Osaamista
ja oivallusta
tulevaisuuden
tekemiseen

Ville Aho

Sähköinen portfolio vähäkoodisen ohjelmistokehityksen tekniikalla

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikka

Insinöörityö

24.4.2019

Tekijä Otsikko Sivumäärä Aika	Ville Aho Sähköinen portfolio vähäkoodisen ohjelmistokehityksen tekniikalla 26 sivua 24.4.2019
Tutkinto	Insinööri (AMK)
Tutkinto-ohjelma	Tieto- ja viestintätekniikka
Ammatillinen pääaine	Mediatekniikka
Ohjaaja	Yliopettaja Kari Aaltonen

Insinöörityön tarkoituksena oli portfolion toteutus sekä perinteisiä ohjelmointimenetelmiä käyttäen että vähäkoodisen sovelluskehityksen tekniikalla. Portfolio tehtiin visuaalisen suunnitelman mukaan, joka tehtiin kuvankäsittelyohjelmalla. Perinteisesti ohjelmoimalla portfolio toteutettiin HTML:n, CSS:n ja JavaScriptin avulla, ja vähäkoodisen sovelluskehityksen versio toteutettiin Microsoftin PowerAppsilla.

Portfolio suunniteltiin työnhaun avuksi näyttämään tekijän osaamista ja samalla itse portfolio toimii myös työnäytteenä. Sen helppo käytettävyys ja päivittäminen ovat myös tärkeitä ominaisuuksia.

Vähäkoodinen sovelluskehitys on ajattelutapa, jonka toi yleiseen tietoisuuteen neuvontapalveluihin ja tutkimustyöhön erikoistunut yritys. Sen mukaan vähäkoodisen ohjelmistokehityksen alustat mahdollistavat sovellusten nopean toimituksen mahdollisimman vähäisellä käsin ohjelmoimisella ja mahdollisimman pienellä panostuksella perustukseen, koulutukseen ja käyttöönottoon. Vähäkoodisen sovelluskehityksen suurin etu on sen tarjoama nopeus ja helppous sovelluksen toteuttamiseen. Sovelluksen rakenteen ymmärtäminen on helppoa graafisen käyttöliittymän ansiosta ja raahaa sekä pudota -ominaisuus mahdollistaa sovelluksen toteuttamisen ilman aiempaa ohjelmointikokemusta. Erään arvion mukaan vähäkoodisen sovelluskehityksen markkinat nousevat vuoden 2017 3,8 miljardista eurosta 21,1 miljardiin euroon vuoteen 2022 mennessä.

Työssä vertailtiin portfolion toteutusta perinteisen ohjelmoinnin keinoin ja vähäkoodisen sovelluskehityksen tekniikalla. Vertailu osoitti, että vähäkoodisen sovelluskehyksen avulla portfolio valmistui noin neljä kertaa lyhyemmässä ajassa kuin perinteinen versio, mutta siitä puuttui osa perinteisen version toiminnallisuudesta. Näin suuri ajan säästäminen oli arvokasta. Päivittämisen helppous oli myös iso etu, ja näistä syistä vähäkoodinen tekniikka osoittautui paremmaksi valinnaksi portfolion tuottamiseen.

Tavoitteena oli tehdä portfoliosta yksinkertainen, mutta samalla mielenkiintoinen ja saada katsoja keskittymään itse töihin. Tässä onnistuttiin hyvin, ja valmis portfolio vastasi odotuksia. Sen yksinkertaisuus ja rauhallisuus jättivät hyvin tilaa itse töille.

Avainsanat	Low code, HTML, CSS, JavaScript, Photoshop, valokuvaus

Author Title Number of Pages Date	Ville Aho Creating electronic portfolio using low code technology 26 pages 24 April 2019
Degree	Bachelor of Engineering
Degree Programme	Information and communication technologies
Professional Major	Media Technology
Instructor	Kari Aaltonen, Principal Lecturer

The aim of this thesis was to create a portfolio using traditional programming methods and low code technology. Visual layout for the portfolio was created with a graphics software Adobe Photoshop. Traditionally programming the portfolio was created with HTML, CSS and JavaScript. The low code version was created with Microsoft PowerApps.

The portfolio is designed to show abilities and skills when job-hunting. The portfolio itself also works as a demonstration of skills. Easy usability and ability to update are also important features.

The term low code was coined by Forrester Research. Forrester defines low code as follows “Low-code platforms enable rapid delivery of business applications with a minimum of hand-coding and minimal upfront investment in setup, training, and deployment”. The biggest benefit of low code is the speed and ease to create applications it provides. Understanding an application’s structure is easy because of the graphic user interface. The drag and drop -feature enables creating applications without any previous experiment in programming. According to a PCMag UK’s article the four best low code platforms are Appian, Microsoft PowerApps, Mendix and OutSystems. According to Forrester’s estimation the low code markets will rise from 2017’s 3.8 billion euros to 21.1 billion euros by the year 2022.

HTML is a standard markup langue used for creating web pages. CSS is a stylesheet used for changing the layout of a HTML document. JavaScript is a programming language which is used to create functionality to a HTML document.

The objective of this thesis was to make the portfolio simple but at the same time interesting and keep the viewer’s focus on the photos inside the portfolio. The creator succeeded in that and the portfolio met expectations. The simplicity and calmness of the design left room for the photos.

Keywords	Low code, HTML, CSS, JavaScript, Photoshop, photography

Sisällys

1 Johdanto

Insinöörityön tarkoituksena on sähköisen portfolion toteutus perinteisen ohjelmoinnin ja vähäkoodisen sovelluskehityksen (low code) tekniikoita käyttäen. Tärkeä osa työtä on low coden ja perinteisen ohjelmoinnin vertailu keskenään ja etenkin low coden käytettävyyden ja toimivuuden analysointi. Portfolion tehtävänä on esittää konkreettisesti tekijän osaamista sekä aikaisempia töitä. Yksi sen tärkeimmistä tehtävistä on olla tehokas apuväline työnhaussa ja muodostaa mahdollisimman todenmukainen ja edustava kuva tekijän osaamisesta potentiaaliselle työnantajalle. Itse portfolio toimii myös työnäyttönä ohjelmointi- ja suunnittelutaidoista.

Yksi tärkeimmistä tavoitteista on tehdä portfolion käytöstä mahdollisimman helppoa ja mielekästä. Keskeistä tavoitteen toteutumisen kannalta on tehdä portfoliosta toimiva mahdollisimman monella eri laitteella ja selaimella. Tästä syystä on tärkeää, että portfoliota selaavan käyttäjän käyttökokemuksen tulee olla miellyttävä laitteesta riippumatta. Tärkeää on tehdä portfolion päivittämisestä mahdollisimman helppoa, koska materiaalia kertyy jatkossa lisää ja on syytä myös panostaa portfolion päivittäjän käyttökokemukseen. Siksi käyttöliittymän täytyy olla helppokäyttöinen, looginen ja sulava, visuaalista ilmettä kuitenkaan uhraamatta. Yksi tavoite on saada portfoliota käyttävä henkilö kiinnostumaan ja innostumaan tekijän osaamisesta.

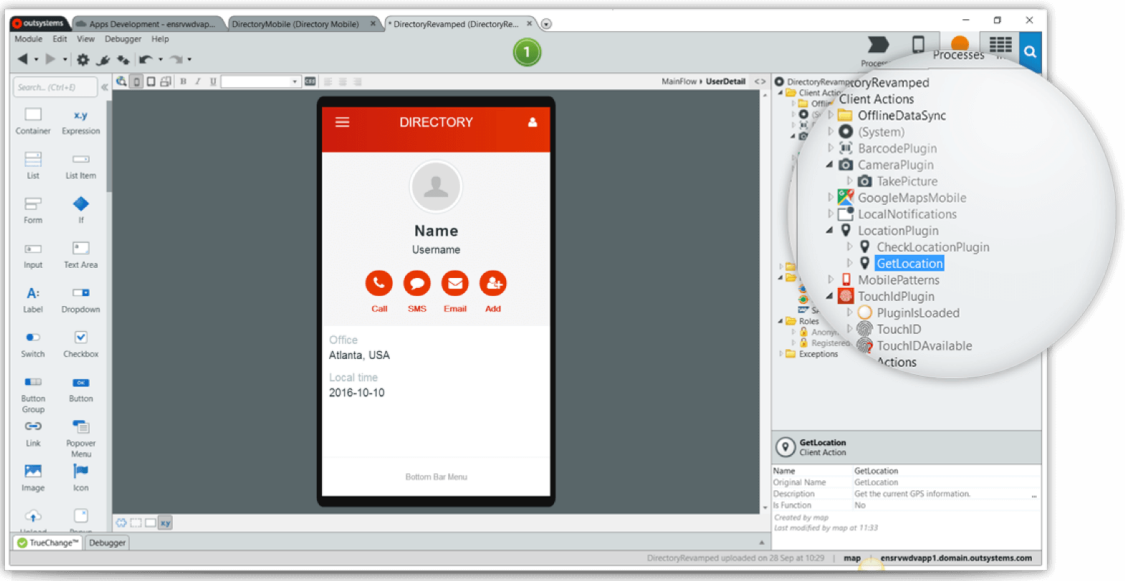
Ensimmäisessä luvussa perehdytään low coden historiaan, vahvuuksiin ja heikkouksiin sekä käyttötarkoituksiin. Toisessa luvussa selvitetään valokuvauksen historiaa, kuvauskalustoa ja kuvankäsittelyä. Portfolion tekemistä perinteisesti ohjelmoiden käsitellään kolmannessa luvussa. Siinä selostetaan HTML-, CSS- ja JavaScript -tekniikoita, joilla portfolio toteutettiin. Neljäs luku koostuu puolestaan portfolion rakentamisesta low code -työkalulla, ja siinä pohditaan low coden käytettävyyttä etenkin henkilökohtaisen portfolion toteuttamisen kannalta. Luvussa vertaillaan myös perinteistä ohjelmointia ja low code -tekniikkaa keskenään.

