiä liittyen kosketukseen tai hiiren käyttöön ja *hover media featurella* voidaan testata leijumisefektin toimivuutta käytettävässä laitteessa (Frain 2022, luku 3). Edellä mainituilla ominaisuuksilla voidaan tehdä hyödyllisiä tarkennuksia CSS-määrityksiin käytettävillä laitteilla tai poistaa määrityksiä käytöstä esimerkiksi kosketusnäytöiltä, jos huomataan niiden aiheuttavan

käytettävyysongelmia (Diaz 2022). Esimerkiksi leijumisefekti ei toimi useinkaan halutulla tavalla kosketusnäytöillä, koska ei ole kursoria, joka leijuisi objektin yllä.

2.2.2 Asemointitekniikat - fluid layout: CSS Flexbox ja CSS Grid

Fluid layout on web-suunnittelussa käytetty joustavan asettelun konsepti, joka sallii verkkosivun elementtien mukauttaa leveyttään ja korkeuttaan suhteessa selainikkunan kokoon. Joustavan asettelun tekniikoita käytetään verkkosivun sisällön yhtäläiseen uudelleen asemointiin kuvaportin koon vaihtuessa. (Christensson 2015; Juviler 2.11.2021.) Verkkosivun elementtien pysyvät (eng. fixed) leveysarvot muunnetaan verrannollisiksi näytön tai selainikkunan leveyteen nähden, jolloin kuvaportin kutistuessa elementit skaalautuvat suhteessa toisiinsa (Juviler 2.11.2021). Yksinkertainen kaava on jakaa kohde-elementin leveysarvo kontekstilla, jolloin saamme tulokseksi suhteutetun suurusarvon. Kontekstilla tarkoitetaan kokonaisuutta, jonka sisällä kohde-elementti on. (Frain 2022, luku 4.)

CSS Flexbox ja CSS Grid ovat kaksi tehokasta joustavan asettelun -moduulia: CSS Flexboxilla voidaan tehokkaasti asetella sisältöä yhteen suuntaan kerralla ja CSS Gridillä kahteen suuntaan kerralla (Frain 2022, luku 3). CSS Flexbox on yksiulotteinen asettelumenetelmä, joka järjestää elementtejä riveihin tai sarakkeisiin (Mozilla Corporation 2023a). Se on CSS-ominaisuus, joka mukauttaa elementtien leveyttä ja/tai korkeutta parhaalla mahdollisella tavalla täyttämään käytettävissä olevan tilan näytöillä. Sen avulla elementit voivat joustavasti laajentua tai kutistua sopimaan pienempiin tiloihin. (Rajora 17.6.2021.) CSS Grid on ruudukkoasetteluun perustuva menetelmä, jossa elementtejä voidaan hallita kahdessa ulottuvuudessa (sekä riveissä että sarakkeissa), mikä mahdollistaa sisällön tarkemman sijoittelun ja auttaa epätavallisten, asymmetristen ja päällekkäisten muotoilujen luomisessa (Goyal 17.8.2021). Näiden tekniikoiden käyttö mahdollistaa tapoja luoda mukautuvia komponentteja, mikä vähentää mediakyselyiden tarvetta (Afif 2021).

2.2.3 Flexible media eli joustava media

Mediatiedostot, kuten kuvat, vaikuttavat negatiivisesti sivuston latautumisnopeuteen, jos niitä ei ole optimoitu riittävän hyvin (Bece 1.10.2019). Mediatiedostot voidaan asettaa skaalautumaan joustavasti, kuten muukin sisältö erilaisilla määrityksillä.

Verkkokehityksessä on järkevä hyödyntää uusia kuvaformaatteja, kuten AFIV ja Webp, koska ne mahdollistavat pakkautuvuudellaan hyvän suhteen kuvanlaadun ja tiedostokoon välillä. Samasta kuvasta voidaan luoda eri kuvaporteille optimaalisia kuvia molemmissa edellä mainituissa tiedostomuodoissa ja määrittää **<picture>**-tagin avulla, missä tilanteissa vaihtoehtoisia kuvia käytetään. Esimerkiksi jos selain ei tue toista kuvaformaattia, niin se osaa valita toisen toimivan vaihtoehdon. Toinen käytäntö on hyödyntää attribuuttia nimeltä **srcset**, jolla voidaan antaa **<img>**-tagin sisällä

määrityksiä, mikä vaihtoehto esimerkiksi samasta kuvasta ladataan vaikkapa resoluution vaihtuessa. Käytännön avulla määritetään **src**:n sisään vakiokuvan polku, joka ladataan varalta, jos selain ei tukisi **srcset** ominaisuutta. (Frain 2022, luku 9.)

2.2.4 Viitekehyksien käyttämisestä

Teknologiat, kuten viitekehykset, voivat helpottaa suunnitteluprosessia, koska niissä on yleensä otettu huomioon mukauttava suunnittelu, asemointitekniikat, median mukautuminen ja mobiili ensin -lähestymistapa. CSS-Viitekehykset sisältävät tyypillisesti valmiiksi mukauttavia ominaisuuksia, jotka vähentävät kehittäjän työtaakkaa, sillä mediakyselyjen luominen käsin ei ole välttämätöntä. Esimerkiksi suosittu Bootstrap käyttää esimääritettyjä CSS-koodeja (GeeksforGeeks 2023).

Bootstrap Grid-ruudukkojärjestelmä on joustava, 12-sarakkeinen asettelukehys, joka perustuu CSS Flexboxiin. Se mahdollistaa responsiivisen asemoinnin, joka sallii rivien ja sarakkeiden luomisen ja sisällön sijoittamisen haluttuihin kohtiin. Sen sijaan edellisessä aliluvussa mainittu CSS Grid tarjoaa tiukempaa hallintaa asettelun suunnitteluun jakamalla verkkosivun tilan riveiksi ja sarakkeiksi, jotka voidaan määrittää kiinteälevyisiksi tai suhteessa käytettävissä olevaan tilaan. Vaikka molemmat järjestelmät tarjoavat mukautuvuutta, CSS Gridillä ei ole esimääritettyjä ruudukon pysäytyspisteitä kuten Bootstrapissä, joka käyttää mediakyselyissä **min-width**-mediaominaisuutta määrittääkseen pysäytyspisteet eri kokoisille näytöille. (Goyal 2021.)

Bootstrap-kehyksessä käytetään kuutta vakiopysäytyspistettä, jotka vastaavat laajaa skaalaa eri näyttöjen kokoja. Bootstrapin CSS pyrkii soveltamaan minimimäärän CSS-määrityksiä, jotta asettelu toimisi pienimmässä pysäytyspisteessä, ja lisää sitten kerroksittain CSS-määrityksiä säätääkseen layoutia suuremmille laitteille. (Bootstrap team s.a.) Mobiili ensin -lähestymistavassa kaikki poikkeukset CSS-määrityksiin lisäävät koodin monimutkaisuutta ja työn tehottomuutta, mikä puolestaan johtaa tarpeeseen lisätä testausta ja tekee koodipohjan ylläpidosta vaikeampaa (Clancey 2019).

Viitaten edellä mainittuun, viitekehysten käyttäminen ei välttämättä ole kivutonta tai aina suositeltavaa, koska oletustyylimääritykset täytyy syrjäyttää CSS-tyyliohjeissa, jos määrityksiä aikoo muuttaa. Muutetuista tyyleistä ja valitsimista (eng. selector) tulee näin ollen tarkemmin osoitettuja, mikä voi johtaa myöhemmin ongelmiin (Frain 2022, luku 14). Esimerkiksi tekstin kirjasinkoon muuttaminen ristiriitaiseksi CSS-viitekehyksen oletustyylimäärityksien ja valitsimien kanssa voi johtaa siihen, ettei kirjasinkoko mukaudu enää näyttökoon vaihtuessa. Tämä tarkoittaa sitä, että kehittäjän on lisättävä korvaavia, eli poikkeavia, mediakyselyjä CSS-tyyliohjeisiin ongelman korjaamiseksi.

Viitekehysten käytön vaikutukset on otettava huomioon kehitystyössä. Kehykset voivat tarjota hyödyllisiä työkaluja, mutta niiden käyttö saattaa myös hankaloittaa projektia, jos sivusto ei mukaudu toivotusti eri näytöillä ja CSS-tyylejä joudutaan toistuvasti ohittamaan käsin ja testausta lisätä.

2.3 Optimointia tarkistuslistojen avulla

Verkkosivuston optimoinnilla pyritään parantamaan suorituskykyä eri osa-alueilla käyttämällä työkaluja, kehittyneitä strategioita ja kokeiluja (Vwo 2023). Verkkosivuston optimoinnin päätavoitteena on varmistaa, että sivusto luo sulavan digitaalisen matkan (Siteimprove 2023). On tärkeää tunnistaa optimoinnin osa-alueet ja ymmärtää niihin liittyviä työkaluja ja käytänteitä hyvin. Sivuston optimointi on kokonaisuus, joka kattaa yhtä lailla sivuston sisällön laadukkuutta kuin sen suorituskykyä eri mittareilla.

Sivustoille voidaan optimoinnin näkökulmasta tehdä erilaisia hyödyllisiä auditointeja ja testauksia useiden verkossa tarjolla olevien työkalujen avulla, koska ne voivat antaa vihjeitä siitä, miten optimointi eri osa-alueilla on onnistunut. Suorituskykyä, turvallisuutta, hakukoneoptimointia, responsiivisuutta ja paljon muuta voidaan testata säännöllisesti, jolloin potentiaaliset ongelmat voidaan havaita ja niihin voidaan reagoida. Käyn aliluvussa läpi valittujen tarkistuslistojen kautta optimoinnin tärkeimpiä osa-alueita järjestyksessä.

2.3.1 Suorituskyvyn optimointi

Suorituskyvyn optimoinnissa lähtökohtana on sivuston ensimmäisen latautumiskerran nopeus. Googlen mukaan käyttäjät usein poistuvat sivustolta, jos sen latautuminen kestää yli 3 sekuntia (Camarena 7.3.2023). Verkkosivuston maksimaalinen suorituskyky on kahden ydinstrategian tulosta – sivuston painolastin keventäminen (eng. reducing page weight) ja jakelun virtaviivaistaminen (eng. streamlining delivery). Ensimmäisen strategian tavoitteena minimoidaan ensinnäkin selaimen lataaman datan määrä. Toisen strategian tavoitteena optimoidaan, kuinka data haetaan ja lähetetään palvelimelta ja ladataan selaimeen virtaviivaisesti. (Brandon s.a.)

Pagespeedchecklist.com-sivustolla määritellään edellä mainittujen strategioiden koostuvan listasta, joka alkaa pohtimalla tarkkaan resurssien vähentämistä, jolloin paras tasapaino sisällön ja ominaisuuksien välillä näkyy nopeampina latausaikoina ja keskimääräisesti parempana käyttäjäkokemuksena (Brandon s.a.). Seuraavaksi käyn järjestyksessä listan läpi kohta kohdalta aloittaen siitä, kuinka sivuston painolastia kevennetään:

1. **Resurssien vähentämisellä** tarkoitetaan sitä, että tiedostojen kokoa ja määrää vähentämällä sivustojen latautuminen voi olla nopeampaa ja tehokkaampaa. Käytännössä tämä tarkoittaa sitä, että verkkosivustoon liittyvät tiedostot tai kansiot siistitään lopulta tarpeettomiksi tulkituista koodeista, liitännäisistä, tiedostoista tai muusta, mikä voi lopulta enemmän

hidastaa sivustoa kuin hyödyttää sitä. Samalla voidaan pohtia kevyempiä, nopeammin latautuvia vaihtoehtoja. (Brandon s.a.)

2. **Tiedostojen yhdistäminen, pakkaaminen:**
   Yhdistämällä useammat pienemmät CSS- ja JavaScript-tiedostot vähälukuisemmiksi isoiksi tiedostoiksi tuottaa tehokkaampaa HTTP-pakkausta, mikä vähentää edelleen sivustolle siirrettävän datan määrää. (Brandon s.a.)
3. **Minify Code:**
   Kaikki HTML-, CSS- ja JavaScript-koodit voidaan optimoida tiivistettyyn **minified**-muotoon, jolloin koodista poistetaan kaikki tarpeettomat välilyönnit ja tyhjät rivit, mikä pienentää tiedoston kokoa. Tietokoneet ja selaimet pystyvät tulkitsemaan koodin ilman ylimääräisiä välilyöntejä, joten niiden poistaminen nopeuttaa sivun latausaikaa ja vähentää datankäyttöä. (Brandon, n.d.; Monus 14.1.2022.). Yleensä CDN:n kautta hyödynnettävistä resursseista löytyy myös minified-versiot.
4. **Fonttien ja kuvien optimointi:**
   On suositeltavaa käyttää vain muutamia nopeasti latautuvia fontteja ja hyödyntää niiden lataamista nopeuttavia tekniikoita. Esimerkiksi, kun käytetään web-fontteja, kuten Google Fonts -palvelun tarjoamia fontteja, voimme ladata fonttien CSS-resurssit asynkronisesti käyttämällä rel-attribuutin arvoina **Preload** tai **Preconnect**. CDN:n palveluiden sijasta mukautetut fontit voidaan ladata myös paikallisesti. Edellä mainitut tekniikat optimoivat fonttien latausaikaa. (Brandon s.a.; Monus 14.1.2022.)

   Kuvien osalta valitaan pakatut kuvat, joissa on huomioitu tiedostokoko, kuvaformaatti ja sopiva kuvakoko mobiililaitteille ja suuremmille näytöille. Jokainen kuvaformaatti sopii parhaiten eri käyttötarkoitukseen. Optimaalisen kuvan koon pitäisi olla sopivan pieni nopean latauksen varmistamiseksi, mutta riittävän suuri hyvän kuvanlaadun säilyttämiseksi. (Brandon s.a.). Lazy Loading (suom. laiska lataus) on tekniikka, jossa ei-kriittiset resurssit ladataan vasta, kun niitä tarvitaan (Rachel & Andrew 2022). Suurten kuvatiedostojen lataaminen mobiililaitteille voi kuluttaa tarpeettomasti dataa (Hempenius 2022).

   Seuraavat kuusi kohtaa listassa keskittyvät jakelun virtaviivaistamiseen:
5. **Eliminate Render Blocking Resources**:
   Renderöinnin estävät resurssit ovat staattisia tiedostoja, kuten fontteja, HTML-, CSS- ja JavaScript-tiedostoja, jotka ovat välttämättömiä verkkosivun renderöinnille. Apuna tunnistamisessa voidaan käyttää sivuston suorituskykyä testaavaa diagnostiikkaa, joista näkee mitkä resurssit jäävät käyttämättä. (Monus 14.1.2022.) On tärkeää ottaa huomioon sivuston latautumishierarkia, eli järjestys, jossa eri osat sivustosta renderöidään. Hyvänä käytäntönä on priorisoida aluksi ne osat, jotka näkyvät ennen kuin sivustoa vieritetään. Näin eliminoidaan

renderöinnin estävät resurssit, joiden ennenaikainen latautuminen lisää latautumisviiveitä. (Brandon s.a.)

Skriptien latautumisjärjestystä voidaan hallita hyödyntämällä <link>-tagin ominaisuuksia, kuten **async** ja **defer**, jotka mahdollistavat skriptien latautumisen viivästämisen. **Rel**-attribuutin **preload**-arvoa käytetään indikoimaan, että skripti tai resurssi tulisi ladata mahdollisimman nopeasti.CSS-tyylitiedostojen latautumisnopeutta voidaan parantaa vastaavalla tavalla, mutta **type**-attribuutin arvoksi tulee määrittää **stylesheet**. Lisäksi tärkeimmät CSS-tyylit priorisoidaan lisäämällä ne suoraan HTML-dokumentin <head>-osaan. Vähemmän tärkeät CSS-tiedostot voidaan asettaa latautumaan matalammalla prioriteetilla, lisäämällä **onload**-attribuuttiin funktio, joka poistaa media-attribuutin, kun tiedosto on ladattu. Jos skriptien suoritus estetään (esimerkiksi selaimen asetuksista), voidaan käyttää <noscript>-tagia ja sisällyttää siihen <link>-tagi, joka lataa CSS-tyylit tavalliseen tapaan. Tämä varmistaa, että tyylit ladataan, vaikka JavaScript ei olisi käytössä. Edellä kuvatut menetelmät nopeuttavat sivun latautumista. (Brandon s.a.; Monus 14.1.2022.)

