OULUN AMMATTIKORKEAKOULU

Teemu Anttila

Web-sovellus Oulun Rauta-Pojat Oy:lle

Web-sovellus Oulun Rauta-Pojat Oy:lle

Teemu Anttila
Opinnäytetyö
Kevät 2017
Tietojenkäsittelyn tutkinto-ohjelma
Oulun ammattikorkeakoulu

TIIVISTELMÄ

Oulun ammattikorkeakoulu
Tietojenkäsittelyn tutkinto-ohjelma, Web-sovelluskehitys

Tekijä(t): Teemu Anttila
Opinnäytetyön nimi: Web-sovellus Oulun Rauta-Pojat Oy:lle
Työn ohjaaja: Eero Leskinen
Työn valmistumislukukausi ja -vuosi: Kevät 2017 Sivumäärä: 47

Tämän opinnäytetyön tarkoituksena oli toteuttaa uusi verkkosivusto toimeksiantajana toimineelle Oulun Rauta-Pojat Oy:lle vanhanaikaiseksi käyneen sivuston tilalle. Uuden sivuston avulla haluttiin parantaa yrityksen näkyvyyttä ja tarjota asiakkaille uusia yhteydenottomahdollisuuksia. Oulun Rauta-Pojat Oy on Oulun Ruskossa toimiva tilauskonepaja, joka valmistaa asiakkailleen erilaisia metallirakenteita. Lisäksi yritys tekee asiakkailleen muun muassa korjausmaalauksia, hitsausta ja hiekkapuhallusta.

Yrityksen näkyvyyttä lähdettiin parantamaan sivulle sijoitetun kuvagallerian avulla, johon tuotteet saadaan paremmin esille. Koska toimeksiantaja halusi itse hallinnoida kuvagallerian sisältöä, toteutettiin sivuston rinnalle verkkosovellus sen mahdollistamiseksi. Sivulle tarvittiin myös lomake, jonka avulla olisi mahdollista lähettää tarjouspyyntöjä ja työhakemuksia. Sivuston ulkoasusta haluttiin yksisivuinen, ja pitää sen sekä sovelluksen käyttöliittymä mahdollisimman yksinkertaisena.

Raportissa käydään läpi työhön käytettyjä tekniikoita ja niiden soveltamista. Teoriaosuuden jälkeen käydään läpi työn suunnittelu- ja toteutusvaiheet. Lähteinä työssä käytettiin alan tietokirjallisuutta ja sen tukena uudempia verkkolähteitä.

Opinnäytetyön tuloksena syntyi toimeksiantajalle uusi sivusto ja sen sisällönhallintaan tarkoitettu sovellus. Jatkotoimenpiteinä ovat sivuston käyttöönotto ja sen kehitys näkyvyyden ja ulkoasun osalta. Ulkoasun muotoilua ei saatu täysin valmiiksi, sillä sinne sijoitettavaa materiaalia ei ollut vielä käytettävissä työtä tehdessä. Mahdollisia kehitysideoita ovat sisällönhallinnan laajentaminen ja sivuston versiointi eri kielille.

Asiasanat: Verkko-sovellus, sovelluskehykset, käyttöliittymät, graafinen suunnittelu

ABSTRACT

Oulu University of Applied Sciences
Degree Program in Business Information Systems, Web Application Development

---

Author(s): Teemu Anttila
Title of thesis: Web application for Oulun Rauta-Pojat Oy
Supervisor(s): Eero Leskinen
Term and year when the thesis was submitted: Spring 2017 Number of pages: 47

The subject of this thesis was to develop a new website for Oulun Rauta-Pojat Oy as a replacement for the old site. The reason for the site development was to improve the company's visibility and provide customers new contact opportunities. Oulun Rauta-Pojat Oy is an Oulu based company which manufactures metal structures for its customers. In addition, the company is involved in metal refinishing, welding and sandblasting.

An image gallery was needed to improve the visibility of the company's products. Since the company wanted to be able to manage the image content by them self, a web application was created for this purpose. A feedback form was also required to enable sending tender requests and job applications. The site's layout was chosen to be as simple as possible as well as for the interface of the application.

The techniques used for the thesis are reviewed in the theoretical part of the report. After that, the planning and implementation phase of the work is examined. The sources used for the thesis were non-fiction literature with support of online sources.

As a result, a new site was created for the commissioner company with the application for content management. The final layout could not be fully finished during the thesis process, because the needed contents were not yet made available by the company. Further steps in the future will be the deployment of the site and further development of visibility and layout. Future development ideas include expanding content management and localization of the site.

Keywords: web application, framework, user interface, graphic design

SISÄLLYS

1 JOHDANTO 6
2 TEKNIIKAT 8
2.1 HTML 8
2.2 PHP 9
2.3 CSS 11
2.4 JavaScript 13
2.5 Tietokannat ja MySQL 15
2.6 Hakukoneoptimointi 17
3 WEB-SOVELLUSTEN ARKKITEHTUURI 19
3.1 Olio-ohjelmointi 19
3.2 Web-arkkitehtuurit 20
3.3 Sovelluskehykset 23
4 KÄYTTÖLIITTYMÄSUUNNITTELU 26
4.1 Responsiivisuus 27
4.2 Graafinen suunnittelu 29
5 TOTEUTUS 31
5.1 Lähtökohdat ja tavoitteet 31
5.2 Käytetyt työkalut ja ohjelmistot 31
5.3 Hallintapaneeli 32
5.4 Käyttöliittymä ja näkymät 37
6 JOHTOPÄÄTÖKSET JA POHDINTA 43
LÄHTEET 45

1 JOHDANTO

Tämän opinnäytetyön aiheena on uuden web-sivuston suunnittelu ja toteutus toimeksiantajana toimivalle yritykselle. Oulun Rauta-Pojat Oy on Ruskossa toimiva tilauskonepaja, joka valmistaa erilaisia metallirakenteita asiakkaiden tarpeiden mukaisesti. Asiakkaina on sekä toisia yrityksiä, että yksityisasiakkaita. Metallirakenteiden tuottamisen lisäksi yritys tekee myös korjausmaalauksia, hitsausta ja hiekkapuhallusta.

Aihe oli helppo ottaa vastaan, koska olen tehnyt yrityksen kanssa aiemmin yhteistyötä. Jatkettuani opintojani vuoden 2017 alusta, päätin keskittyä web-ohjelmointiosaamiseni kehittämiseen. Työ sopi siis kiinnostukseni kohteisiin erinomaisesti. Pääsin toteuttamaan itsenäisesti kokonaisen projektin, johon minulle annettiin toimeksiantajalta suhteellisen vapaat kädet. Vanha sivusto on käyttökelpoinen, mutta toiminnallisuudeltaan vajaa ja ulkoasultaan sekä käytettävyydeltään vanhanaikainen. Staattiselta HTML-sivulta puuttuvat yhteydenottomahdollisuudet ja tuote-esittelymahdollisuudet.

Toimeksiantajan vaatimuksia olivat kuvagallerian toteuttaminen niin, että kuvien päivittäminen onnistuisi ilman ulkopuolista apua. Myös hakukoneoptimointi oli yritykselle tärkeää näkyvyyden parantamiseksi. Tämän lisäksi haluttiin palautelomake asiakkaita ja työnhakijoita varten. Sivuston ulkoasu ja käyttöliittymä jätettiin minun suunniteltavaksi ja ulkoasuksi valikoitui yksisivuinen etusivu. Työn pääasiallinen tarkoitus yrityksen näkökulmasta on parantaa näkyvyyttä uudella ulkoasulla ja uusilla toiminnoilla. Yrityksen tuotteet saadaan paremmin näkyville ja yhteydenottomahdollisuudet lisääntyvät.

Työ toteutettiin MVC (Model-View-Controller) –arkkitehtuurilla CodeIgniter–sovelluskehystä käyttäen. CodeIgniterilla toteutettu palvelinpuolen sovellus ohjelmoitiin käytettäen PHP-ohjelmointikieltä. Lisäksi sivuston toimintaan tarvittiin MySQL-tietokanta, jolla mahdollistetaan kuvagallerian toimivuus ja käyttäjätilin luominen gallerian hallintaan. Käyttäjätilille tarvittiin kirjautumisominaisuus tietoturvasyistä.

Käyttöliittymään käytettiin HTML-kuvauskielen lisäksi apuna JavaScriptiä ja PHP:tä dynaamisten sisältöjen luomiseen. Käyttöliittymästä tehtiin Twitter Bootstrapin avulla responsiivinen eli eri laittelle muokkautuva. Itse sisällön tuottaminen ei kuulunut tähän opinnäytetyöhön, vaan osaa jo

olemassa olevasta tekstistä tullaan käyttämään sellaisenaan tai muokattuna uudelle sivustolle sopivammaksi. Uuden kuva- ja tekstisisällön tuottaminen jää toimeksiantajan vastuulle tai mahdollisesti myöhempään ajankohtaan tekemällä uusi sopimus sen osalta.

2 TEKNIIKAT

Tässä osiossa käydään läpi opinnäytetyössä käytettyjä tekniikoita, sekä merkkaus- ja ohjelmointikieliä. Myös tavoitteena oleva yrityksen näkyvyyden parantaminen käsitellään hakukoneoptimoinnin osalta.

2.1 HTML

HTML:n, eli Hypertext Markup Languagen, kehitys alkoi vuonna 1989 Euroopan hiukkasfysiikan tutkimuslaitoksessa (CERN). World Wide Webin kehittäjänä tunnettu Tim Berners-Lee alkoi kehittää tapaa, jolla fyysisesti eri paikoissa sijaitsevia dokumentteja voitaisiin ladata näkyviin tekstimuotoisena yhdelle keskusyksikölle verkon kautta ilman, että itse dokumenttia tarvitsisi siirtää kokonaisuudessaan. Tämä linkittämisen idea kehittyi myöhemmin protokollaksi, jota kutsutaan hypertekstiksi. Berners-Leen kehitystyö perustui jo olemassa olevaan SGML-kieleen ja HTML:stä saatiin selainriippumaton. Erityisesti hypertekstilinkit tekivät keksinnöstä käytännöllisen. (Addison Wesley Longman 1998, hakupäivä 7.5.2017.)

HTML on merkintäkieli, jolla kuvataan www-sivujen loogista rakennetta. HTML-dokumentti sisältää yleensä vain tekstiä, mutta sillä voidaan viitata muunlaisiin dokumentteihin niin, että selain osaa esittää esimerkiksi kuvia, ääntä tai videoita. HTML-dokumentti voidaan jakaa kahteen osaan: loogisen rakenteen kertovaan merkkaukseen ja selaimessa näkyvään tekstisisältöön. (Korpela & Linjama 2005, 70.)

HTML koostuu elementeistä, jotka sisältävät alkumerkinnän, sisällön ja loppumerkinnän (Kuvio 1). Alku- ja loppumerkinnät ilmoittavat, että selaimessa näkyvä sisältö määritellään niiden välissä. Eräät elementit voivat olla myös niin sanotusti tyhjiä, jolloin ne sisältävät ainoastaan alkumerkinnän. Tässä tapauksessa merkinnällä on ainoastaan muotoilua koskeva määrite. (Korpela & Linjama 2005, 72.)

KUVIO 1. Esimerkki yksinkertaisesta HTML-sivun koodista ja sen tulostuksesta selaimessa.

Alun perin web-dokumentit olivat staattisia eli muuttumattomia. Sivun sisältöä voitiin muuttaa vain tekemällä muutoksia sivun merkintään. Ainoastaan hyperlinkkien avulla käyttäjä kykeni vaikuttamaan sisältöön selaimen kautta. Staattisten sivujen muuttaminen voi olla työlästä, mutta niillä on edelleen käyttötarkoituksensa. Jos sivuja ei tarvitse päivittää usein, yleensä yksinkertainen staattinen sivu on edelleen toimiva ratkaisu. (Rantala 2005, 3.)

Web-dokumentit voivat sisältää myös muuttuvaa eli dynaamista sisältöä. Esimerkiksi haettaessa tietokannasta haluttua sisältöä, vaikkapa verkkokaupan tuotesaldoa, on sivuston dynaamisuus ehdotonta. Dynaamisuuden luomiseen on olemassa erilaisia ratkaisuja, jotka vaativat lähes poikkeuksetta ohjelmointia. Kun sivuston rakenne koostuu pääosin dynaamisesta elementeistä, kyseessä on sivuston sijaan sovellus. Dynaamisuus voidaan luoda esimerkiksi PHP-skripteillä. (Rantala 2005, 4.)

2.2 PHP

PHP eli rekursiivinen akronyymi lauseesta: ”PHP: Hypertext Preprocessor” on Rasmus Lerdorfin kehittämä ohjelmointikieli, joka oli alun perin kokoelma www-pohjaisten sovellusten tekemistä helpottavia C-kielisiä rutiineja. Lukuisten lähdekoodin uudelleenkirjoituskertojen ja versiointien jälkeen vuonna 2000 PHP 4.0 oli jo ilmestyessään täysiverinen ohjelmointikieli. (The PHP Group 2017, hakupäivä 7.5.2017.)

PHP on tulkattava ohjelmointikieli, joka tarkoittaa sitä, että PHP-koodi ajetaan joka kerta kun palvelin lähettää sivun pyynnön tehneelle selaimelle. Koodin ajo tapahtuu aina palvelimella ennen sivun lähettämistä. Itse koodi ei näy selaimessa ollenkaan HTML-koodin seassa, vaikka se olisikin upotettu sen sisään. Selaimessa näkyy sen sijaan koodin suorittama tulostus osana sivua. (Heinisuo 2003, 16–17.) Vaikka PHP on kehitetty nimenomaan dynaamisten sovellusten

kehittämiseen, ei selaimesta nähtävän tiedon perustella voida päätellä onko sivu toteutettu staattisesti vai dynaamisesti (Rantala 2005, 17).

PHP-kielen avulla voidaan käyttää useimpia olemassa olevia tietokantoja. PHP:n ollessa avoin ohjelmisto, on lähdekoodi saavavilla vapaasti kaikille. (Rantala 2005, 4.) PHP sisältää tyypillisimmät ohjelmointikielien ominaisuudet: ehto- ja toistolauseet, muuttujat ja funktiot. PHP on oliopohjainen eli sovellukset on mahdollista toteuttaa luokkina perinteisemmän funktioiden käyttämisen sijaan. (Heinisuo 2003, 17.) HTML-koodin sisään upotettu PHP-koodi erotetaan merkinnöillä ”<?php” ja ”?>”. Komentorivin lopetuskohta määritellään puolipisteellä. (Rantala 2005, 17.)