2 Vähäkoodinen sovelluskehitys

Tässä luvussa perehdytään vähäkoodiseen sovelluskehitykseen eli low codeen. Käydään läpi, mitä low code -tekniikalla tarkoitetaan ja mistä se on peräisin, kerrotaan sen heikkouksista ja vahvuuksista ja vertaillaan neljää eri low code -alustaa keskenään. Lopuksi käsitellään hieman low coden tulevaisuutta ja potentiaalista yleistymistä.

2.1 Low code -tekniikka

Low codella (tunnetaan myös no codena) ei ole virallista määrittelyä, koska se on enemmänkin ajattelutapa kuin jokin konkreettinen mitattava asia. Tämän ajattelutavan myötä se on yleistynyt ja useita eri low code -alustoja on muodostunut. [2.] Low code -alustat nopeuttavat ja helpottavat ohjelmistokehitystä huomattavasti. Niiden avulla koodin käsin kirjoittamisen määrä vähenee ja joissain tapauksissa sitä ei tarvitse tehdä yhtään. Low code -alustat mahdollistavat ohjelmoinnin graafisen käyttöliittymän valmiiden elementtien ja toimintalogiikoiden kautta. Ohjelmoija siis rakentaa sovelluksen low code -alustan käyttöliittymällä, ja tämä työkalu luo valmiin koodin ohjelmoijan valintojen mukaan. [1.] Kuvassa 1 nähdään OutSystemsin low code -alustan käyttöliittymä ja esimerkkisovellus mobiilinäkymässä.



Kuva 1. OutSystemsin low code -alustan käyttöliittymä [3].

Neuvontapalveluihin ja tutkimustyöhön erikoistunut yritys Forrester Research keksi termin ”low-code” vuonna 2014 kuvaamaan ohjelmointialustoja, jotka keskittyvät ohjelmoinnin yksinkertaisuuteen ja helppokäyttöisyyteen. Toinen, myös tutkimustyöhön ja neuvontapalveluihin erikoistunut yritys Gartner puolestaan auttoi Forresteria tuomaan low coden yleiseen tietoisuuteen. Näiden yritysten kuvaukset low codesta ovat lähes identtiset. [4; 5; 6.]

Forresterin mukaan low code -alustat mahdollistavat liiketoiminnan sovellusten nopean toimituksen vähäisellä käsin ohjelmoimisella ja mahdollisimman pienellä sijoituksella perustukseen, koulutukseen ja käyttöönottoon. Gartnerin mukaan low code -ohjelmointi sekä kuvaa alustoja, jotka erottavat itsensä koodista, että tarjoaa integroidut työkalut sovelluksen toimituksen nopeuttamiseen. [5.]

2.2 Vähäkoodisen sovelluskehityksen vahvuudet ja heikkoudet

Low coden suurin etu on kehitystyön nopeutus. Tämän myötä sovellusta päästään testaamaan selvästi tavallista aikaisemmin ja sen esittely asiakkaille päästään myös toteuttamaan tavallista nopeammin. Tämän seurauksena saadaan asiakaspalautetta ja asiakkaan muutostoiveisiin päästään reagoimaan tehokkaammin ja nopeammin, mikä puolestaan auttaa pitämään asiakkaan tyytyväisenä ja vähentää ylimääräisen työn määrää huomattavasti. Myös kustannustehokkuus kasvaa, koska sovelluksen kehittämiseen vaadittava aika vähenee. Tämä myös mahdollistaa sellaiset kehityshankkeet, jotka eivät aiemmin kustannussyistä olleet järkeviä toteuttaa, esimerkiksi eri käyttäjäryhmille räätälöinti tai pienelle joukolle käyttäjiä tarkoitetut toteutukset. [1.]

Toinen low coden vahvuus on sen tarjoama helppous koodin tuottamiseen. Sen avulla vähemmän kokeneet ohjelmoijat saavat tuotettua tasalaatuista koodia helpommin ja nopeammin. Tämä voi olla erityisen toivottu ominaisuus nykypäivän startup-yrityksille, joiden henkilöstö ei ole tarpeeksi suuri kattamaan yrityksen ohjelmointitarpeita. Samalla sovellusta ja sen rakennetta on helpompi ymmärtää ja muuttaa graafisen käyttöliittymän ansiosta. [1.]

Low codea ei ole tarkoitettu korvaamaan yrityksen koko ohjelmistokehitystä, eikä se toimi sellaisissa suurissa järjestelmissä, joita varten on jo olemassa valmiit ohjelmistopaketit.

Vaativat ohjelmistoprojektit, kuten esimerkiksi toisiinsa kytkettyjen ja räätälöityjen järjestelmien kehittäminen, eivät myöskään toimi low code -tekniikalla. [1.]

2.3 Vähäkoodisen sovelluskehityksen alustat

PCMag UK on julkaissut artikkelin, jossa vertaillaan keskenään kymmentä eri low code -alustaa. Tähän lukuun on valittu neljä parhaan arvosanan saanutta alustaa, joihin syvennytään tarkemmin.

PCMag UK:n artikkelin testeissä jokaista low code -alustaa testattiin sekä keskivertoyrityshenkilön että kokeneen sovelluskehittäjän näkökulmasta. Tämän tarkoituksena oli selvittää, kuinka hyvin testattava alusta selviää eritasoisista vaatimuksista riippuen rakennettavan sovelluksen tarpeista. Keskivertoyrityshenkilön näkökulmasta testaus toteutettiin rakentamalla sama yksinkertainen aikataulusovellus kaikilla low code -alustoilla. Kokeneen ohjelmoijan näkökulmasta testi toteutettiin rakentamalla hieman monimutkaisempi sovellus yhteystietojen hallintaan jokaisella alustalla. Jokaisen alustan osalta pyrittiin myös vastaamaan muutamaan yksinkertaiseen kysymykseen. Kysymykset olivat seuraavat: Onnistuimmeko rakentamaan toimivan perussovelluksen? Olivatko lomakkeisiin perustuvat sekä raahaa ja pudota -tekniikalla toimivat käyttöliittymät helpompikäyttöisiä ja aikaa säästäviä verrattuna perinteiseen ohjelmointiin? Mitä räätälöintiin liittyviä toimintoja ja lisäominaisuuksia alusta tarjosi? Vaatiko alusta yhtään koodin kirjoittamista sovellusta rakennettaessa? Jos kyllä, niin kuinka paljon ja missä asiayhteydessä? [7.]

Artikkelin mukaan kaikki alustat ovat erittäin tasaväkisiä käytettävyyden, toiminnallisuuden ja yleisten low code -ominaisuuksien osalta sekä keskivertoyrityshenkilön että kokeneen ohjelmoijan kannalta. Testeissä edukseen erottuivat kuitenkin neljä parasta alustaa, jotka olivat Appian, Microsoft PowerApps, Mendix ja OutSystems. Toimittajan valintapäätöksiä jaettiin kaksi, joista toinen meni veteraani Appianille keskivertoyrityshenkilöiden käyttöön ja toinen uudelle tulokkaalle Microsoftin PowerAppsille ohjelmoijien ja tehokäyttäjien käyttöön. Aivan niiden perässä tulivat Mendix ja OutSystems. [7.]

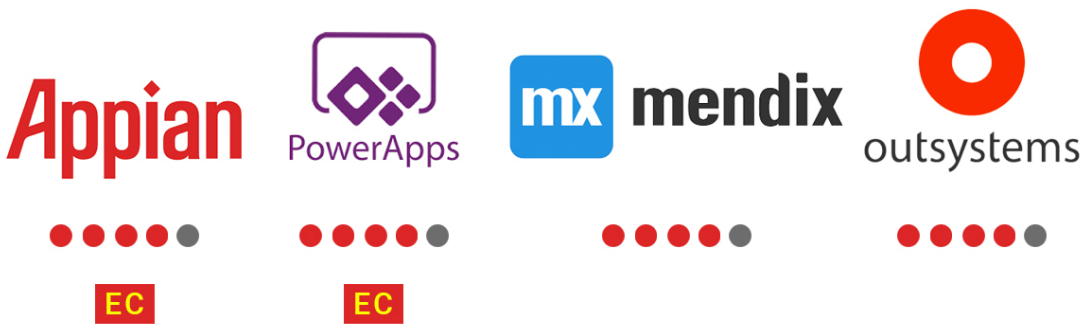
Appianin vahvuuksia olivat sen tarjoama intuitiivinen ohjattu kokemus ilman koodin kirjoittamista. Se erotti sovelluksen rakentamisen ja räätälöinnin selviksi prosesseiksi

yrityshenkilöille ja ohjelmoijille. Myös sisäänrakennettu yhteistyömahdollisuus, natiivit mobiilisovellukset sekä raahaa ja pudota -työkalun toimivuus olivat syitä Appianin korkeaan arvosanaan. Heikkouksia olivat sen hinta, osittainen käyttöliittymän vanhanaikaisuus ja ohjelmointitietämyksen tarve virheiden korjaamisessa. [7.]

Microsoftin PowerAppsin vahvuuksia olivat tehokas visuaalinen käyttöliittymä, yhteys Office 365:n kanssa ja laaja valikoima käyttöliittymäobjekteja ja valmiiksi rakennettuja mallineita. Myös hyvät mobiili- ja tablettikehitysmahdollisuudet ja kehittynyt työnkulun automatisointi sisäänrakennettuna Microsoft Flow'n kanssa olivat PowerAppsin vahvuuksia. Heikkouksia puolestaan olivat hieman pidemmät latausajat, datalähteiden luominen ja niihin yhdistämisen ongelmallisuus ja vaikeus säätää käyttöliittymän elementtejä. [7.]