6. **Resource Hints eli resurssivihjeet:**
   Resurssivihjeet ovat HTML-koodin osia, jotka edistävät sivun latausprosessia ennakoivasti käynnistämällä verkkoyhteydet ja tiedostojen lataukset etukäteen. Ne antavat selaimelle etumatkaa ohjeistamalla valittujen tiedostojen lataamista jo ennen kuin selain tyypillisesti löytäisi ja lataisi ne. Resurssivihjeitä käytetään HTML-koodissa <link>-elementin sisällä määrittelemällä **rel**-attribuutin arvoksi esimerkiksi **preload**, **preconnect**, **pre-fetch** tai **pre-render**. (Brandon s.a.; Jackson 30.9.2022.)
7. **Web hosting** eli verkkoisännöinti koostuu kokoelmista palveluita, jotka tallettavat ja jakavat verkkosivuja Internetissä mukaan lukien web-serverien ohjelmistot ja laitteistot. Serveriasetuksilla ja resurssivalinnoilla on isompi vaikutus sivuston latautumiseen, kun verkkoliikenne kasvaa tai sivuston sisältö monimutkaistuu tai resurssien käyttö voimistuu. Verkkoisännän valinnalla voi olla suuri merkitys sivuston latautumisnopeuteen ja siihen, kuinka kauan sivusto pysyy katkeamatta saavutettavana. (Brandon s.a.; Jackson 6.2.2018.)
8. **Content Delivery Network (CDN) eli sisältöverkko:**
   CDN on verkkoisännöinnin lisämuoto, jolla tähdätään parantamaan tehokkuutta välimuistilla ja jakamalla tiedostoja vakaasta ja maantieteellisesti laaja-alaisesta palvelimien verkostosta. Nopeushyödyn lisäksi sisältöverkot voivat väliaikaisten verkkoliikennepiikkien aikana tarjota parempaa luotettavuutta ja sietokykyä. CDN-palvelut auttavat tallettamaan ja lataamaan resursseja sivustolle nopeasti, mutta latausnopeus on riippuvaista käyttäjän sijainnista suhteessa CDN-serverin sijaintiin nähden. (Brandon s.a.; Jackson 6.2.2018.)
9. **HTTPS** ja **HTTP/2** protokollat auttavat parantamaan verkkosivujen latausnopeutta ja turvallisuutta. HTTPS lisää yhteyteen turvallisuuskerroksen autentikoimalla ja salaamalla

datanvaihdon esimerkiksi verkkokauppaostosten yhteydessä. HTTP/2 on päivitetty versio HTTP-protokollasta. Sen merkittäviä parannuksia on kyky siirtää useita tiedostoja samanaikaisesti, mikä parantaa sivuston latausaikaa. HTTP/2 tehostaa sivuston suorituskykyä ja parantaa CDN-servereiden turvallisuutta. (Brandon s.a.; Baumgartner 17.2.2023.) HTTPS suojaa sivustoa monenlaiselta hakkeroinnilta (Kaismala 2022).

10. **HTTP-pakkaus (eng. HTTP compression):**
    Tiedostojen kokoa voidaan pienentää ja latausnopeutta kasvattaa asettamalla palvelimelle HTTP-pakkaus päälle. Se toimii parhaiten tekstipohjaisiin kooditiedostoihin toisin kuin mediatiedostoihin, jotka on jo pakattu. (Brandon s.a.)
11. **Välimuistin hallinta (eng. Cache Control)**
    Välimuistin asetusten määrittäminen auttaa vähentämään tarpeettomien tiedostojen uudelleenlataamista. Selaimet on suunniteltu tallentamaan ja käyttämään aiemmin ladattuja tiedostoja, mutta välimuistia hallitsemalla voidaan tehostaa tätä prosessia. Välimuistia säädellään määrittelemällä ajanjakso, jonka ajan välimuistiin tallennetut tiedostot säilytetään. (Brandon s.a.)

2.3.2 Turvallisuus

Turvallisuuden optimointi on äärimmäisen tärkeä osa verkkosivujen kehittämistä, koska sen laiminlyöminen voi altistaa käyttäjät tietoturvariskeille. Käsittelen lyhyesti turvallisuuden optimoimisesta muutamia oleellisia menetelmiä, joilla suojata verkkosivuja, koska aihekokonaisuus on erittäin laaja. Luvussa oletetaan perusymmärrystä aiheesta.

HTTPS on yhdistelmä HTTP-protokollaa ja SSL/TLS-protokollaa. Verkkosivuston turvaamiseksi on hankittava SSL/TLS-sertifikaatti, joka todentaa verkkosivuston aitouden, salaa liikenteen ja luo turvallisen yhteyden verkkosivuston ja palvelimen välillä. Salatun HTTPS-yhteyden puuttuminen voi alentaa sivuston näkyvyyttä hakukoneissa. (Kaismala 2022.)

SQL-injektio on kyberhyökkäys, jossa käytetään SQL-koodia manipuloimaan sivuston backend-tietokantaa. Pahimmassa tapauksessa hyökkääjä saa hallinnointioikeudet verkkosivun tietokantaan. Cross-site scripting (XSS) hyökkäyksissä suojattuihin ja luotettuihin verkkosivuihin lisätään vahingollisia koodeja, joille sivuston käyttäjät altistuvat ja pahimmassa tapauksessa hyökkääjät pääsevät käsiksi käyttäjien tietoihin sivustolla. Clickjacking on käyttöliittymäperustainen hyökkäys, jossa huijataan verkkosivuston käyttäjiä klikkaamaan haitallisia linkkejä. Hyökkääjät voivat esimerkiksi luoda aidosta verkkosivustosta iframe-upotetun valesivuston, jonka päälle he upottavat haitalliset linkkinsä. (Mozilla Corporation 2023c.)

Verkkosivustojen tietoturvaa ja yksityisyyden suojaa voidaan parantaa HTTP-otsikoilla, jotka sisältävät ehtoja, joita palvelin asettaa selaimelle. X-Frame Options -otsikolla pyritään estämään clickjacking-hyökkäyksiä. X-XSS-Protection estää yleisimmät ja perustavanlaatuiset XSS-hyökkäykset, mutta ei kaikkia. HSTS-otsikko on suunniteltu estämään salaamattomien yhteyksien käyttö HSTS-metodia tukevissa selaimissa. (Viljanen 16.3.2019.) Content Security Policy (CSP) on tietoturvaotsikko, jota suositellaan sivustoille, joilla käsitellään käyttäjätietoja (Kitamura, Nalpas & Janc 2022). CSP-otsikon lisääminen sivulle tarjoaa lisäsuojaa sivustolle estämään luvattoman koodin syötön, kuten XSS-hyökkäykset. Sivustolla olevia haku- ja kirjoituskenttiä on järkevä suojata validoimalla ja niin sanotusti sanitoimalla syötteet mahdollisesta koodista, jotka voivat altistaa XSS-hyökkäyksille ja SQL-injektioille. (OWASP 2021a.)

Cross-Site Request Forgery (CSRF) on hyökkäys, joka ilmenee, kun vahingollinen verkkosivusto aiheuttaa käyttäjän selaimen tekemään jotain ei-toivottua, kun käyttäjä on autentikoitu ensin luotetulla sivustolla (KirstenS 2023). CSRF-hyökkäykseltä voidaan suojautua esimerkiksi generoimalla toisiaan vastaavia CSRF-tokeneita, joiden validointi suojaa CSRF-hyökkäyksiltä: CSRF-tokenit suojaavat sivuston käyttäjiä, koska ilman validoitua tokenia, eli valtuutusta, hyökkääjä ei voi luoda backend-serverille valideja pyyntöjä (OWASP 2021b).

Subsource Integrity (SRI), eli aliresurssien integriteetti on turvallisuusominaisuus, jota käytetään varmistamaan, että verkkosivun lataamat resurssit, kuten skriptit tai tyylitiedostot, eivät ole muuttuneet latauksen aikana. Tämä tapahtuu määrittämällä kryptografinen tiiviste (eng. hash), eli uniikki merkkijono, jota haetun resurssin on vastattava. SRI-tiiviste lasketaan tiedoston sisällöstä. Tiivisteitä voidaan luoda esimerkiksi komentorivillä käyttämällä OpenSSL-kirjastoa syöttämällä tiedoston nimi ja sopiva komento. (Mozilla Corporation 2023c.)

2.3.3 SEO eli hakukoneoptimointi

Hakukoneoptimointi on digitaalisen markkinoinnin kulmakivi, jolla määritetään pitkälti hakukoneiden kautta sivustolle saapuva kävijämäärä. SEO kuvaa kaikki prosessit ja tekniikat, jotka auttavat verkkosivustoa esiintymään korkealla hakukoneiden tuloksissa. (Pavlik 27.3.2023.) Hakukoneoptimointi on laaja kokonaisuus, jota käsittelen seuraavaksi tiiviin listan avulla ja pyrin mainitsemaan tärkeimmät kohdat, vaikka suuressa kokonaisuudessa kaikilla yksityiskohdilla on merkitystä. Kokonaisuutena SEO voidaan lyhyesti jakaa tekniseen, sivuston sisäiseen, sisällölliseen ja sivuston ulkopuoliseen hakukoneoptimointiin sekä hakusanojen tutkimiseen. Mainitsemani listan kohdat perustuvat hakukoneoptimointiin erikoistuneen Semrush.com-sivuston tarkastuslistaan, mutta en noudata sen järjestystä orjallisesti. Semrush-sivun ohjeistus hakukoneoptimointiin on hyvin laaja ja perinpohjainen.

1. **Tekninen SEO**

   Tekninen SEO muodostaa optimoinnille vankan pohjan ja varmistaa, että sivusto kyetään hakukoneiden puolesta selaamaan (eng. crawling) ja indeksoimaan läpi. Sivuston indeksointiprosessia voidaan nopeuttaa ja seurata isoimpien hakukoneiden SEO-työkaluilla, kuten Bing Web-masters Tools ja Google Search Console. Sivustokartan tekeminen ja lataaminen auttavat hakukoneita löytämään, selaamaan ja indeksoimaan sivuston kaiken sisällön. Robots.txt-tiedostolla voidaan kertoa hakukoneelle, mitkä sivut ja tiedostot ovat selattavissa. Käyttämällä SEO-työkaluja voimme seurata sivuston suorituskykyä ja havaita ongelmia, kuten virheelliset linkit tai duplikaattisivujen indeksoituminen. (Silva 14.12.2022.)
   On suositeltavaa käyttää sivustolla HTTPS-protokollaa, koska se vaikuttaa merkittävästi hakukoneoptimointiin, sillä esimerkiksi Google pitää sivuston turvallisuutta yhtenä tärkeänä kriteerinä sijoittelussaan (Ait Bahajji & Illyes 7.8.2014). Sivuston suorituskyvyn optimointi vaikuttaa merkittävästi hakukoneoptimointiin, koska käyttäjät eivät jää odottelemaan hitaille sivuille. Edellisen lisäksi sivuston responsiivisuus ja erityisesti mobiiliystävällisyys ovat tärkeitä elementtejä SEO:n kannalta, koska Google on virallisesti siirtynyt mobile first -indeksointiin. Sivuston URL-osoitteiden tulisi olla rakenteeltaan SEO-ystävällisiä ja selkokielisiä, koska hakukoneet osaavat tulkita ja selata niitä paremmin. (Silva 14.12.2022.)

2. **Sivuston sisäinen ja sisällöllinen SEO**

   Ilman laadukasta, hyödyllistä sisältöä ja hyvää käyttökokemusta, verkkosivusto saattaa menettää sijoituksiaan hakutuloksissa ja siten orgaaninen verkkoliikenne vähentyy. Sisäisessä hakukoneoptimoinnissa keskitytään tunnistamaan ja korjaamaan sivuston metatageihin, otsikoihin, avainsanoihin ja muuhun sisältöön liittyvät ongelmat parantaakseen sen näkyvyyttä hakukoneissa. Sisältöauditointi on tapa havaita parannusmahdollisuuksia sisäisessä SEO:ssa. (Silva 14.12.2022.)

3. **Sivuston ulkopuolinen SEO**

   Ulkopuolisessa hakukoneoptimoinnissa keskitytään parantamaan näkyvyyttä sivuston ulkopuolella ja tehostamaan sen suorituskykyä suhteessa kilpailijoihin. Tämän tavoitteen saavuttamiseksi pyritään lisäämään sivustoon johtavien niin sanottujen backlinkkien määrää ja vahvistamaan brändin ja sivuston näkyvyyttä eri alustoilla, kuten sosiaalisessa mediassa. Google Business -tilin avaaminen ja optimointi on hyödyllistä, koska se auttaa yritystä tavoittamaan asiakkaitaan paremmin hakukoneiden ja Google Mapsin kautta. (Silva 14.12.2022.)

4. **Hakusanojen tutkiminen**

   Hakusanojen ja niiden yhdistelmien tutkiminen voi parantaa sivuston näkyvyyttä hakukoneissa suhteessa kilpailijoihin ja näin houkutella lisää asiakkaita. Prosessi sisältää tärkeiden orgaanisten hakusanojen löytämisen, järjestämisen ja priorisoinnin, jonka pohjalta

luodaan oma hakusanoihin pohjautuva strategia. Kilpailijoiden tunnistaminen on olennaista, koska heidän käyttämänsä hakusanat voivat parantaa sekä heidän että optimoitavan sivun sijoitusta hakutuloksissa. Hakusanojen tutkiminen ei rajoitu vain yksittäisiin sanoihin tai sanapareihin, vaan kattaa myös pidemmät sanaketjut tai lauseet, jotka voivat itse asiassa parantaa sivuston sijoittumista hakutuloksissa. (Silva 14.12.2022.)

2.4 Ketterät menetelmät

Esittelen luvussa lyhyesti ketteriä menetelmiä ja siihen liittyviä menettelyjä, kuten ketterä vaatimusmäärittely, käyttäjätarinat ja tuotejono -tekniikat.

Ketterät menetelmät ovat ohjelmistokehityksen lähestymistapa, joka keskittyy nopeaan reagointiin ja jatkuvaan parantamiseen. Menetelmien tavoitteena on toimittaa asiakkaalle säännöllisesti ja rivakasti vaatimuksia täyttäviä ohjelmistoversioita. (Koski s.a.) Lyhyesti tarkoituksena on tehdä kehittämisprosessista tehokkaampi ja mukautuvampi sekä siten tuottaa asiakkaille lopulta parempia lopputuloksia (Tivix 8.2.2022). Säännöllinen keskustelu ja kasvokkain käytävä kommunikointi ovat tärkeä osa tiedonvälitystä. Yksinkertaisuus ja keskittyminen olennaiseen ovat ketterille menetelmille tunnusomaisia piirteitä. Tiimien vahvuus on kyky itseorganisoitua, arvioida jouhevasti tekemistään ja mukauttaa toimintaansa tarpeiden ja jatkuvan parantamisen mukaisesti. (Koski s.a.)

Ketterä vaatimusmäärittely on lähestymistapa, jossa vaatimukset muodostuvat ja tarkentuvat joustavasti iteraatioiden eli toistettavien työvaiheiden myötä. Toisin kuin perinteisessä vaatimusmäärittelyssä, ketterän lähestymistavan alussa tehdään juuri sen verran määrittelyä, että kehityksen suunta saadaan selville. Iteraatioiden aikana valmistuu uusia ohjelmistoversioita, joita asiakkaat voivat testata ja käyttää. Tämän käytön perusteella saadaan arvokasta palautetta ja mahdollisesti uusia vaatimuksia, jotka ohjaavat kehitystä eteenpäin. Ketterä vaatimusmäärittely eroaa perinteisestä vaatimusmäärittelystä ollen joustava ja iteraatioiden kautta rakentuva jatkuva prosessi. (Lilja 2021.)

Käyttäjätarinan tehtävä on kuljettaa asiakkaan vaatimukset kehittämisprosessin läpi määrittelystä, ohjelmointiin ja aina tuotantoon saakka. Käyttäjätarinat ovat vaatimuksia, joilla määritellään kehitettävän ohjelman sisältämät toiminnot siten, että kaikki kehittämiseen osallistuvat ymmärtävät ne selkeästi ja väärinymmärryksiltä vältyttäisiin. (Leffingwell 2011, osa 2–3.)

Kehitysjono koostuu tunnistetuista käyttäjätarinoista, jotka tulee toteuttaa ennen ohjelmiston valmistumista. Ketterien menetelmien kehitysjono on joustava kokonaisuus, koska se täydentyy ja muuttuu iteraatioiden myötä mukautuen projektin tarpeisiin ja oivalluksiin. (Leffingwell 2011, osa 2.)

3 Freesi Fiilis -sivuston kehitystyö

Kuvaan luvussa tarkemmin toimeksiannon ja kehitystyön eri osa-alueita sekä oppimispolkuani. Toimeksiantoni oli kehittää uudet, modernit ja sisältömuokattavat kotisivut yritykselle, Freesi Fiilis. Freesi Fiilis on vuodesta 2013 asti toiminut kauneudenhoito- ja hierontapalveluja tarjoava pienyritys Helsingin Töölössä. Heidän edellinen verkkosivustonsa oli toteutettu kotisivukoneella, ja ne toimivat yrityksen kotisivuina aina uusien sivujen julkaisuun asti. Vanhahtava sivusto ei ollut käyttöliittymältään tai ulkoasultaan moderni, eikä toiminut mobiiliystävällisesti.

Tarkoitukseni oli suunnitella freesifiilis.fi-sivuston ulkoasu ja käyttöliittymä mahdollisimman laajasti itsenäisesti, mutta yhteistyössä toimeksiantajan kanssa. Minulla oli asiantuntijan ja kehittäjän rooli, eli esitin toimeksiantajalle alustavia suunnitelmiani, jotka perustuivat tietoon ja ymmärrykseeni web-kehityksestä. Kehitin sivustoa itsenäisesti, mutta toimeksiantajan toiveita kuunnellen. Annoin parhaan osaamisen mukaisesti paljon painoarvoa sivuston eri osa-alueiden optimoinnille ja testaukselle, jotta lopputuloksena olisivat mahdollisimman laadukkaat kotisivut. Pyrin panostamaan erityisesti pääsivuun, joka palvelee parhaiten käyttäjiä yhdestä paikasta. Ajatuksena oli, että sivusto olisi tyyliltään maltillisen minimalistinen. Sivusto tarjoaisi käyttäjälle juuri sopivasti olennaista informaatiota, jotta sivusto pysyisi helposti silmäiltävänä ja ymmärrettävänä.

