Jyri Herka

Yrityksen kuvakirjaston parantaminen

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tieto- ja viestintätekniikan tutkinto-ohjelma

Insinöörityö

16.5.2021

Tiivistelmä

Tekijä:	Jyri Herka
Otsikko:	Yrityksen kuvakirjaston parantaminen
Sivumäärä:	29 sivua
Aika:	16.5.2021
Tutkinto:	Insinööri (AMK)
Tutkinto-ohjelma:	Tieto- ja viestintätekniikka
Ammatillinen pääaine:	Mediatekniikka
Ohjaajat:	Managing Partner Jesse Jokinen Lehtori Ilkka Kylmäniemi

---

Insinöörityön tarkoituksena oli parantaa yrityksen koulutustyökalun olemassa olevaa kuvakirjastoa. Hakutoiminnallisuuden lisääminen kuvien nimillä ja tageilla, kuvien jakaminen sivuihin, kansiorakenteen luominen, kuvien liikuttaminen kansioiden välillä, kuvien lisääminen kansioihin ja kuvan muuntaminen png-muodosta jpg-muotoon olivat tämän insinöörityön tuloksia.

Tekijänoikeus on asia, joka syntyy tekijälle heti teoksen valmistumishetkellä. Kuvien oikeuksista säädetään tekijänoikeuslailla, mikä tarkoittaa sitä, että kaikkia kuvia ei voi vapaasti käyttää. On olemassa kuitenkin poikkeuksia tekijänoikeussuojattuihin kuviin, jos niiden käyttämiseen ei liity ansiotarkoitusta. Internetissä on paljon sivustoja, jossa on tekijänoikeusvapaita kuvia ja niitä voi hyödyntää kuka tahansa.

Lokaalit ympäristöt toimivat kehittämisen tukena ja varmistavat sen, että omat kehitettävät asiat pysyvät erillään virallisesta versiosta niin palvelimelta kuin tietokannasta. Lokaalien ympäristöjen pystyttämiseen löytyy useita alustoja.

Kuvakirjaston parantaminen sisälsi frontendia ja backendia, palvelinpuolta ja tietokantaa. Insinöörityön tuotosta testattiin yrityksessä ja kirjattiin siitä löytyneet korjattavat asiat. Kehitysehdotuksia myös tuli kyselyn ja testaamisen myötä, esimerkiksi useamman kuvan lisääminen samaan aikaan ja kuvan lisääminen kuvakirjastoon vedä ja pudota -tyyppisesti omalta tietokoneelta. Päämääränä on saada tuotos korjauksien jälkeen käyttöön.

Avainsanat: kuva, kuvakirjasto, HTML, Jquery, PHP, tekijänoikeus

Abstract

Author:	Jyri Herka
Title:	Improvement of the company's image library
Number of Pages:	29 pages
Date:	16 May 2021
Degree:	Bachelor of Engineering
Degree Programme:	Information and Communication Technology
Professional Major:	Media Technology
Instructors:	Jesse Jokinen, Managing Partner
	Ilkka Kylmäniemi, Senior Lecturer

The purpose of this thesis was to improve the existing image library in the company's tool for education purposes. Searching images with tags and names, dividing images into separate pages, creating folder structure, moving images between folders, adding images to folders and converting png-images to jpg-images were the results of this thesis.

Copyright is an automation that comes to the creator after the work is finished. The right to use images is restricted by Finnish copyright law which means that not all images are accessible for free use. There are exceptions to this law, if the purpose to use the copyrighted image is not commercial. There are a lot of sites on the internet that include royalty free images and anyone can use them.

Local environments support development and ensure that the features that are being developed are separate from the official versions including server and database. There are several platforms to set up a local environment.

Improvement of the image library included frontend and backend, server and database. The improvement of the image library was tested by the company and the found bugs were listed. Further improvement suggestions also arrived through testing and inquiry. The goal is to get the improved image library in use after fixing the bugs.

Keywords: image, image library, HTML, Jquery, PHP, copyright

Sisällys

Lyhenteet

Ajax:	*Asynchronous JavaScript And XML.* Joukko tekniikoita, jotka lisäävät verkkosivujen vuorovaikutteisuutta.
API:	*Application Programming Interface*. Ohjelmointirajapinta.
HTML:	*HyperText Markup Language*. Merkintäkieli, jolla hahmotellaan sivujen rakenne.
JSON	*JavaScript Object Notation.* Objekti, jota käytetään datan säilyttämiseen ja siirtämiseen.
PHP	*PHP Hypertext Preprocessor.* Palvelinpohjainen ohjelmointikieli.

1 Johdanto

Insinöörityön aiheena oli parantaa pienyrityksen nimeltä Apprix koulutustyökalun, Builderin, kuvakirjaston toimivuutta. Kuvakirjaston parantaminen nähtiin aiheelliseksi, koska sieltä puuttui useita kuvien järjestettävyyttä ja haettavuutta helpottavia tekijöitä, ja päämääränä oli edistää kyseisiä ominaisuuksia erilaisin lisäyksin. Kuvien organisoinnin helpottuessa ja haettavuuden syntymisen myötä kaikilla, jotka koulutustyökalun kuvakirjastoa käyttävät, tulee olemaan astetta parempi käyttökokemus.

Tavoitteena oli kehittää parannettu kuvakirjasto entistä muokaten ja siihen ominaisuuksia lisäten, ja lopputulos laitettaisiin yrityksen testattavaksi. Kuitenkin kohderyhmänä olivat kaikki, jotka kuvakirjastoa käyttävät: yrityksen sisäiset henkilöt, tekninen henkilöstö, sisällöntuottajat ja asiakkaat. Tavoitteisiin kuuluivat muun muassa kuvien haettavuus nimellä ja tagilla ja kuvanäkymän järjestäminen, johon kuuluivat kuvien jakaminen sivuihin, kansioiden luominen ja niihin kuvien laittaminen. Builderin koulutuksissa elementtien elävöittämisessä ja koulutusten selkiyttämisessä keskeisessä roolissa ovat kuvat, joten kuvien nopea löytyminen nopeuttaa myös koulutusten tekemistä.

Kuvakirjaston parantamisen valmiiksi saattaminen vaati työskentelyä niin frontendin kuin backendin koodissa. Builderin oma tietokanta ja jQueryn erilaiset metodit olivat keskeisessä osassa, samoin palvelinpuoli ja se, kuinka tiedostot liikkuvat siellä PHP-ohjelmointikielen avulla.

Insinöörityön raportissa tutkitaan kuvia yleisesti ja niiden oikeuksia, kuvia ja kuvakirjastoja internetissä, lokaaleja ympäristöjä kehittämisessä, yrityksen koulutustyökalun rakennetta ja koodia ja lopuksi itse insinöörityötä eli kuvakirjaston parantamista. Etenemistapa raportissa on yleistasolta teknisempään puoleen ja sanastoon.

2 Kuvat, oikeudet ja lokaalit kehitysympäristöt

2.1 Kuvien hankkiminen ennen ja nyt

Ihmiset ovat kautta aikain tarvinneet projekteihinsa kuvia. Ennen internetin saapumista, pääosin 1970-luvulla toimittajat tarvitsivat kuvia lehtiartikkeleihinsa. Näinä aikoina ei ollut kuvapankkeja tarjolla, joten toimittajilla oli mukanaan valokuvaaja, jonka tehtävänä oli ottaa artikkeliin tarvittavat valokuvat. Valokuvaajilla oli vapaat kädet aina intimiteettisuojalain tuloon asti, jolloin se, mitä tai ketä kuvataan, ei ollut enää yhtä vapaata. Intimiteettisuojaa rikotaan kuvien osalta, kun on otettu kuva, joka loukkaa kuvassa olevia ihmisiä.

Suomessa lähestulkoon kaikissa julkisissa paikoissa saa valokuvata ja esimerkiksi taloja, patsaita ja ihmisiä saa valokuvata, mutta esimerkiksi kotirauhan suojaamia paikkoja ja yksityistilaisuuksia ei saa kuvata ilman lupaa. [1.] Kuvausoikeus perustuu Suomen perustuslain toisen luvun 12. pykälään, jossa sanotaan, että jokaisella on sananvapaus ja oikeus julkistaa ja vastaanottaa tietoja, mielipiteitä ja muita viestejä kenenkään ennakolta estämättä. [2.]

Nyt modernisoituneessa maailmassa suurin osa luettavista artikkeleista sijaitsee internetissä, vaikka painettu lehti pitääkin pintansa. Kuvien tarve elävöittämään tekstiä ei ole kuitenkaan muuttunut. Kuvapankkien määrä internetissä on suuri, ja monet niistä ovat tekijänoikeusvapaita, esimerkiksi Pixabay ja Unsplash. Unsplash on sivusto, johon pääasiassa valokuvaajat voivat laittaa ottamiaan kuvia muiden käytettäväksi. Unsplashilla on myös oma API-rajapinta, joka palauttaa dataa JSON-muodossa ja johon täytyy erikseen rekisteröityä. [3.] Omia verkkosivuja tehdessä kuvien helppo upottaminen koodiin ilman, että se vie tiedostotilaa, on kehittämisen kannalta helpottava tekijä, vaikkakin tällöin sivusto on enemmän riippuvainen kolmannen osapuolen sivuston pystyssä pysymisestä. Monesti tekijänoikeusvapaissa sivustoissa, esimerkiksi Pixabayssa, voi omantunnon mukaan maksaa alkuperäisen kuvan ottajalle pienen palkkion ja myös kuvan ottajan Instagram-tili kuvien yhteydessä

löytyy, jotta henkilöä voi seurata. Pixabayta tarkastellaan tässä insinöörityössä luvussa 2.3.