Muistiin tallennettaviin tietoihin viitataan muuttujilla, joiden arvo voi vaihtua ohjelmaa suoritettaessa rajattoman monta kertaa. PHP-kielessä muuttujia ei tarvitse määritellä etukäteen, vaan niiden tyyppi ja tarvitsema tila määräytyy siihen sijoitettavan tiedon mukaan. Tietotyyppi määrittää muuttujan sisältämää tietoa, tilanvarausta ja miten sitä voidaan käyttää. Tietotyypit voidaan jakaa kahteen eri kategoriaan: yksinkertaisiin, jolloin muuttujalla on kerrallaan täsmälleen yksi ainoa arvo, ja rakenteisiin, jolloin muuttujaan voi yhtäaikaisesti liittyä useampi arvo. PHP-kielen yksinkertaisia tietotyyppejä ovat totuusarvo (boolean), kokonaisluku (integer), liukuluku (float) ja merkkijono (string) (Kuvio 2). Rakenteisia tietotyyppejä ovat taulukko (array) ja olio (object). (Rantala 2005, 25–27.)

```
$tosi = TRUE;
$kokonaisluku = 3;
$desimaali = 3.4;
$merkkijono = 'Tämä on merkkijono';
```

KUVIO 2. Esimerkki yksinkertaisista tietotyypeistä PHP-kielellä.

PHP:n avulla taulukkotyyppiin voidaan rakentaa erilaisia tietorakenteita. Yksinkertaisten tietotyyppien tapaan taulukkomuuttujan kokoa ei tarvitse määritellä etukäteen. Taulukko voidaan luoda kuvion 3 mukaan array-käskyllä tai sijoittamalla arvoja muuttujaan (Rantala 2005, 33.)

```
public function lisaa() {
        $data = array(
                'id'                    =>      '',
                'sukunimi'              =>      '',
                'etunimi'               =>      '',
                'lahiosoite'            =>      '',
                'postitoimipaikka'      =>      '',
                'postinumero'           =>      ''
        );
```

KUVIO 3. Esimerkki taulukon luomisesta tyhjillä arvoilla array-käskyllä.

Muuttujien ja funktioiden palauttamien arvojen vertailu voidaan tehdä if-ehtolauseen avulla (Heinisuo 2003, 54). Kuvio 4:n ylimmällä rivillä oleva muuttujien $a ja $b vertailu määrittelee mitä ohjelma seuraavaksi tekee. Jos $a:n arvo on suurempi kuin $b:n, tulostetaan ruudulle ylempien aaltosulkeiden välissä oleva print-komento. Mikäli a$ ei ole suurempi kuin $b, siirrytään alempaan lohkoon.

```
if ($a > $b)
{
    print "a on suurempi kuin b";
}
else
{
    print "a ei ole suurempi kuin b";
}
```

KUVIO 4. Muuttujien vertailu if-ehtolauseella.

PHP:n valinta tämän opinnäytetyön tekemiseen oli varsin helppo. Vaikka vaihtoehtoja onkin olemassa, on PHP vaihtoehdoista tutuin. PHP on myös erittäin suosittu: yli 80 prosenttia palvelinpuolen sivustoista käyttää sitä (Web Technology Surveys 2017, hakupäivä 24.5.2017). Useimmat web-palvelimet tukevat PHP-kieltä suoraan ja sitä voidaan käyttää myös tietokantojen manipuloimiseen. Muuttujien tyyppiä ei tarvitse määritellä, mikä helpottaa huomattavasti työn tekemistä.

2.3 CSS

CSS (Cascading Style Sheets) tarkoittaa yleisesti tyyliohjetta, jonka avulla web-dokumentin esitysasua muotoillaan. Eri selaimet voitat esittää saman web-sivun eri tavoilla eikä tyyliohje kumoa tätä. Tyyliohje esittää ulkoasua koskevia ehdotuksia, jotka selain ominaisuuksiensa mukaan

toteuttaa. Käytännössä tyylisäännöt muokkaavat siis sivun graafisia ominaisuuksia. (Korpela & Linjama 2005, 300.)

CSS:n kehitys alkoi vuonna 1994, kun Håkon Wium Lie näki tarpeelliseksi mahdollisuuden muotoilla web-sivuja. HTML-sivun rakenteen ja sivun ulkoasun erottaminen oli ollut tavoitteena jo HTML:n synnystä asti, mutta yleistä tapaa määritellä sivun tyyliä ei vielä ollut olemassa. Ratifioidessaan tulevaisuuden määritelmiä HTML:lle vuonna 1995, W3C (World Wide Web Consortium) otti CSS:n mukaan tavoitteenaan tehdä siitä suositellun tavan ulkoasun määrittelyyn. (Bos 2016, hakupäivä 11.5.2017.)

Yleisesti sivun asettelu tarkoittaa sitä, että HTML-dokumentissa määritellään sisältö div-merkintöjen sisään ja sen jälkeen div-elementit asetellaan oikeille paikoilleen CSS:n avulla. Div-elementtien sisältämien luokka– (class) ja tunnistemääritteiden (id) avulla voidaan muotoilu kohdistaa vielä tarkemmin. CSS mahdollistaa mediakyselyiden avulla myös responsiivisen suunnittelun. Esimerkiksi rinnakkain asetetut elementit näytetään kapealla näytöllä allekkain. Osa CSS:n määrittelyistä tulee merkitä erikseen eri selaimille, jotta niiden toiminta voidaan varmistaa. Esimerkiksi siirtymä-ominaisuus (transition), jolla voidaan muuttaa elementin ominaisuus toiseksi määritellyssä ajassa, täytyy merkitä selainkohtaisilla etuliitteillä (Kuvio 5) (Lehdonvirta & Korpela 2013, 88–89.)

```
 body {
     position: relative;
}

 nav.navbar {
  -webkit-transition: all 0.4s ease; /* Safari */
  transition: all 0.4s ease;
}

.navbar-inverse {
   background-color: #222;
   border-color: transparent;

}
.text {
   color: #DDD;
}
```

KUVIO 5. Esimerkki CCS-tyylitiedoston määrittelyistä sisältäen selainkohtaiset määritteet.

Samalle sivulle voidaan kohdistaa useita tyyliohjeita ja ne kaikki voivat vaikuttaa sivun ulkonäköön. Tästä johtuen voi eri ohjeiden välille syntyä ristiriitoja. Nämä päällekkäisyydet kuitenkin ratkeavat

loogisesti, sillä tarkemmin määritelty sääntö kumoaa yleisemmän ja myöhempi sääntö aiemman. (Korpela & Linjama 2005, 301.) Päällekkäisyyksistä on usein myös hyötyä eikä niiden olemassaoloa voida aina edes välttää. Käytettäessä valmiita tyylioppaita, joilla määritellään sivun ulkoasu yhtenäiseksi, tarvitaan usein yksittäisen elementtien kohdalle tarkennuksia tai alkuperäisten sääntöjen kumoamista.

Tyylisäännöstö voidaan liittää HTML-sivuun usealla eri tavalla. Suositeltavin tapa on viitata erilliseen .css –päätteiseen tiedostoon käyttämällä HTML:n link-merkintää. Viittaus voidaan tehdä myös HTML-dokumentin style-elementissä, joka kirjoitetaan dokumentin head-osaan. Yksittäiseen elementtiin viittaus voidaan tehdä viittaamalla siihen itse elementin merkinnässä. Yleismäärittelyt on hyvä tehdä erilliseen viitattavaan tiedostoon joko paikallisesti tai sitten viittaamalla valmiiseen tyylitiedostoon verkossa. (Korpela & Linjama 2005, 305.)

Twitter Bootstrap on CSS-tyyliopas, joka sisältää valikoiman käyttöliittymän rakennuspalikoita lähes kaikenlaiseen mahdolliseen sivun muotoiluun. Mukana on myös responsiivisen käyttöliittymän toteuttamiseen tarvittavia määritteitä. Nimensä mukaisesti sosiaalisen median palvelu Twitter käyttää Bootstrapia itsekin, mutta se on käyttökelpoinen lähes minkä tahansa sivun muotoiluun. Bootstrap auttaa yhtenäistämään selainten käyttämiä oletuksia sivujen muotoilun osalta ja sen tuottamia ulkoasuja voidaan vapaasti muokata tarpeiden mukaan. (Lehdonvirta & Korpela 2013, 93–95.)

2.4 JavaScript

JavaScript on useimpien selainten tukema oliopohjainen ohjelmointikieli. Nimensä mukaisesti kyseessä on skriptikieli, mikä tarkoittaa sitä, että koodi tulkitaan eikä sitä käännetä erikseen tulkin avulla. JavaScriptin vahvuuksia ovat käyttöjärjestelmäriippumattomuus, nopeus ja vapaa syntaksi. (Saarikumpu 2016, hakupäivä 17.5.2017.) JavaScriptin haittana ovat sen mukana tulevat ongelmat tietoturvan kanssa. Vaikka JavaScript itsessään ei sisällä tietoturva-aukkoja, eri selainten toteuttama kielen käsittely on ollut ongelmallinen (2Kmediat 2017, hakupäivä 17.5.2017.)

JavaScriptin kehitys alkoi Netscapen toimesta vuonna 1995, kun Brendan Eich kehitti HTML:n sekaan upotettavan ja suoraan tulkattavan skriptikielen. JavaScriptin piti alun perin olla vain

väliaikainen ratkaisu, mutta se on vakiinnuttanut paikkansa yleisesti toimivimpana ratkaisuna: JavaScript mahdollistaa muun muassa web-sivun sisällön päivittämisen ilman, että koko sivua tarvitsee ladata uudelleen. (Försström 2015, hakupäivä 15.5.2017.)

Yleisimmin JavaScriptiä käytetään niin sanotun asiakaspuolen ohjelmoinnissa. Skriptit eli komentojonot suoritetaan ja tulkitaan asiakaskoneelle. Tulkitseminen tarkoittaa sitä, että selain sekä lukee, että suorittaa koodia samanaikaisesti. Tästä syystä mahdolliset ohjelmointivirheet tulevat esiin käyttäjän selaimessa eikä palvelimella. (2Kmediat 2017, hakupäivä 17.5.2017.)

JavaScript on mahdollista kytkeä kokonaan pois päältä tai estää sen ajaminen halutuista osoitteista. Tästä syystä on tärkeää suunnitella JavaScriptiä käyttävät sovellukset niin, ettei toiminta riipu siitä onko käyttäjällä JavaScript päällä. (2Kmediat 2017, hakupäivä 17.5.2017.) Tämän opinnäytetyön JavaScript-ominaisuudet liittyvät lähinnä lisätoimintoihin, jotka lisäävät käyttömukavuutta ja parantavat visuaalista ilmettä. Sovellus toimii kuitenkin myös ilman JavaScriptiä, eikä käyttäjältä jää mitään sisältöä piiloon.

JavaScript upotetaan osaksi HTML-dokumenttia script-elementin avulla (Kuvio 6). Suositeltava tapa on sijoittaa JavaScript-koodi omaksi tiedostokseen ja sisällyttää sen linkki osaksi HTML-sivua. Myös tyyppi-elementtiä (type) on hyvä käyttää, sillä HTML 5:ttä vanhemmat versiot vaativat sen toimiakseen.

```
<head>
    <script src="skripti.js" type="text/javascript"></script>
</head>
```

KUVIO 6. JavaScript-tiedoston linkittäminen HTML-dokumenttiin.

Script-elementin sijoittamiselle ei ole yhtä oikeaa paikkaa, vaan se riippuu suoritettavan koodin käyttötarkoituksesta. Jos koodi on tarkoitus suorittaa heti sivun latautuessa, tulee se sijoittaa kuvio 6:n tapaan head-elementin sisään. Muutoin se voidaan sijoittaa myös sivun runkoon, eli body-elementin sisään. (2Kmediat 2017, hakupäivä 17.5.2017.)

JavaScript-koodin yksinkertaistamiseksi ja ylläpidon helpottamiseksi selainpuolen ohjelmoinnissa käytetään usein jQuery-kirjastoa. Toinen tärkeä seikka jQueryn käytölle on sen kyky huolehtia eri selainten eroista. JQuery tarjoaa korkean tason toimintoja, joissa selainkohtaiset toteutuserot ovat piilotettuna kirjaston koodin sisään. (Lehdonvirta & Korpela 2013, 59.) JQuery julkaistiin vuonna

2006, ja se on nykyään suosituin JavaScript-kirjasto. Suosio perustuu muun muassa syntaksin helppouteen. JQuery soveltuu toimintojen käsittelyyn ja animaatioiden tekemiseen (Williams 2016, hakupäivä 31.5.2017.) Se voidaan linkittää JavaScript-tiedoston tapaan HTML-sivun head-elementtiin niin, että jQuery on järjestyksessä ennen JavaScriptiä.

2.5 Tietokannat ja MySQL

Tietokannalla tarkoitetaan tietovarastoa, joka koostuu kokoelmasta toisiinsa liittyvistä tiedoista. Tietokanta voi olla myös muussa kuin sähköisessä muodossa. Esimerkiksi paperille kirjoitettua ajopäiväkirjaa voidaan pitää tietokantana. (Webopas, Hakupäivä 15.5.2017.) Yleisesti tietokanta on loogisesti yhteenkuuluvien ja tallennettujen tietojen joukko, jota käsitellään tietokantakielellä. Tietokantaan tallennettuja tietoja hallinnoidaan tietokannan hallintajärjestelmällä (TKHJ). Hallintajärjestelmät ovat monimutkaisia ohjelmistoja, jotka avustavat suuresti ohjelmoijia ja käyttäjiä. Tietokantojen käyttämistä perustellaan tietomuutosten helpottamisella, tietoeheyden varmistamisella ja suorituskyvyn parantamisella. Ilman hallintajärjestelmiä käytettäisiin niiden sijaan erillisiä tiedostoja. Esimerkiksi tietojen mahdollisten ristiriitaisuuksien välttämiseksi tietokannat ovat erittäin hyödyllisiä. Asiakkaan osoite voisi olla eri lailla kirjoitettuna eri tiedostoissa ja olisi vaikea tietää mikä osoite on oikea. Vaikka myös tietokannan sisältämät tiedot voivat olla keskenään ristiriidassa, sen avulla tietojen eheyttä on helpompi hallita. (Hovi, Huotari & Lähdenmäki 2003, 4.)

Nykyiset tietokannan hallintajärjestelmät ovat pääosin SQL-pohjaisia relaatiotietokantoja. Ne ovat helppokäyttöisempiä kuin aiemmin käytetyt hierarkkiset järjestelmät. Relaatiotietokannat perustuvat E. F. Coddin vuonna 1970 julkaisemaan relaatiomalliin, joka määrittelee teoreettisen pohjan relaatiotietokannoille. (Hovi ym. 2003, 5–7.)