Mendixin vahvuuksia olivat päivittäinen low code -kokemus, laaja valikoima valmiiksi rakennettuja App Store -mallineita ja -integraatioita, responsiiviset mobiili- ja tablettiesikatselumahdollisuudet, sovellusanalytiikka sekä automatisoitu sovellustestaus. Heikkouksia olivat hinta ja työläämpi opittavuus muihin alustoihin verrattuna. [7.]

OutSystemsin vahvuuksiksi osoittautuivat räätälöity ja ohjattu sovelluksen luominen käyttäjän roolin ja taitotason mukaan, datan offline-varastointi, interaktiivinen opetus sekä mahdollisuus julkaista suoraan App Store ja Google Play -sovelluskauppoihin. Heikkouksia olivat yrityssuunnitelmien hinta ja käyttöliittymän kankeus. Kuvassa 2 nähdään PCMagin testin neljä parasta ehdokasta, niiden kokonaisarvosana (0–5) ja toimittajan valinta palkinnon saaneiksi alustoiksi. [7.]



Kuva 2. PCMag-testauksen neljä parasta low code -alustaa [7].

2.4 Vähäkoodisen sovelluskehityksen tulevaisuus

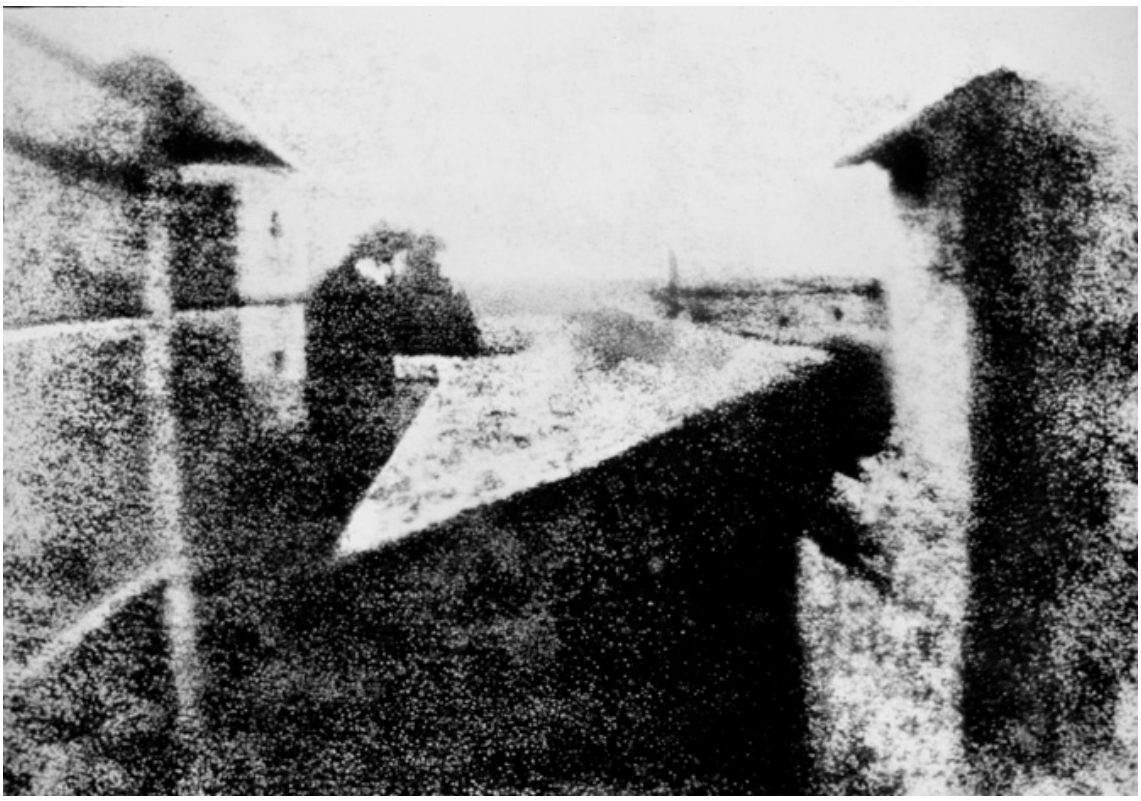
Kissflow'n artikkelin mukaan low code ei ole tulevaisuus, vaan se on jo täällä ja sen mukaan low code -alustat muuttavat liiketoimintaa [5]. Forrester puolestaan arvioi, että low code -markkinat nousevat vuoden 2017 3,8 miljardista eurosta 21,1 miljardiin euroon vuoteen 2022 mennessä. Tämä kasvu todistaa, että yritysten on harkittava low code -alustojen käyttöönottoa tosissaan. [8.]

3 Valokuvaus

Tässä luvussa käsitellään valokuvauksen historiaa ja kerrotaan muun muassa valokuvaprosessin kehittäjästä ja ensimmäisestä valokuvasta. Listataan myös portfoliossa käytettyjen valokuvien kuvauskalusto ja kerrotaan yleisesti kuvankäsittelystä.

3.1 Valokuvauksen historia

Nicéphore Niépce kehitti ensimmäisen valokuvaprosessin vuonna 1824. Tämä prosessi tunnettiin nimellä heliografia. Valokuvauksen alkuaikoina valokuvan muodostaminen heliografialla vei useita päiviä, mutta Niépce ja Louis Jacques Mandé Daguerre onnistuivat kehittämään prosessia vuonna 1832. Yhdessä he paransivat prosessia ja saivat nopeutettua valotusajan yhteen päivään. [9.] Kuvassa 3 nähdään maailman ensimmäinen valokuva, Niépcen ensimmäinen valokuva hänen itse kehittämällään tekniikalla.



Kuva 3. ”Point de vue du Gras”. Nicéphore Niépcen ja samalla maailman ensimmäinen valokuva. [10.]

Daguerre kehitti oman valokuvaprosessinsa vuonna 1838, ja sen nimeksi tuli dragerrotypia. Se oli ensimmäinen prosessi, joka käytti valokuvan kehitystä yhtenä otoksen muodostamisen vaiheena. Valotusaika dragerrotypialla oli ainoastaan 30 minuuttia. [9.]

Näihin aikoihin valokuvat muodostettiin hopealevyille, mutta vuonna 1839 Hippolyte Bayard keksi uuden tavan muodostaa valokuvia paperille. Bayardin tavalla valokuvan muodostuminen vei aikaa puolesta tunnista kahteen tuntiin. [9.]

Fyysikko Hippolyte Fizeau kehitti ensimmäisen kameran vuonna 1841. Tämä oli ensimmäinen valokuvatekniikka, jolla oli mahdollista ottaa muotokuvia. Fizeau käytti kamerassaan lyhyen polttovälin linssiä ja dragerrotypiaa. Tämän ansiosta hän sai lyhennettyä valotusajan vain pariin sekuntiin. Kuvattavien mallien täytyi olla paikallaan vain hetken, mikä mahdollisti kuvien ottamisen ilman häiritsevää liike-epäterävyyttä. [9.]

Ensimmäisen värillisen valokuvan onnistui kehittämään Louis Ducos du Hauron vuonna 1869. Hän muodosti samasta otoksesta keltaisen, punaisen ja sinisen version

käyttämällä eri suodattimia kuvien kehittämisessä. Lopuksi hän yhdisti eri versiot yhdeksi värilliseksi valokuvaksi. [9.]

Ensimmäisen kaupallisen kameran loi Estman Kodak vuonna 1888 ja nimesi sen Kodak Browniksi. Se oli erittäin helppokäyttöinen mustavalkokamera, joka nousi suureen suosioon keskiluokan keskuudessa. Kodak Brownin slogan oli suomennettuna: ”Paina nappia, me teemme loput”. [11.]

Ensimmäisen digitaalisen kameran DS 1P:n toi markkinoille Fuji vuonna 1991. Se oli suunnattu erityisesti journalisteille, eikä niinkään koko kansan käyttöön. Apple loi ensimmäisen kaupallisille markkinoille tarkoitetun digitaalikameran. Sen nimi oli Apple QuickTake ja se oli ensimmäinen kamera, joka oli yhdistettävissä tietokoneeseen kaapelin välityksellä. [12.] Kuvassa 4 nähdään ensimmäinen tietokoneeseen liitettävä kamera Apple QuickTake.

Kuva 4. Maailman ensimmäinen tietokoneeseen liitettävä kamera, Apple QuickTake [13].

3.2 Valokuvauskalusto

Jokainen insinöörityönä tehdyn portfolion kuva on otettu kahdella eri digitaalijärjestelmäkameralla: kameroina olivat Nikonin hieman vanhempi malli D40X ja uudempi D90-malli.

Nikon D90 sisältää 12,3 megapikselin DX-muotoisen CMOS-kuvakennon, kolmen tuuman tarkan nestekidenäytön, automaattitarkennusmoottorin sekä kehittyneen 11 pisteen autotarkennusjärjestelmän. Sen suurin mahdollinen isoherkkyys on 3200, joka mahdollistaa kuvaamisen käsivaralla myös tilanteissa, joissa valoa ei ole juurikaan tarjolla. [14.] Nikon D40X sisältää 10,2 megapikselin DX-kennon ja 2,5 tuuman nestekidenäytön. Suurin mahdollinen isoherkkyys on 1600, joka on selvästi huonompi kuin D90:ssä. Siispä hämärässä kuvaaminen sillä on lähes mahdotonta käsivaralla. [15.]

Kuvissa käytetyt objektiivit olivat Nikonin 40 mm, 50 mm, 55–200 mm ja 18–55 mm sekä Sigman 18–35 mm.