2.2 Tekijänoikeuslaki

Tekijänoikeusvapaat (monesti nimellä ”royalty free”) kuvat ovat tärkeitä niin esitelmille, artikkeleille kuin verkkosivustoillekin. Tekijänoikeus on automaattisesti tekijälle syntyvä asia, joka teokselle tulee sen valmistuttua. Tekijänoikeus ei kuitenkaan ole itsestään selvä asia, vaan teoksen pitää ylittää teoskynnys, millä viitataan siihen, että teoksen on oltava tarpeeksi omintakeinen. Teoksia voivat olla esimerkiksi tietokonepelit, sävellykset tai juuri valokuvat. ETA-alueelta peräisin olevat teokset ovat voimassa 70 vuotta tekijän kuolemasta ja muualta olevat teokset 50 vuotta tekijän kuolemasta [4]. Työntekijän tekijänoikeudet eivät siirry työnantajalle, poikkeuksena tietokoneohjelman tekijänoikeudet. [5.]

Valokuva, joka ylittää teoskynnyksen, on tekijänoikeuslailla suojattavissa. Muut valokuvat suojataan tavallisina valokuvina, ja niiden suoja-aika on lyhyempi. Tekijänoikeusneuvoston mukaan esimerkiksi Kalle Kultalan vuonna 1961

ottama kuva presidentti Urho Kekkosesta Havaijilla suojataan tekijänoikeuslailla (kuva 1).

Kuva 1. Kalle Kultalan vuonna 1961 ottama kuva presidentti Urho Kekkosesta Havaijilla [6].

Jos kyseessä on tekijänoikeuslain pykälän 25 mukainen kuvasitaatti, oikeudenhaltijan lupaa tekijänoikeuden suojaaman teoksen käyttöön ei tarvita. Pykälän 25 nojalla tieteelliseen esitykseen saa lainata kokonaisia kuvia. Tieteellisenä esityksenä pidetään muun muassa yliopiston harjoitus- ja opinnäytetöitä. Jos

käyttö ei ole kaupallista käyttöä, tieteellistä työtä voidaan levittää digitaalisessa julkaisussa [7].

Vuonna 2018 sattui tapaus, jossa valokuvaajan ottamaa kuvaa käytettiin verkkosivustolla hänen suostumuksestaan. Sitten saksalainen opiskelija käytti samaista kuvaa omassa esitelmässään. Kuva myös julkaistiin opiskelijan koulun verkkosivustolla. Valokuvaaja huomasi tämän, vaati koululta korvauksia ja valitti tuomioistuimeen. Euroopan unionin tuomioistuin linjasi, että jos kuvan omistaja on antanut luvan jollekin tietylle sivustolle käyttää kuvaa, kolmas osapuoli ei saa kuvaa käyttää, vaikka se internetissä onkin. Kuva tulee upottaa kolmannen osapuolen sivulle hyperlinkin avulla [8]. Euroopan unionilla on oma tekijänoikeusdirektiivi [9], ja sen jäsenmaiden lait väistyvät aina direktiivien tieltä.

Tekijänoikeuslain pykälä 21 sanoo julkisesta esittämisestä, että julkaistua teosta saa esittää julkisesti esimerkiksi jumalanpalveluksessa ja opetuksen yhteydessä. Teosta saa esittää julkisesti, jos itse teoksen esittely ei ole avain asemassa ja tilaisuutta ei ole järjestetty ansiotarkoituksessa. Jos julkisessa tilaisuudessa on näytillä jäljennös alkuperäisteoksesta, jäljennöksen tekijän nimeä ei saa laittaa niin, että se on sekoitettavissa alkuperäisteokseen. Valokuvaajalla on yksinoikeus muutetun tai muuttumattoman valokuvan määräämiseen julkisesti sitä esittämällä tai valmistamalla siitä kappaleita [10]. Tekijänoikeusrikkomuksista säädetään erikseen lailla.

Yksi tekijänoikeusloukkauksista on piratismi eli luvaton valmistaminen. Se on ilman oikeudenhaltijoiden lupaa tehtyä äänitteiden, elokuvien tai video- ja tietokonepelien valmistamista tai kopiointia. Tekijänoikeuslain rikkomisesta rangaistaan joko tekijänoikeusrikoksena tai -rikkomuksena, ja maksimirangaistus on kaksi vuotta vankeutta. Tekijänoikeusrikokseen

edellytyksiä on kaksi: tekoon on liitettävissä pyrkimys tuottaa tuloja, ja teko aiheuttaa haittaa tai vahinkoa alkuperäisen teoksen oikeuden haltijalle [11].

2.3 CSS-ohjelmistokehykset

Kun tarkastellaan kuvakirjastoja internetissä, yhteistä niille useimmiten on grid- eli ruudukkorakenne ja thumbnail-kuvat eli pikkukuvat. Grid-rakenne on useimmiten toteutettu ohjelmistokehyksellä, joka voi olla esimerkiksi Bootstrap tai Semanctic UI. Ohjelmistokehyksiä on kuitenkin lukematon määrä tarjolla ja niiden tehtävänä on tarjota kehittäjille tyyliltään valmiit ja nopeasti siirrettävät tyylisäännöt omaan projektiin. Tätä monet kehittäjät suosivat, koska kun visuaalisuus on nopeasti valmis, voi keskittyä enemmän projektin teknisiin seikkoihin.

Voidaan ottaa tarkasteluesimerkiksi pixabay.com, tekijänoikeusvapaa kuvakirjasto, johon kuka tahansa voi laittaa ottamiaan tai omistamiaan kuvia. Kun tarkastellaan sivuston käyttämiä koodeja, nähdään, että se ei käytä CSS-ohjelmistokehystä, vaan jQueryn fleximages-liitännäistä, joka on kehitetty yksinomaan Pixabay-sivustolle ja sitä käyttävät myös Flickr ja Google Images. [12.] Tätä kirjastoa voidaan hyödyntää koodissa muun muassa esimerkkikoodin 1 näyttämällä tavalla.

```
$(selector).flexImages({
   key1: value1,
   key2: value2
});
```

Esimerkkikoodi 1. Fleximages-liitännäisen koodin runko Javascript-tiedostossa.

Esimerkkikoodia 1 hyödynnetään Pixabay-sivustolla alkunäkymän koodissa esimerkkikoodin 2 mukaisesti.

```
$('.flex_grid').flexImages({
   rowHeight: 320,
   maxRows: 8,
   truncate: true
});
```

Esimerkkikoodi 2. Liitännäisen avulla asetetut tyylit Pixabay-sivustolla.

Esimerkkikoodista 2 nähdään rivin maksimikorkeus (rowHeight, oletusarvo 180), näytettävien rivien maksimimäärä (maxRows, oletusarvo null) ja se, piilotetaanko viimeinen kuva, jos se ei mahdu riville (truncate, oletusarvo false, toisin sanoen ei päällä automaattisesti), ja luokka, johon koodi vaikuttaa (”flex_grid”).

Yksi suosituimmista ohjelmistokehyksistä on Bootstrap. Sitä käyttävät muun muassa Twitter ja Spotify. Bootstrapilla luodaan valmiiksi responsiivisia sivuja. Ominaista Bootstrapin toiminnalle on, että HTML:ään lisätään esimerkiksi sarakkeen leveyttä määräävät luokat (esimerkkikoodi 3).

```
<div class="container">
   <div class="row">
         <div class="col-sm">
            One of three columns
   </div>
<div class="col-sm">
          One of three columns
 </div>
 <div class="col-sm">
          One of three columns.
  </div>
 </div>
</div>
```

Esimerkkikoodi 3. Bootstrapin grid-logiikka, jossa yksi rivi on jaettu kolmeen sarakkeeseen.

Col-sm tarkoittaa ”column small”, jonka maksimileveys on 540 pikseliä [13]. Ohjelmistokehykset ovat monesti kuitenkin äärimmäisen ylikirjoittavia, joten ”luovalle vapaudelle” tyylin suhteen koodissa ei niinkään aina jää tilaa. Varsinkin tyylit helposti ylikirjoittuvat Bootstrapissa !important-käskyllä, joka määrää tyylit dominoivasti valitulle luokalle, ja lähtökohtaisesti itse tyylejä luodessa ei ole yleisesti hyvän tavan mukaista käyttää tätä käskyä, ellei ole muuta vaihtoehtoa (esimerkkikoodi 4).

```
.container {
     background-color: #000 !important;
}
```

Esimerkkikoodi 4. Sääntö muuttaa luokan container taustavärit mustiksi ja kirjoittaa yli kaikki muut tyylit, jotka yrittävät määrätä toisin.