Relaatiotietokannan rakenteen perusta on taulu (Kuvio 7). Tauluun kootaan asiakokonaisuus, kuten tässä esimerkissä tallennettuihin kuviin viittaavat tiedot. Taulu sisältää sarakkeita, kuten kuvion 7 ylimmältä riviltä löytyvät perusavain id, tiedostonimi, thumb ja niin edelleen. Eri riveillä on esitetty jokaiseen kuvaan viittaavat tiedot. Sarakkeiden tietojen arvot kuuluvat samaan arvojoukkoon, joka tarkoittaa sitä, että niillä on yhteinen tietotyyppi ja niille varattu pituus on sama.

Kunkin taulun tunnistetta kutsutaan perusavaimeksi. Perusavaimen arvo on yksilöivä eli sen tulee esiintyä taulussa vain kerran. Perusavaimen lisäksi taulussa voi esiintyä myös viiteavain, jolla voidaan viitata toisen taulun perusavaimeen. Viiteavaimia tarvitaan tehtäessä liitoksia eri taulujen välille. (Hovi ym. 2003, 8–9.) Tässä opinnäytetyössä tietokanta on sen verran yksinkertainen, ettei viiteavaimia ole käytetty.

id	tiedostonimi	thumb	tallennettu	kuvaus	nimi
48	20170216_211040.jpg	20170216_211040_thumb.jpg	2017-04-24 17:35:54	ieo nei ljkel elhb jkblek	jottain
49	Steel-Industry-Electric-Motors-Industrial-Fan-Blow...	Steel-Industry-Electric-Motors-Industrial-Fan-Blow...	2017-04-25 12:53:51		kaapelit
50	maxresdefault.jpg	maxresdefault_thumb.jpg	2017-04-25 12:54:03		hitsi
51	bg_texture.jpg	bg_texture_thumb.jpg	2017-04-25 13:14:43		tausta
54	red-1618916_960_720.png	red-1618916_960_720_thumb.png	2017-04-26 19:59:06		logo3
59	Steel-Industry-Electric-Motors-Industrial-Fan-Blow...	Steel-Industry-Electric-Motors-Industrial-Fan-Blow...	2017-04-28 17:02:29	juu	sparks
60	logot.png	logot_thumb.png	2017-04-28 17:37:32	logot	logo
61	xFreehold-Welding-Banner630-x-300.jpg.pagespeed.ic...	xFreehold-Welding-Banner630-x-300.jpg.pagespeed.ic...	2017-05-03 10:44:57	teräsrakenteen hitsaus	hitsaus

KUVIO 7. Esimerkki relaatiotietokannan taulusta.

MySQL on monipuolinen, joustava ja suorituskykyinen relaatiotietokantaohjelmisto, jota käytetään laajasti erilaisten www-palvelujen taustalla. Se oli alun perin tarkoitettu ruotsalaisen konsultointiyrityksen MySQL Ab:n sisäiseen käyttöön, jonka kehittämä kyseinen relaatiotietokanta on. MySQL on nykyään ilmainen ohjelma, mutta se kilpailee silti ominaisuuksiltaan kaupallisten tietokantojen kanssa. Sovellukset eivät ikinä käsittele MySQL-tietokantaa suoraan, vaan se tapahtuu palvelinohjelman kautta noudattaen asiakas-palvelin-arkkitehtuuria. (Heinisuo 2003, 36–37.)

SQL-kieli eli standardoitu relaatiotietokantojen kyselykieli, mahdollistaa tietokantojen käsittelemisen käyttöliittymän kautta. SQL:n avulla voidaan taulujen luonnin ja poiston lisäksi käsitellä tauluihin tallennettua tietoa. SQL-kielen käskyt jaotellaan määrittely- ja käsittelykomentoihin sen perusteella, luodaanko uusia tauluja vai käsitelläänkö jo olemassa olevia tietoja. (Heinisuo 2003, 63.) Create, Retrieve, Update ja Delete (CRUD) eli luo, nouda, päivitä ja poista, ovat tietokantojen neljä pääasiallista toimintoa. Ne toimivat käyttöliittymänä tietokantaan ja mahdollistavat tiedon lisäämisen, näyttämisen, muokkaamisen ja poistamisen. (Technopedia 2017, hakupäivä 26.5.2017.) Myös PHP-kieli tarjoaa rakenteet ja komennot, jotka tekevät mahdolliseksi SQL-komentojen välittämisen tietokannalle. Tietojen tallennus tietokantaan onnistuu esimerkiksi HTML-lomakkeen avulla. (Heinisuo 2003, 63.) MySQL-tietokannan tauluissa voidaan käyttää

erilaisia tietotyyppejä. Tietotyyppi määrää, minkä muotoista tietoa taulun kenttään voidaan tallentaa ja miten tieto esitetään käyttäjälle (Heinisuo 2003, 79–80.)

2.6 Hakukoneoptimointi

Hakukoneoptimoinnin tavoitteena on, että sivun tekstisisältö löytyy hyvin Internetin yleisillä hakukoneilla kuten Google, Bing ja Yahoo. Voidaan puhua myös hakukoneystävällisyydestä. Hakukoneoptimointi voidaan ajatella esimerkiksi seuraavilla tavoilla: kuinka korkealle sivusto sijoittuu hakutulosten listassa tai kuinka hyvin löytyy se sivu, jolla haettu tieto on esitetty parhaiten. Useimmille verkkosivuille on olennaista olla löydettävissä hakukoneiden kautta. (Lehdonvirta & Korpela 2013, 130.)

Hakukoneiden toiminta perustuu niistä löytyvään lomakkeeseen, joka muodostaa käyttöliittymän. Sen avulla voidaan avainsanoja käyttämällä pyytää palvelua hakemaan niitä vastaavia sivuja. Usein käytettävissä on myös monipuolisempi hakulomake, jolla voidaan eri tavoin rajata hakuja esimerkiksi julkaisuajankohdan tai käytetyn kielen mukaan. Varsinainen hakupalvelin käsittelee tehdyn haun ja palauttaa tuloksen käyttäjälle näyttäen sivun osoitteen, otsikot ja mahdollisesti tiivistelmän sivun sisällöstä. Haku tehdään tietokannan avulla, johon kerätään automaattisesti tietoa Internetin sivustoista. Automatiikka toimii indeksointirobottien avulla, jotka vierailevat sivuilla joilla se on aiemmin käynyt tai joista sille erikseen kerrotaan. Sivujen lukemisen sijasta ne käsittelevät sivujen tekstisisältöä painottaen mahdollisesti otsikoita leipätekstin ohitse. Robotit painottavat myös tekstin sijaintia eli alussa oleva teksti on tärkeämpää kuin muu. (Korpela & Linjama 2005, 145–146.)

Hakukoneet painottavat sivustojen sisällöstä eniten sieltä löytyvää tekstiä. Jos sivulla on vain kuvagalleria ilman selityksiä, on sivu hakukoneen kannalta sisällötön. Koska hakukoneet ottavat näytteeksi muutamia sanoja sivun alusta, on tärkeä sijoittaa tärkeät asiat heti tekstin alkuun. Ei haittaa, vaikka sivun otsikon ja leipätekstin sisältö toistaisi samaa asiaa. Hakukone voi ottaa tietoa myös HTML-merkkauksen meta-elementistä, joka sijoitetaan sivun head-elementin sisälle (Kuvio 8). Meta-elementtiä voidaan käyttää sivuston kuvauksen esittämiseen. Tässä tapauksessa käytetään name-määritettä, jonka arvo on ”description”. Tätä määritettä käyttäessä kuvauksen tulee liittyä kyseisen sivun, ei koko sivuston sisältöön. (Korpela & Linjama 2005, 146–148.) Myös

keywords-arvoa eli avainsanoja voidaan käyttää, mutta suurimmat hakukoneet eivät enää huomioi näitä ollenkaan tai niillä on enää hyvin vähän painoarvoa (Edwards 2014, hakupäivä 26.5.2017). Lehdonvirran ym. mukaan meta-määreen käytöllä ei voida enää juuri vaikuttaa hakukoneisiin (Lehdonvirta & Korpela 2013, 130). Parempi tapa vaikuttaa hakukonenäkyvyyteen on sijoittaa avainsanat itse sivun tekstiin (Korpela & Linjama 2005, 149.)

```
<html>
<head>
  <meta charset="UTF-8">
  <meta name="description" content="Web tutorials">
  <meta name="keywords" content="HTML,CSS,PHP,JavaScript">
```

KUVIO 8. Esimerkki meta-elementistä hakukoneita varten.

Hakukoneiden indeksointirobotteja voidaan pyytää olemaan indeksoimatta sivua lainkaan. Tämä voidaan tehdä käyttämällä jo mainittua meta-elementtiä määritteellä ”robots” (Kuvio 9) (Korpela & Linjama 2005, 149). Tästä on hyötyä esimerkiksi sivulla jotka ovat vielä testausvaiheessa, tai sivuilla joiden olemassaolo halutaan pitää jostain syystä salassa.

```
<meta name="robots" content="noindex">
```

KUVIO 9. Hakukoneindeksoinnin estäminen.

Jos sivuston sisältö on HTML-elementeissä, mutta luotu CSS:n tai JavaScriptin avulla, se ei ole löydettävissä hakukoneille parhaalla mahdollisella tavalla. Mikäli sisältö on kokonaan luotu JavaScriptin avulla, se ei näy hakukoneille ollenkaan. JavaScriptiä ei voida olettaa suoritettavaksi selaimessa tai hakukoneen tulkitsevan sen sisältöä tekstiksi. (Lehdonvirta & Korpela 2013, 130.) Tätä työtä toteuttaessa lähdettiin jo alun perin siitä lähtökohdasta, että hakukoneoptimointia tarvitseva julkinen etusivu rakennetaan HTML-pohjaisena.

3 WEB-SOVELLUSTEN ARKKITEHTUURI

3.1 Olio-ohjelmointi

Ohjelmointikielet voidaan jakaa proseduraalisiin- ja olio-ohjelmointikieliin. Ensiksi mainitussa ohjelma rakentuu funktioista, joista yksi suorittaa aina yhden asian. Olio-ohjelmat puolestaan koostuvat nimensä mukaisesti olioista, jotka esittävät ohjelmaan liittyvää kokonaisuutta. Tieto ja niiden käsittelemiseen tarvittavat toiminnot yhdistyvät niihin liittyviin olioihin. Monet ohjelmointikielet sallivat molemmat mainitut lähestymistavat. Olio-ohjelmoinnin etuina ovat ohjelmakoodin parempi järjestäminen, laajennettavuus ja uudelleenkäytettävyys. Mitä laajempi ohjelmisto ja mitä suurempi joukko ihmisiä ohjelmistoa tekee, sitä hyödyllisempää on käyttää olio-ohjelmointia. (Rantala 2005, 64.)

Olio-ohjelmointi syntyi Norjassa 1960-luvun lopulla Norsk Regnesentral –laskentakeskuksessa, jossa ensimmäinen oliokieli Simula kehitettiin Ole-Johan Dahlin ja Kristen Nygaardin toimesta. Vuonna 1961 kehitetyssä Simula I –kielessä käytettiin jo olio-ohjelmoinnin piirteitä, ja sen seuraaja Simula67 oli ensimmäinen varsinainen olio-ohjelmointikieli. 1970-luvulla syntyi useita pohjana toimineita kieliä, ja 1980-luvun puolivälissä olioperusteinen ohjelmointi teki läpimurron useiden tahojen kehittäessä sitä. (Koskimies 2000, 26–27.)

Olio-ohjelmoinnissa ongelma jaetaan pieniin osiin todellisen elämän tapaan. Oliot toimivat rakennuskomponentteina, joita voidaan käyttää uudelleen toisissa projekteissa. Ohjelmistokokonaisuus on helpommin hallittavissa, koska ei ole välttämättä tarpeellista tietää kuinka olio on tehty, vaan ainoastaan kuinka oliota käytetään. (Kareinen 2017, Hakupäivä 16.5.2017.)

Vaikka tämän opinnäytetyön ohjelma ei ole merkittävän laaja, on käytettäväksi valittu oliopohjainen lähestymistapa. Aiempi kokemus proseduraalisesta ohjelmoinnista vaikutti liian tutulta, eikä se sovellu niin hyvin tähän työhön. Mahdollinen jatkokehittely helpottuu oliopohjaisen lähestymistavan avulla. Sovelluskehyksenä käytetty CodeIgniter soveltuu myös paremmin olio-ohjelmointiin ja valmiita toimintoja voidaan käyttää uudelleen.

PHP-kieleen on lisätty olio-ohjelmointiominaisuuksia sen käytön yleistyessä. Aiemmin PHP:n suurimpina pidetyt puutteet liittyivät nimenomaisesti olioihin. PHP:n versiossa 5 olio-ominaisuudet ovat jo muiden modernien oliokielten tasolla. (Rantala 2005, 64.) Vaikka PHP:n normaalit muuttujatyypit riittävät periaatteessa kaikkiin tarkoituksiin, saadaan koodin rakenne selkeämmäksi olio-ohjelmoinnin avulla. Olio-ohjelmoinnin peruskäsitteet ovat luokka ja olio. Ensiksi mainittu määrittelee olion tietosisällön ja käsittelytavan. Luokka voidaan ajatella muuttujan tyyppinä ja olio muuttujan arvona. (Laaksonen 2011, hakupäivä 15.5.2017.)

Uutta oliota luodessa käytetään muodostin -metodia. Periyttäminen mahdollistaa jo tehdyn ohjelmakoodin uudelleenkäytön uusien luokkien pohjana. Perivä luokka eli lapsiluokka sisältää kantaluokkansa toiminnan, mutta siihen voidaan lisätä myös uusia ominaisuuksia. Periyttäminen tapahtuu extends -lauseen avulla. (Hällström LTD Partnership 2017, hakupäivä 16.5.2017.) Esimerkiksi käyttäessä CodeIgniter-ohjelmistokehystä, voidaan sen jo sisältämät toiminnot periyttää kehitettävälle sovellukselle (Kuvio 10).

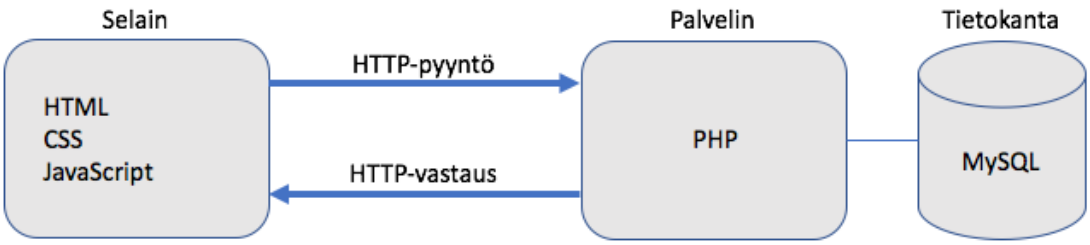
```
class Home extends CI_Controller {
    public function __construct() {
        parent::__construct();

                $this->load->helper(array('form', 'url'));
                $this->load->model('tiedosto_model');
    }
```

KUVIO 10. Luokka Home periytetään CodeIgniterin CI_Controller –luokasta.