Nikkor AF/D 50 mm F/1.8 D on valovoimainen kiinteäpolttovälinen objektiivi Nikonin kameroihin. Se toimii D90:ssä normaalisti tarkennusmoottorin ansiosta [14], mutta D40X-kamerassa se toimii ainoastaan manuaalitarkennuksella sillä siinä ei ole samaa moottoria [16]. Nikkor AF-S DX Micro 40 mm f/2.8G on valovoimainen lähikuvaukseen tarkoitettu makro-objektiivi [17]. Nikkor AF-P DX 18-55 mm f/3.5-5.6G on normaalizoomobjektiivi, joka on tarkoitettu normaalikäyttöön perusharrastelijalle [18]. Nikkor AF-S DX 55-200 mm f/4-5.6G ED VR II on kuvanvakaimella varustettu teleobjektiivi, jolla pääsee lähemmäs kohdetta pitkän zoomin ansiosta [19]. Sigma 18-35 mm f/1.8 DC HSM on monikäyttöinen ja erittäin valovoimainen normaalizoomobjektiivi [20].

3.3 Kuvankäsittely

Tapanani on käsitellä kaikkia ottamiani kuvia, eivätkä portfolion kuvat ole poikkeus tästä, vaan niitä on muokattu Adobe Lightroomilla tai Photoshopilla. Kuvia on käsitelty muun muassa kontrastin, rajauksen, värien, terävyyden ja valotuksen osalta.

Adobe Photoshop on lyhyesti sanottuna kuvankäsittelyohjelma. Se tarjoaa yleiset työkalut valotuksen ja värien muuttamiseen, mutta myös paljon muuta, esimerkiksi tasot, maskit ja suodattimet. [21, s. 3.]

Photoshop sisältää melkein rajattoman valikoiman eri työkaluja, ja se toimii erinomaisesti ratkaisuna erilaisiin ongelmiin. Seuraavassa on muutama mahdollinen käyttötarkoitus Photoshopille: kuvan ehostus, kuvan korjaus, kuvakokonaisuuksien muodostaminen,

taidetehosteiden tekeminen, maalaaminen, tekstin lisääminen kuviin, web-kuvien tekeminen, tulostusvalmiin materiaalin tuottaminen, videomateriaalin korjaus, 3D-objektien ja muotojen tuottaminen, kuvien animoiminen ja tekstuurien lisäys 3D-objekteihin. Vaikka Photoshop onkin monipuolinen sovellus, sillä ei kuitenkaan kannata tehdä kaikkea. Seuraavassa muutama tehtävä, jotka olisi järkevämpää toteuttaa jollain toisella työkalulla: sivujen tai kirjan taittaminen, tekstin tuottaminen, vektoritaiteen tekeminen ja kaavioiden tai muu infografiikan teko. Kuvassa 5 nähdään, miten Photoshopilla on mahdollista esimerkiksi muuttaa totuutta ja tehdä kuvasta paljon mielenkiintoisempi, mikäli taidot vain siihen riittävät. [21, s. 5; 21, s. 6.]

Kuva 5. Vasemmalla kuvan ennen muokkausta Photoshopilla ja oikealla sama kuva muokkauksen jälkeen [22].

Adobe Lightroomissa on paljon samoja toimintoja kuin Photoshopissa, mutta se on suunniteltu enemmän ammattivalokuvaajille ja valokuvauksen tosissaan ottaville harrastelijoille. Sen toiminta keskittyy kuvien pieneen korjaukseen ja ehostukseen. Se myös tarjoaa käyttäjille toimivat työkalut valokuvien esittämiseen ja hallinnoimiseen. Lightroomin nopea ja kätevä tapa käsitellä suuria kuvakokonaisuuksia on sen vahvuus muihin

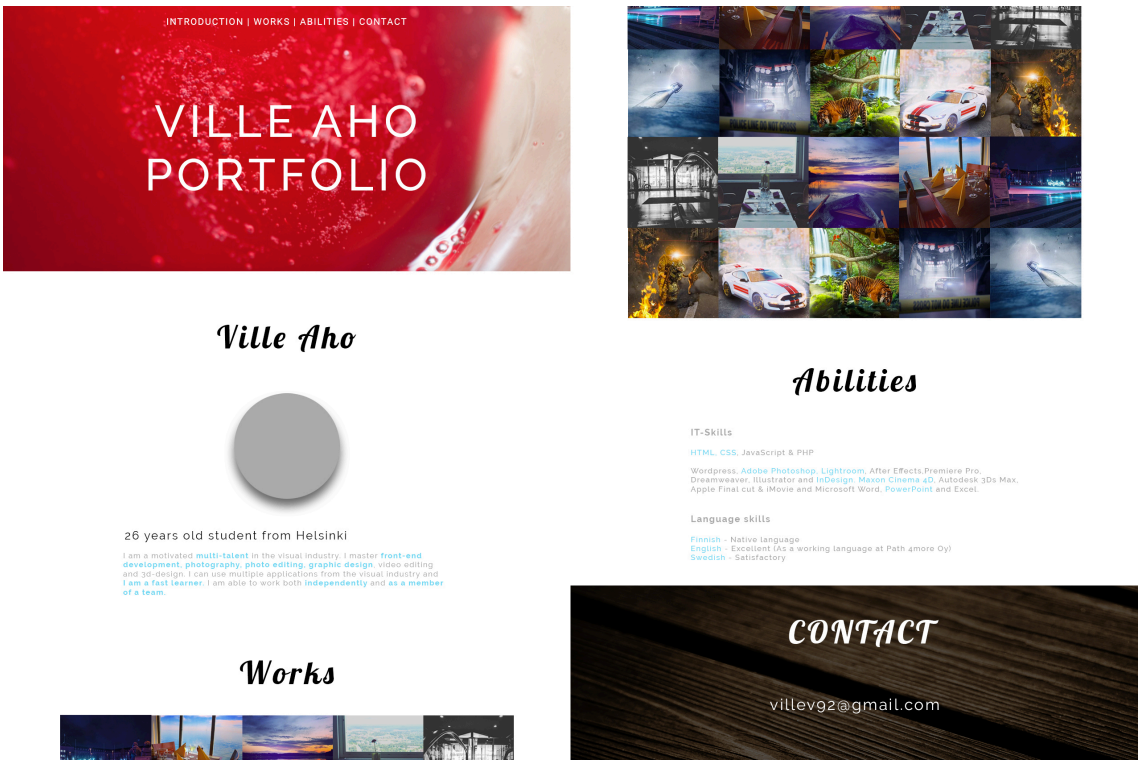
kilpailijoihin verrattuna, etenkin tilanteissa, joissa on useita kuvia käsiteltävänä. Se toimii kuitenkin hyvin myös yhdessä muiden sovellusten, kuten esimerkiksi juuri Photoshopin kanssa. [23, s. 2.] Itse pidän hyvänä tapana käsitellä kuvia ensin hieman Lightroomissa, esimerkiksi korjata valotusta ja kontrastia. Mikäli jokin kuva kaipaa suurempaa käsittelyä, toteutan sen sitten Photoshopissa.

4 Portfolio perinteisesti ohjelmoimalla

Tässä luvussa selostetaan insinöörityön portfolion suunnittelu ja perinteisillä menetelmillä ohjelmoiminen. Käydään läpi visuaalisen suunnitelman tuottaminen Photoshopilla ja portfolion rakentaminen HTML-merkintäkielen, CSS-tyylilajien ja JavaScript-ohjelmointikielen avulla.

4.1 Suunnitelma

Aloitin projektin tekemällä suunnitelman portfolion ulkoasusta, ensin hahmottelemalla perinteisesti kynällä paperille ja tekemällä sitten viimeistellyn version Adobe Photoshopilla. Tämä nopeutti ja helpotti ohjelmointityötä huomattavasti ja auttoi koko portfolion rakenteen hahmotusta. Halusin tehdä portfolion ulkoasusta yksinkertaisen ja tyylikkään, tekemättä siitä kuitenkaan tylsää. Onnistuin mielestäni tavoitteessa hyvin. Kuvassa 6 nähdään työssä luotu visuaalinen suunnitelma: jaoin kuvan kahteen osaan säästääkseni tilaa tässä dokumentissa.



Kuva 6. Portfolion visuaalinen suunnitelma.

4.2 HTML-merkintäkieli ja CSS-tyylilajit

HTML (Hyper Text Markup Language) on web-sivujen luomiseen käytetty standardi merkintäkieli. Sitä käytetään kuvaamaan verkkosivujen rakennetta, sivu rakennetaan HTML-elementtien avulla. HTML-elementit on määritetty valmiiksi, ja ne rakentuvat niin sanotusta alkutagista, elementin nimestä ja lopputagista. Alkutagi rakennetaan pienempi kuin- ja suurempi kuin -merkkien sisään < ja >, ja lopputagi tehdään muuten identtisesti, paitsi välittömästi pienempi kuin -merkin jälkeen kirjoitetaan vinoviiva /. Esimerkkikoodissa 1 on erittäin yksinkertaisen esimerkkisivun HTML-rakenne. [24.]

```
<!DOCTYPE html>
<html>
<head>
<title>Sivun otsikko</title>
</head>
<body>

<h2>Tämä on otsikko</h2>
<p>Tämä on leipätekstiä</p>

</body>
</html>
```

Esimerkkikoodi 1. Esimerkki HTML-koodista [24].

<!DOCTYPE html> -elementti määrittää dokumentin HTML-versioksi HTML5, joka on tätä dokumenttia kirjoittaessa uusin vesio HTML:stä. <html>-elementti tulee olla jokaisen HTML-dokumentin ensimmäisenä elementtinä. <head>-elementti puolestaan sisältää itse dokumenttia sisältävää metatietoa. <title>-elementillä määritetään sivun otsikko, joka nähdään esimerkiksi selaimen välilehden otsikkona. <body>-elementin sisään tulee kaikki sivulla selaimessa näkyvä sisältö. <h2>-elementti muodostaa suuren otsikon. <p>-elementti taas määrittää leipätekstin. [24.]