2.4 Lokaalit ympäristöt kehittämisen tukena

Lokaaleja ympäristöjä käytetään, jotta ei tarvitsisi kehittää projektia suoraan palvelimelle. Käytännössä suoraan palvelimelle kehittäminen on mahdollista ja jopa monissa tapauksissa nopeampaa viedä näyttövalmiiksi, toisin kuin lokaalissa ympäristössä kehitetty osa, sillä lokaalista ympäristöstä siirtäminen saattaa viedä aikaa, eikä aina myöskään versionhallinnan konflikteilta vältytä. Kuitenkin lokaaliin ympäristöön kehittäminen on aina turvallisempaa, koska silloin ympäristö on erillään palvelimen ympäristöstä, jolloin mahdolliset virheet eivät joudu palvelimelle.

Eniten mahdollisia virheitä oman kokemuksen mukaan tapahtuu backend-puolella. Esimerkiksi kun kehitetään suoraan palvelimelle, frontend on monesti anteeksiantavampaa kuin backend. Jos yhteistä kantaa muokkaa moni kehittäjä, voi tulla muuttaneeksi asioita esimerkiksi taulurakenteissa, mikä voi haitata toisen kehittäjän projektin etenemistä. Tietysti tiimin kommunikaatiolla on tässä kriittisen tärkeä rooli, mutta erillisillä lokaaleilla ympäristöillä minimoidaan mahdolliset konfliktitilanteet, vaikka kuten edellä mainittu, eivät lokaalitkaan ympäristöt estä versionhallinnan niin sanottuja konflikteja eli kahden versionhallinnan version koodin päällekkäisyyksiä. Lokaalien ympäristöjen vaihtoehtona voi olla yksinkertaisesti suoraan palvelimelle kehittäminen, mutta tällöin saattaa tulla ongelmia versionhallinnan kanssa. On myös viisasta ladata koodieditoriin liitännäinen, joka synkronisoi keskenään lokaalin ja palvelimen version; esimerkiksi Visual Studio Codeen liitännäisiä löytyy useampia. Esimerkkejä lokaaleista kehitysympäristöistä ovat XAMPP ja Docker.

2.4.1 XAMPP-kehitysympäristö

XAMPP on suosituin lokaali kehitysympäristö, joka auttaa kehittämään projekteja lokaalilla web-palvelimella. Se on saatavilla Windowsille, macOS:lle ja Linuxille yhdellätoista kielellä. XAMPP on lyhenne, ja se koostuu sanoista X – cross platform (järjestelmäriippumaton), A – Apache (avoimeen lähdekoodiin perustuva HTTP-palvelinohjelma), M – MYSQL (relaatiotietokanta), P – PHP (palvelinpuolen ohjelmointikieli) ja Perl (yleismaailmallinen ohjelmointikieli, alun perin kehitetty tekstin käsittelyyn, mutta nykyään käytetään web-kehitykseen) [14].

XAMPP on nopea ja helppo asentaa koneelle, ja sen mukana tulee paneeli, jonka avulla sitä on helppo käyttää. Asennuksen myötä tulee kiinnittää huomiota httpd.conf-tiedostoon, jossa määritellään, mikä polku selaimessa aukeaa, ja myös, mitä porttia käytetään. Portti voi olla mikä vain, kunhan se ei ole varattu muun ohjelman toimesta. Polku selaimessa voi olla esimerkiksi *localhost:3000/Documents*, jolloin selain aukeaa koneen Documents-kansioon ja portti on 3000. Hallintapaneelit ovat melko samanlaiset, kun niitä verrataan macOS-käyttöjärjestelmässä ja Windowsissa, mutta yhtenä suurimmista eroista näiden kahden käyttöjärjestelmän välillä on se, että macOS-käyttöjärjestelmässä viimeisimmät versiot ovat virtuaalikoneita. Tässä on etuna se, etteivät projekti ja sen tiedostot vie omalta tietokoneelta tilaa. Insinöörityöprojektin kehittämiseen käytettiin XAMPP:a.

2.4.2 Docker-kehitysympäristö

Docker on toinen suosittu lokaali kehitysympäristö, johon voidaan myös asentaa oma hallintapaneeli. Dockerissa on mahdollista suorittaa omaa kehitettävää projektia erillisessä kontissa, jossa on kaikki tarvittava sovelluksen suorittamista varten. Kontteja voi myös jakaa muiden henkilöiden kanssa niin, että sovellus toimii kontissa muillekin samalla tavalla. Image on read-only-pohja, joka antaa ohjeet Docker-kontin luomiseen. Imagen luomiseen tarvittaviin askeliin luodaan Dockerfile. [15.] Dockerin kontteja on mahdollista myös laittaa kommunikoimaan

toistensa kanssa, jos projektissa tarvitaan esimerkiksi tietokantaa. Tämä onnistuu laittamalla molemmat tai tarvittaessa useampi kontti saman tietoverkon alle. Tietoverkko pitää luoda erillisellä käskyllä.

Kaikista kolmesta, XAMPP, suoraan palvelimelle kehittäminen ja Docker, Docker on kaikista ketterin kehitysympäristö. Kuitenkaan yrittäessäni yhdistää Builderin tietokantaa sisältävän kontin ja Builderin projektin sisältävän kontin, asiat eivät edenneet kirjautumissivulta eteenpäin. Vaihtoehdoksi jäi XAMPP, vaikka ajoittain siinä on ongelmia versionhallinnan ja tiedostojen oikeuksien kanssa (käytössä oli macOS Big Sur). Dockerin pystyttämiseen kannattaa panostaa tulevaisuudessa.

2.4.3 Kehittäminen suoraan palvelimelle

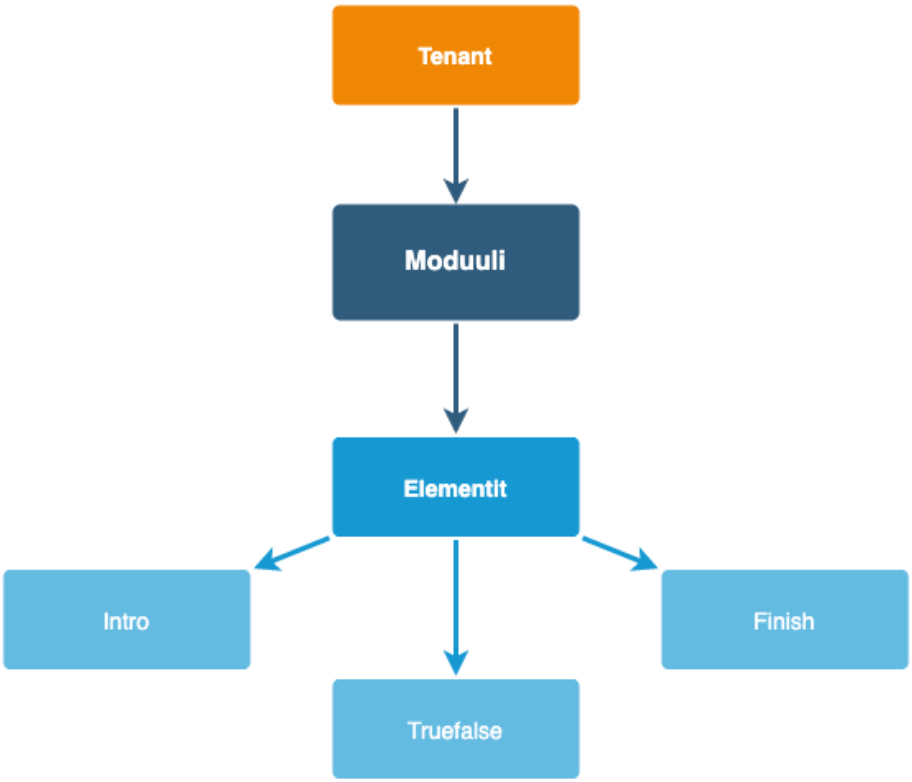
Lokaalien kehitysympäristöjen vaihtoehtona on kehittäminen suoraan palvelimelle. Muutosten tekeminen ja sen jälkeen tiedostojen vieminen palvelimelle on aikaa vievää, ja siksi on olemassa erilaisia liitännäisiä. Visual Studio Code -editoriin löytyy liitännäisiä, joista yksi esimerkki on SSH FS, jonka konfiguraatiotiedostoon voi helposti asettaa tarvittavat tiedot, polun, palvelimen ja portin. Vaikka liitännäisen avulla voi muokata tiedostoja suoraan palvelimelle ja muutokset tulevat versionhallintaan lokaalisti, ei versionhallinnan muutoksista jäänyt jälkeä palvelimen versionhallintaan. Tämä saatiin tietää suorittamalla terminaalissa komento git status.