3.2 Web-arkkitehtuurit

Web-sovelluksessa käsiteltävä tieto ja niiden käsittely on hyvä erottaa siitä, mitä käyttäjälle näytetään ja mitä toimintoja käyttäjällä käyttöliittymässä on. Näin sovelluskehityksen hallinta helpottuu. Lisäksi erilaisia käyttöliittymiä on helpompi tehdä ja testata, sillä ne ovat riippumattomia sovelluslogiikasta. Saman koskiessa laajennuksia ja muutoksia. (Lehdonvirta & Korpela 2013, 66.)

Web-sovelluksen arkkitehtuuri voidaan yleisellä tasolla jakaa kahteen osaan: asiakaspäähän ja palvelinpäähän (Frontend ja Backend). Asiakaspään koodi suoritetaan selaimessa, ja palvelinpään koodi käsitteen mukaisesti palvelimella (Kuvio 11). Näiden välillä kommunikointi tapahtuu HTTP-protokollan avulla. (Ihantola 2016, hakupäivä 18.5.2017.)



KUVIO 11. Web-sovelluksen yleinen arkkitehtuuri tarkennettuna tämän opinnäytetyön tekniikoilla.

HTTP (HyperText Transfer Protocol) on Internetin keskeinen tiedonsiirtoprotokolla eli yhteyskäytäntö. Se määrittelee miten selain ja palvelin viestivät keskenään. Koko Internetin olemassaolo perustuu vahvasti HTTP-protokollaan (Korpela & Linjama 2005, 415.) Palvelin tarjoaa palveluita, joita asiakkaat pyytävät esimerkiksi selaimen kautta. HTTP on tilaton protokolla, joka tarkoittaa sitä, että palvelin ei muista asiakkaan aiemmin lähettämiä pyyntöjä. Tilattomuus yksinkertaistaa palvelimen toteutusta, mutta usein on myös tarvetta säilöä tilaa, kuten esimerkiksi evästeitä. (Ihantola 2016, hakupäivä 18.5.2017.)

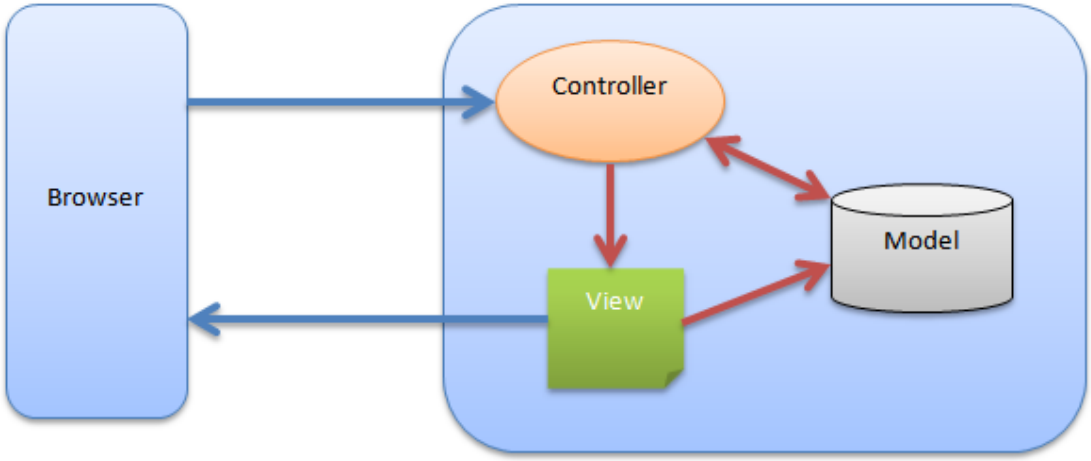
Eväste (cookie) on palvelimen selaimelle lähettämä merkkijono, joka tallennetaan selaimen muistiin tai käytetyn laitteen kiintolevylle. Evästeillä voidaan ratkaista HTTP-protokollan tilattomuuden aiheuttama ongelma. Istunnolla tarkoitetaan kokoelmaa HTTP-pyynnöistä ja –vastauksista, joille on määritelty aloitus- ja lopetustapahtuma. Esimerkkinä evästeiden käytöstä mainittakoon kirjautuminen johonkin palveluun, kuten verkkopankkiin. Käyttäjän kirjautumistieto pysyy voimassa evästeen voimassaoloajan. Istunnon ollessa voimassa käyttäjä voi vierailla myös muilla sivuilla palatakseen takaisin sivustolle, jossa istunto on voimassa. Evästeiden haittapuoli on se, että käyttäjä voi halutessaan estää niiden käytön selaimestaan. Tällöin palvelu tulee käyttökelvottomaksi. Valtaosa verkkopalveluista kuitenkin vaativat evästeiden käyttöä ja jää käyttäjän vastuulle halutaanko niitä hyväksyä. (Rantala 2005, 218–219.)

Autorisointi (valtuuttaminen) tarkoittaa käyttöoikeuksien antamista käyttäjälle. Autentikointi tarkoittaa käyttöoikeuksin varustettujen käyttäjien tunnistamista. HTTP-protokollan tilattomuuden seurauksena selain joutuu lähettämään autentikoinnin jokaisella pyynnöllä ja vastaanottava palvelin tarkastamaan autentikoinnin jokaisella vastauskerralla. (Rantala 2005, 230.) Autentikointi voidaan toteuttaa sekä HTTP:n että PHP:n avulla (Rantala 2005, 230–232). Tässä opinnäytetyössä valittiin kuvagallerian hallintaan tarvittavan kirjautumisen autentikoinnin toteuttamiseksi

CodeIgniter-sovelluskehyksen istuntomekanismit. Niiden toteutus tapahtuu tavallisten PHP:n istuntomekanismien avulla.

MVC-arkkitehtuurissa sovellus jaetaan kolmeen osaan: Malliin (Model), näkymään (View) ja ohjaimeen (Controller). Sen idea on peräisin 1970-luvulta Xeroxin kehittämänä. Ensimmäiset graafiset käyttöliittymät olivat tulleet käyttöön Smalltalk-ohjelmointikielen ohella. Smalltalk oli jo aiemmin mainitun Simulan kanssa ensimmäisiä olio-ohjelmointikieliä. Smalltalkin mukana tuli käyttöön uusi ohjelmistoarkkitehtuuri eli MVC. Vaikka perusta MVC:lle olikin luotu jo 1970-luvulla, se esiteltiin oikeastaan vasta 1988 ilmestyneessä MVC Cookbook –artikkelissa (Vahtolammi 2010, hakupäivä 19.5.2017.)

MVC-mallissa tekninen arkkitehtuuri jaetaan siis kolmeen osaan. Malli huolehtii tietovaraston toiminnasta sen ollessa ainoa sovelluksen osa, joka on yhteydessä tietokantaan. Näkymä ja ohjain ottavat siis yhteyden tietokantaan vain mallin kautta. Näkymä on se osa, joka huolehtii tietojen esittämisestä. Ohjain puolestaan ottaa vastaan sovelluksen pyynnöt. Web-sovelluksessa se lukee tiedot vastaanotetusta HTTP –viestistä (Kuvio 12). MVC-arkkitehtuurin käyttäminen vaatii käytännössä olio-ohjelmointia. (Vahtolammi 2010, hakupäivä 19.7.2017.) Kaikki käyttäjän tekemät toimenpiteet kulkevat aina ohjaimen ja mallin kautta (Lehdonvirta & Korpela 2013, 67).



KUVIO 12. MVC-arkkitehtuuri jaettuna loogisiin osiin.

On tärkeää huomata, että näkymä ja ohjain ovat riippuvaisia mallista. Malli itsessään ei kuitenkaan ole riippuvainen näkymästä tai ohjaimesta. Tämä on yksi merkittävistä sovelluksen jakamisen eduista ja mahdollistaa mallin toteuttamisen ja testaamisen erillään näkymästä. Näkymän ja ohjaimen erottelu on joissakin sovelluksissa toissijaista. Puhuttaessa web-sovelluksista joissa on

olemassa asiakas- ja hallintarajapinta, asia on kuitenkin toisin. Sovelluksen sisältäessä useita eri näkymiä, niiden erottelu on oleellinen asia. (Microsoft 2017, hakupäivä 19.5.2017.) Tähän työhön toteutettiin kaksi erillistä näkymää jotka käyttävät kuvagallerian tietoja ja kolmas kirjatumista varten. Vain hallintapaneelissa tietoa on mahdollista lisätä, muokata ja poistaa. Etusivu on tiedon esittämistä varten.

3.3 Sovelluskehykset

Sovelluskehysten tarkoituksena on helpottaa ja nopeuttaa verkkopalveluiden toteuttamista valmiiden sovelluskomponenttien ja rutiinien avulla. Sovelluskehys ei kuitenkaan ole yleensä valmis ratkaisu, vaan yleensä sovellus rakennetaan sen päälle. Mikäli olemassa olevia valmiita alustoja halutaan käyttää, tulee niiden vastata sovellukseen kohdistuviin tarpeisiin hyvin, koska niiden muokkausmahdollisuudet voivat olla melko rajattuja. Esimerkiksi WordPress on yksi tällainen valmis alusta. Valmiissa alustoissa voi olla mukana ominaisuuksia ja riippuvuuksia, jotka ovat toteutettavan sovelluksen kannalta tarpeettomia. Ilman sovelluskehystä rakennetut palvelut voivat olla toimiviakin, mutta toisaalta niistä voi syntyä vaikeasti ymmärrettäviä ja hankalasti ylläpidettäviä. (Niemi 2015, hakupäivä 19.5.2017.)

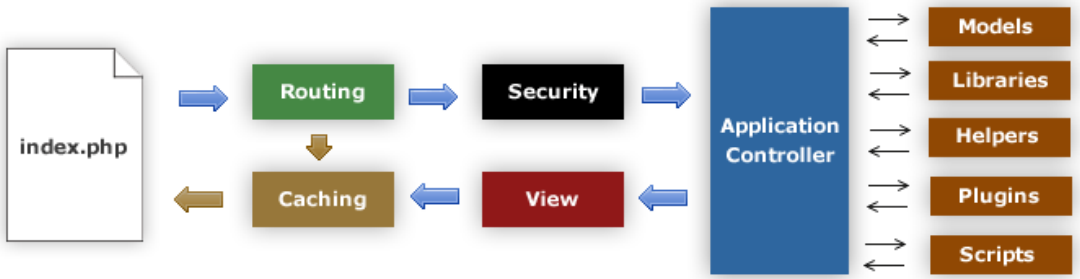
Sovelluskehyksen käytöllä voidaan saavuttaa useita etuja web-palvelujen kehittämisessä, kun verrataan siihen, että kehitys aloitettaisiin puhtaalta pöydältä. Kehykset tarjoavat valmiita komponentteja ja pohjan yleisimmin tarvittavia toimintoja varten, eikä jo toimivia ominaisuuksia tarvitse lähteä kehittämään alusta asti itse. Lähdekoodista tulee helppolukuisempaa, sillä kehitetyn sovelluksen koodi on erillään valmiina tarjotusta pohjasta. Tämä helpottaa huomattavasti myös jatkokehitystä. Myös tietoturva paranee sisäänrakennettujen ominaisuuksien avulla. Yhtenäiset ohjelmointikäytännöt helpottavat kehitystyötä aktiivisen kehittäjäyhteisön tuen avulla. (Niemi 2015, hakupäivä 19.5.2017.)

Sovelluskehyksen valinnassa on muutamia yleisiä laatukriteerejä, jotka on hyvä ottaa huomioon. Kehyksen tulee olla aktiivisesti ylläpidetty ja hyvin dokumentoitu. Sen tulee olla riittävän yleisesti käytetty, jotta apua ratkaisujen etsimiseen on helpompi löytää. Yleisimpien ominaisuuksien sisällytys tai liitettävyys on oleellinen, jotta kehyksestä saadaan irti sen tarjoama hyöty. Kehyksen tulee tukea erilaisia sovellusarkkitehtuureja, jotka mahdollistavat halutun sovelluksen toteuttamisen. Laatukriteerien täyttyessä voidaan tarkempi valinta tehdä useiden eri vaihtoehtojen

välillä. Eri kehykset tukevat eri ohjelmointikieliä, ja halutun sovelluksen toteutus voidaan saada aikaiseksi monella eri kehyksellä ja kielellä. Jokaisella sovelluskehyksellä on omat hyvät ja huonot puolensa. (Niemi 2015, hakupäivä 19.5.2017.)

PHP-kieltä tukevat useat eri sovelluskehykset, joista CodeIgniter, Symfony, CakePHP, Zend ja Laravel ovat tällä hetkellä suosituimpia. Phalcon on toiminnaltaan nopein, mutta sen tunnettavuus on tällä hetkellä vielä heikko verrattuna muihin. Suosituin ja CodersEye –sivuston vertailun voittajaksi on valikoitunut Laravel. Sen parhaita puolia on tuki normaalista relaatiomallista poikkeaville tietokannoille. Laravelin jälkeen toiseksi suosituimpana Googlen hakukonevertailussa on CodeIgniter. (CodersEye 2017, hakupäivä 22.5.2017.) Opinnäytetyössä toteutettu sovellus olisi voitu hyvin rakentaa useammalla eri vaihtoehdolla. Valinnaksi muodostui kuitenkin CodeIgniter, joka oli ennestään tuttu, eikä toisen sovelluskehyksen opettelu ollut välttämätön asia työn toteutusta varten. Se tukee myös sovelluksessa käytettyä MVC-mallia suoraan, vaikkei sen orjallinen seuraaminen välttämätöntä olekaan. Myös vähäisen konfiguraation määrä alentaa kynnystä CodeIgniterin käyttöön. CodeIgniter sisältää myös useita hyödyllisiä kirjastoja ja ominaisuuksia ja se on erinomaisesti dokumentoitu. Esimerkkinä syötteiden tarkastus voidaan toteuttaa CodeIgniterin omilla luokilla. Kattavampaa vertailua ei ollut hyödyllistä tehdä, sillä toteutettu sovellus ei ole merkittävän monimutkainen. Suuremmalla sovelluksella suorituskyky olisi jo merkittävä tekijä ja CodeIgniter suoriutuu MVC-arkkitehtuuria käytettäessä paremmin kuin useimmat kilpailevat sovelluskehykset (CodersEye 2017, 22.5.2017). Mahdollisen jatkokehittelyn osalta CodeIgniter on siksi myös hyvä valinta.