Henkilökohtaisesti tahdoin yhdistää kiinnostukseni web-kehitykseen kehittäjän ja suunnittelijan näkökulmista. Koin ammatillisen kehityksen tärkeäksi projektin aikana ja halusin ymmärtää omia heikkouksia ja vahvuuksiani. Halusin oppia paremmin hyödyntämään web-kehityksen perinteisiä rakennuspalikoita, eli HTML-, CSS- ja JavaScript-kieliä. JavaScript on luotu mahdollistamaan skriptauksen käyttö kaikkiin HTML-dokumentin elementteihin ja yhdessä CSS:n kanssa ne muodostavat yhdistelmän dynaamisten verkkosivujen kehitykselle (Nixon 2014, luku 1). Käytin työkalusarjana Bootstrap 5 -kirjastoa pyrkimyksenäni nopeuttaa kehitystyötä ja toteuttaakseni sivuston responsiivisuutta. Toteutin kehitystyötä paikallisesti, jonka jälkeen siirsin valmiin sivuston web-hotelliin.

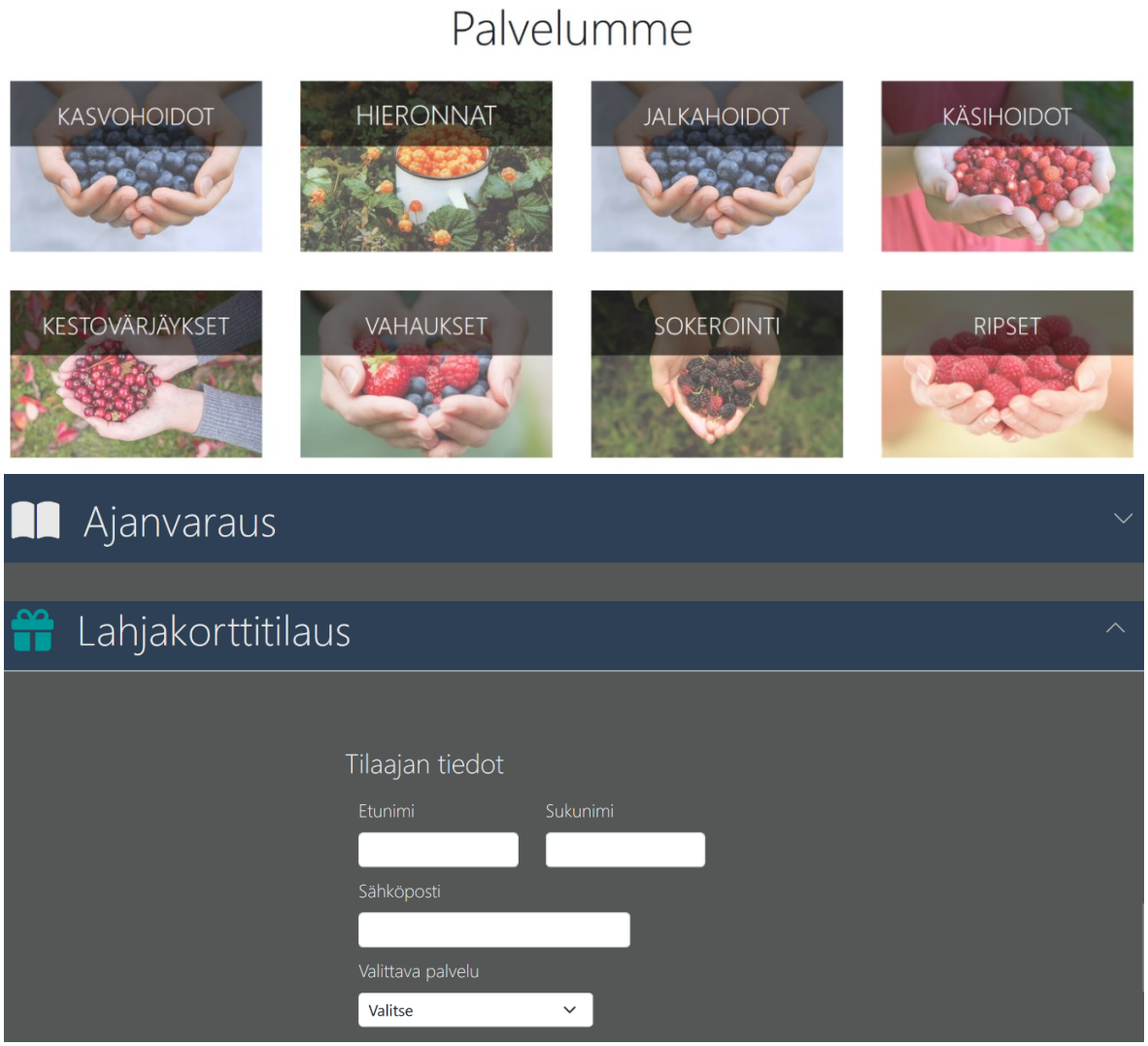
Oppimispolku sisälsi itselleni kartoittamattomia osa-alueita, kuten taustahallinta ja sivustojen erilaiset toteutustavat CMS-järjestelmäarkkitehtuurin näkökulmasta. Edellä mainittujen osa-alueiden parissa opettelin myös suppeasti PHP-kieltä. Opinnäytetyön kirjoittamisprosessille oli tyypillistä jatkuva itsereflektointi ja sitä kautta ymmärryksen uudelleenkehittyminen.

3.1 Projektin käytänteet ja ketterä vaatimusmäärittely

Projektin toteutus noudatti soveltaen ketteriä menetelmiä. Säännöllinen kasvokkain käytävä kommunikaatio, etäviestintä tarpeen mukaan ja muistiinpanojen tekeminen olivat toistuva osa projektia. Viestinnässä käytettiin esimerkiksi WhatsAppiin perustettua ryhmää. Vaatimusmäärittelyä

toteutettiin iteratiivisesti ja vaatimuksia löydettiin toistuvasti läpi prosessin. Vaatimuksia paranneltiin ja priorisoitiin projektin edetessä. Emme luoneet formaalia dokumentaatiota ja validointia tehtiin rennoissa katselmointipalavereissa ja hyväksymistestauksina, jotta varmistuttiin vaatimusten vastaavan toimeksiantajan ja käyttäjän tarpeita.

Ajatustenvaihtoa ja yksinkertaista koodausluonnostelua käytettiin nopeaan ideoiden validointiin, jolloin kehitystyön uusiin vaiheisiin voitiin ryhtyä ketterästi. Esimerkiksi hahmottelin toiminnallisuuksista nopeita esimerkkejä koodaamalla, jonka jälkeen valitsimme sopivan toteutustavan nähtyämme konkreettisesti, kuinka ideat toimivat käytännössä. Seuraava kuva on esimerkki hahmotellusta, jossa testattiin erikseen palveluosion, ajanvarauksen ja lahjakorttitilauksen asettelua ja toimivuutta. Ajanvarauksen ja lahjakorttitilauksen toteuttamista kokeiltiin haitarikomponentilla. Accordion eli haitari on Bootstrapin komponentti, jossa otsikkosisällön alapuolelle haitariin liitetty sisältö laajenee auki tai supistuu kiinni klikattaessa.



Kuva 1. Esimerkki hahmottelusta.

Käytin kehitystyössä tuotteen kehitysjono- ja käyttäjätarina-tekniikoita. Kirjoitin käyttäjätarinat toimeksiantajan esittämien tarpeiden mukaisesti mahdollisimman selkokielisesti ja sovelsin tekniikkaa myös eri käyttäjäroolien näkökulmasta. Nimesin siis useampia käyttäjärooleja riippuen, mitä ominaisuutta tarina havainnollisti. Seuraava kuva on esimerkki käyttäjätarinoista, jotka on toteutettu Excelillä.

High	Asiakas/Käyttäjä	Löytää selkeän osion lahjakorttitilaukselle, tehdä helposti ja selkeästi yhden tai useamman lahjakorttitilauksen, jossa on useampi palvelu valittuna. **#Lahjakortti**	Saavutan vaivattoman kokemuksen tilaamisesta yhdellä -lähetä painikkeen klikkauksella, kun lomake on hetkessä täytetty ja tilaus vahvistettu.
High	Tilauksenkäsittelijä	Tahdon tiedot asiakkaan tekemästä lahjakorttitilauksesta selkeässä, tiiviissä muodossa. Yhdessä tilauksessa voi olla useampi lahjakortti ja palvelu valittuna. **#Lahjakortti**	Tilauksen käsittely nopeutuu, selkeytyy ja helpottuu, kun yksi tilaus sisältää kaiken olennaisen.
High	Sisällönmuokkaaja	Halutessani voin muokata tiettyjen ennalta valittujen osioiden sisältöjä ja koska teen sitä todennäköisesti harvoin, muokkauksen täytyy olla helposti opittavaa ja toteutettavaa. **#CMS** Halutessani voin lisätä palveluita, muokata hinnastoa ja palveluiden kuvauksia nopeasti.	Voin lisätä, muokata, piilottaa tai poistaa sisältöä verkkosivuillamme vaivattomasti ja silloin kun haluan.

Kuva 2. Esimerkki käyttäjätarinoista.

Koostin tarinoista kehitysjonolistan, johon purin kunkin käyttäjätarinan siitä tunnistamieni tehtävien ja alitehtävien perusteella. Jaottelin kehitysjonon teemoittain osiin ja pyrin järjestämään teemat tehtävineen mahdollisimman loogiseen järjestykseen ylhäältä alas. Seuraava kuva on esimerkki kehitysjonolistasta, joka on toteutettu Excelillä.

BACKLOG TASK & USER STORY ID	STORY POINTS	Kuvaus ongelmasta	Tehtävät	STATUS
4	**15**	**#Lahjakorttilaus**	Lahjakorttitilauksen käyttöliittymä ja tehtävät	Valmis
Task 1 Bootstrap Form	2	Vanha puutteellinen lomake ja huono toiminnallisuus. Tehdään uusi Bootsrapillä.	Lahjakorttilomake -Tilaajan tiedot: etunimi, sukunimi, email, lahjakorttien määrä, palveluiden valinta korttiin, lisätiedot, tilauksen vahvistaminen, Lähetä ja lisätiedot painikkeet. -Lisättiin bs-canvas lisätiedoille	Tehty
Task 2 Bootstrap Select ja Jquery	2	Bootstrap 5:n multiselect on epäkäytännöllinen.	Korjataan lisäämällä Bootstrap Select -Selectpicker. Lisää CDN:n Selectistä ja Jquery	Tehty
Task 3 Lomakekenttien validointi ja pakollisuus	2	Käyttäjän on täytettävä pakolliset kentät (Required). Kentät validoitava ja pattern oltava sellainen, ettei lomaketta voi lähettää liian aikaisin/väärillä merkeillä.	Front-end selainvalidointi. Emailin pyyntimuoto ja input-pattern oikeaksi.	Tehty

Kuva 3. Kehitysjonolista.

3.2 Toimeksiantajan lähtötilanne ja alkuperäinen näkökulma heidän ongelmaansa

Kuvailen seuraavaksi projektin lähtötilannetta. Aloitustapaamisen perusteella ymmärsin, että vanhan sivuston ongelmana oli toimeksiantajan näkökulmasta vanhanaikaisuus, ilmeettömyys ja epäkäytännöllisyys, eli huono käyttäjäkokemus. Sivuston muokattavuus oli rajoittunutta ja hieman hankalaa, koska he muokkasivat sivustoa harvoin. Sivuston mobiiliselaaminen oli asiakkaan näkökulmasta kömpelöä, koska sivusto ei ollut riittävän mobiiliystävällinen eikä responsiivinen.

Toimeksiantaja toivoi sivustolle kasvojenkohotuksen, jotta nykyiset toiminnot toimisivat ja asettuisivat sivustolle asiakkaiden näkökulmasta paremmin. Koettiin tärkeäksi, että ajanvaraus ja muut olennaiset toiminnot löytyisivät helposti ja niihin pääsisi siirtymään riittävän monesta paikkaa sivua. Yritys kaipasi myös päivitystä ja selkeyttä lahjakorttitilauksen tekemiseen, koska nykyinen ratkaisu oli melko epäkäytännöllinen ja epäselvä.

Toimeksiantajan toiveena olivat uusien ja vanhojen asiakkaiden näkökulmasta helppokäyttöiset, visuaalisesti miellyttävät, sopivan yksinkertaiset ja selkeät, nykyaikaiset kotisivut, joita voidaan selata älypuhelimella ja tietokoneella. Sivuston rakenne ja sisältö saivat olla siten yksinkertaiset, että ulkoasu olisi pelkistetyn hienovarainen ja sisältäisi heille tarpeelliset toiminnot, joita he ovat tottuneet käyttämään. Syyksi kerrottiin, ettei uusien maksupalveluiden käyttöönotto nykyisten käytäntöjen uudistamiseksi tai korvaamiseksi ole suunnitelmissa, ellei se ole ehdottoman välttämätöntä.

Toimeksiantaja kertoi päivittävänsä itse sivuston sisältöä harvoin, joten olisi tärkeää, että tietyt sivuston osat olisivat heille helposti muokattavissa. Sivuston yläosassa saisi olla pieni asianmukainen osio, johon he voisivat helposti muokata tarpeen mukaan tiedotteita. Sivuston tekstit, kuten palvelujen sisältö saisivat olla muokattavia, mutta niihin tehtävät muutokset tulisivat olemaan harvinaisempia. Sivulla ei nähty tarpeelliseksi olla useita kuvia, eikä myöskään tekstisisällön merkittävää lisäämistä pidetty tarpeellisena. Muuten sivuston tuli kestää ikää useita vuosia muuttumattomana.

Saapuessaan verkkosivulle vanhan asiakkaan tulisi yhdellä klikkauksella päästä ajanvaraukseen tai tekemään lahjakorttitilaus. Uusille asiakkaille tulisi olla useampia mahdollisuuksia avata ajanvaraus, jotta vierailu sivustolla olisi sujuvaa. Palveluita esittelevään osioon tahdottiin asianmukaiset kuvat, jotka toimisivat painikkeina, joista aukeaisi kunkin palvelun laajempi kuvaus. Sivustolla tuli olla yritystä esittelevä osio. Osioista tulisi muotoutua selkeä, hillitty ja kompakti kokonaisuus.

3.3 Kehitysvaiheet

Kehitys- ja suunnittelutyön välinen liikehdintä oli joustavaa ongelmakohtien ratkaisemiseksi. Vaatimuksia päivitettiin jatkuvasti. Verkkosivuston kehitysvaiheet olivat seuraavanlaiset, mutta vaiheiden välillä tapahtui liikehdintää, koska esimerkiksi testausta tehtiin jatkuvasti kehitysvaiheessa:

1. Suunnittelin verkkosivuston ulkoasua ja ratkaisuja toimeksiantajan tarpeisiin. Otin selvää modernin verkkosivuston vaatimuksista, jotka olivat olennaisia toimeksiannon kannalta.
2. Vertailin teknologioita ja perustelin valitsemieni teknologioiden käytön. Valitsin jälkiasennettavan sisällönhallintajärjestelmän. Jatkoin oman tietotaidon kehittämistä.
3. Verkkosivuston kehittäminen paikallisesti, UI/UX ja ulkoasun toteuttaminen. (Front-End)
4. Sivuston optimointia: SEO, suorituskyky, turvallisuus, mukauttaminen
5. CMS-järjestelmän jälkiasentaminen ja valmiin sivuston siirtäminen web-hotelliin. (Back-End)
6. Testausta – Lopputestaus
7. Lopullinen julkaisuvaihe: julkaistun sivuston toiminnallisuuden seuraaminen, käyttäjäpalaute ja SEO

3.4 Alkuvaatimusten toteuttaminen

Tein suunnittelutyötä iteratiivisesti ja ensimmäisessä aihiossa loin sivuston pohjan, navigoinnin, painikkeet. Jaoin sisällön toivottuihin osioihin alkuvaatimusten mukaisesti. Sovimme, että teemme sovellusmaisia piirteitä sisältävän sivuston, jonka tarkoituksena oli parantaa sivuston selkeyttä ja yhdenmukaisuutta. Bootstrap-kirjasto sisälsi valmiita komponentteja verkkosovellusten toteuttamiseen, kuten modaalit, joita voitiin hyödyntää kehitystyössä.

Modaali on käyttöliittymäelementti, joka näyttää sisältöä ilman, että käyttäjän tarvitsee poistua nykyisestä näkymästä tai sivulta. Modaali-ikkuna avautuu yleensä nykyisen ikkunan tai näkymän päälle, usein tummennetun tai epäselvän taustan päälle, joka korostaa modaali-ikkunan etualalla olevaa asemaa. (Wikipedia-artikkeli 8.6.2023.)

"Sovellusmaisilla piirteillä" viittaan siihen, kuinka modaalien sisällä käytettyjä elementtejä on integroitu muualle sivustolle. Esimerkiksi palveluja laajemmin kuvaavat myöhemmin luodut lisäsivut noudattavat samaa visuaalista linjaa kuin modaalit, vaikka eivät itse sisälläkään modaaleja. Modaaleista tutut ristimerkit, jotka sulkevat modaalit, löytyvät myös näiltä lisäsivuilta. Niiden tehtävänä on tässä yhteydessä ohjata käyttäjä takaisin etusivulle.