Vaikka suoraan palvelimelle kehittäminen sisältää omat ongelmansa, koodieditorin liitännäisellä tai ilman, tällä vaihtoehdolla muutokset ovat näytettävissä saman tien. Erillisiä kehitysympäristöjä palvelimella on myös hyvä olla useampia, jotta kaikki pääsevät kehittämään ilman, että kehitysympäristö on mahdollisesti varattu jonkun toisen käyttöön samanaikaisesti.

3 Builder-koulutustyökalu

3.1 Builder käyttöliittymänä

Jotta voidaan ymmärtää Builderia, täytyy myös ymmärtää käsitteet, jotka tulevat vastaan sen myötä. Builder koostuu tenanteista eli asiakkaiden omista osioista. Tenanteissa on moduuleita eli koulutuksia, joissa on erilaisia elementtejä. Moduulissa on vakioina intro-elementti ensimmäisenä ja finish-elementti viimeisenä (kuva 2).



Kuva 2. Tenantin rakenne.

Näiden elementtien paikkaa ei voi vaihtaa, vaan niiden väliin lisätään erilaisia tehtävä- tai infotyyppisiä elementtejä. Elementti voi olla esimerkiksi totta vai tarua -tyyppinen elementti, slider-elementti mielipidemittaukseen tai palaute-elementti. Elementit lisätään moduuliin (koulutukseen) painamalla alasvetovalikko esiin ja pudottamalla se vedä ja pudota -tyyppisesti haluttuun kohtaan. Koulutusta voi tarkastella preview-näkymässä, jossa on mahdollista

hyppiä sivujen läpi CTRL + ALT + vasen/oikea nuoli -yhdistelmällä. Koulutuksen julkaistussa versiossa tämä ominaisuus ei mahdollistu, ja julkaistu versio on lopullinen versio koulutuksesta. Niin moduuleita kuin elementtejäkin voi poistaa, ja moduuleita voi myös duplikoida.

3.2 Builderin rakenne ja tekniikka

Builder käyttää MySQL-tietokantaa ja ohjelmointikielinään jQuerya ja PHP:ta. PHP-kielen ohjelmointikehys on Zend Framework. Builder hyödyntää kehitysympäristöille tarkoitettuja omia ympäristöjä palvelimella (b_dev, b_dev_1, b_dev_2), b_test ja b, joista jälkimmäinen on asiakkaiden käyttämä tuotantopuoli. Uusien ominaisuuksien kehitykseen käytetään lokaalia kehitysympäristöä. Eri ominaisuuksien implementaatiossa pyritään toimimaan niin, että ensimmäisenä ominaisuus laitetaan testattavaksi kehityspuolelle (b_dev-ympäristöt) ja korjataan mahdolliset viat. Tämän jälkeen ominaisuus viedään pull requestin eli pyynnön kautta liittää oma kehityshaara haluttuun haaraan testipuolelle (b_test), joka toimii tuotantopuolen (b) esiasteena. Lopulta uusi ominaisuus liitetään master-haaraan ja viedään tuotantoon. Builder käytti Bitbucketia versionhallinnassaan ja vastikään keväällä 2021 siirtyi Githubin käyttöön. Jokaiselle ominaisuudelle luodaan oma haara.

Pääosin Builderin frontendia käsittelee builder_orig.js-niminen tiedosto, joka käyttää Javascriptin toteutukseen jQuerya. Jotta tämä tiedosto voidaan saada käyttökelpoiseksi, se pitää minifoida eli poistaa suorittamisen kannalta kaikki tarpeeton. Tähän käytetään joko internetistä löytyvää ilmaista minifieria tai omalle koneelleen voi myös asentaa Uglifyjs-minifierin, joka on tarkoitettu minifoitavien tiedostojen luomiseen. Uglifyjs kannattaa asentaa globaalisti, jottei projektiin tule ylimääräisiä tiedostoja (esimerkkikoodi 5).

```
npm install uglify-js -g
```

Esimerkkikoodi 5. Käsky, joka laitetaan terminaaliin, jotta voidaan asentaa Uglifyjs globaalisti.

Kun Uglifyjs on asennettu, voidaan haluttu tiedosto minifoida. Käskyssä ensin tulee tiedosto, joka halutaan minifoida, minkä jälkeen laitetaan –output tai lyhyemmin -o ja sen jälkeen minifoidun tiedoston nimi (esimerkkikoodi 6).

```
uglifyjs builder_orig.js -o builder.js
```

Esimerkkikoodi 6. Builderin pääasiallisen Javascript-tiedoston minifointikäsky Uglifyjs:llä.

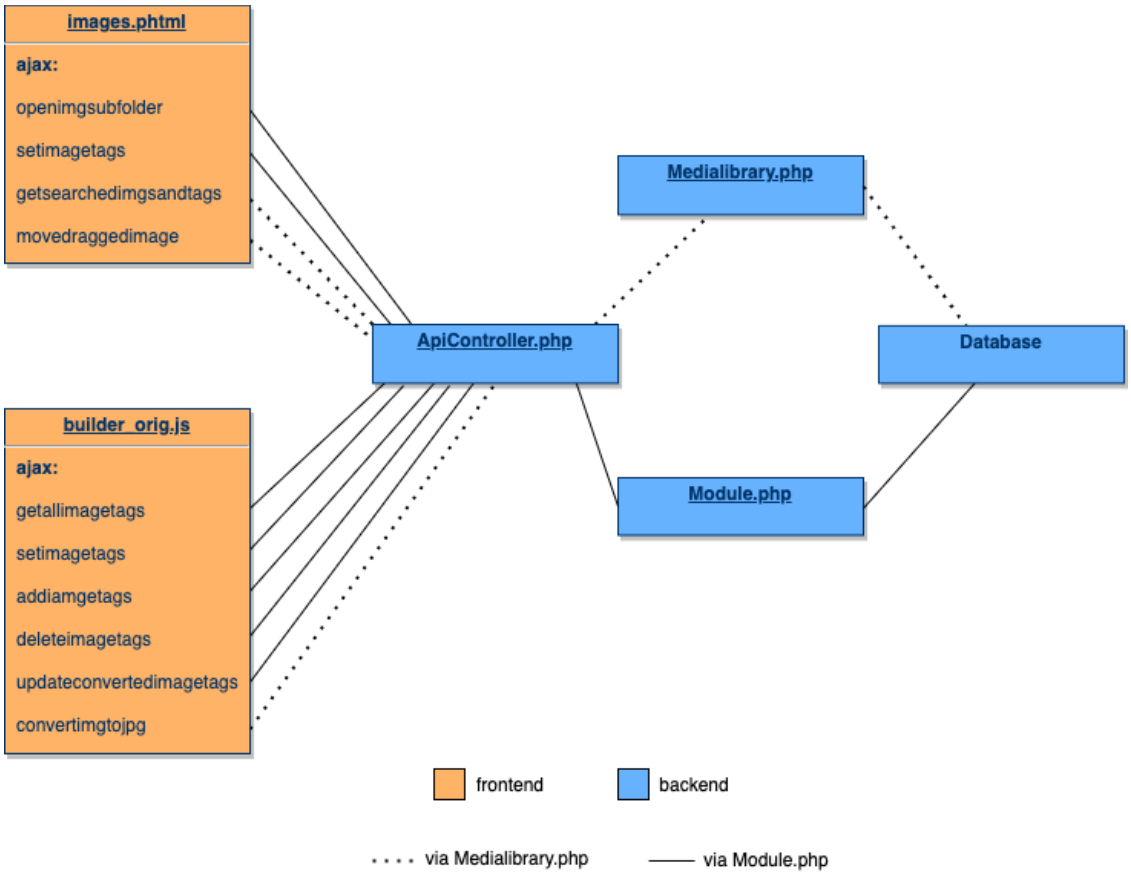
3.3 Zend Controller-ohjelmistokehys

Zend Controller (nykyiseltä nimeltään Laminas) on PHP-ohjelmistokehys, joka on suunniteltu olemaan kevyt, modulaarinen ja laajennettava. Zend-työnkulku sisältää useita eri komponentteja, joista Zend_Controller_Front järjestää tiedostojen työnkulun ja prosessoi kaikki pyynnöt palvelimelta ja on vastuussa pyyntöjen jakamisesta ActionControllereille (Zend_Controller_Action). Zend_Controller_Front kutsuu Zend_Controller_Router_Rewritea päättääkseen, mitä ohjainta ja ohjaimessa olevaa toimintoa tarvitaan lähetykseen. Zend_Controller_Router_Rewrite pilkkoo merkkijonon asettaakseen ohjaimen ja toiminnon nimen pyyntöön, minkä jälkeen Zend_Controller_Front-komponentti lähettää kutsusilmukan, joka kutsuu Zend_Controller_Dispatcher_Standard-komponenttia ja lisää sen pyyntöön lähettämään ohjaimeen ja toimintoon, jotka on määritetty pyynnössä. Kun ohjain on valmis, palautuu hallinta takaisin Zend_Controller_Front-komponentille [16].