CodeIgniterin toiminta voidaan selittää helpoiten vuokaavion avulla (Kuvio 13). Index.php toimii pääohjaimena, joka huolehtii CodeIgniterin toiminnan mahdollistavien resurssien alustamisesta. Routing käsittelee HTTP-pyynnöt ja selvittää mitä niille tehdään. Jos evästeet ovat käytössä, ohitetaan muutoin normaalit ohjelman toimenpiteet ja toimitaan välimuistissa sijaitsevien toimintojenohjausten mukaisesti. Ennen kuin sovellukseen ohjelmoituja toimintoja ajetaan, HTTP-pyynnöt käsitellään vielä tietoturvan varmistamiseksi. Tämän jälkeen ohjain lataa mallin, ydinkomponentit, avustajat ja muut tarvittavat resurssit halutun toiminnan suorittamiseksi. Lopulta näkymä lähetetään selaimelle. Evästeiden ollessa käytössä ne ladataan välimuistiin ennen näkymän lähettämistä selaimelle. (British Columbia Institute of Technology 2017, hakupäivä 22.5.2017.)



KUVIO 13. CodeIgniter-sovelluskehyksen vuokaavio.

CodeIgniterin asetukset voidaan asettaa niille tarkoitettuihin PHP-tiedostoihin. Yleiset asetukset, kuten sovelluksen ja etusivun osoitteet asetetaan config.php –tiedostoon. Tietokannan asetukset tulevat database.php –tiedostoon. Jos sovelluksessa käytetään yleisesti CodeIgniterin omia avustavia luokkia ja kirjastoja, ne voidaan merkata ladattavaksi automaattisesti autoload.php –tiedostoon. CodeIgniterin muodostama selaimessa näkyvä osoite muodostuu sivun perusosoitteesta (example.com/index.php), ohjaimen nimestä (products), metodin nimestä (shoes) ja mahdollisesta metodin parametrista (123) (Kuvio 14). Asetuksilla osoite voidaan myös siistiä poistamalla index.php:n näkyminen osoitteen keskeltä. (Oulun seudun ammattikorkeakoulu 2017, hakupäivä 28.5.2017.) Tässä työssä oleelliset avustavat luokat ja kirjastot liittyvät istuntojen hallintaan, tiedon salaukseen, kuvien manipulointiin, syötteiden tarkastamiseen ja suhteellisen osoitteen luomiseen.

```
example.com/index.php/products/shoes/sandals/123
```

KUVIO 14. CodeIgniterin osoitteenmuodostus.

4 KÄYTTÖLIITTYMÄSUUNNITTELU

Käyttöliittymä tarkoittaa sovelluksen osaa, jonka avulla käyttäjä kommunikoi sovelluksen kanssa. Sen suunnittelulla pyritään varmistamaan käyttäjien tarpeet. Suunnittelu on erilaista riippuen ympäristöstä, johon se kohdistuu. On eri asia suunnitella asianpesukoneen kuin web-sovelluksen käyttöliittymää. Samoin eroa on työpöytä- ja mobiilisovelluksella, vaikka ne voidaankin laskea samaksi asiaksi. Käyttöliittymän on oltava sellainen, että sen käyttö on helppoa ja luonnollista. Käyttäjältä ei tule vaatia erillistä opettelua, vaan toimintojen tulee olla helposti tehtävissä ja nähtävissä. Usein käyttäjien on helpompi omaksua sellaisen sovelluksen käyttö, joka muistuttaa vastaavia ohjelmia. (Kemppainen 2014a, hakupäivä 22.5.2017.)

Peruslähtökohtana on tehdä näkymistä mahdollisimman yksinkertaiset, jotta niissä ei ole mitään ylimääräistä, mutta kaikki tarvittava. Toisiinsa liittyvät asiat on hyvä ryhmitellä lähelle toisiaan ja niiden tulee olla loogisessa järjestyksessä. Yhdessä näkymässä tietojen asettelun ja ryhmittelyn tulee olla loogista, jolloin tiedon etsintä nopeutuu. Mahdollisissa virhetilanteissa, esimerkiksi käyttäjän antaessa vääränlaisen syötteen kenttään, tulee käyttäjälle annetun palautteen olla kohtelias ja helposti ymmärrettävä. Visuaalisilla keinoilla, kuten väreillä ja fonteilla, voidaan helpottaa huomion kiinnittämistä haluttuihin elementteihin tai asioihin. (Kemppainen 2014a, hakupäivä 22.5.2017.)

Käyttöliittymään kohdistuu useita ja myös ristiriitaisia tavoitteita. Käyttöliittymään vaikuttaa erityisesti se kuinka eri tavoitteita halutaan painottaa. Tavoitteiden muodostama ehjä kokonaisuus menee kuitenkin yksittäisten asioiden edelle (Lehdonvirta & Korpela 2013, 84.) Lehdonvirran mainitsemista harkittavista vaatimuksista oleellisimpia tässä työssä ovat toimivuus, mukautuvuus ja käytettävyys.

Toimivuudella tarkoitetaan sovelluksen toimivuutta niissä ympäristöissä, joihin se on ensisijaisesti tarkoitettu. Opinnäytetyön sovelluksen julkinen etusivu on tarkoitettu toimimaan sekä työpöytä- että mobiililaitteissa. Mukautuvuus tarkoittaa käyttöliittymän kykyä mukautua eri laitteiden ja selainten ominaispiirteisiin korostaen laitteiden näytön kokoa. Twitter Bootstrapin avulla voidaan huolehtia näistä molemmista mainituista tavoitteista. Ei ole tarvetta kehittää erillisiä käyttöliittymiä työpöytä- ja mobiilikäyttöön, vaan Bootstrap huolehtii toimivuudesta ja mukautuvuudesta molempien osalta.

Hallintapaneelin osalta tavoitteet ovat hieman erilaiset. Sen ei tarvitse mukautua mobiililaitteisiin, koska käyttötarve on ainoastaan työpöytälaitteelle.

Responsiivinen tekniikka ja sivustojen mobiilikäyttö ovat lisänneet yksisivuisina (one page) toteutettujen sivustojen lukumäärää, varsinkin pienempien sivustojen osalta. Käyttäjät arvostavat nykyään sitä, että kaikki tieto on esitetty yhdellä sivulla. (Marjamäki 2014, hakupäivä 23.5.2017.) Toteutetun etusivun tiedon määrän ollessa suhteellisen pieni, valittiin käyttöön juuri one page – tyylinen esitystapa. Tämä yksinkertaistaa sivua käytettävyydeltään ja tieto esitetään loogisessa järjestyksessä selatessa sivua alaspäin graafisten elementtien tukemana.

4.1 Responsiivisuus

Responsiivinen suunnittelu tarkoittaa sitä, että käyttöliittymä sopii hyvin käytettäville laitteelle. Oleellinen asia on tällöin laitteen näytön koko. (Lehdonvirta & Korpela 2013, 103.) Mobiililaitteet ovat tuoneet uudenlaisia haasteita web-sivustojen rakentamiselle. Isommalle ruudulle tarkoitettu sivusto ei aina toimi tarkoituksen omaisesti pienemmän näytön omaavalla mobiililaitteella. Ongelma ratkaistiin aiemmin tekemällä sivustosta erillinen versio mobiililaitteita varten ja tätä tapaa käytetään edelleenkin. Sivusta voidaan myös tehdä samanaikaisesti toimiva sekä pöytäkoneita että mobiililaitteita varten responsiviisen suunnittelun avulla. (Kuivanen 2014, Hakupäivä 16.5.2017.) Responsiivisen suunnittelun kehittäjänä voidaan pitää Ethan Macottea, joka esitteli idean vuonna 2010. Responsiivinen suunnittelu on tarkoitettu käytettäväksi yhdessä joustavan sommittelun kanssa, joka on Macotten mukaan responsiivisuuden perusta. Elementtien koot tulisi esimerkiksi määritellä mieluummin prosentteina kuin staattisina pikselikokoina, koska tällöin ne mukautuvat paremmin eri laitteille. (Lehdonvirta & Korpela 2013, 115–116.)

Suunnittelussa otetaan yleensä huomioon muutamat eri kokoiset näytöt. Väliin putoavat laitteet huomioidaan niin, että niille käytetään valituista vaihtoehdoista lähinnä sopivinta. Responsiivista suunnittelua voidaan kutsua myös mukautuvaksi suunnitteluksi. Tiedon esityksen muutos käyttäjän toimenpiteen takia ei kuitenkaan yleensä tarkoita responsiivisuutta, vaikka tämäkin voi olla totta esimerkiksi selainikkunan kokoa muutettaessa. Oleellista on se, että sisällön esitys riippuu käytettävän laitteen ominaisuuksista. Joustavalla sommittelulla, jossa elementtien koot muuttuvat

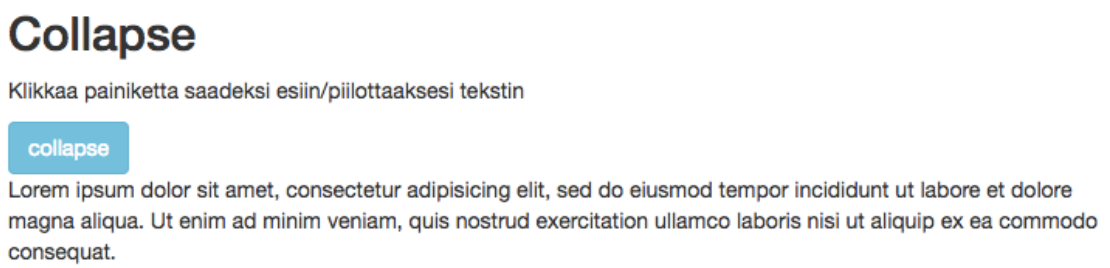
suhteessa selaimen kokoon on etunsa, mutta usein se ei auta, kun tavoitteena on saada sisältö toimivaksi useissa eri laitteissa. Pienillä näytöillä tällainen suunnittelu johtaa sisällön leikkautumiseen. tai elementit muuttuvat liian kapeiksi. Responsiivisuus toteutetaan usein CSS-tyylitiedostojen avulla, mutta se voidaan tehdä myös muilla tekniikoilla (Lehdonvirta & Korpela 2013, 103–105.)

Erilaisiin laitteisiin mukautuminen ei tarkoita koodin kirjoittamista erikseen eri laitteille, vaan laitteiden ominaisuuksien tunnistamista ja mukautumista niihin. Kyse ei ole pelkästään fyysisestä laitteesta, vaan myös sen käyttötavasta. Esimerkiksi puhelimen käytöstä pysty- tai vaakasuunnassa. Tärkeä ominaisuus responsiivisuudessa on selaimen käytettävissä oleva koko. Sen tunnistamista varten eräs tapa on käyttää CSS:n mediakysely-ominaisuutta, jonka avulla voidaan asettaa niin sanottuja breakpointeja eli katkokohtia. Niiden avulla asettelu muuttuu selaimen leveyden mukaan. (Lehdonvirta & Korpela 2013, 105–107.)

Twitter Bootstrapia käyttämällä responsiivisuudesta ei tarvitse huolehtia manuaalisesti määriteltävien katkokohtien tai mediakyselyiden avulla. Valmis CSS-koodi huolehtii automattisesti elementtien asettelusta ruudunleveyden perusteella. Kun selainikkuna käytetyn laitteen mukaan kapenee, elementtejä muun muassa asetellaan allekkain. Suuremmalla ruudulla elementit asetellaan vierekkäin (Kuvio 15). Muutoin staattisen koon omaavat kuvat pienenevät myös automaattisesti. (Lehdonvirta & Korpela 2013, 112.) Bootstrap-projektin tavoitteena on priorisoida mobiililaitteita niin, että sivujen toiminta varmistetaan ensisijaisesti niille. (Kuivanen 2014, Hakupäivä 16.5.2017.)

KUVIO 15. Esimerkki responsiivisuudesta Twitter Bootstrapin avulla.

Osaa Bootstrapin toiminnoista ei voida käyttää pelkästään CSS:n avulla, koska se sisältää myös JavaScript-pohjaisia piirteitä. Esimerkiksi Collapse-ominaisuus, jolla voidaan piilottaa ja tuoda uudelleen näkyviin tietty elementti (Kuvio 16), tarvitsee toimiakseen myös JavaScriptin jQueryn-kirjaston. (Lehdonvirta & Korpela 2013, 96.)



KUVIO 16. Esimerkki Twitter Bootstrapin Collapse-toiminosta.

4.2 Graafinen suunnittelu

Typografia tarkoittaa sivun ulkoasua ja suunnittelua (Juselius 2004, hakupäivä 23.5.2017). Ulkoasun suunnittelu on kiinni käyttäjien erilaisista tarpeista. Käyttäjäkunta vaikuttaa käytettävään typografiaan, tekstin määrään ja väreihin. Web-sivu eroaa vanhemmista tiedon esitystavoista, kuten televisiosta tai painotuotteista, vaikka monet asiat ovatkin edelleen samoja. Typografian osalta merkittävin ero löytyy siitä, että web-sivu on esitysmuoto tiedolle, joka muuttuu selaimen ja käytetyn laitteen mukaan. Sivua ei voida käytännössä muotoilla staattiseksi kuten muissa medioissa. Erona on myös web-sivun dynaaminen esitysmuoto sen sisältäessä interaktiivisia toimintoja ja palautemahdollisuuksia. Pelkästään näyttävä grafiikka ei tarkoita, että sivun typografia olisi kohdallaan. Tutkimusten mukaan käyttäjät kokevat sivun tietosisällön ja käytettävyyden oleellisimmiksi asioiksi. (Korpela & Linjama 2005, 356.) Uudempien trendien mukaan kuvien määrä on kuitenkin lisääntymässä suureksi osin nopeampien verkkoyhteyksien takia, koska latausajat eivät ole enää niin riippuvaisia kuvien koosta (Marjamäki 2014, hakupäivä 23.5.2017).

Tasapaino tarkoittaa erilaisten elementtien välisiä suhteita. Web-sivuilla elementit hakevat parinsa näkymän jostakin toisesta kohdasta: taustakuva suhteessa navigointielementtiin, navigointielementti suhteessa otsikoihin ja niin edelleen. Erilaisten elementtien on hyvä olla ryhmiteltynä ja eroteltavissa omiksi kokonaisuuksikseen, mutta niiden ei tarvitse olla samankokoisia. (Juselius 2004, hakupäivä 23.5.2017.) Mitä enemmän sivustolla on tyhjää tilaa, sitä enemmän käyttäjän huomio kiinnittyy itse sisältöön ja elementteihin. Tekstien ja kuvien välissä

oleva tyhjä tila tekee myös tekstistä helppolukuisempaa. (Kemppainen 2014b, hakupäivä 24.5.2017.)