Kävimme läpi, että sivuston optimointiin tulee liittymään haasteita, mutta pyrin ratkaisemaan ongelmat parhaani mukaan. Olimme toimeksiantajan kanssa yhtä mieltä siitä, että perinteinen useasta HTML-sivusta koostuva verkkosivusto ei olisi lähtökohtaisesti yhtä käytännöllinen tai kiinnostava käyttäjäkokemuksena kuin yhden sivun sivusto.

Yhden sivun sovellus (SPA) on web-sovelluksen tai verkkosivuston tyyppi, joka lataa kaikki tarvittavat resurssit yhdellä sivulatauksella. Tämän jälkeen sovellus ei enää lataa uusia sivuja, vaan päivittää nykyistä sivua dynaamisesti käyttäjän kanssa tapahtuvan vuorovaikutuksen mukaisesti. (Wikipedia-artikkeli 2.6.2023.)

Alun perin sivuston oli siis tarkoitus koostua vain yhdestä verkkosivusta, mutta ymmärsin myöhemmin, että yhden sivun sovelluksen hakukoneoptimointi saattaa olla haastavaa. Tahdoin varmistua, että sivuston SEO onnistuisi varmasti riittävän laadukkaasti, joten sovin lisääväni kaikille palveluille, lahjakorttitilaukselle ja ajanvaraukselle erilliset sivut.

Valitsin alkuvaatimusten mukaiset teknologiat. Tein Bootstrapillä palvelut-osioon kuvapainikkeista koostuvan asemoinnin Bootstrap Grid -tekniikkaa hyödyntäen, jotta asettelu toimii responsiivisesti. Alkuvaiheessa sisällytin sivuun selkeät osiot, jotka olivat ensin latautuva pääkuva-alue, Palvelut, Esittely ja Yhteystiedot, jonka sijoitin sivuston alatunnisteeseen. Käytin Bootstrapiä pitkälti sivuston rakentamisessa, koska kuten aiemmin mainitsin sen tarjoavan paljon valmiita komponentteja, kuten mobiiliystävällisiä modaaleja.

Kehitystyön alkutaipaleella oli jäänyt vielä epäselväksi, kuinka ajanvaraus sijoitetaan sivustolle ja millainen ajankohtaista tietoa sisältävän osion tulisi olla. Samoin ei ollut selvää, kuinka lahjakorttitilaus toteutetaan ja sijoitetaan sivustolle. CMS-järjestelmää ei ollut vielä valittu, mutta kriteerien perusteella vaihtoehtoja oli kartoitettu. Tämän jälkeen lähdimme toimeksiantajan kanssa pohtimaan uusia vaatimuksia ja keskittymään tarkemmin ajanvaraukseen ja lahjakorttitilaukseen.

3.5 Jälkiasennettava CMS

Ennen sivuston kehitystyön aloittamista tutkin erilaisia CMS-vaihtoehtoja. Tavoitteeni oli löytää sellainen järjestelmä, joka tuntuisi kiinnostavalta ja soveltuisi parhaiten sivuston kehittämiseen. Arvioin, että minulla oli riittävästi taitoja ja osaamista viedä kehitysprojekti alusta loppuun. Testasin kokeilumielessä sekä WordPressiä että Drupalia, mutta en kokenut työskentelyn kummankaan alustan parissa tukevan riittävän vapaasti web-kehitykseen liittyviä oppimistavoitteitani. Ensituntumalla edellä mainitut perinteiset, monoliittiset CMS-järjestelmät vaikuttivat sekä kehittäjän että käyttäjän näkökulmasta hieman vaikeaselkoisilta ja aikaa vieviltä. Tämä johtui osittain siitä, että minulla ei ollut aikaisempaa kokemusta kyseisistä teknologioista. Näiden järjestelmien suurimpina etuina pidetään niiden helppokäyttöisyyttä, joka ei vaadi syvällistä koodaustaitoa. Perinteisissä

CMS-ratkaisuissa itse sisällönhallintajärjestelmä määrittelee sivuston rakenteen, mikä asettaa kehittäjälle tietyt rajoitukset. Tunsin, että nämä rajoitukset olisivat esteenä omille oppimistavoitteilleni.

Edellä mainituista lähtökohdista kartoitin mahdollisuutta, kuinka voisin kehittää staattiset verkkosivut valitsemillani teknologioilla ja muuntaa ne vasta jälkikäteen dynaamisesti muokattaviksi. Huomasin tutkiessani aihetta eri keskustelupalstoilla ja verkkosivustoilla, että erilaisia CMS-järjestelmiä on runsaasti, joten lopputyöprojektin rajallisen koon kannalta kaikkien läpikäynti ei olisi ollut mielekästä. Etsiessäni tietoa järjestelmien jälkiasennettavuudesta, tutustuin päätteettömiin CMS-ratkaisuihin. Päätteettömät toteutukset ovat pääpiirteittäin järjestelmiä, joissa verkkosivuston sisältö voidaan toteuttaa minkä tahansa frontend-teknologian avulla. Samaan aikaan erillinen taustahallinta ylläpitää rajapinta-arkkitehtuuria, jonka avulla verkkosivuston sisältö voidaan esittää eri laitteilla.

Päätteettömän CMS:n etuina voidaan nähdä esimerkiksi joustavuus, laajennettavuus ja suorituskyky, koska toteutustapa vapauttaa mahdollisuuksia toteuttaa digitaalinen palvelu haluamillaan teknologioilla, eikä kahlehdi sitä yhden alustan alle (Arola 23.3.2022; Muranen 15.6.2022). Mielestäni siten toteutustavan näkökulmasta päätteettömyys on moniulotteinen vaihtoehto, joka voi olla samalla sen heikkous ja vahvuus. Päätteettömyys saattaa nimittäin rakentua kehitystyön näkökulmasta monimutkaiseksi järjestelmäkokonaisuudeksi, jolloin kaikki työ ja ylläpito vaativat enemmän osaamista työntekijöiltä, kuten tietoperustassa todettiin.

Löysin muutamia jälkiasennettavia CMS-vaihtoehtoja, jotka tuntuivat soveltuvan hyvin tavoitteisiini. Jälkiasennettavaa toteutustapaa voisi mielestäni kuvailla kompromissiratkaisuksi perinteisen monoliittisen CMS:n ja päätteettömän CMS:n välillä. Staattinen sivusto olisi myös mahdollista siirtää WordPress sivustoksi, joista yksi tapa on luoda manuaalisesti WordPress -teema, joka perustuu alkuperäiseen HTML-sivustoon (Weller 26.4.2023). Edellä mainittu toteutustapa ei ole luonteeltaan jälkiasennettava vaan ennemmin jälkikäteen siirrettävä tai sulautettava. Lisäksi on myös tapoja, joilla rakentaa perinteisistä CMS-järjestelmistä päätteettömiä toteutuksia. WordPress REST API on keino, jolla WordPress-sivustosta on mahdollista rakentaa niin sanotusti päätteetön WordPress-sivusto (Hughes 23.3.2022).

Etsin jälkiasennettavaa ratkaisua, joka olisi mahdollisimman yksinkertainen. Päädyin valitsemaan CouchCMS-nimisen sisällönhallintajärjestelmän, koska tämä voitiin jälkiasentaa staattisen sivuston CMS-järjestelmäksi. CouchCMS oli yhteensopiva suunnitelmani kanssa, jotta voisin kehittää verkkosivun selainpuolen haluamallani tavalla. Ratkaisu tuntui projektini suurusluokkaan ja pienen yrityksen tarpeisiin sopivalta, kustannustehokkaalta ja selkeältä ratkaisulta. Tämän jälkeen päätin aloittaa sivuston kehittämisen alkuvaatimusten mukaisesti.

Kävimme toimeksiantajan kanssa läpi heidän tarpeensa ja mukautin sisällönhallinnan myöhemmin heidän toiveiden mukaisesti. Kaikki HTML-sivut muutettiin CouchCMS hallinnoiduiksi sapluunoiksi, eli sisältömuokattaviksi sivuiksi muuntamalla HTML-sivujen pääte PHP:ksi. Käytännössä tämä tarkoitti sitä, että kaikki sivuston sisältämät tekstiosiot ja kuvat ovat muokattavissa. Myös palveluita hintoineen voidaan lisätä, poistaa ja muokata. HTML-sivujen muokattaviin osiin liitettiin XHTML-kaltaisia tageja suoraan koodiin.

Couch-tageja, kuten <cms:template>, <cms:editable>, <cms:show> integroidaan HTML-koodiin. Näitä tageja käytetään määrittämään, mitkä osat sivustosta ovat muokattavissa CMS:n kautta. Lisäksi erilaisia parametreja käytetään muokkaamaan tagin toimintaa ja ne antavat tagille tarkkaa tietoa siitä, mitä sen on tarkoitus tehdä. Selaimen lopulta renderöidessä sapluunasivun, joka sisältää Couch-tageja, jokainen tagi korvataan yksilöllisellä tulosteella. Tagi ei siis koskaan näy HTML:ssä, joka palautetaan selaimelle. (Webzyme Softwares 2017b.)

CouchCMS:n asentaminen merkitsi sitä, että sivuston järjestelmäarkkitehtuurista muodostui monoliittisempi, koska sisällön lopullinen esitystapa on riippuvaisempi taustahallinnasta ja yhdestä alustasta. Sivustoa voidaan kuitenkin edelleen kehittää sisällönhallintajärjestelmän ulkopuolisesti perinteisillä web-kehitysmenetelmillä, mutta lopulta sivuston HTML-sivujen ja sisällön jakaminen kulkee CouchCMS:n läpi, kuten perinteisissä CMS-järjestelmissä.

3.6 Kehitystyön kulmakiviä

Sivuston suunnittelussa ja kehityksessä ei ilmennyt ylipääsemättömiä ongelmia. Eniten vaatimusmäärittelyä vaativat ajanvarausjärjestelmän upottaminen ja lahjakorttitilaus-ominaisuuden tekeminen sivustolle. Pyrin jatkuvasti parantamaan käyttäjäkokemusta pieniä yksityiskohtia viilaamalla, kuten lisäämällä esimerkiksi mobiilinavigointipalkin sulkemisfunktion, kun käyttäjä painaa mihin tahansa palkin ulkopuolella.

Ajanvarauksessa käytettiin Timma-ajanvarauspalvelun tuottamaa komponenttia, joka upotettiin sivustolle. Sen upottaminen modaalin sisään pääsivulla tuotti hieman teknisiä haasteita käyttäjäkokemuksen näkökulmasta, mutta ratkaisin ongelmat muutamalla JavaScript-koodilla ja CSS-tyyleillä. Ongelmana oli, että muu sivusto liikkui taustalla, kun ajanvarausta käytettiin enkä voinut vaikuttaa ajanvarauspalvelun tarjoaman upotetun komponentin CSS-määrityksiin.

Lahjakorttitilaus-toiminnon toteutus vaati teknisesti enemmän työtä, vaikka se ei aluksi vaikuttanut kovin monimutkaiselta. Tämän ominaisuuden jouduin rakentamaan alusta loppuun itse, ja matkan varrella kohtasin monia haasteita. Oli löydettävä ratkaisu siihen, kuinka käyttäjä voi tilata useita lahjakortteja kerralla ja valita useita palveluja samalle lahjakortille. Toimeksiantajan oli kyettävä myös lisäämään tai muuttamaan palveluita. Käyttäjäkokemuksen parantamiseksi lisäsin ostoskorin

kaltaisen yhteenveto-ominaisuuden, jonka kautta käyttäjä voi muuttaa valittujen palvelujen määrää. CMS-järjestelmän jälkiasentamisen jälkeen toiminnallisuutta oli hieman säädettävä, jotta se toimisi paremmin CouchCMS:n tarjoaman lomaketoiminnon kanssa. Lahjakorttitilaus-lomakkeen suunnittelussa hyödynsin Bootstrapin lomakemuotoilua ja Bootstrap Select -liitännäistä, koska Bootstrapin alkuperäinen monivalintaominaisuus ei ollut mielestäni tarpeeksi hyvä.

Kenties teknisesti suurimman haasteen koin, kun paikallisesti kehittämäni sivusto piti lopulta siirtää web-hotelliin. Tämä ei ollut niinkään vaikea tai työläs tehtävä, vaan ennemminkin täysin uusi kokemus. Tehtävän suorittaminen vaati erityistä tarkkuutta, jotta asennus ja oikeiden asetusten määrittäminen onnistuisivat.

3.7 Optimoinnista

Rakentamani verkkosivusto ei ole erityisen monimutkainen, mutta ilman optimointia sen käyttäminen olisi ollut hankalaa, hitaampaa ja jäykempää. Optimointi oli kokonaisuutena projektin työläin ja jatkuvasti toistuva työvaihe, koska sitä tapahtui melkein läpi koko projektin. Lähdin optimoinnissa liikkeelle siten, että kävin läpi osa-alueet yksi kerrallaan valitsemieni tarkistuslistojen mukaan. Tämä teki etenemisestä prosessissa helpompaa, koska pystyin käsittelemään asioita kohta kohdalta. Näin pyrin varmistamaan, että sivusto täyttäisi modernin verkkokehityksen tärkeät vaatimukset systemaattisesti.

Freesi Fiilis -sivuston etusivu on elementtien runsauden vuoksi raskain ladata. Koska pääsivu toimii sivuston sydämenä ja tärkeimpänä palveluportaalina, sen optimointi oli erityisen merkittävää. Jo kehitystyön aikana pyrin varmistamaan, että sivuston keskeiset osa-alueet, kuten käyttäjäkokemus ja saavutettavuus, olivat hyvällä tasolla ennen varsinaista järjestelmällistä optimointia. Selvitin eri lähteiden perusteella, mitkä ovat verkkosuunnittelussa välttämättömiä elementtejä, ja pyrin saattamaan nämä vaatimukset kuntoon. Varmistin, että valitsemani väripaletti ja fontit olivat yhteensopivia ja läpäisivät saavutettavuustestit. Tavoittelin yksinkertaista ja minimalistista ulkoasua, selkeitä osioita ja harkittua sommittelua.

Optimoinnilla on vaikutusta siihen, minkälaisen vaikutelman yritys antaa itsestään ulospäin. Kotisivuilla on varmasti yksi suurimmista merkityksistä monien yrityksien näkyvyyteen. Usein voi törmätä mainintaan, että verkkosivusto on myyntitiimin ainoa myyjä, joka tekee töitä 24 tuntia vuorokaudessa.

3.7.1 Responsiivisuudesta eli mukauttamisesta

Mukauttaminen tekee sivustosta mobiiliystävälliset, parantaa käyttäjäkokemusta ja sen ulkonäköä eri näytönkoilla. Se auttaa välttämään tarpeetonta koon muuttamista, vierittämistä, zoomausta tai

panorointia, jotka voisivat ärsyttää käyttäjiä. Responsiivisuus poistaa myös tarpeen luoda useita eri asetteluja eri näytönkokoja varten, tarjoten siten kustannustehokkuutta, joustavuutta, parempaa käyttäjäkokemusta ja hyötyjä hakukoneoptimoinnin näkökulmasta. (WebFX s.a.) Käytännössä laadukas mukauttava suunnittelu ilmenee siten, että sivustoa suunnitellaan yhdessä rinnakkain sekä pienille kuvaporteille että isoille.

Otin lähtökohdakseni hyödyntää Bootstrap-viitekehystä toteuttaakseni mobiiliystävälliset ja mukautetut sivut, mutta jouduin myöhemmin parantelemaan tiettyjä yksityiskohtia parantaakseni käyttäjäkokemusta. Käytin Bootstrapin ominaisuuksia mukauttamisen saavuttamiseksi, koska se nopeutti työtä. Siten sain fontit ja kuvat skaalautumaan oikein kuvaportin vaihtuessa melko vaivattomasti. Kohtasin kuitenkin joitain haasteita tiettyjen tekstielementtien kanssa, koska muutamat tekstit skaalautuivat mielestäni liian pieniksi mobiililaitteilla. Tämän vuoksi jouduin korvaamaan Bootstrapin CSS-esimäärityksiä mediakyselyillä käsin. Esimerkiksi etusivun ensinäkymän otsikkofonttien koot jouduin määrittelemään erikseen itse, koska en ollut tyytyväinen niiden skaalautumiseen Bootstrapin määrittäminä.

Mobiili ensin -lähestymistapaa noudatin siten, että suunnittelin mukauttamista ja käyttäjäkokemusta rinnakkain mobiililaitteille ja pöytäkoneille. En kuitenkaan ottanut varsinaisesti lähestymistavan mukaisesti omassa CSS-tyyliohjeen vakiomäärityksessä lähtökohdaksi mobiilikokoa, koska käytin Bootstrapiä, joka muodosti sivustolle vakioasettelun. Kirjoitin kuitenkin esimäärityksiä korvaavia mediakyselyitä pienimmästä näyttökoosta ylöspäin. Jos en olisi käyttänyt Bootstrap-kehystä, olisin joutunut miettimään vakioasettelun toteutustapaa ja CSS-määrityksiä eri tavalla mobiililähtöisesti.

Tärkeimmät mediakyselyt koskivat esilatautuvien taustakuvien valikoimista kuvaportin leveyden vaihtuessa. Mediakyselyillä paransin lähinnä käyttäjäkokemusta esimerkiksi piilottamalla elementtejä, jotka olivat turhia pienemmillä näytöillä tai testaamaan hover- eli leijumisefektiä, jolloin kosketusnäytöllisillä laitteilla kyseinen efekti voitiin jättää huomiotta. Pysäytyspisteitä kirjoittaessa on tärkeää huomioida, verrataanko käytettäviä arvoja tarkasti kuvaportin minimileveyteen, maksimileveyteen vai johonkin muuhun mediaominaisuuteen, koska huono logiikka aiheuttaa mediakyselyn suorittamisen ei-toivotuin tuloksin.