CSS (Cascading Style Sheets) määrää, miltä HTML-elementit näyttävät selaimessa. Sitä käytetään luomaan ja muokkaamaan verkkosivun ulkoasua. CSS voidaan sijoittaa joko HTML:n <head>-elementin sisälle, suoraan muokattavan HTML-elementin sisään tai erilliseen .css-tiedostoon, jolloin selain lataa sen käyttöön HTML-tiedossa olevan linkin kautta. Yleisin vaihtoehto aikaisemmista on tehdä sille oma tiedosto, koska sen avulla projektin kansiorakenne pysyy selkeänä ja HTML-dokumentti mahdollisimman lyhyenä ja siistinä, mikä on aina tärkeää ohjelmointityössä. [25.]

CSS-syntaksissa aina ensin valitaan HTML-elementti, jota halutaan muokata. Seuraavaksi kirjoitetaan haluttu ominaisuus ja sen arvo. Syntaksi kirjoitetaan aaltosulkeiden sisään { sekä } ja se lopetetaan aina puolipisteeseen ;. Esimerkkikoodissa 2 nähdään esimerkki, jossa -elementin väri muutetaan siniseksi, teksti asetetaan oikealle ja lisäksi teksti muutetaan isoiksi kirjaimiksi. [26.]

```
span {
      Color: blue;
      Text-align: right;
      Text-transform: uppercase;
}
```

Esimerkkikoodi 2. Yksinkertaisen CSS-syntaksin esimerkki [26].

4.3 JavaScript-ohjelmointikieli

Kun portfolion rakenne oli valmis, lisäsin siihen vielä hieman toimivuutta JavaScriptin avulla. Lisäsin portfolion yläosan navigaation linkkeihin toimivuuden, joka vierittää sivua sen otsikon kohdalle, jonka linkkiä painettiin. Rakensin myös karusellin, jonka avulla kuvia voi selata yksi kerrallaan ”edellinen” ja ”seuraava” -nuolien avulla.

JavaScript on yhdessä käytettävän selaimen kanssa toimiva ohjelmointikieli, jota käytetään muun muassa verkkosivujen ehostamiseen. JavaScriptillä staattisen ja mitäänsanomattoman verkkosivun sisältöä on mahdollista muokata mielenkiintoiseksi, älykkääksi ja interaktiiviseksi. JavaScriptin käyttö voi esiintyä hienovaraisena, kuten esimerkiksi ”Terve maailma!”- tai ”Hyvää iltaa!” -tervehdyksenä sivun käyttäjälle tietokoneen kellonajan mukaan. Toisaalta sen käyttö voi olla myös selvästi esillä olevaa, kuten esimerkiksi kuvakaruselli web-sivulla, jossa JavaScript hoitaa vaihdokset kuvien välillä sekä niiden tehosteet ilman, että sivu latautuu välillä uudelleen. JavaScript ei tietenkään ole ainoa teknologia, joka tekee verkkosivujen ehostamisen mahdolliseksi. Siksi onkin tärkeää ymmärtää JavaScriptin sijainti eri web-työkalujen varastossa. Luvussa 4.2 mainitut HTML ja CSS toimivat usein yhdessä JavaScriptin kanssa, kuten myös monet muut teknologiat. Tästä syystä muidenkin teknologioiden hallitseminen JavaScriptin rinnalla on tärkeää. [27, s. 3.]

Kielenä JavaScript on monikäyttöinen, ja sen avulla voidaan ratkaista monia eri ongelmia. JavaScript toimii hyvin esimerkiksi seuraavanlaisten toiminnallisuuksien toteuttamiseen:

- muuttamaan verkkosivun elementtejä (esimerkiksi painike) interaktiivisiksi sivun ja käyttäjän välillä ja jakamaan käyttäjälle pientä tietokantamaista dataa käyttäjäystävällisen käyttöliittymän avulla
- hallitsemaan monikehyksisiä navigaatioita HTML-dokumentissa
- prosessoimaan dataa ennen sen lähettämistä palvelimelle

- muuttamaan sivun tyylejä ja sisältöä välittömästi käyttäjän toimintojen mukaan
- pyytämään tiedostoja palvelinpuolelta sekä tekemään lue ja kirjoita -komentoja palvelinpuolen skripteihin. [27, s. 13.]

JavaScriptiä ei kuitenkaan ole tarkoitettu kaikkien ongelmien ratkaisemiseen. Ohjelmoijat usein tuhlaavat aikaa yrittäessään käyttää JavaScriptiä tehtäviin, joihin sitä ei ole tarkoitettu. Tämän vuoksi on myös tärkeää tietää, millaisiin tehtäviin JavaScriptiä ei tulisi käyttää; vastaanottamaan tai muuttamaan selaimen asetuksia, pääikkunan ulkoasun toimintoja tai tulostusasetuksia, käynnistämään käyttäjän tietokoneella olevaa sovellusta, kirjoittamaan tai lukemaan käyttäjän tietokoneella olevia tiedostoja (poikkeuksena evästeet), kirjoittamaan suoraan palvelimella sijaitseviin tiedostoihin, ottamaan kiinni suoria datavirtoja palvelimelta uudelleenlähettämistä varten tai lähettämään salaisia sähköpostiviestejä käyttäjältä sivun hallitsijalle. [27, s. 13.]

JavaScript kielenä koostuu yhdeksästä eri osasta: merkkijonot, numerot, muuttujat, totuusarvomuuttujat, operaattorit, funktiot, ehtolauseet, taulukot sekä oliot [28].

Merkkijonot ovat arvoja, jotka koostuvat tekstistä. Ne voivat sisältää symboleita, numeroita, kirjaimia, välimerkkejä tai jopa emojeita. Merkkijono rakentuu lainausmerkkien sisään, ’’ tai ””. Numerot ovat matemaattisiin laskuihin käytettäviä arvoja, eikä niiden käyttö vaadi minkäänlaista erillistä syntaksia. Muuttujat ovat ohjelmoijan nimeämiä arvoja ja ne voivat sisältää mitä tahansa JavaScript-arvoja. Totuusarvomuuttujat ovat joko tosia tai epätosia, ja niitä säilytetään usein muuttujien sisällä. Operaattorit ovat symboleita arvojen välillä, ja ne mahdollistavat erilaisia matemaattisia operaatioita, kuten esimerkiksi yhteen-, vähennys- ja jakolaskuja. Funktiot taas ovat pääohjelman sisällä olevia ohjelmia, joita on mahdollista nimetä ja käyttää uudelleen. Ehtolauseet kontrolloivat JavaScriptin toimintaa sekä määrittävät, mitä koodin osia suoritetaan ja mitä ei. Taulukot ovat laatikon tapaisia arvoja, ja ne voivat säilyttää mitä tahansa muita arvoja. Taulukon sisällä olevat arvot ovat alkioita. Oliot puolestaan ovat arvoja, jotka sisältävät muita arvoja. Nimeämiseen ne käyttävät avaimia, jotka muistuttavat toiminnaltaan muuttujia. [28.]

Esimerkkikoodissa 3 nähdään yksinkertainen JavaScript-ohjelma, joka on rakennettu HTML-dokumentin sisään ohjelman esittämisen helpottamiseksi. Ohjelmassa on HTML-nappi <button>, johon liitetty koodi ottaa nykyisen kellonajan Date() -funktion avulla ja tulostaa sen näkyviin <p>-elementin sisään käyttämällä ”demo”-id:tä.

```
<!DOCTYPE html>
<html>
<body>

<button type="button" onclick="document.getElementById('demo').innerHTML =
Date()">Click me to display Date and Time.</button><p>This is a paragraph.</p>

<p id="demo"></p>

</body>
</html>
```

Esimerkkikoodi 3. Esimerkki JavaScript-koodista [29].

4.4 Portfolion ohjelmointi

Aloitin portfolion ohjelmoimisen HTML:n ja CSS:n avulla seuraamalla tekemääni visuaalista suunnitelmaa. Rakensin portfoliota ylhäältä alas osa kerrallaan, mutta muutaman osion kohdalla vaihdoin järjestystä hieman. Aloitin ylänavigaatiosta ja otsakekuvasta. Navigaatio oli vielä tässä vaiheessa ainoastaan tekstiä, mutta myöhemmin tein siihen toiminnallisuutta, jonka avulla sivulla voi liikkua linkkien kautta. Otsakekuvaksi valikoitui eräs ottamani makrokuva, ja tekstin otsakkeen sisälle tein <h1>-elementtinä, koska se oli ensimmäinen ja suurin otsikko portfoliossa. Seuraavaksi rakensin osion, johon tuli lyhyt esittelyteksti tekijästä, otsikko <h2> ja leipäteksti <p>-elementtinä sekä pyöristetty kuva tekijästä. Seuraavaksi loin kyvyt-osion, koska se oli rakenteeltaan lähes identtinen esittelyosion kanssa. Kopioin esittelyosion ja muutin siitä otsikon sekä leipätekstin sisällöt ja poistin kuvan. Tämän jälkeen tein kuvagallerian, joka tuli esittely- ja kyvyt-osioiden väliin. Taas galleriasta puuttui toiminnallisuus, ja toistaiseksi se oli vain viisi kuvaa vierekkäin ja neljä kuvaa allekkain.