Fonttien muodoilla voidaan luoda tunnelmaa sivuille, joiden sisältö on lähinnä tekstiä. Myös vähemmän tekstiä sisältäville sivuille saadaan luotua ilmeikkyyttä fonttien valinnalla. Fonteilla on kaksi päätyyppiä: groteski ja antiikva (Kuvio 17). Antiikvassa kirjaimet päättyvät pieneen pääteviivaan, joka taas puuttuu groteskista fontista. Antiikva on muodoltaan koristeellinen ja sen viivat ovat usein eri paksuisia. Paintuotteissa sitä käytetään usein leipätekstissä. (Korpela & Linjama 2005, 372–373.)

KUVIO 17. Antiikva- ja groteski-fonttien vertailu.

Näytöltä luettaessa antiikva-fonttien luettavuus on heikompi verrattuna groteskiin (Korpela & Linjama 2005, 372–373). Pääsääntönä voidaan pitää sitä, että antiikvaa käytetään otsikoihin ja groteskia leipätekstiin. Fonttien käytössä tulisi myös muistaa se, että vähemmän on enemmän: liian usean erilaisen fonttityypin käyttäminen tekee sivusta sekavan ja vaikealukuisen. Sääntöä voidaan kuitenkin soveltaa, mikäli halutaan erityisesti korostaa jotain tekstin osaa eri fontilla. (Kyrnin 2017, hakupäivä 23.5.2017.)

5 TOTEUTUS

5.1 Lähtökohdat ja tavoitteet

Opinnäytetyön tavoitteena oli toteuttaa toimeksiantajalle uusi sivusto, joka palvelisi paremmin yrityksen tarpeita verrattuna vanhaan sivustoon. Sivuston tuli olla rakenteeltaan yksinkertainen one page –tyylinen sivu, jonka avulla kaikki asiakkaille haluttu tieto voitaisiin esittää tiiviisti ja tyylikkäästi moderneja tekniikoita hyväksikäyttäen. Vaatimuksina sivustolle olivat kuvagallerian ja yhteydenottolomakkeen toteuttaminen. Kuvagalleriaa tulisi voida päivittää yrityksen toimesta, ja tämä toteutettiin luomalla sivuston ohelle oma hallintapaneeli kirjautumisen taakse. Yhteydenottolomake tarvittiin tuotteisiin liittyvää yhteydenottoa sekä työhakemuksia varten. Työn pääasiallinen tarkoitus yrityksen näkökulmasta on parantaa näkyvyyttä uudella ulkoasulla ja uusilla toiminnoilla. Hakukoneoptimointi oli toteutettu jo vanhalle sivustolle, ja sitä käytetään myöhemmin pohjana uuden sivun optimointiin aiemmin mainituin keinoin.

5.2 Käytetyt työkalut ja ohjelmistot

NetBeans IDE on useille eri ohjelmointikielille ja sovelluskehyksille tarkoitettu avoimen lisenssin integroitu ohjelmointiympäristö. Netbeans on kirjoitettu Java-ohjelmointikielellä ja sen käyttäminen vaatii Javan ajoympäristön. Ohjelmiston kehittäminen alkoi vuonna 1996 Tsekkoslovakiassa ja kehittyi lopulta kaupalliseksi tuotteeksi. Sun Microsystems osti kehitystyöstä vastanneen yrityksen, ja julkaisi NetBeansin vuonna 2000 avoimen lähdekoodin projektina. (Netbeans 2017, hakupäivä 24.5.2017.) NetBeans on ollut tämän opinnäytetyön pääasiallisena kehitysympäristönä ja sitä käytettiin palvelin- ja asiakaspuolen ohjelmointiin. Valitun ohjelmistoympäristön vahvuuksia ovat sen käytön vapaus lisenssinsä vuoksi, sekä hyvä tuki erilaisille ohjelmointikielille ja sovelluskehyksille.

Kehitettävän sivuston paikallisen toteuttamisen ja testaamisen mahdollistamiseksi tarvittiin myös avuksi ohjelmisto, joka toimii virtuaalipalvelimena. Tähän tarkoitukseen käytettiin XAMPP-ohjelmaa, joka on NetBeansin tapaan avoimen lähdekoodin ohjelmisto. XAMPP on suosituin PHP-ohjelmakehityksen ympäristö paikalliseen testaamiseen. Se sisältää tuen PHP:n ja MySQL:n

käyttämiseen ja soveltui näin erinomaisesti työn toteuttamiseen. (Apache Friends 2017, hakupäivä 24.5.2017.) XAMPP sisältää myös phpMyAdmin-sovelluksen, joka on selainpohjainen hallintatyökalu MySQL-tietokannoille (Heinisuo 2003, 70.)

5.3 Hallintapaneeli

Sovelluksen palvelinpuoli ohjelmoitiin ensin, sillä MVC-arkkitehtuurin mukaisesti tietokantaa käyttävästä mallista ovat riippuvaisia sekä ohjain, että näkymä (Microsoft 2017, hakupäivä 19.5.2017.) Aluksi suunniteltiin tarvittava tietokanta, ja sen sisältämät taulut tietotyyppeineen. Tämän jälkeen suunniteltiin sovellus, joka käyttää hyväkseen tietokantaa. Kun CodeIgniter – projekti oli luotu NetBeans-ympäristöön ja tarvittavat asetukset tehty, aloitettiin itse ohjelmointityö.

Tarvittava tietokanta sisältää kaksi taulua. Taulut luotiin kuvagalleriaa varten ja kirjautumisen mahdollistavia käyttäjätietoja varten Heinisuon (2003, 64) ohjeiden mukaisesti SQL-kielellä, käyttäen phpMyAdmin-sovellusta (Kuvio 18). Tiedostot-taulussa id-kenttä on automaattisesti jokaisella lisäyksellä kasvava kokonaisluku, joka toimii avaimena tallennetulle riville. Tiedostonimi-kenttää käytetään linkkinä tallennettuun kuvaan, ja se on maksimissaan 255 merkkiä pitkä merkkijono. Thumb-kenttää käytetään linkkinä luotavaan esikatselukuvaan. Tallennettu-kenttä toimii aikaleimana. Kuvaus-kenttään voidaan tallentaa kuvagalleriassa näkyvä jokaiselle kuvalle yksilöity teksti. Käyttäjä-taulussa on tiedostot-taulun mukaisesti tunnisteena id-kenttä, ja käyttäjänimelle varattu 100-merkkiä pitkä oleva merkkijonokenttä tunnus. Lisäksi salasana-kentän sisältö salataan.

```
CREATE TABLE `tiedostot` (
 `id` int NOT NULL AUTO_INCREMENT,
 `tiedostonimi` varchar(255) NOT NULL,
 `thumb` varchar(255) NOT NULL,
 `tallennettu` datetime NOT NULL,
 `kuvaus` varchar(255) NOT NULL,
 `nimi` varchar(255) NOT NULL,
 PRIMARY KEY (`id`)
);

CREATE TABLE `kayttaja` (
`id` int NOT NULL AUTO_INCREMENT,
`tunnus` varchar(100) NOT NULL,
`salasana` varchar(255),
PRIMARY KEY (`id`)
);
```

KUVIO 18. Tietokannan luominen XAMPP-ohjelman phpMyAdmin-käyttöliittymän kautta.

Sovellukseen luotiin kaksi erillistä mallia: tiedosto_model ja kirjaudu_model. Ensimmäinen sisältää metodit kuvatiedostojen tietojen käsittelyä varten. Jälkimmäisellä hallitaan kirjautumistietoja.

Metodi luodaan public function –käskyllä, jonka perään annetaan metodille nimi (Kuvio 19). Kun metodille halutaan välittää parametrina muuttuja, kirjoitetaan se sulkuihin nimen perään. (Heinisuo 2003, 58–59). Esimerkkinä metodi hae, jolle välitetään muuttuja $id. Metodilla haetaan sille välitetyn avaimen eli $id:n perusteella haluttuun kuvaan liittyvät tiedot tietokannan taulusta tiedosto. Return-käskyllä kysely palautetaan sen hakeneelle metodille takaisin ohjaimeen. Lisaa-metodille välitetään oliona muuttuja $data, joka sisältää käyttäjän kuvatiedostoon liittyvät syötteet. Metodi lisää syötteet tauluun tiedostot. Tietokantaa käsitellään suoraan PHP-kielen avulla, kuten Heinisuo toteaa olevan mahdollista (Heinisuo 2003, 63).

```
public function hae($id) {
            $this->db->where('id',$id);
            $query = $this->db->get('tiedostot');
            return $query->row();
    }

public function lisaa($data){
    $insert = $this->db->insert('tiedostot',$data);
    return $insert?true:false;
}

public function muokkaa($data) {
            $this->db->where('id',$data['id']);
            $this->db->update('tiedostot',$data);
    }
```

KUVIO 19. Tiedosto_model –mallin hae–, lisaa– ja muokkaa–metodit.

Sovellus sisältää kolme ohjainta: Tiedosto, joka käsittelee käyttäjän syötteitä, ja Kirjaudu, joka käsittelee käyttäjän kirjautumissyötteet. Nämä kaksi ohjainta muodostavat jo käsiteltyjen mallien kanssa hallintapaneelin palvelinpuolen sovelluksen. Kolmas ohjain Home, toimii julkisen etusivun ohjaimena.

Kirjaudu-ohjaimen tarkasta_kayttaja –metodi ottaa vastaan käyttäjän antaman käyttäjätunnuksen ja salasanan ja asettaa ne muuttujiin $tunnus ja $salasana. Tämä jälkeen metodi vertaa annettuja muuttujia tietokantaan kirjaudu_model –mallin avulla. Jos tunnus ja salasana täsmäävät ehtolauseessa, metodi alustaa istunnon session-kirjaston avulla. Jos syöte on virheellinen, siirrytään ehtolauseen else-lohkoon, ja viedään näkymään mukana form validation –kirjaston avulla selkeä kehotus tunnuksen ja salasanan uudelleensyöttämisestä (Kuvio 20).

```php
public function tarkasta_kayttaja() {
    $tunnus = $this->input->post('tunnus');
    $salasana = $this->input->post('salasana');

    $kayttaja = $this->kirjaudu_model->tarkasta_kayttaja($tunnus,$salasana);

    if ($kayttaja) {
        $this->session->set_userdata('kayttaja',$kayttaja);
        return TRUE;
    }
    else {
        $this->form_validation->set_message('tarkasta_kayttaja','Käyttäjätunnus tai salasana virheellinen!');
        return FALSE;
    }
}
```

KUVIO 20. Kirjautumistietojen tarkistusmetodi.

Tiedosto-ohjain tarkistaa isset-toiminnolla onko käyttäjä kirjautunut sovellukseen. Jos käyttäjä yrittää päästä hallintapaneeliin kirjautumatta, hänet ohjataan takaisin kirjautumiseen redirect-komennolla. Mikäli kirjautumisistunto on voimassa, ladataan käyttöön kirjastosta syötteiden tarkastus (form validation), avustajat syötteille ja suhteellisille osoitteille (form ja url), sekä malli (Kuvio 21). Form validation –kirjaston toiminta on esitelty kuviossa 20.

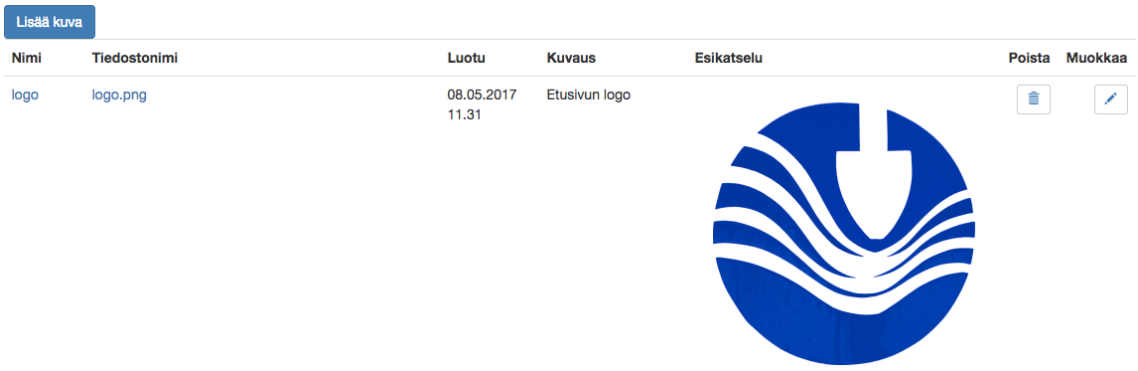
```php
class Tiedosto extends CI_Controller {
        public function __construct() {
                parent::__construct();

                if (!isset($_SESSION['kayttaja'])) {
                    redirect('kirjaudu');
                }
                $this->load->library('form_validation');
                $this->load->helper(array('form', 'url'));
                $this->load->model('tiedosto_model');
```

KUVIO 21. Tiedosto-ohjaimen alustus.

Alustuksen jälkeen sovellus siirtyy oletusmetodiin index ja lataa mallin kautta tietokantaan tallennettujen kuvien tiedot olioon, joka esitetään näkymässä taulukkomuotoisena listana. Sen jälkeen ladataan näkymät ja sovellus jää odottamaan käyttäjän toimenpiteitä (Kuvio 22). Ladatuista näkymistä muodostuu selaimeen hallintapaneelin näkymä, josta voidaan lisätä, muokata ja poistaa kuvia (Kuvio 23).

```php
    public function index() {

    $data['files'] = $this->tiedosto_model->get_rows();
    $this->load->view('template/header');
    $this->load->view('template/nav');
    $this->load->view('tiedostot_view', $data);
    $this->load->view('template/footer');
}
```

KUVIO 22. Tiedosto-ohjaimen index-metodi.



KUVIO 23. Hallintapaneelin näkymä.

Kun käyttäjä painaa lisää kuva –painiketta, avautuu selaimeen lomake, johon annetaan kuvalle halutut tiedot, sekä valitaan itse tallennettava kuvatiedosto. Kuva tallentuu painamalla Tallenna-nappia (Kuvio 24).

KUVIO 24. Hallintapaneelin lisää kuva –toiminnon näkymä.