3.4 Kuvakirjasto Builderissa

Builderin koulutusten (moduulien) elementteihin voi lisätä kuvia kuvakirjastosta kaikkiin niihin elementteihin, joihin niitä on mahdollista lisätä. Builderissa kuvakirjastoa ohjataan images.phtml-tiedostosta ja builder_orig.js-tiedostosta. Itse kuvakirjaston dialogi luodaan builder_orig.js-tiedostossa. Monesti tämä

aiheuttaa hankaluuksia, sillä esimerkiksi kuvakirjaston kehyksen ympärillä olevan dialogin klikkitapahtumat eivät ole käytettävissä suoraan images.phtml-tiedostossa. Builder_orig.js-tiedostossa luodaan dialogi-ikkuna kuvakirjaston kehyksen ympärille. Kuitenkin builder_orig.js-tiedoston valitsimiin on mahdollista päästä käsiksi images.phtml-tiedostosta $(window.parent.document).find()-valitsimella. Näissä kahdessa tiedostossa tapahtuu insinöörityön frontend (kuva 3). Kuvakirjasto avataan aina elementin oikeassa yläkulmassa sijaitsevasta kuva-kuvakkeesta.



Kuva 3. Insinöörityössä käytetyt tiedostot, lisätyt kutsut ja se, mitä kautta ne menevät.

Jqueryn dialogi on ikkuna, joka sisältää otsikon ja sisältöalueen (kuva 4). Dialogissa voi käyttää erilaisia valintoja (options), metodeja ja tapahtumia. Valintoihin lukeutuvat muun muassa title, draggable, resizable ja position. Title spesifioi dialogi-ikkunan otsikon ja on oletusarvoltaan null, ja tällöin otsikko myös

näkyy. Draggable on tyypiltään totuusarvomuuttuja, joten se voidaan asettaa vain kahteen arvoon: true tai false [17].

Kuva 4. Kuvakirjaston näkymä Builderissa ennen opinnäytetyön tekemistä.

Kun on asetettu arvo true, ikkuna on raahattavissa. Resizable on tyypiltään myös totuusarvomuuttuja, ja kun arvo on asetettu true, dialogi-ikkuna on skaalattavissa. Position spesifioi dialogin paikan sivulla, kun se aukeaa ja on tyypiltään objekti (esimerkkikoodi 7). Oletusarvoisesti dialogi on keskitetty.

```
$('.selector').dialog({
   title: 'Dialog Title',
   draggable: true,
   resizable: false,
   position: { my: 'left top',
               at: 'left bottom',
               of:  $('.container')
             }
});
```

Esimerkkikoodi 7. Dialogi-ikkunan ehtoja.

Metodeja dialogissa voivat olla esimerkiksi close ja open, jotka sulkevat ja avaavat dialogin (esimerkkikoodi 8), ja destroy, joka poistaa kaikki dialogin toiminnallisuudet. Tapahtumia voivat olla esimerkiksi drag ja focus. Drag toteutuu, kun dialogia raahataan hiiri pohjassa, ja focus toteutuu, kun hiiren kohdistin kohdistetaan dialogiin.

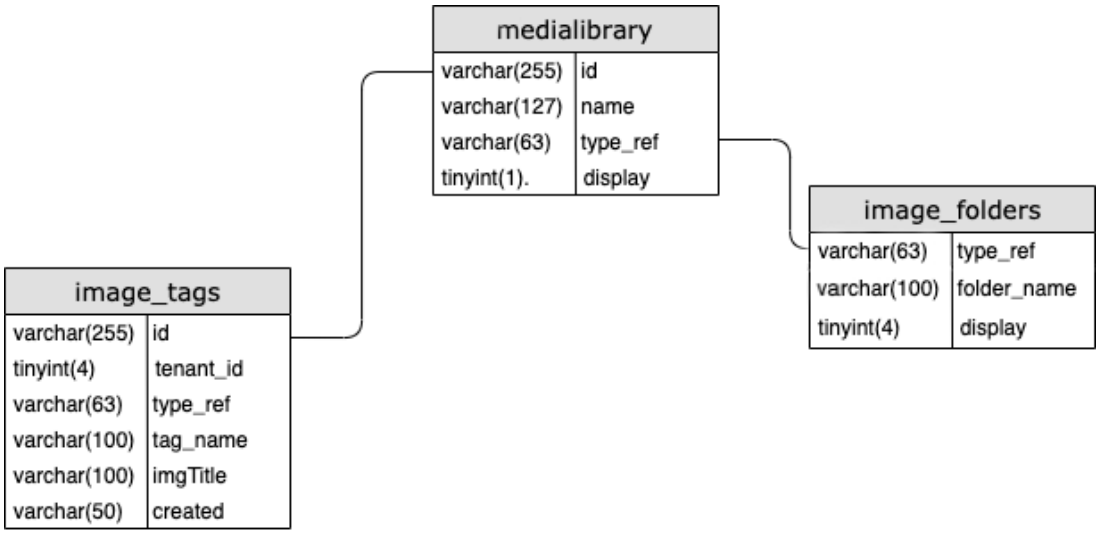
```
$('.selector').dialog('close');
```

Esimerkkikoodi 8. Dialogin sulkemiskäsky.

4 Kuvakirjaston parantaminen

4.1 Tarpeet

Builderin kuvakirjastosta puuttuu monia käyttömukavuutta edistäviä tekijöitä: nimet kuvissa (tai mahdollisuus nähdä nimet helposti), kuvien hakutoiminto, varmistaminen ennen kuvan poistamista (aiheuttaa helposti vahinkopoistoja) ja mahdollisuus rajata kuvia, joita haluaa nähdä, esimerkiksi pysty- tai vaakakuvat. Luvuissa 4.2–4.5 havainnollistetaan, mitä lopputyössä kehitettiin, mitä se vaati teknisesti – mitä ohjelmointikieliä, tietokantarakennetta, tiedostoja, ongelmia, joita tuli matkan varrella ja mitä kehitetään tulevaisuudessa. Lähtökohtana oli, ettei muuteta olemassa olevia tietokantatauluja, ja näin ollen pystyin olemaan varma siitä, että työ pysyy täysin erillisenä projektina tietokannan suhteen. Työhön täytyi luoda kaksi uutta taulua tietokantaan: image_tags, joka säilyttää kuvien tagit, ja image_folders, jossa on listattu tenanteille luodut kansiot (kuva 5).



Kuva 5. Builderin kuvakirjaston parannusta varten tietokantaan lisätyt taulut ja olemassa ollut taulu medialibrary.

4.2 Hakutoiminnallisuus nimillä ja tageilla

Insinöörityössä oli ensimmäiseksi tarkoituksena luoda hakutoiminnallisuus kuvien nimillä ja tageilla. Builderin tietokannassa oli jo taulu nimeltään medialibrary, johon kuvat kannassa tallennetaan. Medialibrary-taulussa on sarake nimeltä id, joka on muotoa *img_u38uMSJl/uploaded/kuva.png*. Kuvan id:stä nähdään tenantin id (*u38uMSJl*), kansio, jossa kuva sijaitsee (uploaded), ja kuvan nimi (kuva.png). Tästä sarakkeesta medialibrary-taulusta oli helppo parsia kuvan nimi kyselyyn.

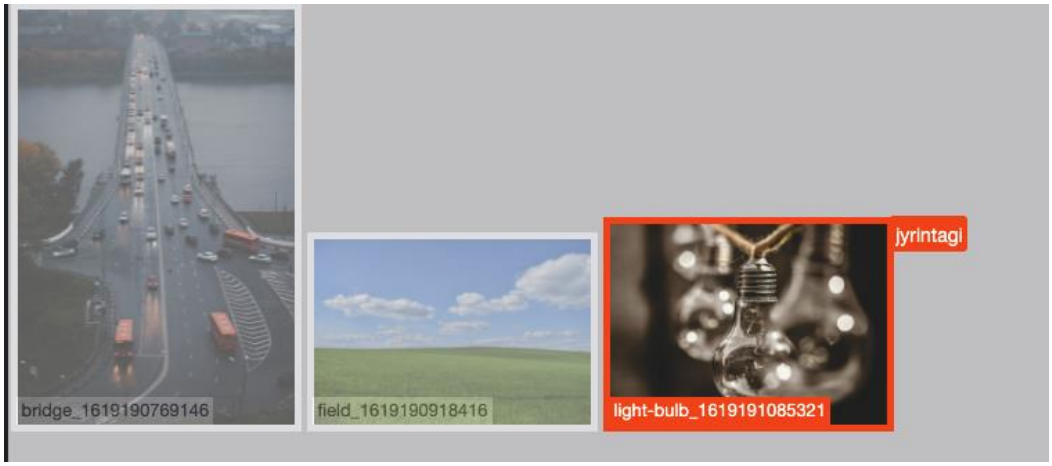
Kun kuvan nimi oli noudettavissa, piti seuraavaksi miettiä, mihin kuvan tagit tallennetaan, ja syntyikin niille oma taulu nimeltään image_tags. Aluksi kuvan nimien ja tagien haku oli erillisinä Ajax-kutsuina keyup-handlerissa hakukentässä. Toisin sanoen enteriä painaessa nämä kaksi hakua toteutuivat, mutta koska Ajax-kutsut ovat asynkronisia, eivät kutsut aina toimineet kuten piti: hakutuloksia saattoi jäädä noutamatta. Lopulta ratkaisu oli yhdistää kahden taulun hakutulokset yhteen eli tässä tapauksessa image_tags-taulun ja medialibrary-taulun hakutulokset LEFT JOIN-käskyllä. Tässä käskyssä on oltava kaksi taulua yhdistävää saraketta ja kyseisiä tauluja yhdistää sarake id.