Siinä mielessä en siis noudattanut mobiili ensin -lähestymistapaa, että olisin itse aloittanut vakioasettelun suunnittelun pienimmästä mobiilikoosta, jonka päälle yksityiskohdat kerrostuisivat. Suunnittelin mobiili- ja pöytäkonenäkymiä rinnakkain ja etenin virtaviivaisesti molemmista suunnista keskelle kohti keskikokoisia näyttökokoja. Minunhan ei varsinaisesti tarvinnut tehdä itse vakio asettelua, koska käytin Bootstrapia, mutta jouduin tekemään useampia muutoksia yksityiskohtiin, jotta sivusto näytti visuaalisesti johdonmukaiselta ja miellyttävältä.

Verkkosivustojen mukauttamista voidaan rakentaa konkreettisesti web-kehityksen ydinrakennuspalikoiden, kuten HTML-, CSS- ja JavaScript-kielien osaamisella. Edellä mainituista rakennuspalikoista muodostuvaa kokonaisuutta hallitsemalla web-kehittäjä pystyy omaksumaan itse tekniikoita, jotka ovat välttämättömiä laajan mukauttamisen saavuttamiseksi. Esimerkiksi CSS tarjoaa paljon eri työkaluja mukauttavaan suunnitteluun, kuten Math() funktiot.

HTML tarjoaa sivuston perusrakenteen, jota parannetaan ja muunnellaan CSS:n ja JavaScriptin avulla. CSS:ää käytetään esittämisen, muotoilun ja asettelun hallintaan, kun taas javascriptiä käytetään erilaisten elementtien toiminnan ohjaamiseen. (Cox 21.7.2021.)

Sivuston asettelua on hyvä testata samanaikaisesti kehitystyön aikana ottaen huomioon mukauttaminen ja sisällön skaalautuminen kokonaisuutena. Tein sivustolle ajoittain mukauttamistestausta, joista sain lopulta hyvät tulokset. Eniten testausta tein browserstack-sivuston tarjoamalla palvelulla, jolla voitiin tehdä testausta eri laiteselainyhdistelmillä.

Verkkosivun mukauttamisen testaaminen tarkoittaa visuaalisen johdonmukaisuuden tarkistamista eri laitteilla ja selaimilla, erilaisten sivun elementtien testaamista eri näytön kokojen mukaan ja verkkosivuston nopeuden varmistamista eri laiteselainyhdistelmillä. Tämä auttaa varmistamaan, että verkkosivusto tarjoaa saumattoman käyttäjäkokemuksen. (Kumar 2022.)

3.7.2 Sivustorakenne ja SEO

Alun perin en aikonut luoda perinteistä useasta sivusta koostuvaa sivustoa, koska mielestäni käyttäjän näkökulmasta perinteinen sivustorakenne voi ilmetä kankeampana tai jopa sekavana käyttökokemuksena kuin yhden sivun rakenne, jos sitä ei ole mietitty kunnolla. Päädyin poimimaan molempien ratkaisujen parhaita puolia ja välttämään sudenkuopat.

Usean sivun sivustorakenteessa on riskinä sivustokokonaisuuden hitaampi latautumisaika ja se, että käyttäjät saattavat ärsyyntyä, jos heidän täytyy navigoida sivulta toiseen turhaan yksinkertaisen vastauksen perässä. Toisaalta yhden sivun sivuston asettelusta voi muodostua myös hämmentävä ja sekavan hajanainen, jos sivulla esitetään liikaa sisältöä. (LeSuer 2022.)

Ymmärsin tutkittuani aihetta, että yhden sivun sivustojen hakukoneoptimointi olisi ollut haastavaa. Lyhyesti yhden sivun sivuston heikkouksina SEO:n näkökulmasta voidaan pitää rajallisempaa hakusanojen kohdentamista, yksityiskohtaisen sisällön puutetta ja edistyneempien SEO-strategioiden toteuttamisen haasteellisuutta (Heitzman 18.4.2022). Hakukoneoptimointi on tärkeää, koska se voimistaa verkkosivuston näkyvyyttä, mikä kasvattaa liikennettä sivustolla lisäten mahdollisuuksia muuntaa prospektit potentiaalisiksi asiakkaiksi (Digital Marketing Institute 18.10.2021).

Päädyin hyvän SEO:n ja käyttäjäkokemuksen saavuttamiseksi kompromissiin. Halusin, että sivuston pääsivu tarjoaa käyttäjälle helposti ja tiiviisti tietoa sekä sovellusmaista käytettävyyttä. Yhden sivun sivuston SEO:n rakentamisen muodostuttua ongelmalliseksi, muunsin sivustorakennetta siten, että pääsivu säilytti alkuperäisen käyttäjäkokemuksen, mutta rakensin uudet yksityiskohtaisemmat osiot kaikista palveluista erillisille sivuille. Osioihin voitiin tuottaa kattavammin sisältöä ja informaatiota, mikä on hyvä käyttäjälle ja tärkeää SEO:n kannalta. Samoin jaoin ajanvarauksen ja lahjakorttitilauksen omiksi sivuiksi.

Pääsivulta löytyy kuitenkin edelleen omat modaalinsa kullekin palvelulle, josta löytyy tiivistetty paketti olennaista tietoa ja lisätietoa-linkki, joka ohjaa kunkin palvelun omalle lisäsivulle. Myös ajanvaraus ja lahjakorttitilaus löytyvät modaalien alta ja nämä muodostavat paremman käyttäjäkokemuksen, kun erillisten sivujen välillä ei tarvitse liikkua. Tieto säilyy yksinkertaisella tavalla modaaleissa, vaikka ne suljettaisiin ja avattaisiin uudelleen, koska sivustoa ei tarvitse ladata uudelleen. Eli käyttäjä saattaa palata tekemään lahjakorttitilausta tai ajanvarausta, vaikka sulkisi modaalit väliaikaisesti tutkiakseen pääsivun sisältöä. Pääsivu on siten sivuston keskus, josta löytyy kaikki tärkeimmät toiminnot.

Selitin toimeksiantajalle, että uudistukset olivat kannattavia ja heidän etujensa mukaisia. Heillä ei varsinaisesti ollut mitään sanottavaa vastaan, mutta hieman päänvaivaa nostatti ajatus siitä, että heidän tarvitsi keksiä SEO-ystävällistä sisältöä kullekin lisäsivulle. He kuitenkin ymmärsivät ratkaisun tuomat edut.

Pyrin tekemään tietoperustassa esittelemäni SEO-tarkistuslistan mukaiset muutokset sivustolle ja panostamaan SEO-ystävällisyyteen, kuten lisäämään kaikki tarvittavat metatagit ja analysoimaan hakusanoja ja niiden yhdistelmiä. Kannustin toimeksiantajaa panostamaan teksteihin, jotka olisivat vähintään 150–300 merkkiä, ja että niissä hyödynnettäisiin tärkeimpiä avainsanoja.

Varsinaisen hakukoneoptimoinnin eri hakukoneita ajatellen pystyin aloittamaan sivuston julkaisun jälkeen. Ensimmäiseksi julkaisun jälkeen lisäsin sivuston Googlen Search Console -palveluun. Se on Googlen kehittämä SEO-palvelu, joka sisältää työkaluja ja mittareita parantamaan optimoinnin tuloksia (Google s.a.). Oikeastaan ainoa selkeä ongelma, jonka kohtasin kyseisessä vaiheessa, oli että palvelu indeksoi samat sivut kahdesti. Tämä johtui siitä, että palvelu ymmärsi lataamani sivustokartan jälkeen, että jokainen sivu sisältää erikseen sivun, joka päättyy kenoviivaan ja sivun, joka ei pääty kenoviivaan. Onnistuin korjaamaan ongelman lisäämällä 301-uudelleenohjauksen, jotta URL-osoitteet vastasivat sivustokartan mukaista muotoa, johon kävijä tai hakukonebotti uudelleenohjataan. Kun olin tehnyt mielestäni riittävät toimenpiteet ja korjaukset Search Consolessa, lisäsin sivustokartan vielä Bing-hakukoneen Webmasters Toolsissa, jotta sivuston indeksointi

saatiin käyntiin myös toisessa hakukoneessa. Lisäsin myös robots.txt-tiedoston, jossa määritin mitkä hakukonebotit saavat selata sivustoa ja mitkä sivut saa indeksoida.

SEO:n tulokset alkavat ilmeisesti näkyä kunnolla vasta 3–6:en kuukauden kuluttua, joten en voi varsinaisesti ottaa kantaa suuremmin, kuinka optimointi tulosten valossa näyttäytyy. Suoritin kuitenkin hakusanatutkimuksen pohjalta erilaisilla hakusanayhdistelmillä kokeiluja ja huomasin, että Freesi Fiilis nousi Googlessa kärkipäähän monilla kombinaatioilla. Otin alueelliseksi lähtökohdaksi Töölön kaupunginosan ja valitsin jonkin yrityksen tarjoaman palvelun, jonka lisäksi saatoin lisätä vielä yhden tai kaksi tarkentavaa sanaa (esimerkkinä: stressi hieronta töölö, rentouttava hieronta töölö tai mobilisoivat tekniikat hieronta töölö). SEO:lla pyrin parantamaan sivuston erottautumista kilpailijoista.

Toimeksiantaja on kertonut, että sivuston julkaisun jälkeen ajanvarauksissa olisi näkynyt kasvua. On silti hankala sanoa, onko uudella verkkosivustolla voinut olla merkittävää vaikutusta vai johtuuko muutos muusta syystä, kuten asiakkaiden ostokäyttäytymisen yleisenä piristymisenä vaikeampien taloudellisten aikojen keskellä.

3.7.3 Suorituskyvystä

Nopeuden optimoinnilla varmistetaan, että sivut latautuvat nopeasti, mikä parantaa käyttäjäkokemusta ja lopulta konversiota, eli astetta, jolla käyttäjät toteuttavat toivotun toiminnan sivustolla (Cloudflare 2023). Sivun latautumisnopeuden koin erityisen tärkeäksi tavoitteeksi, koska se on käyttäjäkokemuksen kannalta ensimmäinen heikkouden merkki ja tekee huonoa SEO:lle. Ennen sivuston julkaisua pyrin testaamaan sivustoa säännöllisesti kehitysvaiheessa Google Chromen Lighthouse-työkalulla. Lighthouse pisteyttää sivustoa arvioimalla eri osa-alueita, kuten tehokkuus, esteettömyys, SEO ja parhaat käytännöt (Chrome Developers 2022). Käsittelen luvussa vain suorituskykyä tehokkuuden, eli sivuston latautumisnopeuden näkökulmasta.

Lighthouse on yksinkertainen keino testata sivustoa paikallisesti Chrome-selaimella. Pöytäkoneiden optimoinnin näkökulmasta Lighthouse-arvion tulokset olivat jo projektin alkuvaiheessa erinomaiset, eli lähes sadassa pisteessä, mutta mobiilioptimointi osoittautui selkeästi hankalammaksi. Sivuston julkaisun jälkeen kokeilin muitakin työkaluja, joilla testata suorituskykyä, jotta näin olivatko tulokset linjassa Lighthousen kanssa. En huomannut merkittävää eroa.

Jokainen verkkosivusto on ainutlaatuinen omien tavoitteidensa, tyylinsä ja toimintojensa suhteen, eikä yleispätevä laadunmäärittely välttämättä sovellu kaikkiin sivustoihin. Toisinaan sivusto saattaa tarvita erityisiä suunnitteluratkaisuja ja toiminnallisuuksia parhaan mahdollisen käyttäjäkokemuksen takaamiseksi. Erityisratkaisut voivat vaikuttaa Lighthouse-arvion tulokseen negatiivisesti. Tässä kontekstissa on tärkeää huomioida oikea tasapaino, koska joissakin tapauksissa paremman

käyttäjäkokemuksen takaaminen voi tarkoittaa sitä, että on oltava valmiita hyväksymään hieman alhaisemmat Lighthouse-arvioinnin tulokset. (Snell 2022.)

Lighthousen antamia tuloksia ei siis yksinään kannata pitää vahvana indikaattorina sivuston optimoinnin onnistumiselle. Se ei esimerkiksi korvaa aidon käyttäjäkokemuksen antamaa tietoa tai palautteesta saatua tietoutta. Lighthousen avulla voidaan kuitenkin oikeinkäytettynä löytää puutteita eri osa-alueilla, joita voidaan parhaan mukaan korjata. Lighthouse testaa suorituskykyä mobiililaitteilla 3G-yhteydellä, mikä ei myöskään suoranaisesti anna täysin aitoa kokonaiskuvaa sivuston latautumisnopeudesta, koska valtaosalla Euroopassa on käytössä vähintään nopeampi 4G (Raza 14.6.2022).

Joka tapauksessa testitulokseni mobiililaitteella pyöri 50–75:n pisteen välillä melkein koko kehitystyön ajan ennen kuin aloin tutkia asiaa tarkemmin suorituskyvyn optimoinnin näkökulmasta. Sinänsä tulos ei ollut verrattain edes huono, koska useimpien testaamieni sivustojen mobiilitestien tulokset jäävät tuohon 50–75:n pisteen väliin. Testasin siis useampia verkkosivustoja saadakseni verrokkituloksia. Googlen kehittämän mittarin mukaan hyvin optimoidun sivuston tehokkuutta mittaavan indikaattorin tulisi olla yli 89 pistettä (Chrome Developers 2023). Eli tässä valossa tulos vaati parannusta. Nykyisillä mobiiliyhteyksillä sivustot kuitenkin käytännössä latautuvat yhtä nopeasti kuin pöytäkoneillakin, mutta ei välttämättä katvealueilla tai muualla päin maailmaa alueilla, joilla mobiiliyhteydet voivat olla huonot.

Seuraavaksi aloitin varsinaisen suorituskyvyn optimoinnin. Tulkitsin Lighthousen antamia vihjeitä ongelmista ja etenin kohta kohdalta useampien tarkistuslistojen mukaisesti prosessissa. Ensimmäiset selkeät ongelmat liittyivät joustavaan (eng. flexible) mediaan, eli kuvien optimointiin ja niiden latautumiseen sivustolla. Pakkasin kaikki kuvat kahteen tai kolmeen eri kokoon ja muutin tiedostot **webp**-tiedostomuotoon käyttäen hyödyksi aluksi Inkscape-sovellusta ja myöhemmin squoosh.app -verkkosivustoa. Sitten määritin mediakyselyillä, että sopiva optimoitu kuva latautuu kuvaportin vaihtuessa. Kuten tietoperustassa todettiin, WEBP-tiedostomuodon etuna on hyvä pakkautuvuus ja kuvanlaadun säilyminen silti suhteellisen hyvänä. Edelleen olisi järkevää vielä erikseen tarjota vaihtoehtoinen tiedostonmuoto, jos käyttäjän laite ei tukisikaan webp-muotoista tiedostoa, mutta tätä en tehnyt. Samalla tiivistin sivuston CSS- ja JavaScript-tiedostoja ja pyrin karsimaan turhaa koodia.

Ehkä suurin suorituskykytestausta kuormittava indikaattori on ensimmäisten elementtien latautumisnopeus ennen kuin sivua vieritetään, kuten tietoperustassa kerrotaan. Eli kuvien, fonttien ja tarvittavien skriptien sekä CSS-tyylien latautumisjärjestys vaikuttaa olennaisesti siihen, kuinka kuormittunut latautuminen tulee olemaan. Jos sivuston heti näkyvässä osassa on kuva, niin tämä kannattaa määrittää esiladattavaksi **rel**-attribuutin arvolla **preload**. Tämän lisäksi sivustolla välittömästi

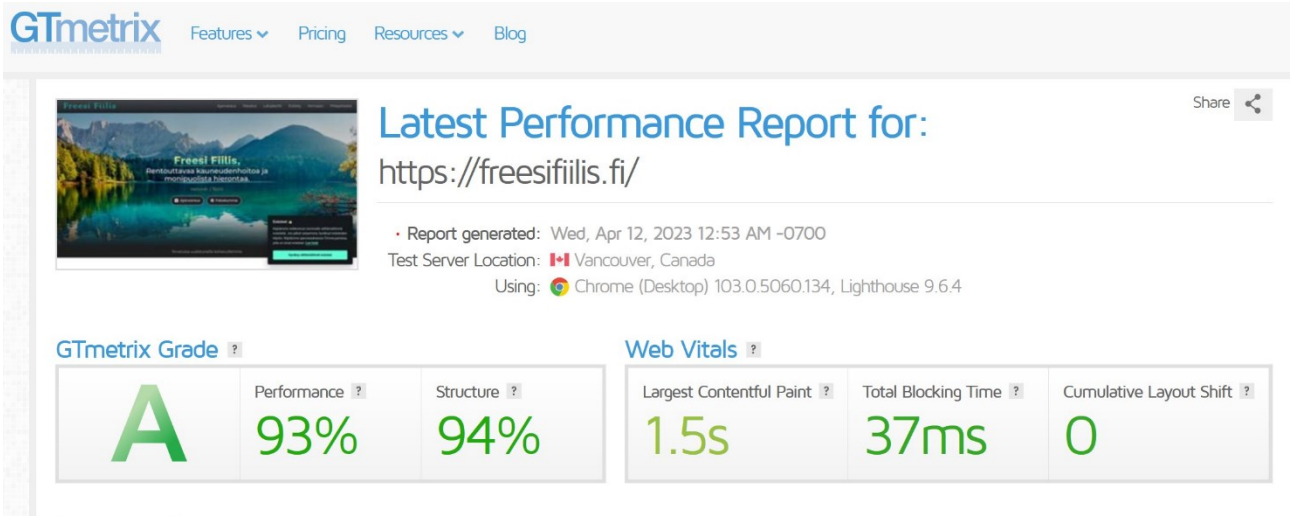
tarvittavien CDN-resurssien latautumisen optimoimiseksi kannattaa luoda esiyhteys niiden alkuperään **preconnect**-arvolla, jolloin puhutaan niin sanotuista resurssivinkeistä. Esimerkiksi kuvaa isännöivälle sivustolle voidaan muodostaa esiyhteys ja tämän jälkeen vielä määrittää sieltä ladattava kuva esilatautumaan. Nopeita CDN-palveluita voidaan hyödyntää muutenkin, koska ne saattavat parantaa suorituskykyä. Toiseksi on järkevä määrittää vähemmän tärkeät kuvat tai videot latautumaan vasta, kun lataaminen on tarpeellista. Edellä mainittu voidaan tehdä lisäämällä kuvaan viittaavaan HTML:n koodiin **loading**-attribuutin arvoksi ”**lazy**”, jolloin tiedostot latautuvat kokonaan vasta, kun niitä tarvitaan, eli silloin kun käyttäjä liikkuu mediaelementin kohdalle sivustolla.