Olin tehnyt valitsemistani kuvista jo valmiiksi 200 x 200 pikselin kokoiset kuvakkeet galleriaa varten. Lopuksi tein yhteydenotto-osion, joka oli yksinkertainen: <h2>-elementtinä otsikko ja <h3>-elementtinä sähköpostiosoite, taustana valokuva puulattiasta. Nyt portfolion rakenne ja ulkoasu olivat valmiita. Tämän jälkeen lisäsin portfolioon toiminnallisuutta JavaScriptin kautta. Lopuksi lisäsin muutaman säännön CSS-koodiin mobiililaitteille parantamaan portfolion responsiivisuutta. Määritin gallerian kuvat täyttämään koko sivun ja asetin niiden kooksi 50 %, jolloin niitä mahtui kaksi vierekkäin, suurensin karusellin edellinen- ja seuraava-nuolia sekä sulje-kuvaketta. Kaiken kaikkiaan portfolion rakentamiseen kului aikaa noin neljä kahdeksan tunnin päivää.

5 Portfolio vähäkoodisella sovelluskehityksellä

Tässä luvussa käydään läpi portfolion rakentaminen Microsoftin PowerApps low code -alustalla. Käydään läpi PowerApps-alustaa ja pohditaan muun muassa sen käytettävyyttä ja tehokkuutta. Lopuksi vertaillaan portfolion perinteisesti ohjelmoimista ja low code -alustalla rakentamista keskenään.

5.1 Alustan valinta

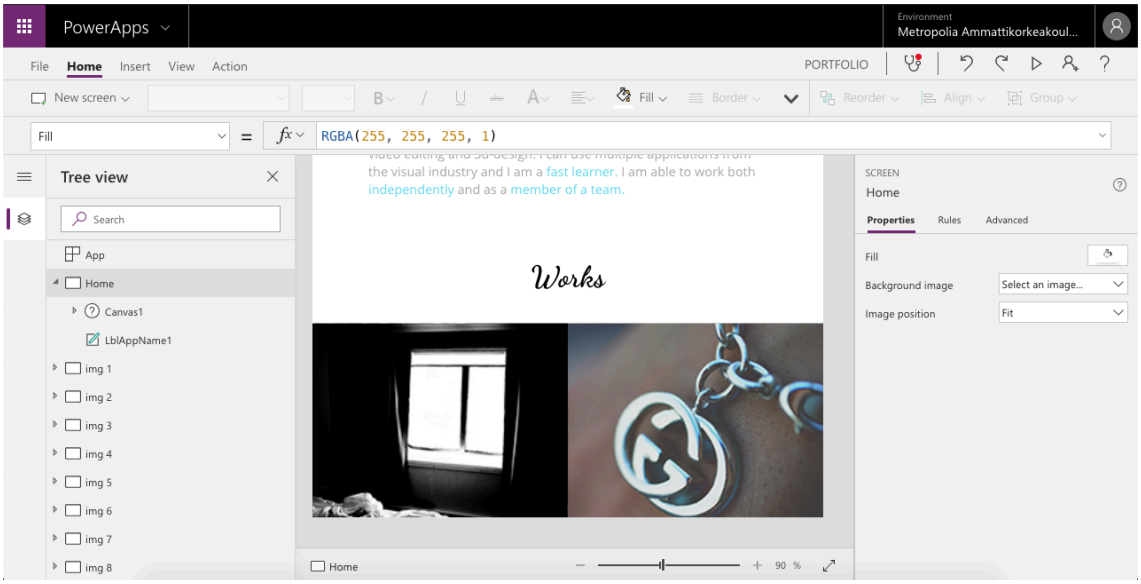
Sopivan low code -alustan valintaan kului aikaa suhteellisen paljon. Aloitin etsinnän hakemalla ilmaisia alustoja, ja löysin artikkelin, jossa listattiin kymmenkunta parasta ilmaista low code -alustaa. Kokeilin kolmea eri alustaa, mutta mikään niistä ei tuntunut soveltuvan portfolion tekoon. Ne olivat kankeita käyttöliittymiltään, ja tarjolla olevat työkalut olivat erittäin rajattuja ja niitä oli aivan liian vähän. Tässä vaiheessa ymmärsin, että portfolion toteutus ilmaisella alustalla ei tulisi onnistumaan. Siispä aloin etsiä maksullisia alustoja. Tätä dokumenttia tehdessä törmäsin PCMagin artikkeliin [7], jossa vertaillaan parhaita low code -alustoja. Alustat eivät ole ilmaisia, mutta niissä kaikissa on tarjolla ilmainen kokeilujakso. Päätin kokeilla yhtä testin parhaista alustoista, Microsoftin PowerAppsia, ja huomasin, että Metropolian tunnuksilla pääsin käyttämään koko alustaa ilmaiseksi.

5.2 Vähäkoodisen sovelluskehityksen alusta PowerApps

Ilmaisten alustojen tapaan Microsoftin PowerApps on selainpohjainen low code -alusta. Kirjauduin sisään Metropolian tunnuksella ja pääsin heti rakentamaan portfoliota. Sain hyvän ensivaikutelman alustasta: käyttöliittymä oli nykyaikainen ja tuttu muista Microsoftin ohjelmista, ja se myös näytti sisältävän enemmän työkaluja ja toimintoja, kuin aiemmin kokeilemani ilmaiset alustat.

Kuvassa 7 nähdään PowerAppsin käyttöliittymä kotinäkymässä. Vasemmalla olevassa palkissa nähdään projektin näkymät ja sen avulla pystytään muokkaamaan projektin rakennetta. Keskellä näkyy esikatselu projektista ja siitä pääsee liikuttamaan elementtejä raahaa ja pudota -toiminnon avulla. Oikealla on valitun elementin asetukset, ja siitä pääsee muun muassa vaihtamaan tekstin väriä, säätämään elementin sijaintia tai kokoa ja

määrittämään elementeille esimerkiksi reunaviivan. Keskellä esikatselun yläpuolella pystyy rakentamaan elementeille funktioita. Funktioita käytin esimerkiksi kuvagallerian karuselliin. Funktioiden yläpuolella on työkalupalkki, jossa voi muokata valitun elementin asetuksia. Se sisältää jonkin verran samoja asetuksia, kun oikeanpuolinen asetusnäkymä, mutta joitain asetuksia voi muuttaa vain yläpalkin kautta. Työkalupalkin vasemmassa reunassa on kohta, josta pääsee lisäämään uuden näkymän projektiin; esimerkiksi kaikki karusellin kuvat ovat omia näkymiään. Asetuspalkin yläpuolella on monista Microsoftin ohjelmista tuttu palkki, josta pääsee muun muassa avaamaan ja tallentamaan projektin ja vaihtamaan PowerAppsin eri näkymien välillä.



Kuva 7. PowerAppsin käyttöliittymä

Kaiken kaikkiaan PowerApps on erittäin toimiva low code -alusta. Se täytti lähes kaikki tarpeet portfolion suhteen, ja sen käyttö oli helppoa ja sujuvaa. Opin sen käytön nopeasti, enkä tarvinnut apua käyttöliittymän käyttöön, vaan lähdin työstämään portfoliota heti alustaan kirjauduttuani. Yksi ongelma alustan suhteen oli, että projektin pystyi luomaan vain mobiilille ja tabletille. Toinen pieni heikkous oli, että en voinut ladata projektiin omia fontteja. Tästä syystä jouduin käyttämään eri fontteja, kuin alkuperäisessä suunnitelmassani, mutta tämä ei kuitenkaan ollut suuri ongelma, sillä löysin hyvin saman tyyppiset fontit suunnittelemieni tilalle. Huomasin, että alustassa pystyi käyttämään perinteistä ohjelmointia: esimerkiksi olisin voinut käyttää CSS-osaamista hyödyksi tehosteiden muodossa. En kuitenkaan käyttänyt aiempaa ohjelmointiosaamista yhtään portfoliota

rakentaessa, vaan tein kaiken suoraan PowerAppsin käyttöliittymän työkaluilla. Sain tehtyä low code -portfoliosta lähes identtisen suunnitelman kanssa.

5.3 Portfolion rakentaminen

Rakensin low code -portfolion ylhäältä alas osa kerrallaan, koska PowerAppsilla osioiden uudelleenjärjestely tuntui hieman vaikealta ja yhden osion siirtäminen sotki helposti muita osioita.

Aloitin yläosan otsakekuvasta, latasin valitsemani kuvan PowerAppsiin ja lisäsin sen päälle tekstialueen. Valitsin tekstin koon ja yritin löytää suunnitelmassa käyttämäni kirjasimen. En löytänyt samaa kirjasinta, mutta valitsin mahdollisimman samannäköisen, ja se toimi hyvin. Jätin yläpalkin navigaation tekemättä, koska ymmärtääkseni sen toiminnallisuutta ei voinut toteuttaa PowerAppsilla. Myös otsakekuvan vaihtuminen, jonka toteutin perinteisen ohjelmoinnin versioon, sai jäädä pois samasta syystä.

Seuraavaksi siirryin työstämään esittelyosiota, joka oli helppo rakentaa. Valitsin otsikkokirjasimeksi mahdollisimman samanlaisen kirjasimen suunnitelman kanssa, ja onnistuin taas löytämään hyvin samannäköisen kirjasimen. Seuraavaksi lisäsin kuvan ja säädin sen ympyrän muotoiseksi oikean reunan asetuspalkista. Lopuksi rakensin esittelyosion tekstiosan. Lisäsin otsikon sekä leipätekstin ja korostin valitut sanat vaaleansinisellä värillä.

Seuraavaksi vuorossa oli kuvagalleria. Koska tein projektin mobiililaitteelle, jaoin gallerian kuvat kahteen kuvaan riviä kohti. Lisäsin kaikki kuvat galleriaan ja jätin sen vielä toistaiseksi ilman toiminnallisuutta. Tämäkin osio valmistui nopeasti PowerAppsin tehokkaan käyttöliittymän ansiosta.