Nyt kaksi taulua yhdistettäessä hakutulos katsoo samanaikaisesti niin nimiä kuin kuviin asetettuja tageja. Images.phtml-tiedostossa katsotaan Ajax-kutsun tulosta ja verrataan niitä HTML:sta löytyviin näkyvissä oleviin kuviin, joiden luokan nimi on ”lib-image”, ja jQueryn valitsimella ne ovat helposti valittavissa: $(”.lib-image”), ja kuvan src:stä parsitaan kuvan nimi. Tageja haettaessa palautuu Ajax-kutsusta kuvien nimiä, sillä kysely katsoo tagien rivistä kuvien nimet. Kuvat, jotka eivät vastaa hakutuloksia, piilotetaan.

Hakukentän ollessa tyhjä ja enteriä painettaessa kaikki kuvat tulevat näkyviin. Jos halutuloksia ei ole, tulee sivulle teksti ”No results found”. Jotta tageja voidaan lisätä kuviin, täytyi niille lisätä oma dialogi-ikkuna, jossa niitä voi lisätä ja poistaa. Dialogia ei voi luoda images.phtml-tiedostosta, joten se täytyi luoda builder_orig.js-tiedostosta. Tämä aiheutti aluksi ongelman, sillä kuvat sijaitsevat

kehyksessä, jonka images.phtml luo, ja kuten edellä mainittu, kehys on dialogi-ikkunan sisällä, jonka builder_orig.js-tiedosto luo. Täytyi siis olla valitsin, joka hakee images.phtml-tiedostosta kuvakirjastossa sijaitsevan kuvan luokan ”lib-image”.

Kun tekninen toiminnallisuus kuvien tageille oli mietitty, kuvien visuaalinen ilmaantuvuus tuli mietinnän kohteeksi: Miten lisätyt tagit tulisivat näkymään kuvissa? Loppuratkaisuksi tuli lisätä ne kuvien oikealle puolelle oransseina merkkeinä, jotka animoituvat yksi kerrallaan. Animaatio toteutettiin jQueryn animaatioilla ja silmukalla (kuva 6). Aiemmin kuvakirjastossa, kun kuvan oli valinnut, ei valintaa voinut enää poistaa klikkaamalla esimerkiksi kuvasta pois. Yhtenä parannuksena tämä otettiin myös huomioon, ja valinnan voi nyt poistaa klikkaamalla kuvien taustaa. Myös valittua kuvaa nyt painotetaan visuaalisesti säätämällä alemmas ei-valittujen kuvien läpinäkyvyyttä.



Kuva 6. Kuvakirjastossa valittu kuva, jossa näkyy lisätty tagi ja kuvan nimi.

Tageja lisätessä kenttä antaa myös ehdotuksia sen mukaan, mitä on kirjoittanut, ja tämä helpottaa olemassa olevien tagien lisäämistä ja myös sitä, että mahdollisesti vaikeat sanat kirjoitetaan yhteneväisellä tavalla. Tämä toteutettiin jQueryn autocomplete-toiminnallisuudella, jossa input-hakukenttä valitaan valitsimella ja autocomplete antaa alasvetovalikon hakukentän alle samankaltaisista vaihtoehdoista kuin hakukentän arvo. Alasvetovalikon vaihtoehdot on asetettu arrayna eli taulukkona, jonka jQueryn autocomplete

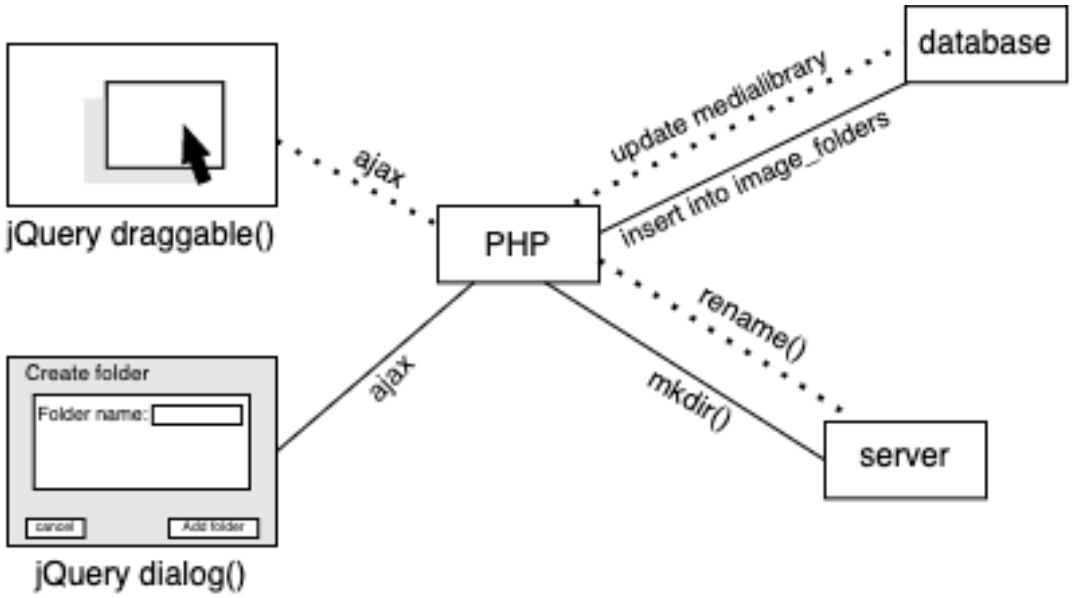
poimii source-valitsimella. Yksi pienistä parannuksista oli myös kuvan poistamisen varmistaminen. Ennen kuvakirjastossa kuvan pystyi poistamaan ilman varmistusta. Tähän mennessä vahinkopoistoja on tullut muutamia. Varmistus tapahtuu confirm-metodilla. Metodi antaa alert-ikkunan OK- ja peruuta-valinnoilla. Kun valintaa OK painetaan, metodi palauttaa true, ja peruuta-valintaa painettaessa false. Confirm-metodia tukevat kaikki selaimet. [18.]

4.3 Kansiot kuvakirjastossa

Yksi tarpeista oli saada kansiot kuvakirjastoon. Toteutus lähti liikkeelle frontendista: miten kansioita voisi lisätä kuvakirjastoon ja miltä ne siellä näyttäisivät. Kansio lisätään kuvakirjaston yläreunassa olevasta rataskuvakkeesta, jonka päälle hiiren viedessä tulee esiin alasvetovalikko ”Create new folder”. Sitä klikkaamalla aukeaa dialogi, jossa on kenttä, johon kirjoitetaan kansion nimi. Klikkaamalla painiketta ”Add folder” tapahtuu kysely, joka lisää kansion tiedot image_folders-tauluun, luo palvelimelle kansion ja lisää kansion kuvakirjaston alkuun frontendissä. Images.phtml-tiedostossa ollut scandir antoi varoituksen uploaded-kansion sisällä olleista alikansioista, joten piti luoda kaksi erillistä silmukkaa, joista toinen katsoo kuvat ja toinen katsoo kansiot ja lisää HTML:t kehykseen. Kaksoisklikkaamalla alikansiota se aukeaa ja näkyy myös sen mahdollinen sisältö. Alikansion sisällä kansiokuvake näkyy alussa, mutta kansion nimenä on kolme pistettä merkkinä siitä, että siitä klikatessa palaa takaisin juureen. Alikansion kaksoisklikkaus juuressa vaati oman Ajax-kutsun, jolla skannataan sen sisältö.

Ominaisuus, joka haluttiin myös kuvakirjastoon, oli kuvien siirtäminen juuresta alikansioon ja päinvastoin. Alikansioon on mahdollista ladata kuvia, kuten juureenkin, mutta toive oli, että kuvia saisi kansioiden välillä liikutettua vedä ja pudota -tyyppisesti. Frontendissa tämä mahdollistui jQueryn draggable- ja droppable-toiminnoilla, joissa kansiokuvakkeeseen asetettiin droppable ja kuvakirjaston kuvaan draggable. Pudottaessa kuva alikansioon tapahtuu animaatio, jossa kuva kutistuu kansion sisään. Kun frontend oli valmis, täytyi

miettiä, kuinka saada kuvat palvelimella liikkumaan kansioiden välillä. Lopulta tämä mahdollistui PHP:n funktiolla rename, joka muuttaa palvelimella olevan kuvan polun (kuva 7).