Samalla on syytä miettiä latautumishierarkiaa kaikkien resurssien osalta, kuten tietoperustassa todettiin. Skriptit, joiden latautuminen ei ole aluksi tärkeää, voidaan määrittää latautuviksi myöhemmin esimerkiksi attribuuteilla **DEFER** ja **ASYNC**. Skriptit kannattaa asettaa HTML-koodiin vasta **<body>**-tagin loppuun, koska muuten ne estävät yleisesti sivuston ensimmäistä renderöintiä. Samoin CSS-määritysten latausjärjestys kannattaa suunnitella niiden tärkeyden perusteella. Esimerkiksi välittömästi tarvittavat CSS-tyylit voitaisiin lisätä suoraan HTML-dokumentin <head>-osioon sisäisinä tyylitietoina, jolloin ne ladataan heti. Vähemmän kiireelliset tyylitiedot voidaan sen sijaan jättää latautumaan myöhemmin erillisistä CSS-tiedostoista. Lighthouse huomauttaa aiheeseen liittyvästä ongelmakokonaisuudesta niin sanottuina renderöinnin estävinä resursseina, jotka tulisi eliminoida. Sain edellä mainituilla muutoksilla testituloksen nousemaan 75–85 pisteeseen mobiilitestissä.

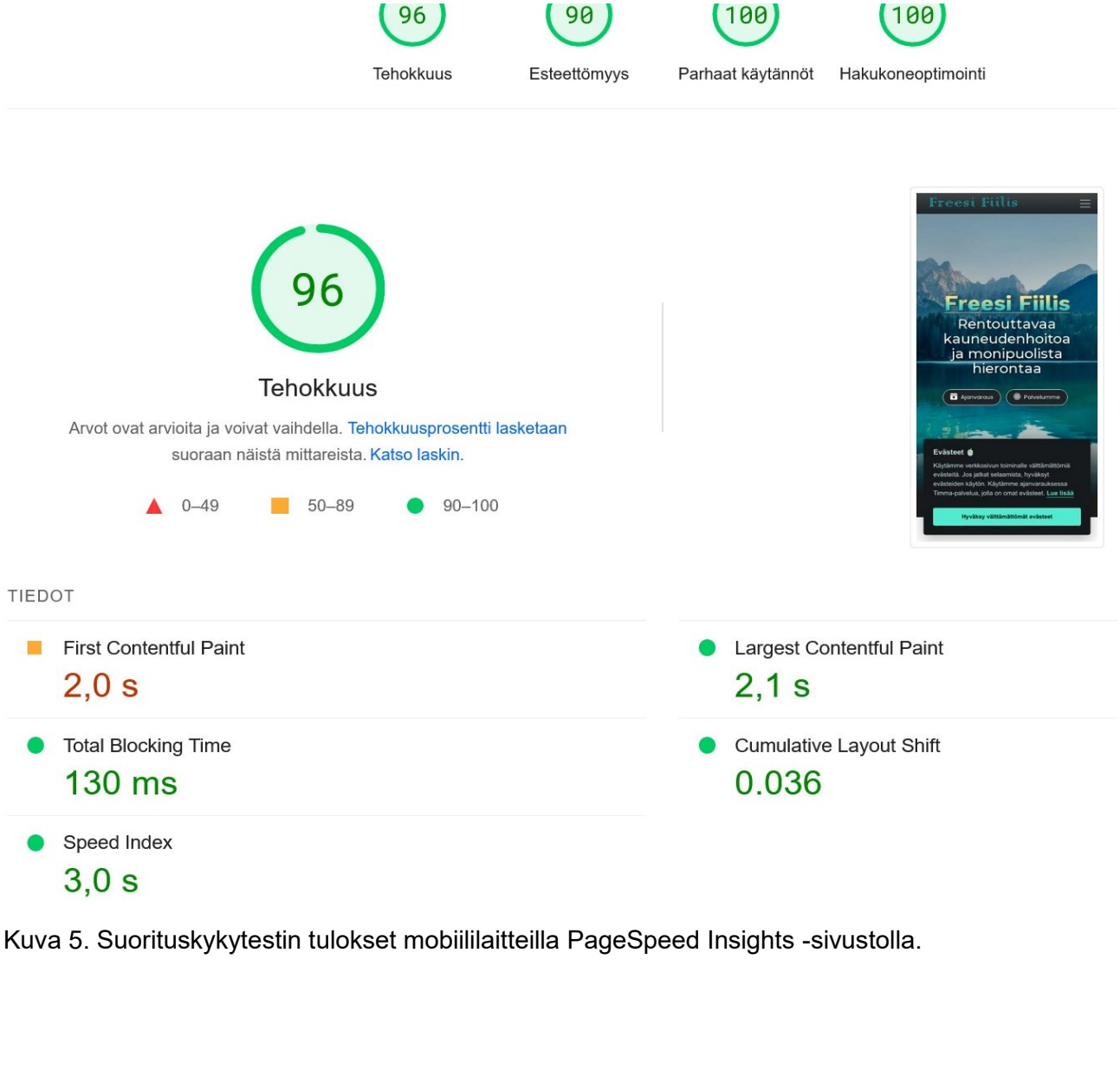
Ihmettelin tässä vaiheessa, että miten käyttämäni Google Fonts -resurssi näyttäytyi erityisesti renderöintiä estävänä. Tutkin asiaa ja tämä on ilmeisesti melko yleinen ongelma. Selaimet kohtelevat oletuksena kaikkea HTML-sivun <head>-osiosta löytämäänsä renderöintiä estävinä, mutta sen sijaan <body>-osion alimmaisena kuvia, mediatiedostoja ja script-tageja käsitellään ei-estävinä resursseina (Monus 14.1.2022). Ongelman korjaamiseen on muutama tapa. Yksi tapa on ladata fontit paikallisesti (Monus 14.1.2022). Toinen tapa on ladata font-CSS asynkronisesti ja lisäämällä sekä **preload** että **preconnect** nimiset resurssivinkit, jotta fontit voidaan ennakkoladata (Brandon s.a).

Lopullisessa toteutuksessa, ratkaistakseni aiemmin mainitsemani ongelman, käytin hieman muunneltua ratkaisua. Ennakkolatauksen lisäksi muutin CDN-linkin kohderesurssin (Google Fonts) latausmäärityksiä käyttäen **preload**-arvoa ja määrittämällä **as**-attribuutin arvoksi **style**. Tämän jälkeen fonttien CSS-tyylit ladattiin asynkronisesti. Tätä varten loin erillisen skriptin, jossa muutin rel-attribuutin arvoa **preload**-arvosta **stylesheet**-arvoksi. Ratkaisu oli tarpeellinen turvallisuusoptimoinnin näkökulmasta. Joka tapauksessa kyseinen muutos vaikutti siihen, että lopulta suorituskyvyn osalta testituloksien pisteet nousivat 90–100 pisteeseen Lighthousella ja tarkemmin vielä

PageSpeed Insightisilla. Seuraavat kuvakaappaukset kertovat PageSpeed Insights ja GTmetrix -sivustojen testituloksista kehitystyön lopputestausvaiheessa.



Kuva 4. Suorituskykytestin tulokset tietokoneilla GTmetrix-sivustolla.



Kuva 5. Suorituskykytestin tulokset mobiililaitteilla PageSpeed Insights -sivustolla.

3.7.4 Turvallisuudesta

Turvallisuus on tärkeä optimoinnin osa-alue ja kävin prosessin läpi kehitystyössä. Jouduin kuitenkin opinnäytetyöhön tekemään rajauksia, jonka vuoksi käsittelen aihetta suppeasti.

Hakkeroitujen verkkosivujen vierailijoille voidaan levittää haittaohjelmia, mikä voi vahingoittaa pysyvästi yrityksen asiakassuhteita. Lisäksi hakukoneet rankaisevat turvattomia sivustoja, mikä vaikuttaa SEO-sijoitukseen. Sivuston joutumisesta kyberhyökkäyksen kohteeksi voi koitua merkittäviä mainehaittoja. (Talalaev 4.3.2021.)

Hyviä sivuston turvallisuuden peruskäytäntöjä on pitää sivustoon liittyvät ohjelmistot ja liitännäiset päivitettyinä ja luoda säännöllisiä varmuuskopioita sivustosta. Päivitykset usein sisältävät turvallisuusparannuksia ja haavoittuvuuksien korjauksia. Sivuston elintärkeät toiminnot, kuten sivuston hallinta ja mahdollisten käyttäjätilien hallinta tulisi suojata vahvalla autentikoinnilla. Serveritasolla tulisi huolehtia siitä, että web-hotelli on luotettava ja tärkeät turvallisuuteen liittyvät määritykset pysyvät ajan tasalla. Palvelimen asetuksiin kirjautuminen tulee suojata kunnolla. (Hendricks s.a.) Lisäksi on hankittava SSL/TLS-varmenne HTTPS-salausta varten, kuten tietoperustassa todettiin.

Aloitin optimoinnin tekemällä alkuperäiselle sivustolle muutamia turvallisuusauditointeja, kuten observatory.mozilla.org sivustolla. Testien perusteella lähdin korjaamaan puutteita ennaltaehkäisevästi uudistetuille kotisivuille ja viimeistelin kuntoon edellä mainittuja peruskäytäntöjä. Tein testejä uudelleen ennen ja jälkeen julkaisun sekä hioin mahdollisia puutteita.

Korjasin puutteita lisäämällä tietoperustassa mainittuja HTTP-otsikoita, kuten HSTS-, X-Frame Options- ja X-XSS-Protection-otsikon. Lisäsin Content Security Policy -otsikon <meta>-tagissa, jotta luvaton koodin syöttö olisi vaikeampaa. Sanitoin varmuuden vuoksi käyttäjäsyötteet, vaikka CouchCMS hoitaa validoinnin ja HTML-sanitoinnin lomakkeissa. CouchCMS ilmeisesti luo ja käsittelee myös CSRF-tokenit ja validoi ne taustahallinnassa (Webzyme Softwares 2017c). Yritin varmuuden vuoksi generoida erilliset CSRF-tokenit ja luoda oman validoinnin, mutta metodi ei toiminut muuta kuin kirjautuneena pääkäyttäjäksi. Lopuksi tarkistin, että sivuston käyttämät resurssit sisälsivät SRI-tiivisteen ja loin puuttuville tiedostoille ja resursseille omat uniikit tiivisteet.

4 Loppupohdinta

Mielestäni saavutin asettamani tavoitteet ja toimeksiantaja sai tarpeet täyttävän tuotoksen. Tein lopulta suuren työn sen eteen, että yksinkertaisesta sivustosta saatiin optimoidut ja onnistuneesti julkaistut verkkosivut. Kannoin projektin läpi itsenäisesti alusta loppuun ja näin paljon vaivaa eri kehitystyön osa-alueilla. Halusin oppia projektin aikana mahdollisimman laajasti, jotta kokisin kehittyväni ammatillisesti. Opin paljon, mutta samalla ymmärsin, kuinka loputonta oppiminen on. Koin tärkeäksi oivaltaa, että kykenen oppimaan adaptiivisesti, mukautumaan ja kehittymään IT-työssä.

Ymmärrän paremmin, mitä kaikkea laadukkaiden modernien verkkosivujen kehittäminen alusta loppuun asti vaatii. Vaikka tekisin työtä huomattavasti tehokkaammin nykykokemuksella, jokaisen osa-alueen hiominen on työlästä. Monella osa-alueella olisi edelleen parannettavaa. Esimerkiksi sivusto ei ole ulkoasultaan erityinen tai teknisesti monimutkainen, mutta olen kuitenkin käyttänyt uusia teknologioita ja miettinyt toteutusta modernien vaatimusten mukaisesti. Jouduin opettelemaan oikeastaan kaikilla osa-alueilla uusia asioita, eikä minulla ollut juurikaan kokemusta taustahallinnasta, enkä ollut aiemmin julkaissut verkkosivustoa siirtämällä sitä webhotelliin.

Kehityin mielestäni jokaisella osa-alueella, mutta koin oppimispolkuni kokonaisuutena pintaraapaisuksi web-kehitykseen, koska jokaisen osa-alueen hallitseminen yksin oli vaativa tehtävä. Toki projekti oli suhteellisen pieni ja siksi hallittavissa. Kehitystyön aikana ja sen jälkeen sain jatkuvasti uusia oivalluksia webkehitykseen ja optimointiin liittyen. Jatkuva oppiminen on osa työtä ja voin edelleen tehdä sivustolle korjauksia ja parannuksia.

Julkaisuvaiheen lähestyessä huomasin jännittäväni ja stressaavani enemmän, että onnistunko lopulta tavoitteissa. Muutaman rankan työpäivän tai -viikon jälkeen, kun suurin optimointityö oli tehty ja sivusto julkaistu, paine vähitellen keveni harteiltani. Seurasin sivuston toimivuutta, tein muutamia korjauksia ja aloitin hakukoneoptimoinnin, mutta tärkeintä oli nähdä, että sivusto toimi oikein ja oli käytettävissä. Tavoitteeni oli saada näkyväksi se, että olin onnistunut ennen kaikkea suorituskyky- ja hakukoneoptimoinnissa. Tulokset näillä osa-alueilla puhuivat puolestaan ja olin tyytyväinen, vaikka hienosäädettävää ja paranneltavaa voitaisiin löytää loputtomasti lisää.

Oli olemassa useita vaihtoehtoisia toteutustapoja, joilla olisin voinut luoda sivuston ja vastata toimeksiantajan tarpeisiin. Valintani eri teknologioiden ja toteutustapojen välillä ei ollut perustunut niiden paremmuuden vertailuun, vaan pikemminkin mitkä ratkaisut palvelisivat parhaiten oppimistavoitteitani. Halusin sopivan monipuolisen läpileikkauksen web-kehitykseen, enkä kokenut WordPressin tarjoavan samanlaista oppimiskokemusta. Tällä tavoin kykenin suunnittelemaan ja täyttämään toimeksiantajan vaatimukset joustavasti. Sivustoratkaisusta muodostui kompakti kokonaisuus, jonka kaikki suunnitellut ominaisuudet ovat käytössä. Ratkaisu on kevyt ilman

tarpeettomia lisäelementtejä tai turhaa painolastia. Tärkeintä oli, että toimeksiantaja ilmaisi olevansa tyytyväinen lopputulokseen. Uusi verkkosivusto on toimeksiantajan mukaan tuonut yritykselle lisää asiakkaita.

4.1 Tutkimusetiikka

Kohdensin toiminnalliseen opinnäytetyöhön liittyviä tutkimuseettisiä periaatteita ja pyrin säilyttämään työn tieteellisen eheyden. Tarkoitukseni oli koota korkealaatuista tietoa tietoperustaan, jota hyödynsin myöhemmin työn empiirisessä osuudessa. Valitsin aineistoni lähdekritiikkiä noudattaen ja pyrin tukemaan työssäni esittämiäni argumentteja useilla eri lähteillä tarpeen mukaan.

Kohdatessani haasteita jälkiasennettavan CMS-järjestelmän suhteen, jossa kattavan ja laadukkaan tiedon löytäminen oli haastavaa, jouduin luovemmin yhdistelemään tietoa eri lähteistä. Tämä tarkoitti verkkokeskustelujen seuraamista ja CMS-järjestelmien omien dokumentaatioiden hyödyntämistä. Tällä tavoin paransin esittämäni tiedon luotettavuutta ja relevanssia.