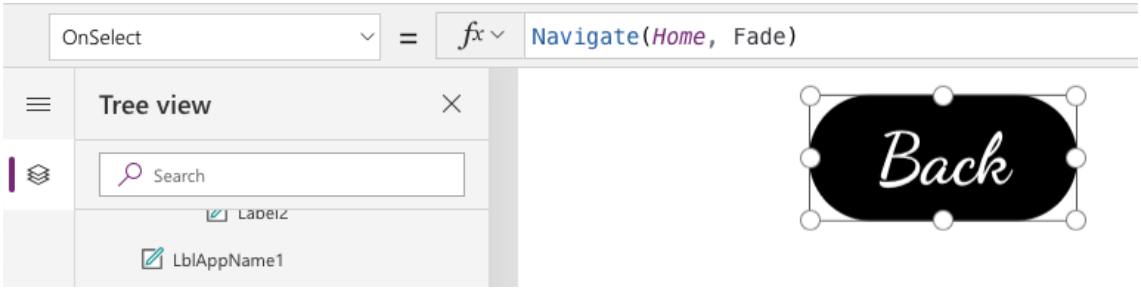
Gallerian jälkeen tein osion kyvyistä ja taidoista. Tämä osio oli nopea toteuttaa: kopioin esittelyosion, muutin tekstit ja korostin valittuja sanoja.

Seuraavaksi rakensin portfolion alaosan. Lisäsin valitun kuvan, jota jouduin tummentamaan Photoshopilla, sillä en saanut sitä tehtyä suoraan PowerAppsilla. Lopuksi lisäsin valitut tekstit ja valitsin niihin sopivan kirjasimen, koon ja värin.

Tämän jälkeen jäljellä oli enää portfolion toiminnallisuus. Koska navigaation ja otsakekuvan toiminnallisuudet oli mahdotonta toteuttaa PowerAppsilla, gallerian karuselli oli ainoa portfolion toiminnallisuus. Jouduin rakentamaan karusellin hieman työläämmällä tavalla, kuin odotin. Rakensin ensimmäiselle kuvalle uuden näkymän, jonka nimesin img 1:ksi. Lisäsin näkymään ensimmäisen kuvan ja tein siitä koko ruudun täyttävän. Seuraavaksi lisäsin napin, jonka tekstiksi kirjoitin ”back”, ja vasemmalle sekä oikealle osoittavat nuolet. Seuraavaksi määritin ”back”-napin viemään takaisin kotinäkymään, kun sitä painetaan. Koska kyseessä oli kuva 1 määritin vasemmalle osoittavan nuolen vievän viimeiseen kuvaan eli img 20 -näkymään ja oikealle osoittavan nuolen toiseen kuvaan eli img 2 -näkymään. Näitä näkymiä ei ollut vielä tässä vaiheessa olemassa, mutta linkityksen pystyi tekemään jo valmiiksi, ja kun esimerkiksi img 20 -niminen näkymä valmistui, linkitys toimi halutusti. Sitten testasin uutta toiminnallisuutta ensin menemällä karuselliin kotinäkymästä painamalla ensimmäistä kuvaa ja sitten menemällä takaisin kotiin painamalla takaisin-nappia. Molemmat toiminnallisuudet käyttäytyivät toivotulla tavalla.

Nyt ensimmäisen kuvan näkymä oli valmis. Seuraavaksi monistin img 1 -näkymän ja nimesin sen img 2:ksi. Vaihdoin näkymän kuvan toiseksi kuvaksi ja muutin vasemmalle osoittavan nuolen viemään img 1 -näkymään ja oikealle osoittavan nuolen img 3 -näkymään. Takaisin-painike vei jo valmiiksi kotinäkymään, joten siihen minun ei tarvinnut muuttaa mitään. Seuraavaksi testasin edellinen ja seuraava -nuolien toiminnallisuutta. Kaikki toimi edelleen toivotusti. Tämän jälkeen monistin img 1 -näkymän 18 kertaa ja toistin äskeisen prosessin kaikille näkymille. Lopuksi testasin koko karusellia ja totesin, että kaikki toimi juuri kuten halusin. Latasin myös PowerApps -mobiilisovelluksen, jonka avulla pääsin kokeilemaan portfoliota älypuhelimella. Myös siinä oli kaikki kunnossa. Low code -versio portfoliosta oli valmis, ja aikaa siihen kului noin yksi kokonainen kahdeksan tunnin päivä.

Kuvassa 8 nähdään takaisin-painikkeen toiminnan lisäys PowerAppsilla. Keskellä ylhäällä funktiopalkissa määritetään navigaatio kotinäkymään, ja Fade lisää siirtymiseen pienen animaation. Vasemman yläreunan palkissa valitaan, milloin funktio toteutetaan: tässä esimerkissä OnSelect eli kun nappia painetaan.



Kuva 8. Toiminnan lisääminen PowerAppsilla

5.4 Perinteinen ohjelmointi ja vähäkoodinen sovelluskehitys

Ennen tämän projektin alkua perinteisen ohjelmoinnin osaamiseni oli melko hyvällä pohjalla. HTML- ja CSS-kokemusta oli jo muutamalta vuodelta, ja JavaScript oli tuttu kursseilta. Low codesta minulla ei ollut minkäänlaista osaamista tai edes tietoa ennen tätä projektia. Siispä olin melko luottavainen, että portfolio perinteisesti ohjelmoimalla valmistuisi nopeammin kuin low code -versio. Totuus oli kuitenkin toinen: low code -versio valmistui noin 24 tuntia nopeammin kuin perinteinen versio. Vaikka tiesin, että low code -tekniikka oli suunniteltu juuri nopeuttamaan ohjelmointiprosessia, oli tämä ero mielestäni silti yllättävää, ottaen huomioon aikaisemman ohjelmointikokemukseni.

Perinteisellä ohjelmoinnilla rakensin portfolion osio kerrallaan hieman vaihtelevassa järjestyksessä ja low code -version rakensin järjestyksessä ylhäältä alas osio kerrallaan. Kuten aiemmin mainitsin, perinteisen ohjelmoinnin portfolioon kului aikaa noin 32 tuntia ja low code versio valmistui noin 8 tunnissa. Tässä kohtaa on tärkeää mainita, että perinteisestä versiosta tuli jonkin verran toiminnallisempi ja sen ulkoasu vastasi täysin suunnitelmaani, kun taas low code versiosta jäi puuttumaan ylänavigointi ja otsakekuvan vaihtuminen eikä sen ulkoasu ollut aivan suunnitelman mukainen. Ulkoasu kuitenkin vastasi noin 90-prosenttisesti suunnitelmaa eivätkä puuttuvat toiminnallisuudet olleet kovin oleellisia. Low code -version käyttö tietokoneella oli vaikeaa, koska sen pystyi rakentamaan vain mobiili- ja tablettilaitteita varten. Perinteinen versio toimi mainiosti kaikilla kolmella.

Ylläpito PowerAppsilla oli helppoa, ja uskon, että portfoliota pystyisi päivittämään kuka tahansa perus IT-taidot omaava projektin ulkopuolinen henkilö. Samaa ei voi sanoa perinteisesti ohjelmoidusta versiosta. Vaikka koodi onkin suhteellisen yksinkertainen ja

kommentoitu uskon, että sen päivittäminen ilman ohjelmointitaustaa tuottaisi ongelmia. Perus HTML- ja CSS-taidot omaavalta henkilöltä päivitys oletettavasti onnistuisi, mutta se olisi ehkä hieman hitaampaa low code -versioon verrattuna. Kaiken kaikkiaan low code -versio on mielestäni tarpeeksi hyvä vaihtoehto, mikäli tietokoneversion puute ei ole ongelma. PowerApps täytti kaikki kriteerini, ja portfolio valmistui noin neljä kertaa lyhyemmässä ajassa kuin perinteinen versio. Näin suuri ajan säästäminen oli arvokasta. Päivittämisen helppous oli myös iso etu ja näistä syistä low code -tekniikka oli mielestäni parempi valinta portfolion tuottamiseen.

6 Yhteenveto

Insinöörityössä perehdyttiin portfolion suunnittelemiseen, perinteisesti ohjelmoimiseen ja low code -alustalla toteuttamiseen. Low code on ajattelutapa, jonka Forrester Research keksi vuonna 2014. Forrester määrittelee low code -termin seuraavasti: vähäkoodisen ohjelmistokehityksen alustat mahdollistavat sovellusten nopean toimituksen mahdollisimman vähäisellä käsin ohjelmoimisella ja mahdollisimman pienellä panostuksella perustukseen, koulutukseen ja käyttöönottoon. Työssä saatiin selville vertailun kautta, että neljä parasta low code -alustaa ovat Appian, Microsoft PowerApps, Mendix ja OutSystems. Työssä tutustuttiin tarkemmin PowerAppsiin ja tultiin siihen tulokseen, että se tarjoaa hyvät työkalut ja loogisen käyttöliittymän. Sen ainoa heikkous on, että sovelluksen voi rakentaa vain puhelimelle ja tabletille.

Insinöörityössä selvisi, että ensimmäinen valokuvaprosessi oli nimeltään heliografia ja sen keksi Nicéphore Niépce vuonna 1824. Työssä selvitettiin myös, että ensimmäisen tietokoneeseen liitettävän kameran kehitti Apple ja sen nimeksi tuli QuickTake. Työssä tultiin siihen tulokseen, että Adobe Photoshop on paras kuvankäsittelyohjelma, kun käsiteltävässä kuvassa on paljon muokattavaa. Adobe Lightroom on parempi vaihtoehto, kun kuvia on paljon ja niihin pitää tehdä vain pieniä korjauksia, esimerkiksi värin, kontrastin tai valotuksen osalta.

Tärkein osa työtä oli itse portfolion toteutus, johon panostettiin paljon. Lopuksi työssä vertailtiin portfolion toteutusta perinteisen ohjelmoinnin ja low code -tekniikan kannalta. Tultiin siihen tulokseen, että low codella portfolion toteutus oli noin 24 tuntia nopeampaa, mutta siitä puuttui osa perinteisen version toiminnallisuudesta.