Kuva 7. Kuvakirjaston kansion luominen ja kuvan siirtäminen vedä ja pudota -tyyppisesti

Jos rename-funktiota käyttäessä uusi tiedostonimi on olemassa, se kirjoitetaan yli, kun taas kansion uudelleen nimeämiseen käytettäessä funktio antaa varoituksen. [19.] Kuvan polun muuttamiseen tarvittiin oma Ajax-kutsu, jossa parametreina olivat kuvan vanha polku, uusi polku, kuvan vanha id ja uusi id, sillä kuvan id-kenttää medialibrary- ja image_tags-tauluissa oli päivitettävä, koska id-sarakkeen arvo tulee tauluihin osittain kuvan polusta. Kansioita on mahdollista poistaa siinä missä kuviakin, ja se tapahtuu klikkaamalla kuvakirjaston alareunasta nappia ”remove selected from library”. Kun kansion on poistanut kuvakirjastosta, se ei konkreettisesti poistu palvelimelta, vaan image_folders-tauluun tulee display-sarakkeen arvoksi kansion kohdalle 0 ja näin ollen kansio on helppo palauttaa muuttamalla sen arvoksi 1, joka

tarkoittaa, että kansio on nähtävissä kuvakirjastossa. Kuvakirjaston kuvat toimivat samalla periaatteella.

4.4 Sivutus ja hakuehdot

Kuvakirjasto tarvitsi myös sivutuksen eli kuvien jakamisen omiin sivuihin ja mahdollisuuden selata sivuja klikkaamalla sivun numeroa tässä tapauksessa kuvakirjaston yläpuolella. Sivutus toteutui puhtaasti frontend-toteutuksella, jossa määritetään silmukalla maksimimäärä yhdelle sivulle.

Sivutuksen lisäksi toteutettiin alasvetovalikko, jossa vaihtoehtoina on kuvan määrän muuttaminen sivulla. Tässä alasvetovalikossa on kolme arvoa, jotka ovat kuvien määrä jaettuna yhdellä, kahdella ja neljällä. Yksi tarve kuvakirjaston kehittämisessä oli myös hakuehtojen lisääminen. Hakuehdot aukeavat klikkaamalla kuvakirjaston oikeassa yläkulmassa olevaa hampurilaiskuvaketta, jolloin vaihtoehtoikkuna animoituu auki. Vaihtoehdot, jotka ikkuna tarjoaa, ovat seuraavat: näytä vain pystysuorat tai vaakasuorat kuvat, piilota nimet kuvissa, järjestä uusimmasta vanhimpaan ja vanhimmasta uusimpaan. Pysty- ja vaakasuorien kuvien näyttäminen tapahtuu vertaamalla kuvien korkeutta ja leveyttä toisiinsa – käydään koodissa kuvakirjaston kuvat läpi ja tarkistetaan, kumpi arvoista on suurempi. Kuvien nimien piilottaminen tehtiin vain jQueryn hide-komentoa käyttäen.

Kuvien järjestäminen uusimmasta vanhimpaan ja päinvastoin olikin haastavampaa. Koska medialibrary-taulussa ei ole aikaleimaa, milloin kuvat on lisätty enkä halunnut muokata olemassa olevia tauluja, täytyi siis löytää tapa järjestää kuvat niiden lisäysaikojen mukaisesti. Lopulta ratkaisuksi tuli katsoa kansiossa olevien kuvien ajat PHP:n filemtime-funktiolla. Luodaan array, johon laitetaan kuvien nimet ja lisäysajat. Tämän jälkeen luodaan PHP-funktio, joka järjestää kuvat aikojen mukaan ja palauttaa järjestetyt arvot. Nyt kuvien nimet ovat järjestyksessä ajan mukaan ja ne asetetaan HTML:ään diveiksi, joiden luokka on ”img-time-result”. Kun valitaan kuvien järjestäminen uudelleen uusimmasta vanhimpaan tai päinvastoin, käydään läpi kuvakirjastossa olevat

kuvat ja kuvat asetetaan sivulle uudelleen ”img-time-result”-luokan arvojen mukaisesti verraten niitä kuvien nimiin.

Kuvien järjestäminen lisäämisajan mukaan olisi ollut helpompaa toteuttaa, jos medialibrary-taulussa olisi vaikka created-sarake, johon aina kun tauluun tulee uusi rivi, lisättäisiin aikaleima PHP:n NOW()-komennolla. Tällöin päivämäärät olisi helppo hakea Ajaxilla. Tulevaisuudessa voisi luoda skriptin, joka käy läpi uploaded-kansion kuvat tenant kerrallaan, kerää kuvatiedostoista päivämäärät ja laittaa ne kyselyllä kantaan oikeisiin kohtiin. Tällöin kyselyssä, jolla noudetaan tulokset halutulla arvolla, voi laittaa ”order by kolumnin_nimi asc”, jolloin haluttu järjestys hakutuloksessa on valmis eikä Ajaxissa frontendissa tarvitse sitä tehdä.

4.5 Kuvan muuntaminen jpg-kuvaksi

Kuvakirjastoon haluttiin myös toiminnallisuus, jossa on mahdollista muuntaa png-kuvia jpg-kuviksi. Tarvittiin yksi Ajax-kutsu builder_orig.js-tiedostoon lisää, ja siihen annettiin parametreiksi kuvan polku ja tenantin id. Medialibrary.php-tiedostossa täytyi myös funktion alussa tehdä tarkistus, onko kuva png PHP:n omalla funktiolla exif_imagetype vertaamalla sitä ehtolauseessa arvoon IMAGETYPE_PNG.

Tämän jälkeen kuva luodaan jpg:ksi PHP:n omalla funktiolla imagecreatefrompng. Funktio ei kuitenkaan poista aiempaa png-kuvaa palvelimelta, joten uudelle jpg-kuvalle täytyi myös luoda oma rivi medialibrary-tauluun ja päivittää taulun riviä, jossa png oli eli asettaa display arvoon 0. Kun kuva muutetaan, täytyi myös kiinnittää huomioita siihen, onko kuvalla tageja, koska tagien tuli säilyä samassa kuvassa, vaikka kuvamuoto muuttuikin. Tälle toiminnallisuudelle oli selkeämpää luoda oma ajax-kutsu builder_orig.js tiedostoon, jossa parametreina ovat tenantin id, kuvan polku ja kuvan title. Module.php-tiedostossa on kysely, joka päivittää image_tags-taulun kentät vastaamaan muutettua jpg-kuvaa.

4.6 Testauksen tulokset

Insinöörityön tulos laitettiin b_dev_2-ympäristöön testattavaksi. Luotiin koulutusmoduuli, jossa oli ohjevideo kuvakirjaston testaamiseen ja linkit tehtävään, johon pystyi lisäämään alitehtäviä löydetyistä virheistä, ja kyselyyn, johon pyydettiin vastaamaan testaamisen jälkeen. Kysely toteutettiin Google Formsilla, ja siinä kysyttiin henkilön työnkuvaa yrityksessä, mitä hyvää uudet ominaisuudet tuovat, mitä voisi parantaa ja kehitysehdotuksia.

Kyselyyn vastasi ja testaukseen osallistui noin kaksi kolmasosaa pienyrityksestä eli 10 henkilöä, ja vastaajien joukossa oli koodareita, sisällöntuottajia, graafikko ja yhteyspäälliköt. Insinöörityö sai kiitosta kuvakirjaston käytettävyyden selkeyttämisestä, käyttömukavuuden parantamisesta ja kuvien organisoinnin ja haettavuuden mahdollisuuksista ja helpotuksesta.

Itse huomasin heti ensimmäisenä lokaalista kehitysympäristöstä laittaessani insinöörityötä b_dev_2-ympäristöön, että kaikilla kuvilla Builderissa ei ole medialibrary-taulussa tietokannassa id-kenttää. Tämä johtunee siitä, että kuvia ei ole laitettu upload-toiminnallisuuden kautta, vaan ne on suoraan laitettu palvelimelle. Hakutoiminnallisuus on sidonnallinen kuvan id-kenttään medialibrary-taulussa, joten laitoin testattavaksi yhden tietyn tenantin, johon latasin noin 50 kuvaa Pixabay-sivulta upload-toiminnallisuuden kautta. Jatkossa hakutoiminnallisuutta voisi hioa niin, että laittaa kuvahaun Ajax-kutsuun ehdon, jossa verrataan hakutulosta myös frontendissa kuvien nimiin.

Korjaustehtäviä tuli useita. Yksi korjattavista asioista oli muun muassa kansion nimeämiseen liittyvät asiat. Kun kansion nimeen laittoi välilyöntejä tai heittomerkin, koodi ei toiminut täysin oikein. Toisin sanoen polku esimerkiksi rename-funktiossa, kun siirretään kuva vedä ja pudota -tyyppisesti ja funktiossa, jossa avataan kansio ei toiminut, mikä aiheutti sen, ettei kuvien siirtäminen kyseisiin kansioihin toiminut. Tämän voisi korjata jatkossa regexillä eli kaavalla, jolla haetaan merkkijonoista haluttuja yhdistelmiä, jolla estetään välilyönnit ja

muut erikoismerkit. Tällöin yrittäessä luoda kansiota pitäisi todennäköisesti tulla ikkuna, jossa kerrotaan käyttäjälle, mitä merkkejä kansion nimi ei voi sisältää. Toisaalta jos tiettyjen merkkien rajoittaminen koetaan liian rajoittavaksi, voi myös laittaa koodiin ehtoja siitä, minkä merkkien pitäisi muuttua polussa tietyissä tapauksissa. JQueryssä tämä voisi tapahtua replace-käskyllä, johon annetaan ensimmäiseksi parametriksi mitä halutaan muuttaa ja toiseksi parametriksi millaiseksi halutaan muuttaa. Myös sivutuksen ja haun yhteispelissä löydettiin korjattavaa. Haettaessa tyhjää, jolloin kaikki kuvat sivulla tulevat näkyviin (tämä oli toiminnallisesti tarkoitettukin näin), sivutustoiminnallisuutta ei huomioida ja kaikki kuvat tulevat näkyviin kerralla.