Kirjoitustyön alkuvaiheessa raportointini muodostui pääosin vapaamuotoisesta itsereflektiosta, jonka koin tukevan oppimistani. Kirjoitustyön edetessä loppuvaiheisiin editoin tekstiäni saamani palautteen perusteella huomioiden paremmin tieteellisen esitystavan vaatimukset, kuten asiakielisyyden ja lähdemateriaalin asianmukaisen käytön. Paransin tieteellisen ja käytännöllisen näkökulman vuoropuhelua, mutta pidin tärkeänä oman, henkilökohtaisen ääneni säilyttämistä kehitystyöni raportoinnissa. Tavoitteenani oli luoda autenttisempi kuva työprosessistani.

Projektin alusta lähtien kävin toimeksiantajan kanssa säännöllisesti läpi projektin etenemistä, ja varmistin, että raportoinnissani otettiin huomioon toimeksiantajan näkökulmat ja kokemukset. Arvostin toimeksiantajan mielipiteitä ja päätöksiä, ja olin varovainen, etten loukkaisi ketään tai esittäisi ajattelemattomia mielipiteitä. Työskentelyni oli läpinäkyvää, ja tein avoimesti näkyväksi tekemäni päätökset ja työskentelyn edistymisen. Pyrin viestimään näistä asioista selkeästi ja ymmärrettävästi, jotta toimeksiantaja pysyi ajan tasalla projektin etenemisestä.

4.2 Yhteenveto omasta oppimisesta ja tuotoksen hyödynnettävyydestä

Olin luottavainen, että osaamiseni olisi riittävä toimeksiantajan tarpeiden ja vaatimusten mukaiseen verkkosivuston toteutukseen ja julkaisuun. Tiesin projektin alussa, että edessäni oli paljon uutta opittavaa, mutta olen tulkinnut yhdeksi vahvuudekseni kyvyn oppia työn ohessa. Tämän projektin kautta sain konkreettista kokemusta selainpuolen ja taustahallinnan kehityksestä, mukauttavasta web-suunnittelusta ja projektinhallinnasta.

Vaikka projektin laajuus mahdollisti monipuolisen oppimisen, en voinut omaksua kaikkea syvällisesti projektin puitteissa. Tästä syystä kuvaan tätä kokemusta pikemminkin pintaraapaisuna web-kehityksen maailmaan, mikä ei kuitenkaan vähennä sen antamaa arvokasta kokemusta ja tietämystä. Opinnäytetyöprosessi oli minulle merkittävä, mutta ajoittain työläs ja haastava askel ammatillisessa kehityksessäni. Prosessi ei ainoastaan laajentanut itsenäistä teknologista osaamistani, vaan myös kehitti kykyjäni hallita monimutkaisia projekteja. Projekti piti sisällään tehtäviä, jotka vaativat rohkeutta hypätä tuntemattomaan ja vastata tehtävistä, joiden mahdollinen epäonnistuminen painoi mieltäni.

Itseluottamukseni ja motivaationi pysyivät suhteellisen vakaina koko projektin ajan, vaikka kohtasinkin hetkiä, jolloin edessäni oli suurempia haasteita. Tällaisia haasteita ilmeni esimerkiksi lahjakorttitilauksen kehittämisen, CMS-ratkaisun integroinnin ja sivuston julkaisun yhteydessä. Eteneminen pysähtyi ajoittain ja uusien ratkaisujen keksiminen vaati aikaa, mutta en kokenut minkään kehitystyön osa-alueen lopulta tuntuneen ylivoimaiselta haasteelta. Näistä kokemuksista huolimatta olen kuitenkin tyytyväinen saavutettuun lopputulokseen ja uskon, että nämä haasteet ovat edistäneet ammatillista kasvuani. Arvelen, että minulla on nyt paremmat valmiudet omaksua uusia teknologioita ja kehittää nykyistä osaamistani edelleen.

Projektin aikana pyrin omaksumaan ketteriä menetelmiä projektinhallintaani, jossa olen huomannut enemmän kehittämiskohtia. Tunnistan, että osa näistä puutteista johtuu kokemattomuudestani IT-alan projektityössä ja menetelmäosaamisen taidoissa. Vaikka projektinhallintani oli joustavaa, se ei aina tuntunut täysin hallitulta. Ketterien menetelmien luonteeseen kuuluu, että projektin vaatimukset saattavat muuttua iteraatioiden myötä. Tämä vaatii jatkuvaa sopeutumista muuttuviin olosuhteisiin ja kehitysvaiheisiin. Tässä suhteessa projektinhallintani onnistui, sillä kykenin mukautumaan vaatimuksiin omalla tavallani. Kuitenkin ammattimaisen kehittymisen näkökulmasta näen, että tulevaisuudessa minun olisi hyödyllistä saada enemmän kokemusta tiimityöskentelystä. Tämä auttaisi minua ymmärtämään paremmin, miten ketterät menetelmät toimivat käytännössä ja miten niitä voidaan tehokkaasti hyödyntää osana tiimiprojekteja.

Pysyin kehitystyössä pääosin suunnitellussa aikataulussa, vaikka alkuperäinen aikataulu osoittautui liian optimistiseksi ja julkaisu siirtyi lopputalvesta kevääseen 2023. Aikataulun venyminen johtui paitsi työskentelyn aikavaatimuksista, myös toimeksiantajan tarpeesta lisäaikaan siirtymän valmistelussa. Opinnäytetyön kirjoittamisen aikataulu pysyi pitkälle kevääseen suunnitellussa, mutta seminaarin ja sen jälkeisen palautteen perusteella venytin kirjoitusprosessia noin kolmella viikolla, koska pyrin huolellisesti parantamaan raporttiani työskennellessäni samalla muiden päällekkäisten kiireiden keskellä.

Vaikka projekti oli full-stack-kehityskokonaisuus, suurin osa kehitystyöstä kohdistui selainpuolen toteutukseen. Yllättäen minun täytyi kuitenkin syventyä huomattavasti taustahallinnan opiskeluun, joka aluksi tuntui haasteelliselta, mutta projektin mittapuussa hallittavalta. Kykenin rakentamaan kaikki toimeksiantajan toivomat ominaisuudet. Lahjakorttiosio, CMS-järjestelmän jälkiasennus ja mukauttaminen sekä sivuston ulkoasun luominen olivat ehkäpä työläimmät osa-alueet optimoinnin ohella. Lahjakorttitilauksen tekemisestä muodostui yksinkertainen, mutta käytettävyydeltään onnistunut kokonaisuus, joka täytti sille asetetut vaatimukset ja tarpeet. Sivusto on mukautettu niin, että toimeksiantaja voi hallita sen sisältöä, mutta ei suuremmin muuttaa sivuston ulkoasua tai osioiden järjestystä. Tämä oli kuitenkin tiedostettu valinta, joka käytiin läpi vaatimusmäärittelyssä.

Toteutustapana päätteetön CMS toimi kimmokkeena sille, että ymmärsin voivani tehdä kehitystyön haluamillani teknologioilla, mutta minulta vei aikaa käsittää, etten tulisi tekemään varsinaisesti headless-ratkaisua, joka siis perustui rajapinta-arkkitehtuuriin. Pitkään opinnäytetyön kirjoittamisen aikana tulkitsin, että ajatus päätteettömyydestä kattaisi kaikki toteutustavat, joissa frontend-kehitystyö voitiin tehdä erikseen, ottamatta kantaa perustuuko sisällön esittäminen rajapinta-arkkitehtuuriin vai ei. Tulkitsin, että toteutustavassani voisi olla kyse päätteettömyydestä, koska rakensin sivuston käyttöliittymän erikseen, eli sivuston sisällön esitystapa ei ollut sidonnainen yhdestä ja samasta alustasta.

Opinnäytetyön aikana minun oli sanoitettava toteutustapani ymmärrettäväksi uudelleen itselleni ja muille. Erilaisia CMS-järjestelmiä on olemassa runsaasti ja jälkiasennettavuudesta löytyvän tiedon niukkuus vaikeutti prosessia. Kolutessani aihetta läpi eri keskustelupalstoilla ja verkkosivustoilla, tunsin välillä olevani syvällä kaninkolossa, sillä mitä enemmän syvennyin aiheeseen, sitä enemmän erilaisia CMS-järjestelmiä tuntui löytyvän. Pyrin erottelemaan jälkiasennettavalle CMS-ratkaisulle ominaisia erityispiirteitä ja sitä kautta tarjoamaan tietoa toteutustavasta, mikä voisi hyödyttää myös opinnäytetyön lukijoita.

Onnistuin melko hyvin tekemään sivustosta responsiivisen ja kirjoittamaan mediakyselyitä suhteellisen vähäisellä vaivalla. Tunnistan kuitenkin puutteita CSS:n ja mukauttavan web-suunnittelun osaamisessani. Pyrin paikkaamaan niitä tutustumalla opinnäytetyön aikana Bootstrap-viitekehyksen lisäksi syvällisemmin CSS:n asemointitekniikoihin, joustavaan mediaan ja mediakyselyihin. Käsittelin joustavaa mediaa työssäni suppeammin kuin muita aiheita työn rajauksen vuoksi.

Sivustolla on joitain parannettavia osa-alueita erityisesti CSS-määritysten ja sisällön dynaamisen uudelleenjärjestymisen näkökulmasta. Esimerkiksi Bootstrap-karusellikomponentin osalta ilmeni ongelmia. En alun perin ottanut huomioon, että vaikka karusellin kuvat mukautuvat näyttöikkunan kokoon, niin erikokoiset kuvat eivät mukaudu samalla tavalla. Karusellissa kuvat vaihtuvat automaattisesti, mutta jos kuvien kuvasuhteet ovat erilaisia, ne eivät mukaudu samalla tavalla toisiinsa

nähden kuvan vaihtuessa seuraavaan. Ongelman korjaamiseksi erikokoisten kuvien rajauksen pitää olla yhtenäinen, joka vaatii erityistä huomiota CSS:n avulla toteutettavassa mukauttamisessa.

En ehtinyt korjata edellä mainittua vikaa. Parannukset on hyvä tehdä ensin paikallisesti. Sivuston nopeisiin korjauksiin liittyy myös turvallisuushaaste, sillä jos koodia muutetaan skripteissä tai CSS-tiedostoissa, joilla on yksilöllinen SRI-tiiviste, tiiviste ei enää vastaa alkuperäistä, eivätkä skriptit tai CSS-tiedostot lataudu sivuston latauksen yhteydessä. Jokaisen muutoksen jälkeen SRI-tiivisteet on päivitettävä, mikä tekee korjaamisesta työläämpää. Aiheeseen liittyen olisin voinut parantaa sivuston pakkausta ja tiivistää tiedostoja enemmän, eli muuntaa minified-muotoon, mutta SRI-tiivisteen muuttumisen vuoksi minun on parempi tehdä mahdolliset muutokset ajan kanssa. Testeissä on ilmennyt myös muita pienempiä korjattavia asioita koodissa, jotka eivät merkittävästi vaikuta sivuston suorituskykyyn.

Sivuston visuaalinen ilme on suhteellisen yksinkertainen, eikä se sisällä erityisen ainutlaatuisia suunnitteluelementtejä. Tämä johtuu osittain aikarajoitteista, jotka estivät minua keskittymästä syvällisemmin graafisen ilmeen kehittämiseen. Toisaalta minimalistinen lähestymistapa saattaa olla hyödyllinen, sillä liian huomiota herättävät suunnitteluelementit voivat heikentää käyttäjäkokemusta ja mahdollisesti hidastaa sivun latausaikoja. Päätin tietoisesti pitäytyä yksinkertaisessa suunnittelussa tarjotakseni parhaan mahdollisen yhteensopivuuden eri selainten kanssa, sillä kaikki CSS-ominaisuudet ja efektit eivät välttämättä toimi kaikissa selaimissa. Toivoisin tulevaisuudessa kehittäväni graafisen suunnittelun taitojani paremmin hyödynnettäväksi webkehityksessä. Esitin yhtenä parannusehdotuksena toimeksiantajalle, että työntekijöiden valokuvat voisivat olla hyödyllinen lisä sivustolle yrityksen esittelyosiossa, mutta tätä ei päädytty toteuttaa projektin aikana.

Alkuperäinen tavoitteeni opinnäytetyön kirjoittamiselle oli kerätä käyttökelpoista tietoa web-kehityksestä suomen kielellä. Työn rajaus tuntui alussa hankalalta, koska kehityksen kenttä on laaja, enkä ollut täysin varma siitä, mitkä aiheet olisivat riittävän kiinnostavia ja hyödyllisiä sekä itselleni että muille. Työn kirjoittamisen edetessä huomasin raportin sisällön tarkentuvan vähitellen ja osa alkuperäisistä ajatuksistani kehittyi eri suuntiin kuin olin alun perin suunnitellut. Tämä johtui siitä, että ymmärrykseni aiheesta syveni ja muuttui prosessin aikana. Esimerkiksi alkuperäiset ajatukseni kirjoittaa responsiivisesta websuunnittelusta sulautuivat osaksi optimointiteemaa ja päätteetön CMS-järjestelmä muuntui jälkikäteen asennettavaksi toteutustavaksi, kun ymmärrykseni aiheesta kehittyi.

Kirjoitusprosessin suunnitteluvaiheessa olin kunnianhimoinen siinä mielessä, että halusin opinnäytetyössäni koota ja jakaa laadukasta, oppimista edistävää tietoa verkkosivuston kehittämisestä. Tavoitteeni oli samalla kuvata yksityiskohtaisesti omaa toteutustapaani ja sitoa tietoperustani siihen. Olisin voinut keskittyä vain yhteen optimointiin liittyvään osa-alueeseen, mutta halusin antaa

kokonaisvaltaisen kuvauksen oppimispolustani, joka kattaa kaiken CMS-toteutuksesta verkkosivuston optimointiin. Opinnäytetyöni antaa muille web-kehittäjille yleiskuvan käyttämästäni toteutustavasta ja tarjoaa ohjeistuksia, jotka ovat hyödyllisiä verkkosivujen kehitystyössä. Raporttini ei ehkä tarjoa syvällistä opetusmateriaalia optimoinnista tai CMS:n asentamisesta, mutta se tarjoaa yleisluontoista tietoa hyvistä käytännöistä. Tämä tieto voi toimia lähtökohtana syvemmälle tutustumiselle aiheisiin seikkaperäisten oppaiden kautta.

Sivuston optimointi on jatkuvan parantamisen osa-alue. Optimointitarpeita voi ilmetä, kun sivustoa muokataan tai päivitetään. Uusien elementtien lisääminen sivustolle saattaa vaikuttaa sivuston suorituskykyyn. Optimoinnissa tekemäni työ oli mielestäni laadukasta, mutta jälkikäteen ymmärrykseni aiheesta on syventynyt ja olen oppinut uusia menetelmiä, joilla sivustojen suorituskykyä voisi edelleen parantaa. Myös hakukoneoptimointi saavutti hyvän lähtötason. Sivusto latautuu nopeasti, keskimäärin kahdessa tai kolmessa sekunnissa, mutta sivustoon tehdyt muutokset julkaisun jälkeen ovat ajoittain heikentäneet suorituskykyä. Optimoinnin vaikutus sivuston käyttäjäkokemukseen ja suorituskykyyn oli selvästi havaittavissa, mikä vahvisti työn tarpeellisuuden. Vaikka uskon saavuttaneeni optimoinnissa hyvän tuloksen, olen tietoinen siitä, että työtä voisi edelleen kehittää.

Toimeksiantaja sai projektista sekä taloudellista hyötyä että parannusta yritysilmeeseen. Optimoinnin myötä sivuston sijoittuminen hakutuloksissa parani huomattavasti aiempaan nähden, mikä voi lisätä asiakaskäyntejä. Vaikka raporttini ei ole suoranainen opas, uskon sen tarjoavan lukijalle perustietoa verkkosivuston kehitysmenetelmistä, jälkiasennettavasta sisällönhallintajärjestelmästä ja sivuston optimoinnista. Projektin myötä minulle tarjoutui luonteva tapa opiskella verkkokehitystä, mikä paransi mahdollisuuksiani kehittää ammatillista osaamistani entisestään. Saavutin projektissa lisää itsevarmuutta itsenäiseen työskentelyyn ja rohkeutta kohdata haasteita, minkä uskon hyödyttävän minua tulevaisuudessa.

Lähteet

ahadb (nimimerkki) 2022. Flat file CMS systems. Github. Readme-kirjoitus. Luettavissa: https://github.com/ahadb/flat-file-cms. Luettu: 6.6.2023.

Afif T. 2021. Responsive Layouts, Fewer Media Queries. CSS-tricks. DigitalOcean. Luettavissa: https://css-tricks.com/responsive-layouts-fewer-media-queries/. Luettu: 6.6.2023.

Maria, A. 21.9.2017. Couch CMS and the lazy developer. Dev Community-yhteisön jäsenen blogi. Luettavissa: https://dev.to/antoinette0x53/couch-cms-and-the-lazy-developer. Luettu: 3.3.2023.

Arola, N. 23.3.2022. Miksi valita Headless CMS? Hybriditoimisto Nitron blogi. Luettavissa: https://www.nitro.fi/matkapaivakirja/miksi-valita-headless-cms. Luettu: 18.4.2023

Bahajji, Z. & Illyes, G. 7.8.2014. HTTPS as a ranking signal. Google kehittäjien blogi. Luettavissa: https://developers.google.com/search/blog/2014/08/https-as-ranking-signal.
Luettu: 16.4.2023.

Baumgartner, S. 17.2.2023. HTTP/2 - What Does It Mean for a CDN? Keycdn-sivuston blogi. Luettavissa: https://www.keycdn.com/blog/http2-cdn. Luettu: 20.5.2023.