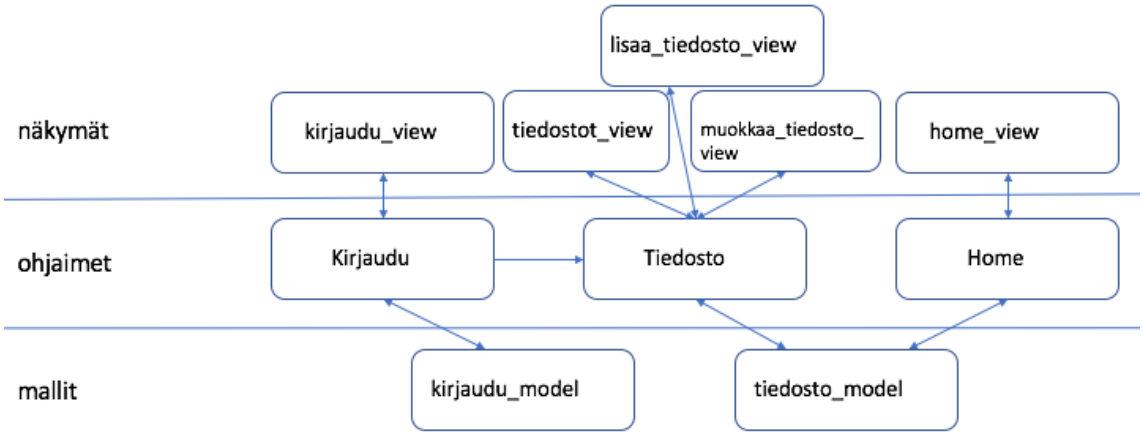
Kuvalle annettujen tietojen ja itse kuvatiedoston tallentaminen tapahtuu lisaa-metodin avulla. Ensiksi tarkistetaan ehtolauseella, onko tiedosto valittu ladattavaksi. Tämän jälkeen form validation -kirjaston avulla tarkistetaan, ovatko vaaditut kentät täytetty. Mikäli molemmat ehdot täyttyvät, annetaan kuvan manipulaatioon (image_lib) ja tiedoston lataukseen (upload) tarvittaville avustimille asetukset. Kuvasta luodaan myös pienennetty esikatselukuva. Tämän jälkeen itse tiedot tallennetaan tietokantaan tiedosto_model –mallin kautta (Kuvio 25). Kun käyttäjä on tehnyt haluamansa toimenpiteet, voidaan kirjautua ulos ja samalla tuhotaan istuntomuuttuja.

```php
$uploadPath = 'uploads/';
$config['upload_path'] = $uploadPath;
$config['allowed_types'] = 'gif|jpg|jpeg|bmp|png';

$this->load->library('upload', $config);
$this->upload->initialize($config);
if($this->upload->do_upload('userFile')){
    $fileData = $this->upload->data();
    $config['image_library'] = 'gd2';
    $config['source_image'] = 'uploads/'.$fileData['file_name'];
    $config['create_thumb'] = TRUE;
    $config['maintain_ratio'] = TRUE;
    $config['width']         = 300;
    $config['height']       = 300;

    $this->load->library('image_lib', $config);

    $this->image_lib->resize();
    $data = $this->upload->data();

    $uploadData['tiedostonimi'] = $fileData['file_name'];
    $uploadData['thumb'] = $data['raw_name'].'_thumb'.$data['file_ext'];
    $uploadData['tallennettu'] = date("Y-m-d H:i:s");
    $uploadData['kuvaus'] = $this->input->post('kuvaus');
    $uploadData['nimi'] = $this->input->post('nimi');
```

KUVIO 25. Kuvatiedoston latausmetodi.

Koska tieto hallintapaneelin olemassaolosta on hyvä pitää vain palvelua käyttävän yrityksen tiedossa, sinne ei ole linkkiä missään julkisella etusivulla. Vaikka pääsy hallintapaneeliin on rajattu käyttäjätunnuksen taakse, tietoturva paranee tällä menettelyllä. Myös hallintapaneelin sivujen indeksointi on estetty Korpelan ohjeiden mukaisesti, jotta hakukoneet eivät näytä niitä hakutuloksissa (Korpela & Linjama 2005, 149.)

Kuviossa 26 on esitelty yhteenvetona sovelluksen eri osat ja niiden väliset yhteydet jaettuna MVC-arkkitehtuurin mukaisesti.



KUVIO 26. Sovellus jaettuna MVC-arkkitehtuurin mukaisiin osiin.

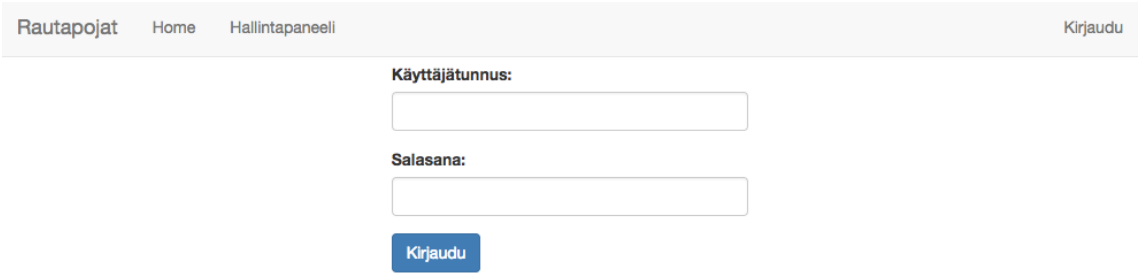
5.4 Käyttöliittymä ja näkymät

Koska etusivulle tulevia kuvia, ja osia tekstistä ei tässä vaiheessa vielä ollut vielä saatavilla, etusivun esittely koskee suurimmassa osin sen rakennetta ja sen sisältämiä elementtejä. Itse ulkoasu muodostuu lopulliseen muotoonsa myöhemmässä vaiheessa. Hallintapaneelin palvelinpuolen valmistuttua, alettiin rakentaa sen käyttöliittymää ja ulkoasua. Jokainen toteutettu näkymä jaettiin rakenteellisesti kolmeen tiedostoon: yläviitteeseen (header), pääsisältöön ja alaviitteeseen (footer). Yläviite sisältää html- ja body-merkintöjen alut, sekä viitteet CSS- ja JavaScript –tiedostoihin. Myös navigointi-elementti sisällytettiin yläviitteeseen, jotta se pysyy samanlaisena jokaisessa näkymässä. Pääsisältöön ladataan kunkin sivun vaihtuva sisältö omana tiedostonaan, kuten näkymä tallennetuista kuvista, tai kuvanlisäyslomake. Tämän lisäksi mukana viedään haluttu tieto tietokannasta $data-muuttujan avulla. Alaviite sisältää body- ja html-merkintöjen pääteet, nimenomaisessa järjestyksessä. CodeIgniter mahdollistaa tämän tyyppisen sivumalliratkaisun, joka helpottaa ohjelmointityötä ja ohjelmointikoodin jäsentelyä. Näin samaa koodia ei tarvitse kirjoittaa erikseen jokaiseen vaihtuvaan sisältösivuun, ja viittaukset CSS- ja JavaScript –tiedostoihin ovat automaattisesti käytössä jokaisessa näkymässä (Kuvio 27).

```
$data['files'] = $this->tiedosto_model->hae();

$this->load->view('template/header');
$this->load->view('tiedostot_view', $data);
$this->load->view('template/footer');
```

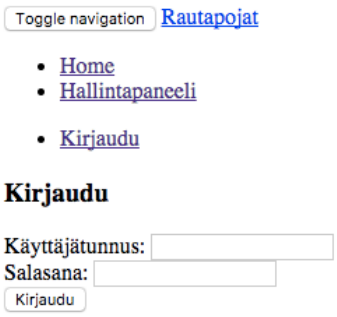
KUVIO 27. CodeIgniter –sivumallin esimerkki.

Kuviossa 28 on esitelty hallintapaneelin kirjautumissivu. Sen yläosassa näkyvä navigointipalkki pysyy samanlaisena myös muilla hallintapaneelin sivuilla. Oikean yläkulman Kirjaudu-linkki on näkyvissä ainoastaan testausvaiheessa, sillä oikeassa käytössä käyttäjä tietää olevansa kirjautuneena. Vasemmalla oleva Home-linkki vie julkiselle etusivulle, josta voidaan esimerkiksi tarkistaa palveluun lisätyn kuvan näkyvyys. Hallintapaneelin käyttöliittymä pidettiin mahdollisimman yksinkertaisena ja havainnollisena. Siinä on ainoastaan kaikki tarvittava, ja käyttäjän katse hakeutuu automaattisesti tärkeisiin kohteisiin. Tätä efektiä saatiin vahvistettua myös elementtien ympärille jätetyllä tyhjällä tilalla, kuten Juselius esittää (Juselius 2004, hakupäivä 23.5.2017.)



KUVIO 28. Hallintapaneelin kirjautumissivu muotoiltuna Bootstrapin avulla.

Alla olevassa kuviossa 29 on vertailuna sama kirjautumissivu ilman Bootstrap CSS -muotoilua. Kuva havainnollistaa hyvin sitä, mikä tyylitiedostojen käytön merkitys on. Vaikka sivu on käytettävissä, joutuisi muotoilemattoman ulkoasun kanssa käyttämään enemmän aikaa oikeiden elementtien etsimiseen ja sivun hahmottamiseen.



KUVIO 29. Hallintapaneelin muotoilematon kirjautumissivu.

Kuviossa 30 on esitelty koodi, jolla tulostetaan hallintapaneelin etusivulla näkyvät kuvatiedostot ja niihin liittyvät tiedot. Esimerkissä on käytetty PHP-kieltä HTML-sivuun upotettuna. Foreach-silmukan avulla käydään oliosta $files läpi kaikki sen sisältämät tiedot, ja tulostetaan ne taulukkomuodossa. Silmukka on if-ehtolauseen sisällä, koska jos tietokanta olisi tyhjä, tai siihen ei saataisi jostain syystä yhteyttä, tulostuisi käyttäjän näytölle epämiellyttävän näköinen virheilmoitus. Epämiellyttävyydellä tarkoitetaan tässä yhteydessä käyttökokemuksen miellyttävyyttä. Nyt tyhjä taulukko aiheuttaa ainoastaan else-lohkon tulostaman ”Tiedostoja ei löytynyt” –viestin.

```
    </tr>
    <?php if (!empty($files)): foreach ($files as $file): ?>
            <?php $id = $file['id']; ?>

            <td><a href="<?php echo base_url('uploads/' . $file['tiedostonimi']); ?>"><?php echo $file['nimi']; ?>
            <td><a href="<?php echo base_url('uploads/' . $file['tiedostonimi']); ?>"><?php echo $file['tiedostonim
            <td><?php echo date($this->util->format_sqldate_to_fin($file['tallennettu'])); ?></td>
            <td><?php echo $file['kuvaus']; ?><td>
            <td><a href="<?php echo base_url('uploads/' . $file['tiedostonimi']); ?>"><img src="<?php echo base_url
            <td><?php echo anchor("tiedosto/poista/$id", "<button type='submit'  type='button' p class='btn btn-de
            <td><?php echo anchor("tiedosto/muokkaa/$id", "<button type='submit'  type='button' p class='btn btn-de
        <?php endforeach;
    else: ?>
        <p>Tiedostoja ei löytynyt</p>
<?php endif; ?>
```

KUVIO 30. Hallintapaneelin etusivun taulukon tulostava koodi.

Saman sivun lähdekoodia tarkastelemalla ei voida kuitenkaan päätellä, onko tulostus tehty dynaamisesti PHP:n avulla, sillä se näyttää vain HTML-merkkauksen (Kuvio 31). Tämä ei sinänsä ole työn kannalta merkittävä asia, mutta osoittaa Rantalan mainitseman ominaisuuden liittyen PHP-kieleen (Rantala 2005, 17.) Lisäksi tämä estää käyttäjää näkemästä palvelinpuolen koodia.

```
            <td><a href="http://localhost/ci_rautapojat_template/uploads/logo.png">logo</a></td>
            <td><a href="http://localhost/ci_rautapojat_template/uploads/logo.png">logo.png</a></td>
            <td>08.05.2017 11.31</td>
            <td>Etusivun logo<td>
            <td><a href="http://localhost/ci_rautapojat_template/uploads/logo.png"><img
src="http://localhost/ci_rautapojat_template/uploads/logo_thumb.png"></a></td>
            <td><a href="http://localhost/ci_rautapojat_template/tiedosto/poista/64"><button type='submit'  type='button' p
class='btn btn-default btn-sm pull-right'><span class='glyphicon glyphicon-trash'></span></a></td>
            <td><a href="http://localhost/ci_rautapojat_template/tiedosto/muokkaa/64"><button type='submit'  type='button' p
class='btn btn-default btn-sm pull-right'><span class='glyphicon glyphicon-pencil'></span></a></td></tr>
```

KUVIO 31. Kuvan tulostuksen lähdekoodi selaimessa.

Etusivun suunnittelussa lähtökohdat olivat sen responsiivisuus ja selkeys. Sivun elementit muokkautuvat eri näytöille, ja muotoilussa on käytetty Twitter Bootstrapia. Etusivun navigointipalkki toimii muutoin kuten hallintapaneelissa, mutta siihen on tehty muutamia tarkentavia määrityksiä. Sivun ollessa selattuna ylös asti, navigointipalkki on läpinäkyvä. Kun sivua selataan alaspäin, palkki muuttuu läpinäkymättömäksi, ja mustaksi taustaltaan (Kuvio 32). Tarkoituksena on tuoda yläreunan navigointi enemmän esille sivua selatessa. Samoin sivun ulkoasu tulee paremmin esiin, kun sivu on selattuna ylös eikä navigointipalkki vie huomiota logosta tai taustakuvasta. Kuvassa näkyy vielä testausvaiheessa väliaikaisesti nimetyt linkit ja myöhemmin poistettava hallintapanelin linkki. Näillä linkeillä voidaan selata määriteltyyn sivun kohtaan. Kun näytön koko kapenee, linkit katoavat näkyvistä ja siirtyvät pudotusvalikon alle.

KUVIO 32. Etusivun navigointipalkki muuttuu alaspäin selatessa läpinäkymättömäksi.