Tyylikorjauksiakin tuli toiminnallisuuskorjauksien ohella. Esimerkiksi kansioiden kuvakkeet menivät hakusuodatusikkunan päälle, mikä on korjattavissa lisäämällä hakusuodatusikkunan tyylissä z-indeksiä. Kansiot myös menivät toistensa päälle, kun niitä oli kahdella rivillä, mikä johtui siitä, että niiden tyyleissä on position ”static”. Alkuperäinen idea tässä oli, että kansiot seuraavat käyttäjää, jotta kuvien sijainnin muuttaminen raahaamalla olisi helpompaa, mutta tähänkin pitää katsoa parempi ratkaisu. Kuvien nimissä liian pitkä nimi meni kuvasta yli, ja tämä on korjattavissa tyyleissä truncate-komennolla. Taustahaalistus jäi myös päälle, kun valittu kuva poistettiin.

Kehitysideoita tuli myös testauksessa useita: muun muassa useamman kuvan lataaminen samaan aikaan, jottei yksi kerrallaan tarvitsisi ladata, kuvan tiputtaminen vedä ja pudota -tyyppisesti suoraan oman tietokoneen tiedostosta kuvakirjastonäkymään, kuvien ja kansioiden näkymän erottaminen omiksi erillisiksi näkymikseen, kuvan muuttaminen jpg:ksi automaattisesti lisäyksen yhteydessä nyt tällä hetkellä olevan napin painalluksen sijaan, hakukenttään x-kuvake, joka pyyhkisi haun pois, kuva ladattaessa näkymä siirtyisi kuvakirjastoon; eikä elementtiin, johon valittu kuva on asettunut taustakuvaksi, ja juuressa tapahtuvan haun ulottuminen myös alikansioissa sijaitseviin kuviin. Kehitysideat luonnollisesti toteutetaan vasta virheiden korjauksen jälkeen.

5 Yhteenveto

Insinöörityönä valmistui parannettu versio Builderin kuvakirjastosta, johon lisättiin toivottuja ominaisuuksia: hakutoiminnallisuus kuvien nimillä ja tageilla, mahdollisuus lisätä kuviin tageja, kuvien sivutus, kansioiden luominen ja niiden välillä kuvien liikuttaminen vedä ja pudota -tyyppisesti, mahdollisuus lisätä kuvia alikansioihin, kuvien haun rajaaminen pystysuoriin ja vaakasuoriin kuviin, kuvien järjestäminen uusimmasta vanhimpaan ja päinvastoin ja muuntaa png-kuvia jpg-kuviksi. Myös kuvakirjaston yksittäisiä käyttömukavuuteen liittyviä asioita parannettiin: kuvan valinnan poistaminen taustaa klikatessa, valitsemattomien kuvien läpinäkyvyyden laskeminen ja varmistaminen ennen kuvan poistamista.

Insinöörityö sisälsi frontendia, backendia, tietokantaa ja palvelinpuolta. Olin aiemmin kehittänyt Builderin frontend-puolta, joten oli mielenkiintoista ja hyödyllistä opetella myös sen backend-puolta ja huomata, miten se toimii Builderissa. Myös tiedostojen liikuttaminen palvelimella PHP:n avulla opittiin insinöörityötä tehdessä. Näistä opeista on varmasti hyötyä tulevaisuudessa omalla osalla Builderin kehityksessä.

Monen henkilön testatessa tuli hyviä huomioita ja paljon sellaista, mitä ei enää itse näe, kun on tuotosta jo katsonut melko paljon. Toki testatessa ihmiset kiinnittävät huomiota eri asioihin: koodareiden huomio kiinnittyy enemmän teknisiin seikkoihin, sisällöntuottajien ja graafikon huomio ulkomuotoon ja käyttömukavuusseikkoihin. Testauksessa löydettyjen virheiden korjaamisen jälkeen ensimmäinen versio kuvakirjaston parannuksesta voidaan uuden testauksen kautta julkaista asiakkaiden käytettäväksi. Kehitysideoiden työstäminen alkaa tämän jälkeen.

Internetissä on laaja valikoima tekijänoikeusvapaita kuvia sisältäviä kuvakirjastoja ihmisten käytettäväksi. Tekijänoikeudet syntyvät luojille automaattisesti, ja niitä tulee lain mukaan kunnioittaa. Projekteille voi valita valmiita tyylikehyksiä, jotta voi keskittyä enemmän tekniseen puoleen. Lokaalit ympäristöt helpottavat kehittämistä ja tekevät siitä turvallisempaa kehittämisen

pysyessä erillään virallisesta ympäristöstä. Insinöörityön tilaajayritys testasi parannettua kuvakirjastoa, jonka päämäärä on mennä korjausten myötä asiakkaiden käyttöön ja kehittyä eteenpäin tulevaisuudessa.

Lähteet

1 Valokuvaajan oikeudet ja velvollisuudet. Verkkoaineisto. Raahen kameraseura. <http://www.raahenkameraseura.fi/wp-content/uploads/2013/03/Kuvaajan-oikeudet-ja-velvollisuudet.pdf>. Luettu 19.4.2021.

2 Suomen perustuslaki. 11.6.1999/731.

3 Unsplash Developers. Verkkoaineisto. Unsplash. <https://unsplash.com/developers>. Luettu 11.5.2021.

4 Kuvaoikeuksien ABC. Verkkoaineisto. Kuvasto Ry. <https://kuvasto.fi/kuvaoikeuksien-abc/>. Luettu 5.2.2021.

5 Tekijänoikeuden syntyminen. Verkkoaineisto. Tekijanoikeus.fi. <https://tekijanoikeus.fi/tekijanoikeus/syntyminen/>. Luettu 5.2.2021.

6 Varjus, Seppo. 2019. 11 ikimuistoista kuvaa: Suomen presidenttien vierailuilta Yhdysvaltoihin on jäänyt unohtumattomia muistoja. Verkkoaineisto. Iltasanomat. <https://www.is.fi/kotimaa/art-2000006256209.html>. Julkaistu 2.10.2019. Luettu 11.5.2021.

7 Hasu, Roosa. 2021. Lakimies, Tekijänoikeuden tiedotus- ja valvontakeskus ry, Espoo. Sähköpostikeskustelu 29.4.2021.

8 Dahlbom, Taika. 2018. EU-tuomioistuin linjasi tekijänoikeuksista: internetissä julkaistuja kuvia ei saa julkaista uudelleen – edes koululaisten esitelmissä. Verkkoaineisto. Helsingin Sanomat. <https://www.hs.fi/kulttuuri/art-2000005784145.html>. Julkaistu 8.8.2018. Luettu 5.4.2021.

9 Direktiivi (EU) 2019/790.

10 Tekijänoikeuslaki. 1995. 446/24.3.1995.

11 Tekijänoikeuden loukkaukset. 2006. Verkkoaineisto. Tekijänoikeuden tiedotus- ja valvontakeskus ry. <https://tekijanoikeus.fi/wp-content/uploads/2015/03/tekijanoikeuden_loukkaukset.pdf>. Luettu 27.4.2021.

12 FlexImages. Verkkoaineisto. Pixabay.com. <https://goodies.pixabay.com/jquery/flex-images/demo.html>. Luettu 29.3.2021.

13 Build fast, responsive sites with Bootstrap. Verkkoaineisto. Bootstrap team. <https://getbootstrap.com/>. Luettu 4.4.2021.

14 XAMPP Tutorial. Verkkoaineisto. JavaTpoint. <https://www.javatpoint.com/xampp>. Luettu 16.4.2021.

15 Docker overview. Verkkoaineisto. Docker Inc. <https://docs.docker.com/get-started/overview/>. Luettu 17.4.2021.

16 Zend_Controller Basics. Verkkoaineisto. Zend. <https://framework.zend.com/manual/1.12/en/zend.controller.basics.html>. Luettu 25.4.2021.

17 Dialog Widget. Verkkoaineisto. OpenJS Foundation. <https://api.jqueryui.com/dialog/>. Luettu 19.4.2021

18 Window confirm() Method. Verkkoaineisto. W3schools. <https://www.w3schools.com/jsref/met_win_confirm.asp>. Luettu 24.4.2021.

19 Rename. Verkkoaineisto. PHP. <https://www.php.net/manual/en/function.rename.php>. Luettu 28.4.2021.