Visuaaliselta ilmeeltään portfolio on hillityn ammattimainen, mikä sitoo sen työnhaun tyyliin ja kiinnittää katsojan huomion itse kuviin. Kirjasintyyppeinä on kolme fonttia Google Fonts -kirjasinvalikoimasta, yksi otsakekuvan isoon otsikkoon, toinen tekstien otsikkoihin ja kolmas leipäteksteihin.

Mielekkäintä portfoliota tehdessä oli valokuvaus ja kuvankäsittely. Myös portfolion visuaalisen suunnitelman tekeminen oli mukavaa ja pidin myös portfolion ohjelmoinnista perinteisiä menetelmiä käyttäen, vaikka se osoittautuikin hitaaksi ja aikaa vieväksi prosessiksi. Portfolion toteuttaminen molemmilla tavoilla oli oppimisen kannalta kaikista tärkeintä. Vaikka olinkin aikaisemmin työskennellyt HTML:n ja CSS:n kanssa ja tiesin JavaScriptin perusteet, sain niistä paljon uutta tietoa ja opin käyttämään niitä tehokkaammin ja nopeammin portfoliota tehdessä. Low codesta en ollut kuullut ennen tämän projektin aloitusta. Uutta tietoa ja kokemusta low codesta kertyi paljon. Myös kiinnostukseni low codea kohtaan lisääntyi ja uskon, että tulen käyttämään tämän projektin myötä saamiani oppeja tulevaisuuden projekteissa.

Olen tyytyväinen molempien portfolioiden lopputuloksiin, ja uskon niiden olevan hyödyksi tulevaisuudessa töitä hakiessa. Myös itse portfoliot toimivat työnnäytteinä, ja siksi olenkin tyytyväinen, että sain tehtyä niistä esittelykelpoisia.

Lähteet

1 Tapanainen, Tero. 2018. Mitä on low-code ja kuka sitä tarvitsee? Verkkoaineisto. <https://www.ecraft.com/fin/blog/2018/2/15/mita-on-low-code-ketteraa-sovellus-kehitysta>. 15.2.2018. Luettu 24.1.2019.

2 Rands, Kevin. 2018. Why low code platforms are the future of app development. Verkkoaineisto. <https://www.cio.com/article/3250490/development-tools/why-low-code-platforms-are-the-future-of-app-development.html>. 22.1.2018. Luettu 24.1.2019.

3 Moore, Madison. 2016. OutSystems 10 extends platform, MOSS supports four open-source projects, and TIBCO's open-source IoT solution available—SD Times news digest: Oct. 4, 2016. Verkkoaineisto. <https://sdtimes.com/android/outsystems-10-extends-platform-moss-supports-four-open-source-projects-tibcos-open-source-iot-solution-available-sd-times-news-digest-oct-4-2016/>. 4.10.2016. Luettu 12.2.2019

4 About Forrester. 2019. Verkkoaineisto. Forrester. <https://www.forrester.com/marketing/about/about-us.html>. Luettu 24.1.2019.

5 The History of Low-Code Platforms: How Development Changed Forever. 2018. Verkkoaineisto. Kissflow. <https://kissflow.com/low-code/history-of-low-code-development-platforms/>. 2.5.2018. Luettu 24.1.2019.

6 For 40 years, Gartner has helped clients make the right decisions and stay ahead of change. 2019. Verkkoaineisto. Gartner. <https://www.gartner.com/en/about>. Luettu 24.1.2019.

7 Marvin, Rob. 2018. The Best Low-Code Development Platform of 2019. Verkkoaineisto. <https://uk.pcmag.com/cloud-services/89789/the-best-low-code-development-platforms>. 10.8.2018. Luettu 2.3.2019.

8 So THAT'S Why No-Code Development Is Getting So Much Attention. Verkkoaineisto. Kissflow. <https://kissflow.com/no-code/>. Luettu 26.1.2019.

9 The history of photography. Verkkoaineisto. Maisons des illustres. <http://www.photo-museum.org/photography-history/>. Luettu 17.10.2018.

10 Niépce, Nicéphore. 1824. Point de vue du Gras. Verkkoaineisto. <http://www.photo-museum.org/>. Luettu 17.10.2018.

11 Tolmachev, Ivan. 2010. A History of Photography Part 1: The Beginning. Verkkoaineisto. <https://photography.tutsplus.com/articles/a-history-of-photography-part-1-the-beginning--photo-1908> 15.3.2010. Luettu 18.10.2018.

12 Tolmachev, Ivan. 2010. A History of Photography Part 3: Going Digital. Verkkoaineisto. <https://photography.tutsplus.com/articles/a-history-of-photography-part-3-going-digital--photo-4003> 4.11.2010. Luettu 18.10.2018.

13 QuickTake was Apple's first doomed foray into digital photography. 2015. Verkkoaineisto. Cult of Mac. <https://www.cultofmac.com/329173/quicktake-was-apples-first-doomed-foray-into-digital-photography/> Luettu 22.10.2018.

14 Nikon D90 Yleiskuva. Verkkoaineisto. Nikon. <https://www.nikon.fi/fi_FI/product/discontinued/digital-cameras/2015/d90#overview>. Luettu 22.10.2018.

15 Nikon D40X Yleiskuva. Verkkoaineisto. Nikon. <https://www.nikon.fi/fi_FI/product/discontinued/digital-cameras/2009/camera-body-d40x#overview>. Luettu 22.10.2018.

16 Nikon Nikkor AF-D 50mm F/1.8 D. Verkkoaineisto. Verkkokauppa.com. <https://www.verkkokauppa.com/fi/product/15317/hcqb/Nikon-Nikkor-AF-D-50mm-F-1-8-D?gclid=Cj0KCQjw6rXeBRD3ARIsAD9ni9CO-jzK1tFpg3cGwYP3wKvOXd_ui-tISe_geI3Imqv0kvrBmmuUGzsJ8aAuVvEALw_wcB>. Luettu 22.10.2018.

17 Nikon AF-S DX Micro NIKKOR 40 mm f/2.8G. Verkkoaineisto. Verkkokauppa.com. <https://www.verkkokauppa.com/fi/product/49319/dcgjn/Nikon-AF-S-DX-Micro-NIKKOR-40-mm-f-2-8G?gclid=Cj0KCQjw6rXeBRD3ARIsAD9ni9BVzs09Rj9vqMRGX96SrLxUPsGfM_SWSe6rjDHA8mvCnqO2UxYEtHcaAiSHEALw_wcB>. Luettu 22.10.2018.

18 Nikon AF-P DX NIKKOR 18-55mm f/3.5-5.6G -objektiivi. Verkkoaineisto. Verkkokauppa.com. < https://www.verkkokauppa.com/fi/product/56707/hkvcq/Nikon-AF-P-DX-NIKKOR-18-55mm-f-3-5-5-6G-objektiivi?gclid=Cj0KCQjw6rXeBRD3ARIsAD9ni9BlGPtie5_8vJv2gBWu5XmklycnUvo8nSeSIpyyeP63IRTB-0FimAAaAo2ZEALw_wcB>. Luettu 22.10.2018.

19 Nikon Nikkor AF-S DX 55-200mm f/4-5.6G ED VR II objektiivi. Verkkoaineisto. Verkkokauppa.com. <https://www.verkkokauppa.com/fi/product/5059/fhsbh/Nikon-Nikkor-AF-S-DX-55-200mm-f-4-5-6G-ED-VR-II-objektiivi?gclid=Cj0KCQjw6rXeBRD3ARIsAD9ni9AlyP59aTeHwZAH-HLIHL7_0tMXBLUadP5BeLtVAqg_mGW_oktvAoqcaAncNEALw_wcB>. Luettu 22.10.2018.

20 Sigma 18-35 mm F1.8 DC HSM | A objektiivi. Nikon. Verkkoaineisto. Verkkokauppa.com. <https://www.verkkokauppa.com/fi/product/27352/dmhgt/Sigma-18-35-mm-F1-8-DC-HSM-A-objektiivi-Nikon?gclid=Cj0KCQjw6rXeBRD3ARIsAD9ni9Awu1MAg2cEVQ3Y7VR_a3xEHhRn5F1PbyV5da9kDRoZe03jrtCK-rkaAvxLEALw_wcB>. Luettu 22.10.2018.

21 DaNae, Dayley, Lisa & Dayley, Brad. 2013. Photoshop CC Bible. E-kirja. Hoboken: Wiley.

22 Asabin, Max. 2017. Sleepless. Verkkoaineisto. <https://www.deviantart.com/maxasabin/art/Sleepless-659653311>. Luettu 22.10.2018.

23 Coalson, Nat, & Sylvan, Rob. 2013. Lightroom 5: Streamlining Your Digital Photography Process. E-kirja. Hoboken: Wiley.

24 HTML Introduction. w3Schools.com. Verkkoaineisto. <https://www.w3schools.com/html/html_intro.asp>. Luettu 25.10.2018.

25 CSS Introduction. Verkkoaineisto. w3Schools.com. <https://www.w3schools.com/css/css_intro.asp>. Luettu 25.10.2018.

26 CSS Syntax and Selectors. Verkkoaineisto. w3Schools.com. <https://www.w3schools.com/css/css_syntax.asp>. Luettu 25.10.2018.

27 Goodman, Danny; Morrison, Michael; Novitski, Paul, & Schupak, Benjamin. 2010. JavaScript Bible. E-kirja. John Wiley & Sons.

28 Learn. 2010. Verkkoaineisto. JavaScript.com. <https://www.javascript.com/learn/strings>. Luettu 23.10.2018.

29 Try it yourself editor. Verkkoaineisto. w3Schools.com. <https://www.w3schools.com/js/tryit.asp?filename=tryjs_myfirst>. Luettu 25.03.2019.