Bece, A. 1.10.2019. Optimizing images for the web - an in-depth guide. Dev Community. Prototyp-yrityksen blogi. Luettavissa: https://dev.to/prototyp/optimizing-images-for-the-web-an-in-depth-guide-4j7d. Luettu: 20.5.2023.

Bootstrap team s.a. Breakpoints. Getbootstrap.com docs. Luettavissa: https://getbootstrap.com/docs/5.0/layout/breakpoints/. Luettu: 25.5.2023.

Boyarko, E. 2.7.2021. Headless or Traditional CMS? Schemes, Use Cases, Pros and Cons. Upsilon-yrityksen blogi. Luettavissa: https://www.upsilonit.com/blog/headless-or-traditional-cms-schemes-use-cases-pros-and-cons. Luettu: 23.4.2023.

Brandon, D. 19.9.2019. Website Templates – The Definitive Guide. WebsiteBuilderInsider-sivuston blogi. Luettavissa: https://www.websitebuilderinsider.com/website-templates/. Luettu: 8.6.2023.

Brandon (nimimerkki) s.a. Get Up To Speed With Website Performance Optimization. Luettavissa: https://pagespeedchecklist.com/. Luettu: 16.4.2023

Camarena, A. 7.3.2023. What Is Page Speed & How to Improve It. Semrush-sivuston blogi. Luettavissa: https://www.semrush.com/blog/page-speed/. Luettu: 17.4.2023.

Christensson, P. 2015. Fluid layout. Techterms.com, fluid layout definition. Luettavissa: https://techterms.com/definition/fluid_layout. Luettu: 16.6.2023.

Chrome Developers 2023. Lighthouse performance scoring. Luettavissa: https://developer.chrome.com/en/docs/lighthouse/performance/performance-scoring. Luettu: 30.5.2023.

Chrome Developers 2022. Overview. Lighthouse. Google. Luettavissa: https://developer.chrome.com/docs/lighthouse/overview/. Luettu: 14.6.2023.

Cloudflare 2023. Why does site speed matter? | Improve webpage speed. Cloudflare. Luettavissa: https://www.cloudflare.com/learning/performance/why-site-speed-matters. Luettu: 29.5.2023.

Cox, L. 21.7.2021. Coding for Web Design 101: How HTML, CSS, and JavaScript Work. Hubspot-sivuston blogi. Luettavissa: https://blog.hubspot.com/marketing/web-design-html-css-javascript. Luettu: 25.5.2023.

Darji, P. 2023. Would you like to learn more about Headless CMS vs Traditional CMS? Syndel-yrityksen webkehitysaiheinen blogi. Luettavissa: https://syndelltech.com/headless-cms-vs-traditional-cms. Luettu: 15.6.2023.

Digital Marketing Institute 18.10.2021. What Is SEO & Why Is It Important? Digital Marketing Institute -sivuston blogi. Luettavissa: https://digitalmarketinginstitute.com/blog/what-is-seo-and-why-is-it-important. Luettu: 20.5.2023.

Diaz, C. 2022. A Guide To Hover And Pointer Media Queries. Luettavissa: https://www.smashingmagazine.com/2022/03/guide-hover-pointer-media-queries. Luettu: 10.6.2023.

Frain, B. 2022. Responsive Web Design with HTML5 and CSS: Build Future-proof Responsive Websites Using the Latest HTML5 and CSS Techniques. Packt Publishing. E-kirja. Luettu: 23.3.2023.

Geeksforgeeks.com 2023. Bootstrap 5. Luettavissa: https://www.geeksforgeeks.org/bootstrap-5/. Luettu: 20.4.2023.

Google s.a. Paranna tehokkuuttasi Google Haussa. Search Console. Luettavissa: https://search.google.com/search-console/about. Luettu: 14.6.2023.

Goyal, A. 17.8.2021. CSS Grid vs. Bootstrap: All You Need To Know. Lambdatest. Web-kehitys blogi. Luettavissa: https://www.lambdatest.com/blog/css-grid-vs-bootstrap. Luettu: 25.5.2023.

Grabaperch.com s.a.(a). Features. Perch CMS. Luettavissa: https://grabaperch.com/features. Luettu: 5.6.2023.

Grabaperch.com s.a.(b). Template reference. Perch. Luettavissa: http://docs.grabaperch.com/templates/. Luettu: 5.6.2023.

Hannon Hill 2023. Templates.Cascade CMS Knowledge base. Luettavissa: https://www.hannonhill.com/cascadecms/latest/design-in-cascade/templates/index.html. Luettu: 8.6.2023

Heitzman, A. 18.4.2022. Single-Page Websites: Are They Good Or Bad For SEO? Search engine journal -sivuston blogi. Luettavissa: https://www.searchenginejournal.com/single-page-websites-good-bad-seo/190095/. Luettu: 29.5.2023

Hempenius, K. 2022. Serve responsive images. Web.dev. Luettavissa: https://web.dev/serve-responsive-images. Luettu: 8.5.2023.

Hendricks, D. s.a. 10 Essential Steps To Improve Your Website Security. IEEE Computer Society -seuran blogi. Luettavissa: https://www.computer.org/publications/tech-news/trends/10-essential-steps-to-improve-your-website-security. Luettu: 15.6.2023.

Hlforr (nimimerkki) 2022: CMS to retrofit into existing website. Avaus viestiketjuun sitepoint-sivuston keskustelupalstalla. Luettavissa: https://www.sitepoint.com/community/t/cms-to-retrofit-into-existing-website/382854. Viitattu: 6.6.2023.

Hughes, J. 23.3.2023. What Is Headless WordPress (And Do You Need It)? Elegant themes -sivuston blogi. Luettavissa. https://www.elegantthemes.com/blog/wordpress/headless-wordpress. Luettu: 5.6.2023.

Jackson, B. 30.9.2022. Resource Hints - What is Preload, Prefetch, and Preconnect? Keycdn-sivuston blogi. Luettavissa: https://www.keycdn.com/blog/resource-hints. Luettu: 20.5.2023.

Jackson, B. 6.2.2018. CDN Hosting vs Traditional Web Hosting. Keycdn-sivuston blogi. Luettavissa: https://www.keycdn.com/blog/content-delivery-networks. Luettu: 20.5.2023.

jimii (nimimerkki) 2014: "Best" CMS to integrate into existing sites? Avaus viestiketjussa Reddit-sivustolla. u/webdev. Luettavissa: https://www.reddit.com/r/webdev/comments/27wbbi/best_cms_to_integrate_into_existing_sites. Viitattu: 6.6.2023.

justbooleandawg (nimimerkki) 2023: Client wants CMS for simple site, what are my options for these specs requested? Avaus viestiketjussa Reddit-sivustolla. u/webdev. Luettavissa:

https://www.reddit.com/r/webdev/comments/102bdzn/client_wants_cms_for_simple_site_what_are_my. Viitattu: 6.6.2023.

Juviler, J. 2.11.2021. What Is Fluid Design and How Is It Used on Websites? Hubspot-sivuston blogi. Luettavissa: https://blog.hubspot.com/website/fluid-design. Luettu: 14.4.2023.

Kosaka, M. 20.9.2018. Inside look at modern web (part 3). Chrome Developers. Luettavissa: https://developer.chrome.com/blog/inside-browser-part3/ Luettu: 15.6.2023.

KirstenS (nimimerkki). 2023. Cross Site Request Forgery (CSRF). OWASP. Luettavissa: https://owasp.org/www-community/attacks/csrf. Luettu: 26.4.2023

Kitamura, E., Nalpas M. & Janc, A. 2022. Security headers quick reference. Web.dev. Luettavissa: https://web.dev/security-headers/. Luettu: 24.5.2023

Korhonen, P. 10.1.2018. Pieni API-sanakirja. CGI:n työntekijän blogi. Luettavissa: https://www.cgi.com/fi/fi/blogi/pieni-api-sanakirja. Luettu: 6.6.2023.

Kaismala, T. 2022. Mikä on HTTPS - kotisivut suojaan hakkereilta. Arvomedia. Luettavissa: https://www.arvomedia.fi/ota-https-salaus-heti-kayttoon. Luettu: 16.4.2023.

Kontent.ai 2023. How to choose a headless CMS for your organization. Kontent.ai Headless CMS. Luettavissa: https://kontent.ai/how-to-choose-headless-cms. Luettu: 18.4.2023.

Koski, J. s.a. Ketterät menetelmät, agile, LEAN ja scrum. Itewiki. Luettavissa: https://www.itewiki.fi/opas/ketterat-menetelmat-agile-lean-ja-scrum. Luettu: 18.4.2023.

Kumar, P. 2022. What kind of Testing is Important in Responsive Design? Browserstack. Luettavissa: https://www.browserstack.com/guide/testing-responsive-design. Luettu: 25.5.2023.

Leffingwell, D. 2010. Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise. Addison-Wesley Professional. E-kirja. Luettu: 20.4.2023.

LePage, P. & Andrew, R. 2019. Responsive Web Design Basics. Luettavissa: https://web.dev/responsive-web-design-basics. Luettu: 24.4.2023.

Lilja, S. 2021. Vaatimusmäärittelyn rooli ketterässä ohjelmistokehityksessä. Ammattikorkeakoulututkinnon opinnäytetyö. Hämeen Ammattikorkeakoulu. Tietojenkäsittelyn koulutus, Hämeenlinnan korkeakoulukeskus. Luettavissa: https://www.theseus.fi/bitstream/handle/10024/505152/Lilja_Sanna.pdf?sequence=2&isAllowed=y. Luettu: 20.4.2023

Maulidina, M. 2023. What Is a WordPress Theme: Different Types and How to Choose One. Hostinger. Luettavissa: https://www.hostinger.com/tutorials/what-is-a-wordpress-theme. Luettu: 7.6.2023.

Martyr2 (nimimerkki) 2022: CMS to retrofit into existing website. Vastaus viestiketjun avaajan viestiin sitepoint-sivustolla. Luettavissa: https://www.sitepoint.com/community/t/cms-to-retrofit-into-existing-website/382854. Viitattu: 6.6.2023

Meelkor (nimimerkki) 2014: Looking for lightweight CMS with only Admin interface and backend library. Avaus viestiketjussa Reddit-sivustolla. r/PHP. Luettavissa: https://www.reddit.com/r/PHP/comments/26wy3g/looking_for_lightweight_cms_with_only_admin. Viitattu: 6.7.2023.

Meshen, L. 2021. CMS Architecture: Traditional, decoupled & headless. Acro Media e-artikkeli. Luettavissa: https://www.acromedia.com/article/cms-architecture-traditional-decoupled-headless. Luettu: 6.6.2023.

Monus, A. 14.1.2022. 9 tricks to eliminate render blocking resources. LogRocket-sivuston blogi. Luettavissa: https://blog.logrocket.com/9-tricks-eliminate-render-blocking-resources/. Luettu: 1.6.2023.

Mozilla Corporation 2023a. Flexbox. MDN web docs. MDN contributors. Luettavissa: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Flexbox. Luettu. 15.4.2023.

Mozilla Corporation 2023b. Using media queris, by Developers. MDN web docs. MDN contributors. Luettavissa https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_media_queries/Using_media_queries. Luettu: 14.4.2023.

Mozilla Corporation 2023c. Website security. MDN web docs. MDN contributors. Luettavissa: https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Website_security. Luettu: 26.4.2023.

Muranen, A. 15.6.2021. Headless CMS -ratkaisulla päihität kilpailijat verkossa. Pixels-nimisen yrityksen blogi. Luettavissa: https://pixels.fi/fi/blogi/headless-cms-ratkaisulla-paihitat-kilpailijat-verkossa. Luettu: 18.4.2023

Nanz, K. 30.10.2021. You Probably Don't Need Media Queries Anymore. Dev Community-yhteisön jäsenen blogi. Luettavissa: https://dev.to/kathryngrayson/you-probably-dont-need-media-queries-anymore-a4j. Luettu: 14.4.2023.

Nixon, R. 2014. Learning PHP, MySQL, JavaScript, CSS & HTML5, 3rd Edition. O'Reilly Media, Inc. E-kirja. Luettu: 7.6.2023.

OWASP 2021a. Cross Site Scripting Prevention Cheat Sheet. OWASP Cheatsheetseries. Luettavissa: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html. Luettu: 26.4.2023

OWASP 2021b. Cross Site Reques Forgery Prevention. OWASP Cheatsheetseries. Luettavissa: https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html. Luettu: 26.4.2023

Pavlik, V. 27.3.2023. How to Learn SEO (Basics, Tips & Free Resources). Semrush-sivuston blogi. Luettavissa: https://www.semrush.com/blog/learn-seo/. Luettu: 16.4.2023

Poirier, H. 2022. When to use a Headless CMS. Agility. Luettavissa: https://agilitycms.com/resources/posts/when-to-use-headless-cms. Luettu: 18.4.2023

Rajora, H. 17.6.2021. CSS Flexbox Tutorial: A Complete Guide. Lambdatest-sivuston webkehitys aiheinen blogi. Luettavissa: https://www.lambdatest.com/blog/css-flexbox-tutorial. Luettu: 26.5.2023.

Raza, A. 14.6.2022. Why mobile speed is lower on Google Page Speed Insights results than on the actual device. Level Up Coding -blogi. Luettavissa: https://levelup.gitconnected.com/google-pagespeed-insights-is-broken-for-mobile-mode-causes-and-way-forward-a9cb967a735d. Luettu: 16.6.2023.

Snell, C. 2021. Google Lighthouse Scores: How Important Are They, Really? Digital Impulse. Luettavissa: https://www.digitalimpulse.com/heartbeat/google-lighthouse-scores-how-important-are-they-really. Luettu: 1.6.2023.

Silva, C. 14.12.2022. SEO Checklist: 41 Tips to Optimize Your Website. Semrush-sivuston blogi. Luettavissa: https://www.semrush.com/blog/seo-checklist/. Luettu: 16.4.2023.

Siteimprove 2023. Website Optimization. Luettavissa: https://www.siteimprove.com/glossary/website-optimization/. Luettu: 17.4.2023.

Talalaev, A. 4.3.2021. 5 Reasons Why Website Security Is Important. Patchstack-sivuston artikkeli. Luettavissa: https://patchstack.com/articles/reasons-why-website-security-important. Luettu: 29.5.2023

Tivix 8.2.2022. Agile Web Development: a Complete Guide. Tivix-sivuston blogi. Luettavissa: https://www.tivix.com/blog/agile-web-development. Luettu: 18.4.2023

Valencia, A. 7.2.2022. Top 7 SEO Benefits of Responsive Web Design. Search engine journal -sivuston blogi. Luettavissa: https://www.searchenginejournal.com/seo-responsive-web-design-benefits/. Luettu: 16.4.2023.

Venermo A. s.a. Mikä on CMS? Digitoimisto Folcanin blogi. Luettavissa: https://folcan.fi/mika-on-cms/. Luettu: 18.4.2023.

Viljanen, V. 16.3.2019. Verkkosivun tietoturvaotsikot. Digitoimisto Valkohatun blogi. Luettavissa: https://valkohattu.fi/artikkeli/tietoturvaotsikot. Luettu: 24.5.2023.

Vwo 2023. Website optimization. Vwo-sivuston blogi. Luettavissa: https://vwo.com/website-optimization/. Luettu: 16.4.2023.

Walker, J. 9.3.2023. How to Use CSS Breakpoints and Media Query Breakpoints for Responsive Design. SauceLabs-sivuston blogi. Luettavissa: https://saucelabs.com/resources/blog/css-breakpoints-media-query-breakpoints-responsive-design. Luettu: 16.4.2023.

Wagner, J. & Andrew, R. 2022. Use lazy loading to improve loading speed. Web.dev. Luettavissa: https://web.dev/lazy-loading. Luettu: 24.4.2023.

Webzyme Softwares 2022. Simple CMS for web designers. CouchCMS. Luettavissa: https://www.couchcms.com/. Luettu: 5.6.2023

Webzyme Softwares 2017a. Templates. CouchCMS. Luettavissa: https://docs.couchcms.com/concepts/templates.html. Luettu: 5.6.2023.

Webzyme Softwares 2017b. Tags. CouchCMS. Luettavissa: https://docs.couchcms.com/concepts/tags.html. Luettu: 5.6.2023.

Webzyme Softwares 2017c. DataBound Forms. CouchCMS. Luettavissa: https://docs.couchcms.com/concepts/databound-forms.html. Luettu: 4.4.2023

Weller, N. 26.4.2023. Converting HTML Sites to WordPress Sites. Elegantthemes-sivuston blogi. Luettavissa: https://www.elegantthemes.com/blog/tips-tricks/converting-html-sites-to-wordpress-sites. Luettu: 6.6.2023

Wikipedia-artikkeli 8.6.2023. Modal window. Wikipedia. Luettavissa: https://en.wikipedia.org/wiki/Modal_window. Luettu: 15.6.2023.

Wikipedia-artikkeli 2.6.2023. Single-page application. Wikipedia. Luettavissa: https://en.wikipedia.org/wiki/Single-page_application. Luettu: 15.6.2023.

Wpbeginner s.a. What is: Template. Beginners guide for WordPress. Luettavissa: https://www.wpbeginner.com/glossary/template/. Luettu: 7.6.2023.

Wiśniewski, K. 12.10.2021. Headless CMS vs Monolith CMS – What You Need to Know. Xfive-sivuston blogi. Luettavissa: https://www.xfive.co/blog/headless-vs-monolith-cms. Luettu: 5.6.2023.