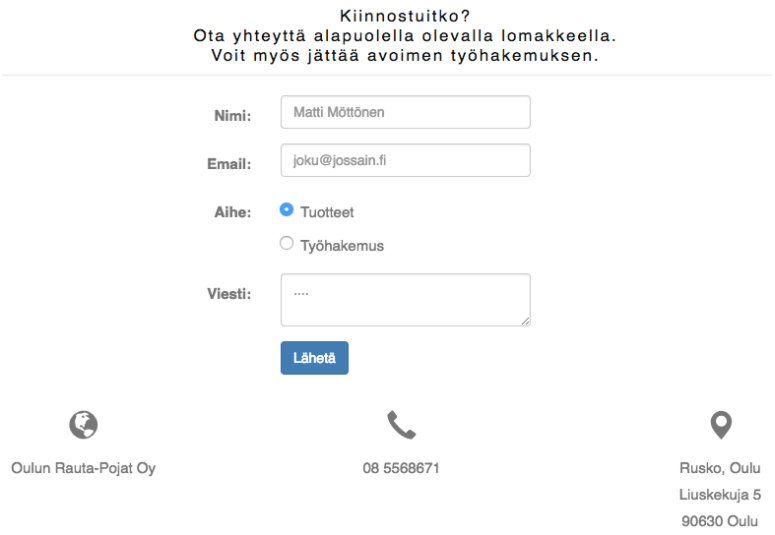
Efekti on luotu käyttämällä JavaScriptin jQuery-kirjastoa. Skriptillä muokataan CSS-määritettä ottamalla käyttöön, tai poistamalla käytöstä navigointipalkin taustaväri. Skripti havaitsee kohdan, jossa sivua selataan ja asettaa määrityksen sen mukaan (Kuvio 33).

```
$(window).scroll(function() {
if ($(document).scrollTop() > 50) {
  $('nav').addClass('navbar-inverse');
} else {
  $('nav').removeClass('navbar-inverse');
}
```

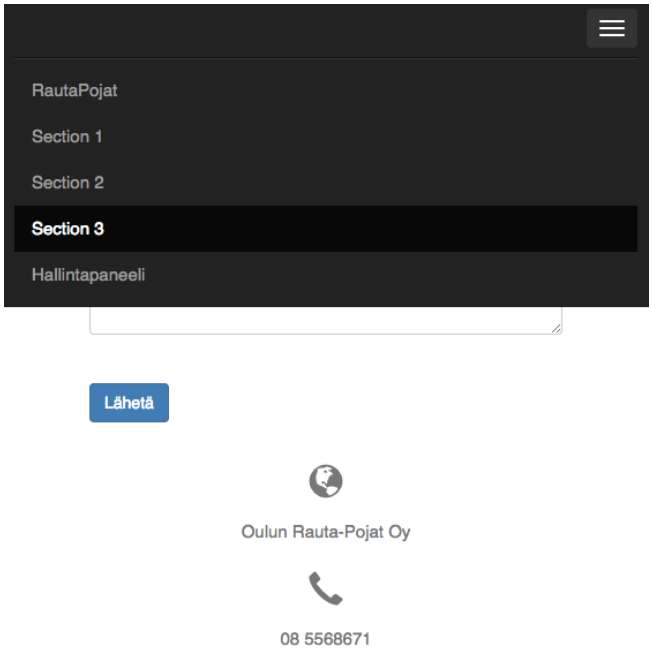
KUVIO 33. Navigointipalkin taustavärin muokkaus jQueryn avulla.

Etusivun kuvagalleria on niin sanottu karuselli, jossa näytettävä kuva vaihtuu määritellyn ajan välein. Lopullista gallerian esitysmuotoa ei kuitenkaan ole vielä valittu, ja sen muotoilu tapahtuu myöhemmin. Koska gallerian toiminta perustuu MVC-mallin mukaisesti vain malliin ja ohjaimeen, voidaan valittu esitystapa muuttaa tarvittaessa myöhemmin varsin helposti muokkaamalla vain etusivun näkymää. Tämä on yksi MVC-arkkitehtuurin hyvistä ominaisuuksista, sillä näkymän muutos ei aiheuta toimenpiteitä ohjaimeen tai mallin. Näkymään tehtävät muutokset ovat lähinnä kosmeettisia, esitystapa voidaan muuttaa vaihtamalla html-elementit PHP-koodien ympärille.

Yhteydenottolomake sijaitsee etusivulla alempana yritys- ja tuote-esittelyjen jälkeen. Käyttöön on valittu groteski fontti, sen selkeämmän luettavuuden takia. Samaa fonttia käytetään myös muualla sivun leipäteksteissä. Lomakkeen muotoilu ja palautteen lähetysnappi on pidetty samankaltaisena kuin muuallakin sivulla. Ainoa ero sovelluksen muihin lomakkeisiin on se, että kenttien selitteet on siirretty kenttien vasemmalle puolelle. Syynä tähän on pienempi tilankäyttö pystysuunnassa, ja helpompi hahmotettavuus kenttien määrän ollessa suurempi. Lomakkeelle annetut syötteet tarkistetaan sekä asiakas- että palvelinpuolella. Jos jonkin kentän sisältö on tyhjä tai virheellinen, annetaan käyttäjälle tästä ilmoitus. Lomakkeen alapuolella olevat symbolit antavat vielä selkeän yhteenvedon yrityksen yhteystiedoista (Kuvio 34). Myös ne toimivat responsiivisesti, ja siirtyvät allekkain näytön ollessa kapeampi (Kuvio 35).



KUVIO 34. Palautelomake ja symbolit yhteystietojen yhteydessä.



KUVIO 35. Palautelomake näkyvissä pudotusvalikon alla kapealla näytöllä.

6 JOHTOPÄÄTÖKSET JA POHDINTA

Opinnäytetyön tarkoituksena oli toteuttaa toimeksiantajalle toimineelle Oulun Rauta-Pojat Oy:lle uusi sivusto. Vanhan sivuston suurimpina puutteina olleet sivuston ulkoasun vanhanaikaisuus ja toiminnallisuuksien puuttuminen. Mainittujen puutteiden korjaamisen lisäksi tavoitteena oli parantaa yrityksen näkyvyyttä. Näiden tarpeiden pohjalta tavoitteita tarkennettiin käsittämään uuden ulkoasun toteuttaminen ja toiminnallisuuksien mahdollistamiseksi rakennettava sovellus. Sivustolle haluttiin kuvagalleria sekä palautelomake, ja ulkoasun tuli mukautua eri laitteille sopivaksi. Sovelluksen tarkoituksena oli mahdollistaa toimeksiantajan omatoiminen kuvagallerian hallinnointi. Palautelomakkeella tuli olla mahdollista lähettää yrityksen sähköpostiin työhakemuksia ja tuotekyselyitä. Näkyvyyttä pyrittiin parantamaan kuvagallerian kautta sekä hakukoneoptimoinnin avulla. Sivuston ulkoasuksi valikoitui yksisivuinen malli, jossa kaikki tieto esitetään tiivistetysti, mutta mitään pois jättämättä.

Tavoitteissa onnistuttiin suhteellisen hyvin. Opinnäytetyöprosessiin käytettävissä olleen varsin rajatun ajan takia hakukoneoptimointi jäi vielä osiltaan myöhempään vaiheeseen, koska sivustolle tulevaa materiaalia ei vielä ollut juuri käytettävissä. Samasta syystä sivuston etusivun ulkoasu on vielä osittain kesken. Sivuston rakenne on valmis, mutta sinne tuleva materiaali vaikuttaa osaltaan muotoiluun sekä tekstin että kuvien osalta. Myös kuvagallerian esitystapa valitaan myöhemmin. Edellä mainitut toimenpiteet ovat kuitenkin varsin helposti toteutettavissa, sillä kyse on vain sivuston elementtien määrittelystä ja muotoilusta. Varsinaista ohjelmointityötä ei enää tarvitse tehdä.

Sivuston asettelu mukautuu hyvin erilaisten näyttökokojen omaavilla laitteille. Tätä toteutettaessa käytetty CSS-tyyliopas Twitter Bootstrap osoittautui hyväksi työkaluksi. Määrittelyjen tekeminen oli varsin yksinkertaista, ja vaikka muutamat elementit tarvitsivat yksilöllistä muotoilua, ei suuria ongelmia syntynyt. Hallintapaneelin muodostava sovellus ohjelmoitiin palvelinpuolen osalta PHP-kielellä, joka oli minulle jo ennestään tuttu, mutta osaaminen syventyi työn aikana. Sovelluskehykseksi valitun CodeIgniterin osaaminen parani myös huomattavasti. Asiakaspuolen toteuttamisen apuna käytetty JavaScript tuotti aluksi hieman ongelmia. Tietopohjan kerääminen ja erilaisten ratkaisujen kokeileminen kuitenkin auttoivat tavoitteiden saavuttamiseen.

Tietopohja kerättiin suurimmaksi osin muutamasta pääteoksena käytetystä kirjasta, joita voisi luonnehtia tietojenkäsittelyn alan huomioon ottaen ikääntyneiksi. Teoksista kuitenkin löytyi edelleen käyttökelpoista materiaalia, jota täydennettiin ja tuettiin uudempien verkkolähteiden avulla. Näin voitiin osoittaa, että web-ohjelmoinnin liittyvät perusperiaatteet pysyvät pääpiirteitään samoina.

Jatkotoimenpiteinä hakukoneoptimointi tehdään loppuun ja sivuston ulkoasu saa lopullisen muotonsa. Lisäksi sovellus siirretään käyttöön oikealle palvelimelle. Sivusto on tällä hetkellä vain suomenkielinen, mutta lokalisaatio olisi toteutettavissa sovelluskehyksen sisältämien kirjastojen avulla, mikäli tämä tarpeelliseksi koetaan. Muita mahdollisia jatkokehittelyn kohteita voisivat olla myös muun sisällön dynaaminen esitystapa kuvagallerian tapaan ja edelleen parannettu hakukonenäkyvyys. MVC-arkkitehtuurin mukainen ohjelmointitapa mahdollistaa sisällönhallinnan myös muun sisällön osalta ilman, että sovellusta tarvitsisi ohjelmoida uudelleen. Lisäominaisuuksia voidaan toteuttaa käyttäen hyväksi jo olemassa olevia metodeja.

LÄHTEET

2Kmediat. 2017. JavaScript-opas. Hakupäivä 17.5.2017, http://www.2kmediat.com/jscript/.

Addison Wesley Longman. 1998. Raggett on HTML 4 1998. Hakupäivä 7.5.2017, https://www.w3.org/People/Raggett/book4/ch02.html.

Apache Friends. 2017. What is XAMPP? Hakupäivä 24.5.2017, https://www.apachefriends.org/.

Bos, B. 2016. A Brief history of CSS until 2016. Hakupäivä 11.5.2017, https://www.w3.org/Style/CSS20/history.html.

British Columbia Institute of Technology. 2017. CodeIgniter Overview. Hakupäivä 22.5.2017, https://www.codeigniter.com/user_guide/overview/index.html.

CodersEye. 2017. 11 Best PHP Frameworks for Modern Web Developers. Hakupäivä 22.5.2017, https://coderseye.com/best-php-frameworks-for-web-developers/.

Edwards, C. 2014. Is the Meta Keyword Tag still used by Google, Bing and Yahoo? Hakupäivä 26.5.2017, https://chrisedwards.me/seo/keyword-meta-tag-google/.

Försström, M. 2015. JavaScriptin lyhyt historia. Hakupäivä 15.5.2017, https://fraktio.fi/blogi/javascriptin-lyhyt-historia/.

Heinisuo, R. 2003. PHP ja MySql. Helsinki. Talentum Media Oy.

Hovi, A., Huotari, J & Lähdenmäki, T. 2003. Tietokantojen suunnittelu & indeksointi. Jyväskylä. Docendo Finland Oy.

Hällström LTD Partnership. 2017. PHP:n perusteet. Hakupäivä 16.5.2017, http://hallstrom.fi/php_opas/php_2.html.

Ihantola, P. 2016. Web-ohjelmointi. Hakupäivä 18.5.2017, http://www.cs.tut.fi/~seitti/2015/luennot/.

Juselius, U. 2004. Typografia. Hakupäivä 23.5.2017, http://www.phpoint.fi/ulrikaj/www/typo.htm.

Kareinen, A. 2017. Olio-ohjelmointi. Hakupäivä 16.5.2017, http://www2.uef.fi/fi/anja.kareinen/olio-ohjelmointi.

Kemppainen, M. 2014a. Käyttöliittymäsuunnittelua käytännössä. Hakupäivä 22.5.2017, https://www.provianet.fi/kayttoliittymasuunnittelua-kaytannossa/.

Kemppainen, M. 2014b. Verkkosivusuunnittelun trendejä. Hakupäivä 24.5.2017, https://www.provianet.fi/verkkosivusuunnittelun-trendeja/.

Korpela, J & Linjama, T. 2005. Web-suunnittelu. Jyväskylä. Docendo Finland Oy.

Koskimies, K. 2000. Oliokirja. Helsinki. Satku.

Kuivanen, I. 2014. Responsiiviset sivustot. Hakupäivä 16.5.2017, http://users.metropolia.fi/~kuivi/bfish/bootstrap.php.

Kyrnin, J. 2017. Font Families Basics. Hakupäivä 23.5.2017, https://www.thoughtco.com/font-families-basics-3467382.

Laaksonen, A. 2011. PHP-ohjelmointi: Osa 14 - Olio-ohjelmointi. Hakupäivä 15.52017, https://www.ohjelmointiputka.net/oppaat/opas.php?tunnus=php_14.

Lehdonvirta, P & Korpela, J. 2013. HTML5 Sovellusalustana. Helsinki. RPS-yhtiöt.

Marjamäki, S. 2014. Viisi trendiä vuodelle 2015. Hakupäivä 23.5.2017, https://www.poutapilvi.fi/artikkelit/viisi-trendi%C3%A4-vuodelle-2015.

Microsoft. 2017. Model-View-Controller. Hakupäivä 19.5.2017, https://msdn.microsoft.com/en-us/library/ff649643.aspx.

NetBeans. 2017. Welcome to the NetBeans Community. Hakupäivä 24.5.2017, https://netbeans.org/about/index.html.

Niemi, T. 2015. Sovelluskehys – Verkkoräätälin työkalupakki. Hakupäivä 19.5.2017, https://www.sofokus.com/blogi/sovelluskehys-verkkoraatalin-tyokalupakki/.

Oulun seudun ammattikorkeakoulu. 2017. CodeIgniter –perusteet. Sisäinen Lähde. Hakupäivä 28.5.2017
https://moodle.oamk.fi/pluginfile.php/128658/mod_resource/content/2/CodeIgniter_perusteet.pdf.

Raittila, A. 2016. Hakukoneoptimointi lyhyesti. Hakupäivä 26.5.2017, http://nettibisnes.info/hakukoneoptimointi/.

Rantala, A. 2005. Web-ohjelmointi. Jyväskylä. Docendo Finland Oy.

Saarikumpu, O. 2016. JavaScript-kielen alkeet – osa 1. Hakupäivä 15.5.2017, http://weppipakki.com/js/opas/alkeet1.htm#mojs.

Technopedia. 2017. Create, Retrieve, Update and Delete (CRUD). Hakupäivä 26.5.2017, https://www.techopedia.com/definition/25949/create-retrieve-update-and-delete-crud.

The PHP Group. 2017. History of PHP. Hakupäivä 7.5.2017, http://php.net/manual/en/history.php.php.

Vahtolammi K 2010. MVC- Malli, peruskauraa frameworkkien käyttäjille. Hakupäivä 19.5.2017, https://vahtolam.wordpress.com/2010/09/23/mvc-malli-peruskauraa-frameworkkien-kayttajille/.

Webopas. 2017. Mikä on tietokanta. Hakupäivä 15.5.2017, http://www.webopas.net/mika_tietokanta.html.

Web Technology Surveys. 2017. Hakupäivä 24.5.2017, https://w3techs.com/technologies/details/pl-php/all/all.

Williams, O. 2016. The most popular JavaScript library, jQuery, is now 10 years old. Hakupäivä 31.5.2017, https://thenextweb.com/dd/2016/01/14/the-most-popular-javascript-library-jquery-is-now-10-years-old/.